# Postgres Pro Enterprise Manager 2.2 #

User Documentation for Postgres Pro Enterprise Manager 2.2.

## Postgres Pro Enterprise Manager Overview #

Postgres Pro Enterprise Manager (PPEM) is your all-in-one solution for streamlined database infrastructure administration. PPEM delivers an intuitive interface for managing and monitoring the Postgres Pro DBMS.

PPEM tools allow you to perform a wide range of routine DBMS maintenance tasks, ensuring seamless management of your Postgres Pro ecosystem across all operational areas — from backup and recovery to performance tracking, monitoring, troubleshooting, and beyond.

To get started with PPEM, refer to the Quick Start guide.

For more information about PPEM components and functionality, see the Architecture and Features sections.

Information about new releases and functionality upgrades can be found in the What is New section.

For more information on working with PPEM, refer to the following sections:

- Hardware and Software Requirements — server specifications for PPEM deployment and resource calculation guidelines
- Supported Platforms and Operating Systems — the list of supported platforms and operating systems
- Compatibility — the list of supported Postgres Pro Standard and Postgres Pro Enterprise versions, as well as browsers
- Security — access rights, networking, authentication, and RBAC
- Considerations and Limitations — important aspects to consider when using PPEM
- Installation and Setup — guides to various installation and configuration options
- Using PPEM — detailed step-by-step operating instructions
- Upgrade and Migration — instructions for upgrading to new versions
- Troubleshooting — self-diagnosis and troubleshooting guides

## Features #

PPEM enables you to perform the full cycle of Postgres Pro database deployment, operation, and maintenance tasks:

- **Automatically discovering DBMS instances**

  PPEM discovers active DBMS instances and automatically adds them to the web application, determines the backup tools used, and synchronizes the backup configuration with the web application. Automatic discovery features minimize the time required to reconfigure DBMS instances to operate with PPEM and help to quickly start using PPEM, as well as to facilitate migration from other management tools.

- **Installing and starting DBMS instances**

  PPEM allows you to deploy new DBMS instances using a browser. You can deploy single and cluster DBMS configurations, as well as deploy from backups.

- **Configuring DBMS instances**

  For new and existing DBMS instances, PPEM provides tools to view and set configuration parameters. This eliminates the need to connect to the instance and directly edit configuration files, accelerating the setup process.

- **Viewing and managing internal objects of DBMS schemas**

  PPEM allows you to view the internal structure of DBMS instances (tablespaces, databases, tables, indexes, and other objects) and also provides tools for creating new objects.

- **Tracking internal activity of DBMS instances**

  PPEM interacts with the statistics subsystem and provides information about the DBMS internal activity: active and background processes, query execution statistics, wait events, and locks. PPEM can also be used to cancel or forcefully terminate DBMS processes.

- **Visualizing running processes and analyzing query plans**

  PPEM integrates with `pgpro-otel-collector` that collects DBMS statistics and activity logs and provides metrics. PPEM acts as a consumer of this telemetry and provides graphs for evaluating and analyzing the DBMS performance.

- **Profiling and integrating with pgpro_pwr**

  PPEM integrates with `pg_profile` and `pgpro_pwr`. It also provides a convenient interface for working with snapshots and reports.

- **Planning, starting, and monitoring routine operations**

  PPEM includes tools for service tasks such as vacuuming, collecting scheduler statistics, reindexing, and others. These operations can be performed either manually or automatically on a regular basis using the task scheduler.

- **Backup and restore tasks**

  PPEM can automatically discover backup tools (pg_probackup), integrate with them, and provide viewing and management tools. PPEM allows you to perform backups both manually and on schedule, as well as create new DBMS instances from backups. Using the point-in-time recovery (PITR) allows for more flexible backup restoration tasks.

- **Directly accessing the DBMS instance console**

  PPEM allows you to connect to the instance console via the web version of psql. This tool is designed for experienced administrators and allows them to quickly connect to the DBMS and interact with it directly.

## Architecture #

This section describes the PPEM architecture and contains the following subsections:

- Description of Components
- High-Level Architecture
- Manager and Agent Architecture
- Monitoring Architecture
- Backup Architecture

### Description of Components #

The Postgres Pro Enterprise Manager architecture comprises the following components:

#### Web Application #

The web application is a set of components that provide a graphical user interface accessible through a web browser. The web application is installed on the same server as the PPEM manager. The user performs various actions using the browser, and the web application transforms these actions into requests and sends them to the manager. The manager processes requests and returns them to the web application. The web application transforms the response into various representations and displays it in the browser.

#### Manager #

The manager is a service that runs in the background. The manager accepts requests for DBMS infrastructure maintenance and has its own scheduler for performing regular service tasks. The manager accepts requests from the web application and transforms them into operations according to the designed logic.

To perform operations, you may need the following:

- Various service data, which is usually (but not necessarily) stored in the PPEM repository
- Executing instructions on the PPEM agent side

The manager reports the operation results to the web application upon completion.

> **Note**
>
> The operations performed by the manager can be synchronous or asynchronous. In case of synchronous operations, the manager is forced to wait until the operation finishes to obtain the response. In case of asynchronous operations, the manager immediately receives a response that the operation has been queued for execution. In this case, the manager needs to periodically check the operation status, but, in most cases, the user will receive a notification upon the operation completion.

#### Repository #

The repository is a database on a dedicated DBMS instance with a set of schemas and tables where the manager stores service data necessary for operation. When the manager starts, it establishes a connection with the repository and reads and writes data from and to it during operations. Repository availability is critical, since the manager cannot continue operating if the repository is unavailable.

### Agents #

Agents are services that must run on managed DBMS servers. Agents receive control instructions from the manager and, depending on the instruction type, perform various actions both in the operating system and in the DBMS instance. These actions may include running commands, creating directories and files, executing queries, etc. During or after execution, the execution result is sent to the manager to be recorded in the repository and/or subsequently processed according to the operation logic.

Agents also have an internal service task scheduler that regularly performs background tasks and sends their results to the manager.

One agent is enough on a standalone server to serve one or more DBMS instances.

### DMBS Instance #

A DBMS instance is a PPEM-managed object, i.e., the Postgres Pro DBMS in various editions (Postgres Pro Standard, Postgres Pro Enterprise). Several DBMS instances can be combined into clusters. These are typically streaming replication clusters.

Only one PPEM agent should interact with a single DBMS instance. Avoid scenarios where multiple agents work with the same DBMS instance, as this can lead to confusion and ambiguous behavior. From the DBMS instance standpoint, the agent is a regular application software that connects to the instance via the SQL interface using a predetermined DBMS account.

### External Services #

PPEM can use various external services to extend its features and capabilities. All these services are optional, and interaction with them is configured separately.

**Note**

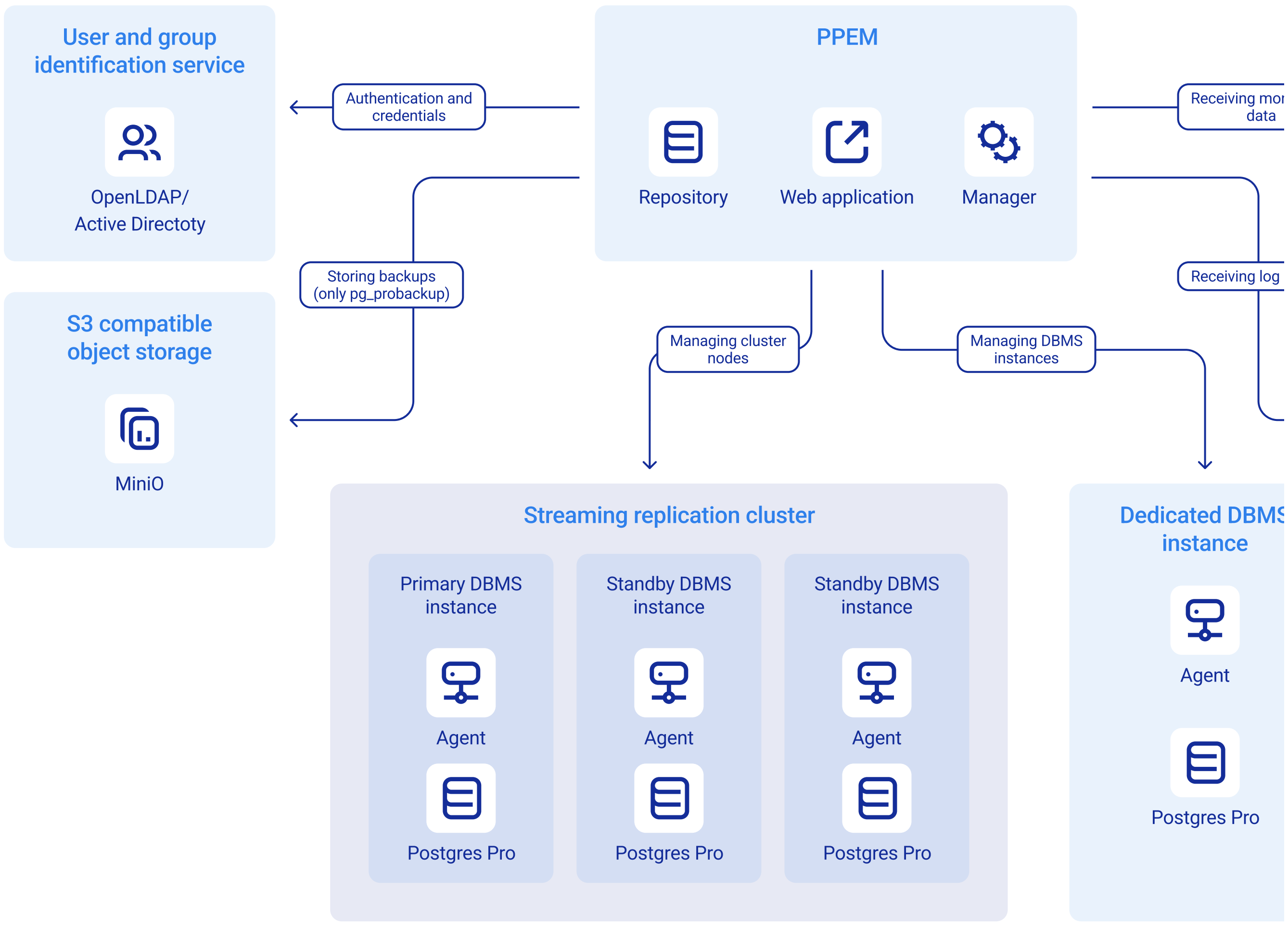
PPEM does not include tools for administrative management of external services (e.g., resource and configuration control).

The following external services are supported:

- **LDAP Directory** is a directory of users and groups that can be used to authenticate users in PPEM. PPEM supports OpenLDAP and Microsoft Active Directory.
- **S3-compatible object storage** — it is used by PPEM to store backups made with the `pg_probackup` utility.
- **OTLP-compatible metrics storage system** — a metrics storage system that supports the push model with the ability to record metrics via the OTLP protocol (for example, Victoriametrics) and/or the pull model capable of fetching metrics via the HTTP protocol (for example, Prometheus). PPEM can use such a system to receive metrics that are written there by the `pgpro-otel-collector` collector. PPEM supports Prometheus and Victoriametrics.
- **OTLP-compatible log storage system** — a metrics storage system capable of receiving logs via the OTLP protocol. PPEM can use such a system to receive DBMS logs that are written there by the `pgpro-otel-collector` collector. PPEM supports Elasticsearch.

## High-Level Architecture #

The diagram below illustrates an example of a high-level architecture and interaction between components:



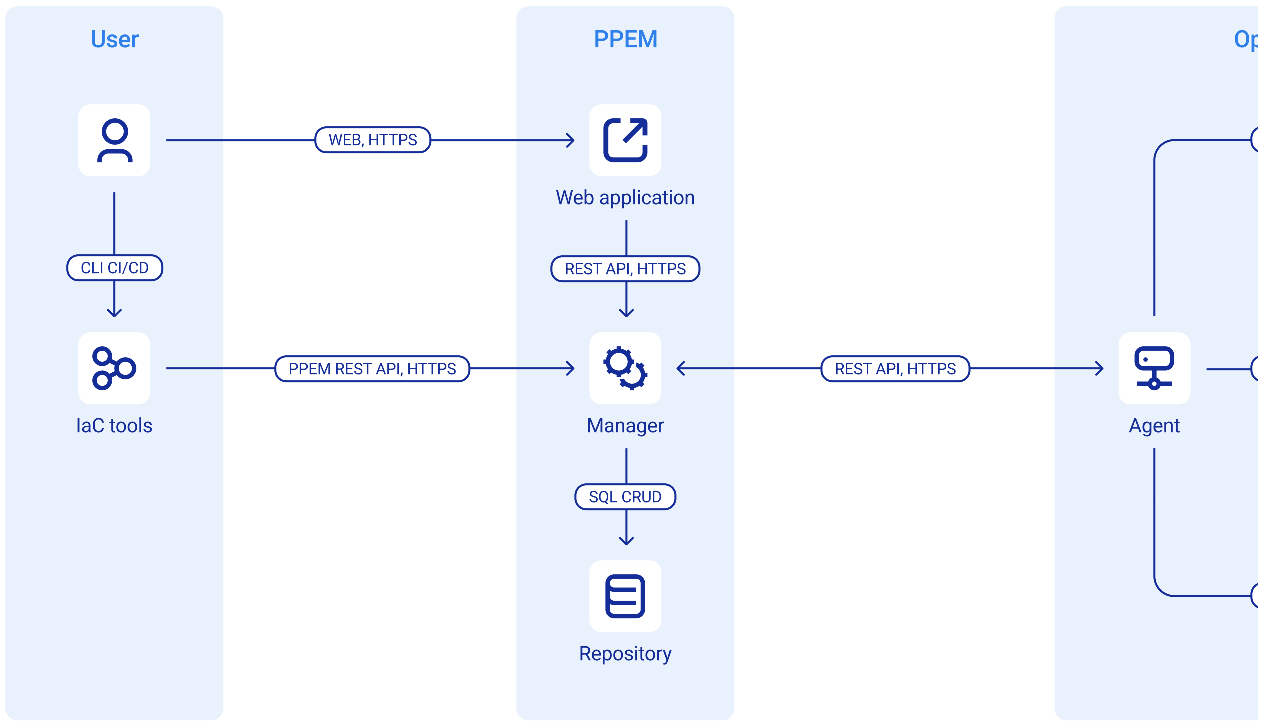
Where:

- **PPEM** is a DBMS infrastructure management service comprising the following components: **web application**, **manager**, **repository**, and **agents**.

- **A dedicated DBMS instance** is a PPEM-managed object that represents the Postgres Pro DBMS in various editions.
- The **replication cluster** is several DBMS instances combined into a cluster. This is typically a streaming replication cluster.
- **User and Group Identification Service** is a user and group directory that can be used to authenticate users in PPEM. This is an optional **external component**.
- **S3-compatible object storage** is a system for storing backups made with `pg_probackup`. This is an optional **external component**.
- **OTLP-compatible monitoring system** is a monitoring system capable of receiving metrics via the OTLP protocol or fetching them via HTTP. PPEM can use this system to receive DBMS performance metrics. This is an optional **external component**.

## Manager and Agent Architecture #

The diagram below illustrates an example of an architecture and interaction between the manager and the agent:



Where:

- The **user** can work with PPEM both through the web application in a browser and through automation tools (IaC), which can interact with PPEM via the REST API.
- In PPEM, the main graphical user interface is the **web application**. The web application is closely associated with the **manager**, from which it receives data and sends user control instructions. The manager provides an API for obtaining data and managing the DBMS infrastructure. The manager stores the intermediate state of the infrastructure in the **repository** database and interacts with the **agent**, which manages the **DBMS instance**.
- **Operating system** is the working environment in which the DBMS instance and the PPEM agent are started. The agent interacts with the manager: sends environment data (OS and DBMS instance information) and receives control instructions.

## Monitoring Architecture #

To provide monitoring functions related to Metrics and Logs (hereinafter Telemetry), PPEM uses the `pgpro-otel-collector` collector from Postgres Pro.

Working with telemetry can be organized in two ways:

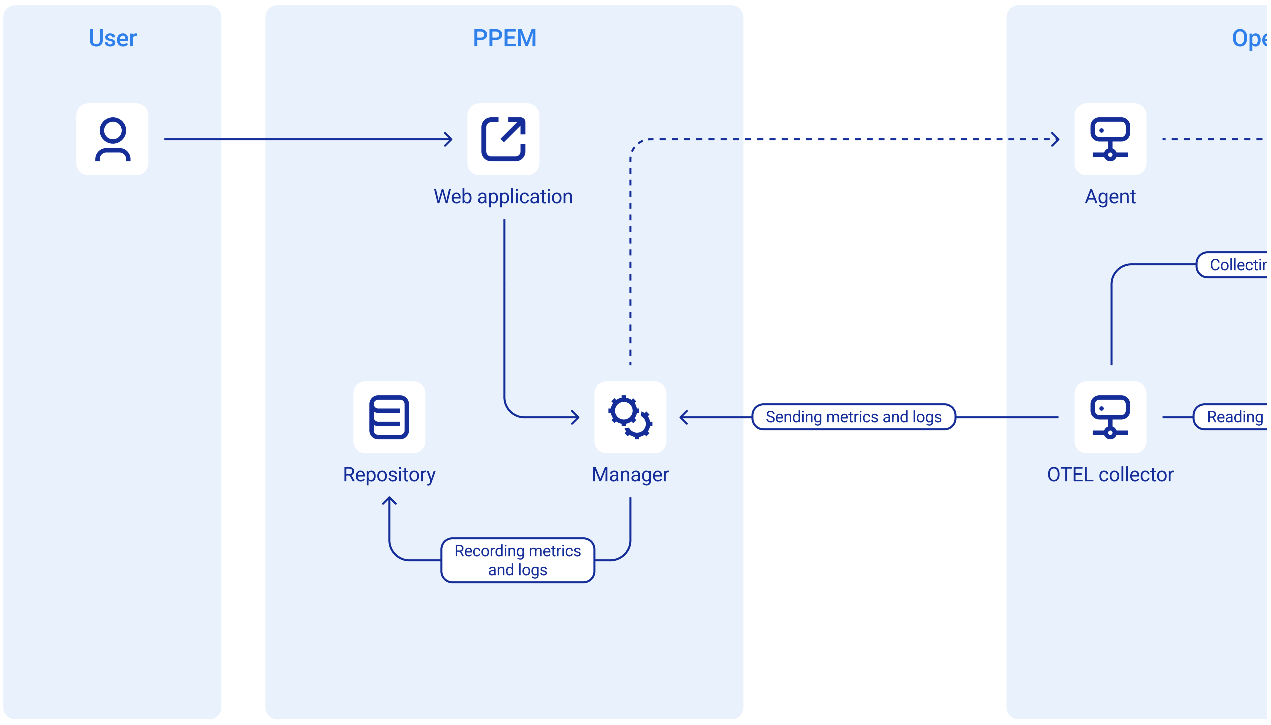
- **Internal storage**. The **repository** databases are used to store telemetry: metrics are stored in the separate `monitoring` schema, and logs are stored in the `logs` schema.
- **External storage**. External data storages (in relation to PPEM) can be used to store telemetry.

In both cases the `pgpro-otel-collector` collector is the required component. The collector gathers statistics and logs, generates telemetry data, and, depending on the selected mode, exports data to the manager or to an external storage system.

> **Warning**
>
> Using the internal storage is not recommended in production environments, as it has limited scalability and potential performance issues if data volumes grow. Writing and reading large amounts of telemetry can negatively impact PPEM overall performance when accessing the repository database.

### Internal Storage #

The diagram below illustrates an example of a monitoring architecture that uses an internal storage:

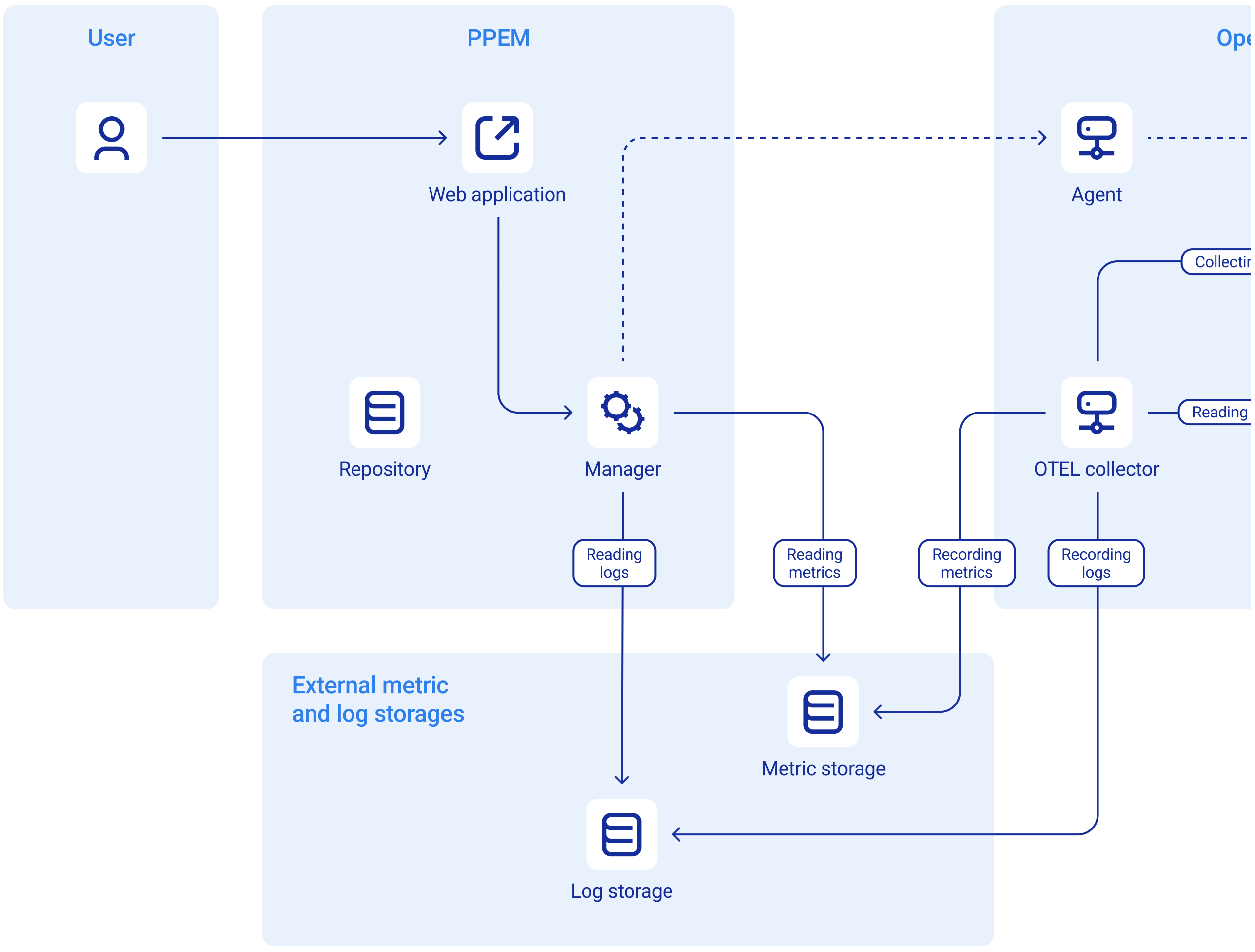


Where:

- The **user** works with PPEM through the web application in a browser where they view graphs of metrics and logs.
- In PPEM, metrics and logs are stored in the **repository**. The **manager** receives metrics and logs from the collector and stores them in repository databases. It also requests metrics and logs from the repository databases and returns them to the consumer via the REST API. The **web application** queries the manager for metrics and logs and then visualizes the received data.
- In the **operating system**, the `pgpro-otel-collector` collector connects to the DBMS instance to receive metrics, reads the DBMS log files, processes the received data, and sends it to the manager via the REST API. The **agent** is a standalone component that provides control functionality and does not collect metrics or logs.

## External Storage #

The diagram below illustrates an example of a monitoring architecture that uses an external storage:

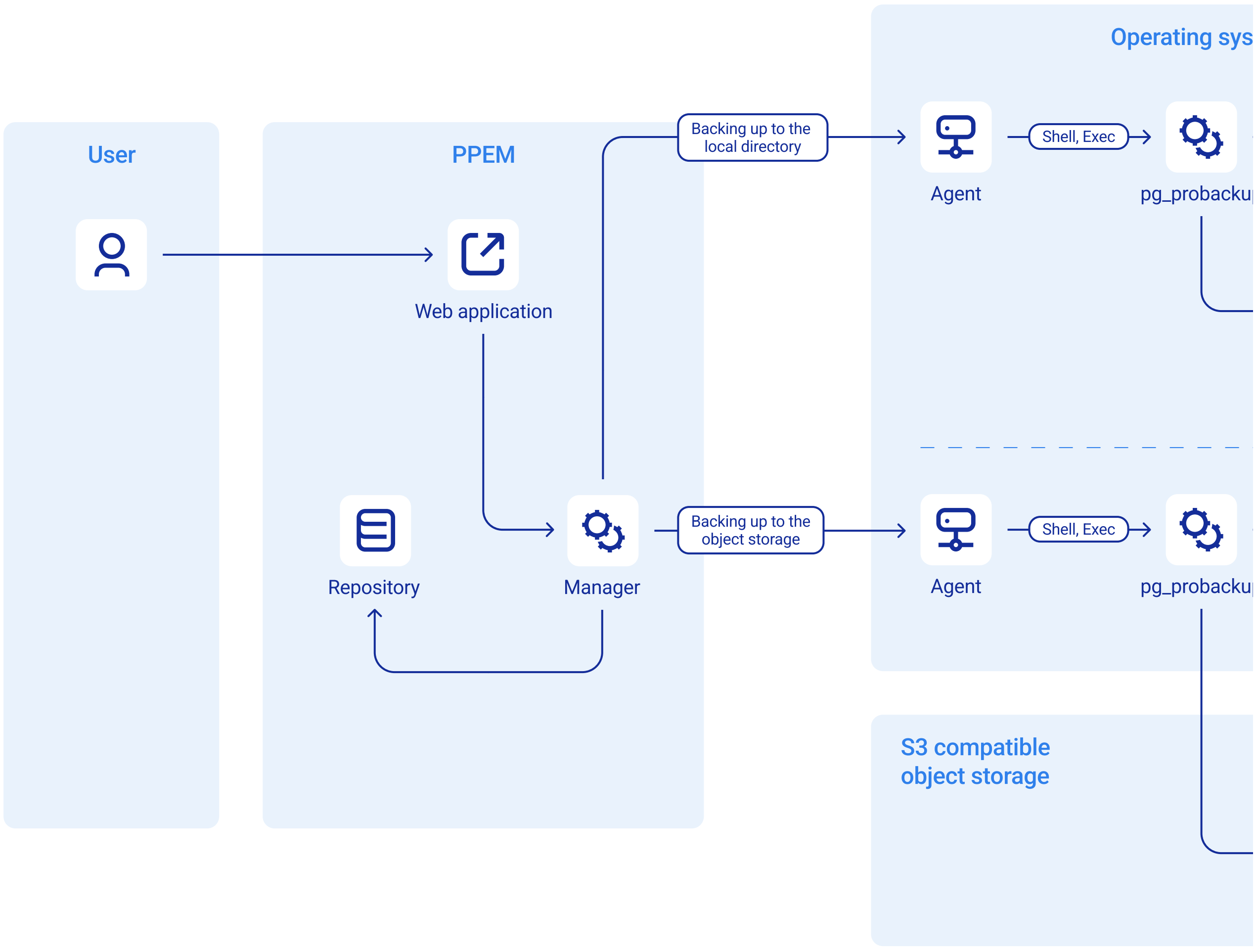


Where:

- The **user** works with PPEM through the web application in a browser where they view graphs of metrics and logs.
- In PPEM, the **manager** requests metrics and logs from external storages and returns them to the consumer via the REST API. The **web application** queries the manager for metrics and logs and then visualizes the received data.
- In the **operating system**, the `pgpro-otel-collector` collector connects to the DBMS instance to receive metrics, reads the DBMS log files, processes the received data, and sends it to external storages via the REST API. The **agent** is a standalone component that provides control functionality and does not collect metrics or logs.
- **External storages for metrics and logs**:
  - A dedicated monitoring system can act as a metrics storage. Prometheus, Victoriametrics, and OTLP-compatible storages are supported.
  - A separate system with OTLP-compatible storage can serve as a log storage. Elasticsearch is supported.

### Backup Architecture #

The diagram below illustrates an example of a backup architecture:



Where:

- The **user** utilizes backup and restore functionality via the web application.
- In PPEM, the **web application** sends requests to the **manager**. The manager sends backup and restore requests to **agents**.
- In the **operating system**, agents perform backup or restoration using the `pg_probackup` utility. A backup can be saved either to a local directory or to an S3-compatible object storage. The backup directory must be preconfigured to work with `pg_probackup`. In case of recovery, the backup can be obtained from either the local directory or from the object storage.
- **S3-compatible object storage** is an external service. Specific object storages are supported by means of `pg_probackup`, the support does not depend on PPEM.

## Hardware and Software Requirements #

This section provides hardware and software requirements for PPEM, as well as information on resource calculation when scaling the DBMS.

### Minimum hardware and software requirements #

Below are the minimum hardware and software requirements for servers where the PPEM manager and agent will be deployed.

The following can act as a server:

- Bare metal server
- A Xen/KVM/VMWare-based virtual machine
- Container environment in privileged mode.

**Warning**

Using PPEM at minimum requirements does not provide fault tolerance and is recommended only for pilot or experimental projects, low-load environments without high availability and fault tolerance requirements.

**For the manager:**

One server with the following minimum configuration:

- CPU — 2 pcs.
- RAM — 4 GB
- SSD — 40 GB
- Platform — x86_64 or any listed in the Supported Platforms and Operating Systems section
- Operating system — Linux or any listed in the Supported Platforms and Operating Systems section

**For the agent:**

One server with the following minimum configuration:

- CPU — 2 pcs.
- RAM — 2 GB
- SSD — 10 GB
- Platform — x86_64 or any listed in the Supported Platforms and Operating Systems section
- Operating system — Linux or any listed in the Supported Platforms and Operating Systems section

### Resource Calculation #

**For the manager:**

The required resources for the manager for 10 DBMS instances with 10 thousand objects in a separate instance:

- CPU — 2 pcs.
- RAM — 2 GB
- SSD — 40 GB

The minimum configuration of the server where the manager will be deployed (up to 10 instances with 10 thousand objects in a separate instance):

- CPU — 2 pcs.
- RAM — 4 GB
- SSD — 40 GB

Resource calculation table:

| Number of instances | CPU, pc. | RAM, GB | SSD, GB |
|---|---|---|---|
| 10 | 2 | 4 | 40 |
| 30 | 4 | 12 | 80 |
| 70 | 8 | 28 | 160 |
| 150 | 16 | 60 | 320 |
| 230 | 24 | 92 | 480 |
| 310 | 32 | 124 | 640 |

**For the agent:**

The required resources for the agent for 1 DBMS instance (up to 100 thousand instance objects):

- CPU — 1 pc.
- RAM — 512 MB
- SSD — 20 GB

The minimum configuration of the server with the DBMS instance where the PPEM agent will be deployed (up to 100 thousand instance objects):

- CPU — 2 pcs.
- RAM — 2 GB
- SSD — 20 GB

Resource calculation table:

| Number of DBMS objects | CPU, pc. | RAM, GB | SSD, GB |
|---|---|---|---|
| 100 thousand | 2 | 2 | 10 |
| 200 thousand | 3 | 4 | 20 |
| 400 thousand | 4 | 8 | 20 |

## Supported Platforms and Operating Systems #

The PPEM manager and agent support x86_64-based Linux operating systems.

PPEM supports installation and operation on the following Linux distributions:

- Alt Linux 10
- Alt Linux 11
- Alt Linux SP 8.2
- Alt Linux SP 8.4
- Alt Linux SP 10
- Astra Linux Smolensk 1.7
- Astra Linux Smolensk 1.8
- Debian 11
- Debian 12
- Red OS 7.3
- Red OS 8.2
- Red Hat Enterprise Linux 8
- Red Hat Enterprise Linux 9
- ROSA 2021.1
- SUSE Linux Enterprise Server 15
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

  **Note**

  PPEM operation has not been tested with distributions other than those listed above, and proper operation on them is not guaranteed.

  PPEM operation on non-x86_64 platforms has not been tested, and proper operation on such platforms is not guaranteed.

## Compatibility #

### Compatibility with Postgres Pro #

The PPEM manager supports the following versions of Postgres Pro Standard and Postgres Pro Enterprise:

- 17
- 16
- 15
- 14

The PPEM agent supports the following versions of Postgres Pro Standard and Postgres Pro Enterprise:

- 17
- 16
- 15
- 14
- 13
- 12

### Compatibility with pg_probackup #

The PPEM agent supports the following versions of pg_probackup:

- 2.8

### Compatibility with pgpro-otel-collector #

The PPEM agent supports the following versions of pgpro-otel-collector:

- 0.3

### Browser Compatibility #

To get the most out of the PPEM web application, consider the following:

- Use a screen with a minimum resolution of 1280×720. A resolution of 1440×900 or higher is recommended for best experience.
- Use the following web browsers:
  - Chrome 87 or later
  - Firefox 85 or later
  - Opera 73 or later
  - Safari 14 or later

## Security #

### Manager and Agent #

The manager is a standard application software and does not require privileged access to operating system features. The manager service can operate fully when running under a non-privileged operating system user.

To work with the repository, the manager needs a separate database where service information will be stored. Also, the DBMS user with the following rights is required:

- The right to `LOGIN` to the instance;
- The right to connect to the database, which will be the repository
- The owner of this database
- Without restrictions on access rights within this database (to perform migrations in the data schema)

The agent is a standard application software that requires the following for full operation:

- Access to operating system features
- Access to the managed DBMS instance

To implement most features, the agent only requires the access level of a non-privileged operating system user. There is a small number of features that require privileged access. To maintain this functionality, additional system configuration and granting of the necessary rights are required. Without the required configuration and permissions, the agent will be unable to perform operations, which will adversely affect PPEM functionality. It is recommended to complete all necessary configurations before running the agent.

Access to the managed DBMS instance can be divided into the following parts:

- **Access to files and directories of the DBMS instance**, which is provided using operating system access levels — the user on whose behalf the agent is running must have access to the main data directory.

  **Note**

  By default, the main data directory is initialized by the `postgres` owner with `0600` rights, so most DBMS installations restrict access to this configuration. Therefore, the optimal operational approach is to run the agent under the `postgres` system user.

- **Access to the SQL interface of the DBMS instance**, for which the agent requires the DBMS user with the following rights:
  - The right to `LOGIN` to the instance
  - The right to connect to all instance databases
  - A member of the `pg_monitor` and `pg_signal_backend` roles

### Networking #

Networking between the manager and agent can be initiated by either side:

- The manager can send control instructions to agents, and agents return the execution results in response.
- Agents can send to the manager requests to register themselves and DBMS instances as well as to update the state of instances. The manager, in turn, must send a response to the agents' requests.

The manager and agent use the HTTPS protocol for communication. By default, the manager uses the tcp/8080 port, and the agent uses the tcp/8081 port. This traffic direction should be taken into account when configuring network access rules. The address and port are set in the main configuration files of the manager and agent.

For secure data transfer, TLS configuration is recommended.

### User-to-Manager Authentication #

To work with the manager, user authentication and authorization are required. Authentication can be performed using:

- Built-in PPEM tools — user and group data is stored in the repository and managed by the PPEM administrator.
- An external LDAP directory (OpenLDAP or Microsoft Active Directory). If you use the external directory, the manager must be configured to work with this directory. All user and group data is stored in an external directory and managed by a dedicated LDAP directory administrator.

### Manager-to-User Authentication #

The manager and agent API is secured by authorization. To execute API requests, the manager and agent perform mutual authentication and issue access tokens for subsequent authorization. The tokens have a limited lifetime — the manager and agents autonomously track token expiration and renew them when necessary.

## Authentication and Authorization Operation Scheme #

There are three interaction types that require authentication and authorization during the manager's operation:

- User - Manager
- Manager - Agent
- Agent - Manager

All interactions are carried out via the HTTP or HTTPS protocol depending on the PPEM parameters.

### User - Manager #

In this interaction type, users work in the web application.

#### User - Manager: Authentication #

The user sends a `POST /v1/sessions` API request to the manager's endpoint to obtain access and refresh tokens for subsequent operations. The API request contains user credentials.

In the basic scenario, credentials are checked against the repository. When using LDAP authentication, credentials are first checked against the directory service, then if they are not found, they are checked against the repository.

#### User - Manager: Authorization #

Upon successful authentication, subsequent HTTP/HTTPS requests from users to the manager include the `Authorization` header, where the `Bearer` keyword is followed by a space and the access token text. For example:

```
headers:
  Content-Type: application/json
  Authorization: "Bearer eyJhbG..."
```

The granted access is determined using the role-based access control (RBAC) model in accordance with the roles assigned to the user:

- You can assign roles to PPEM users directly or through PPEM groups they are members of.
- You can assign roles to LDAP users only through PPEM groups they are members of.

  To add an LDAP user to a PPEM group, the administrator must map the distinguished name (DN) of the user's LDAP group to the PPEM group name. The user will then be automatically added to the PPEM group when logging in to the web application .

  The user group composition is periodically checked against the LDAP server and updated if necessary.

  For more information about mapping LDAP groups to PPEM groups, see Integration with OpenLDAP and Active Directory .

### Manager - Agent #

In this interaction type, the manager sends API requests to agents to perform various operations.

#### Manager - Agent: Authentication #

The manager sends a `POST /v1/sessions` API request to the agent's endpoint to obtain access and refresh tokens for subsequent operations. The API request is made via the URL for connecting the agent to the manager from the repository and contains two parameters:

- `name`: The agent name from the repository.
- `api_key`: The API key for connecting the agent to the manager from the repository.

#### Manager - Agent: Authorization #

Upon successful authentication, subsequent HTTP/HTTPS requests from the manager to agents include the `Authorization` header, where the `Bearer` keyword is followed by a space and the access token text. For example:

```
headers:
  Content-Type: application/json
  Authorization: "Bearer eyJhbG..."
```

The agent verifies the authenticity of the manager's access token. To do this, the agent generates a token based on the known values of the `name` and `api_key` parameters. If the generated and received tokens match, the authentication is successful. If the authenticity is confirmed and the requested endpoint is found, further work is allowed.

### Agent - Manager #

In this interaction type, agents send reports on the API command execution and updates about serviced instance objects to the manager.

#### Agent - Manager: Authentication #

The agent sends a `POST /v1/sessions` API request to the manager's endpoint to obtain access and refresh tokens for subsequent operations. The API request is made via the URL for connecting the agent to the manager, which is specified in the `ppem-agent.yml` agent configuration file using the `agent.manager.url` parameter. The API request contains two parameters:

- `name`: The agent name, which is specified in the `ppem-agent.yml` agent configuration file using the `agent.name` parameter.
- `api_key`: The API key for connecting the agent to the manager, which is specified in the `ppem-agent.yml` agent configuration file using the `agent.manager.api_key` parameter.

#### Agent - Manager: Authorization #

Upon successful authentication, subsequent HTTP/HTTPS requests from the agent to the manager include the `Authorization` header, where the `Bearer` keyword is followed by a space and the access token text. For example:

```
headers:
  Content-Type: application/json
  Authorization: "Bearer eyJhbG..."
```

The manager verifies the authenticity of the agent's access token. To do this, the manager generates a token based on the known values of the `name` and `api_key` parameters. If the generated and received tokens match, the authentication is successful. If the authenticity is confirmed and the requested endpoint is found, further work is allowed.

The granted access is determined by the rules specified in the PPEM code for how the agent should access manager resources.

### Lifetime of Access and Refresh Tokens #

The lifetime of access and refresh tokens is limited. It can be specified in the manager's `ppem-manager.yml` or agent's `ppem-agent.yml` configuration file using the `jwt.lifetime.access` and `jwt.lifetime.refresh` parameters.

When the access token expires, the manager or agent starts responding to the token owner with the `401 Unauthorized` error and the `ERR_AUTH_TOKEN_EXPIRED` code, for example:

```
{
  "error":{
    "code":"ERR_AUTH_TOKEN_EXPIRED",
    "title":"token is expired"
  }
}
```

To get a new access token, the token owner should send a `PUT /v1/sessions` API request to the endpoint with the refresh token that was received along with the expired access token. This will result in new access and refresh tokens that can be used for further work.

If the refresh token has also expired, the token owner must re-authenticate.

### Role-Based Access Control (RBAC) Model #

#### Key Concepts #

Access control in PPEM is implemented via the RBAC (Role-Based Access Control) model, which defines access restriction rules through roles and privileges. The model establishes the following basic conventions:

- *Object* — a resource to which access should be granted or restricted.
- *Subject* — a person (user) or an automated agent.
- *Privilege* — a permission granting access to perform an operation with the object.

- *Role* — a job function or title defined at the authorization level.

The basic conventions define the following extended conventions:

- *Access rule* — a combination of a role, privileges, and their relationships.
- *Session* — an association between a subject and a role.
- One subject can have several roles.
- One role can belong to several subjects.
- One role can have several privileges.
- One privilege can belong to several roles.

  **Note**

  Privileges are not assigned to subjects directly but are acquired by subjects through roles.

## RBAC in PPEM #

Fundamental principles:

- PPEM functionality is composed of *plugins*.
- Each plugin helps to manage a specific *resource*.
- Resources can be managed by basic *CRUD operations*: create, view, edit, and delete.
- Additionally, resource management may include various *RPC operations*. Their names depend on the resource management operation.
- *Privileges* allow subjects to perform certain operations. All resources are managed through privileges (basic minimum requirements):
  - `<RESOURCE>\_create`
  - `<RESOURCE>\_view`
  - `<RESOURCE>\_edit`
  - `<RESOURCE>\_delete`
- Privileges are included in *roles*.
- Roles can be assigned to a subject both at the time of subject creation and later.
- When the plugin receives a request from a subject to perform an operation, it checks the subject's permission to access the requested operation.
- Subjects (users) can create their own roles (if they have permissions) and assign these roles to other subjects.
- Privileges and roles may be object-specific. This means that you can specify a role and privilege that allows access to a limited set of objects.
- Binding to an object can be done at the role level (all subjects with this role will have access to the object) or at the user level (only one user will have access to the object).

### Subjects #

The following may act as subjects:

- Users. Roles are assigned to users at creation time. If roles are not explicitly specified, a default role may be assigned.
- Agents. This role is assigned when creating or registering agents.

### Objects #

Objects can be either resources or their representations in the repository, such as servers, agents, instances, users, and user groups.

### Setting Privileges and Roles #

On the first PPEM startup, the repository initialization is performed, during which system tables are populated (privileges and roles).

During initialization, each plugin is assigned its own set of privileges, roles, and "role-privilege" associations. For example, the `accounts` plugin has dedicated privileges and roles for managing access to its objects. Other plugins receive their own sets of privileges and roles to control access to their respective objects.

If users are granted necessary privileges, they can create custom roles, specify privilege sets, link roles to objects, and assign custom roles to subjects.

## Implementation #

Access permissions are implemented by the following set of tables, which are owned by the `accounts` plugin:

- `privileges` — privileges with object classes they regulate access to.
- `roles` — roles with target object classes.
- `role_privileges` — the "role-privilege" association defines relationships between roles and privileges included in them.
- `users` — users of the system (subjects).
- `user_roles` — the "user-role" association that defines relationships between users and roles assigned to them.
- `user_privileges` — the view that displays the "user-privilege" associations.
- `groups` — the groups of the system.
- `group_roles` — the "group-role" association defines relationships between groups and roles assigned to them.
- `group_users` — the "group-user" association defines relationships between groups and users included in them.

### The `privileges` Table #

The table has the following fields:

- `id`: The unique privilege ID.
- `name`: The unique privilege name.
- `title`: The privilege description.
- `class`: The class of objects to which the privilege grants access.
- `source`: The name of the plugin that sets the privilege and then performs the access check.

The table is populated by the PPEM manager using migrations. Each plugin determines its own set of privileges.

An HTTP middleware handler is used to check privileges. Creation, modification, and deletion of privileges by the user is not supported, since the privilege check is implemented in the PPEM manager code.

### The `roles` Table #

The table has the following fields:

- `id`: The unique role ID.
- `name`: The unique role name.
- `title`: The role description.
- `class`: The class of objects to which the privilege grants access. The field value is used as a UI hint to get privileges of the corresponding class and the list of objects of this class.
- `source`: The name of the plugin that sets the role. For custom roles, the `user` value is set.

Roles are created by the manager using migrations. The basic role set is established for the `accounts` plugin.

With the manager API, users can create, modify, and delete custom roles, but they cannot modify or delete roles set by the PPEM manager (system roles).

### The `role_privileges` Table #

The table defines associations between roles and privileges and has the following fields:

- `id`: The unique association ID.
- `role`: The role ID.
- `privilege`: The privilege ID.
- `parametric`: Whether object binding is used.
- `object`: The unique ID of an arbitrary class object.

The object class is determined by the `privileges.class` field value. The object ID declared here restricts the privilege specified in `role_privileges.privilege` to a single object and allows access to the object only by members of a single role specified in `role_privileges.role` (access is granted to users in `user_roles.user` who have `user_roles.role` = `role_privileges.role`).

The combination of `role`, `privilege`, and `object` is a unique key with the `object IS NOT NULL` condition.

> **Note**
>
> A parameterized object-specific “role-privilege” association determines access only to a single object (when specifying `object`). If access is required for N objects, N entries in the `role_privileges` table are created.

### The `users` Table #

The table has no RBAC-related fields.

When creating a user via the manager API, you can specify a list of role IDs that will be assigned to the user.

### The `user_roles` Table #

The table defines associations between users and roles and has the following fields:

- `id`: The unique association ID.
- `user`: The user ID.
- `role`: The role ID.
- `object`: The unique ID of an arbitrary class object.

The object class is determined by the `privileges.class` field value. The object ID declared here restricts the privilege specified in `role_privileges.privilege` to a single object and allows access to the object by the user specified in `user_roles.user`.

> **Note**
>
> A parameterized object-specific “role-privilege” association determines access only to a single object (when specifying `object`). If access is required for N objects, N entries in the `role_privileges` table are created.

### The `user_privileges` View #

This view shows users with their roles and privileges and makes it easier to check privileges of a specific subject:

- `user`: `user_roles.user`
- `role`: `user_roles.role`
- `privilege`: `privilege.name`
- `object`: `user_roles.objects` or `role_privileges.object`

### The `groups` Table #

The table has no RBAC-related fields.

When creating a group, you can specify a list of role IDs that will be assigned to the group and, as a result, to all users included in it.

### The `group_roles` Table #

The table defines associations between groups and roles and has the following fields:

- `group_id`: The group ID.
- `role_id`: The role ID.
- `object`: The unique ID of an arbitrary class object.

The object class is determined by the `privileges.class` field value. The object ID declared here restricts the privilege specified in `role_privileges.privilege` to a single object and allows access to the object by the user specified in `group_roles.role_id`.

> **Note**
>
> A parameterized object-specific “group-role” association determines access only to a single object (when specifying `object`). If access is required for N objects, N entries in the `group_roles` table are created.

### The `group_users` Table #

The table defines associations between groups and users and has the following fields:

- `group_id`: The group ID.
- `user_id`: The user ID.

The “group-user” associations are not object-specific.

## Object-Specific Roles #

In general, it is assumed that roles can include privileges allowing access to all objects of any class.

For finer-grained access control, you can specify both the class and the object ID in `role_privileges.object`. This ensures that the role and privilege apply only to the object with the specified ID. The object can be specified in multiple locations:

- In `role_privileges.object` for a role. In this case, object access is granted to all role members.
- In `user_roles.object` for a user. In this case, access is granted only to a single user.
- In `group_roles.object` for a group. In this case, access is granted only to group users.

## Checking Access Permissions of Subjects #

During the subject authentication, a session itself and session JWT tokens are created. When an access token is created, it includes the `user_id`. When a user makes requests to perform operations, the access token is attached to the request headers.

The manager performs authorization, extracts the user’s `user_id` from the access token and checks for the required privilege in roles. If the privilege is available, access is granted. If the privilege is unavailable, the request is rejected.

See below for details.

The following is required to check that a subject has the permission to perform an operation with an object:

- `user_id` or `agent_id`: The user or agent ID.
- `class`: The class name for the object (resource type). `object`: The object ID (optional).

The client specifies the `Authorization: Bearer` header in the request and attaches the access token.

The server receives the request and extracts the data needed to verify access:

- The `user_id` (or `agent_id`) value is extracted from the access token.
- The `class` value is defined based on the privilege.
- The `object` values:
  - For GET requests of the `/resources` type, the values are extracted from the URL and request parameters ( `ids=?`).
  - For GET requests of the `/resources/objectID` type, the values are extracted from the URL.
  - For PUT requests, the values are extracted from the request body.
  - For DELETE requests of the `/resources` type, the values are extracted from the request body (a separate object field).

To verify that a user has a specific privilege, the handler may need a mapping of operation IDs to their corresponding privileges. The verification is performed via the repository and the `user_privileges` view.

### Considerations and Limitations #

Using security-enhancing tools that add extra restriction layers may interfere with some PPEM features. PPEM's normal operation is neither guaranteed nor recommended when using such tools. These security-enhancing tools include:

- SELinux mandatory access control (MAC) systems. For details, see the official SELinux documentation.
- Information protection suites in Astra Linux Special Edition. For details, see the official Astra Linux documentation.

## What is New #

This section contains key information about PPEM releases, along with functionality updates and improvements.

For migration instructions and recommendations, refer to the Upgrade and Migration section.

For more information about PPEM versions, see the following sections:

- What is New in PPEM 2.1
- What is New in PPEM 2.0

### What is New in PPEM 2.2 #

Release date: 28 July 2025

For migration instructions and recommendations, refer to the Upgrading to PPEM 2.2 section.

PPEM 2.2 introduces the following functional upgrades:

- **Control Center**

  The PPEM user interface now includes the **Control Center** section, which provides quick status assessment for all managed systems. This tool offers a convenient visual overview of all PPEM-controlled instances and allows viewing historical data for key instance metrics in graph format.

  For more information, refer to the Control Center section.

- **Session Wait Profile Visualization**

  The interface for the `pg_wait_sampling` extension is added, providing tools to view and analyze session wait profiles and history. The new section helps to identify dependency issues for queries processing longer than expected.

  For more information, refer to the Viewing Statistics on Wait Events section.

- **Detection of Patroni-Managed Clusters**

  The system now implements automatic detection of clusters managed by the Patroni failover manager, with subsequent display in the **Clusters** section of the web application.

### What is New in PPEM 2.1 #

Release date: 15 April 2025

For migration instructions and recommendations, refer to the Upgrading to PPEM 2.1 section.

PPEM 2.1 introduces the following functional upgrades:

- **Working with Replication Clusters and Integration with the BiHA Failover Manager**

  The ability to work with streaming replication clusters is added. The PPEM interface now supports creating a cluster from an existing instance and displaying it in the corresponding interface section. PPEM 2.1 takes the first step toward full BiHA cluster support. The new version displays existing BiHA clusters and provides functionality to remove nodes from an existing cluster.

- **Managing Data Sources**

  In the **Infrastructure** → Data sources** section of the PPEM web application, you can now create external storages for metrics and message logs collected via pgpro-otel-collector. For metrics storage, PPEM supports integration with Prometheus, while for message log storage, it supports integration with Elasticsearch. Using external storage for metrics and logs is recommended because it reduces the load on the PPEM service repository by offloading additional data storage. It also eliminates the need for DBAs to perform additional scaling or closely monitor the PPEM service repository. External storage systems are specifically designed to handle large volumes of specific data types and can be scaled and managed independently of PPEM.

  For more information about configuring the integration, refer to the Integration with External Data Sources section.

- **Custom SQL Metrics**

  It is now possible to create and view custom SQL metrics based on SQL statements planning and execution statistics collected by the `pgpro_stats` extension. When creating or editing an SQL metric, the user can select a specific database, specify the user on behalf of which the query will be executed, and set the metric collection interval. The list of created metrics will be displayed on the page, and metric values can be viewed by clicking on the metric name.

- **Navigation Panel Redesign**

  The navigation panel is redesigned based on user experience feedback from infrastructure and application DBAs. Pages primarily intended for infrastructure DBAs are now grouped under the **Infrastructure** section, while the **Databases** section, mainly used by application DBAs, is promoted to the top navigation level for convenient and quick access to the database list.

- **Verifying the Consistency of Instance Data Directory**

  The system now supports instance data directory verification using the `checkdb` command included in the `pg_probackup` extension. This command performs a physical verification of all data files in the directory, validating page headers integrity and verifying block-level checksums.

- **Adding the "Console User" Role"**

For flexible control over access to the psql web console, PPEM 2.1 introduces the “Console user” role. Now, only PPEM users with this role can access the psql console from the PPEM interface to perform database operations.

- **Installing and Working with Extensions**

  It is now possible to install and work with Postgres Pro extensions in the PPEM interface. The DBMS instance navigation panel now includes the new **Extensions** section that displays the list of installed extensions that can be deleted, as well as the **Install extension** block for adding extensions to specific databases.

### What is New in PPEM 2.0 #

Release date: 19 February 2025

For migration instructions and recommendations, refer to the Upgrading to PPEM 2.0 section.

PPEM 2.0 introduces the following functional upgrades:

- **Migration to Golang**

  The manager and agent components migrated to Golang, enhancing performance, scalability, and extensibility capabilities to handle increasing workloads and growing numbers of managed systems. The new manager can serve a larger number of agents while consuming less memory and CPU resources. New agents also use less memory and CPU resources during routine operations.

- **Automatic Resource Detection**

  An agent can now work in automatic detection mode. In this mode, an agent searches for DBMS instances, automatically registers them in PPEM, and continuously maintains up-to-date information about each DBMS instance. Automatic detection and registration simplify PPEM deployment in environments with numerous instances and minimize manual operations required to add agents and database instances in PPEM.

- **Integration with `pgpro-otel-collector`**

  Postgres Pro OpenTelemetry Collector is designed to collect metrics and activity logs from DBMS instances and send them to monitoring systems. PPEM now integrates with Postgres Pro OpenTelemetry Collector, enabling the use of collector-gathered metrics for monitoring purposes. Integration can be performed in two ways:

  - The first method involves PPEM receiving metrics and logs from the collector via the OTLP protocol and storing them in the repository internal database.
  - The second method involves integration with external log and metrics storage systems, allowing to get monitoring data from these sources. External storage integration allows PPEM to incorporate into existing monitoring infrastructure and use it for internal monitoring functions.

  For more information, refer to the Monitoring Architecture.

- **API Redesign**

  The API was redesigned, with initial steps taken toward making it publicly available for broader use. The API is designed following REST standards, providing developers with a familiar framework to manage PPEM resources: creating new ones, viewing existing ones, as well as editing and deleting them. Future API publication will enable PPEM integration with automation systems, enhancing automated tools for DBMS infrastructure maintenance.

## Quick Start #

This section explains how to install PPEM on a server according to the all-in-one deployment scheme. Following the provided instructions, you will test the installation process and get a minimal working version of PPEM suitable for demonstration purposes.

When executing the commands provided in this section, consider the following:

- The Debian Linux operating system commands are specified. For other operating system versions, use corresponding commands.
- Standard object names are specified in the commands, for example, `ppem` for the repository database. If required, you can specify different names.

The installation process includes the following steps:

1. Ensure that prerequisites are met.
2. Configure the manager.
3. Configure the agent.

PPEM will be installed. You can update the browser page with the web application and start working.

### Prerequisites #

Before installation, read the following information and perform the required actions:

1. Prepare the server where PPEM will be installed according to hardware and software requirements.

2. Install the Postgres Pro DBMS instance on the server. For more information about installation, refer to the official Postgres Pro documentation.

3. Start a new session under the superuser:

   ```
   $ sudo -s
   ```

### Configure the Manager #

To configure the manager, follow the steps below:

1. Install the repository:

   ```
   # wget -O pgpro-repo-add.sh https://repo.postgrespro.ru/ppem/ppem/keys/pgpro-repo-add.sh
   # sh pgpro-repo-add.sh
   ```

2. Install the manager:

   ```
   # apt install ppem ppem-gui
   ```

   The manager configuration file `ppem-manager.yml` will be downloaded to your local device.

3. Create the DBMS user on behalf of which the manager will connect to the repository database:

   ```
   # sudo -u postgres createuser --pwprompt ppem
   ```

   When executing this command, specify the DBMS user password.

4. Create the repository database:

   ```
   # sudo -u postgres createdb -O ppem ppem
   ```

5. Ensure that the DBMS user can connect to the database:

   ```
   # psql -h localhost -U ppem -d ppem
   ```

   In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file. For more information, refer to the official Postgres Pro documentation on the `pg_hba.conf` configuration file.

6. In the `ppem-manager.yml` manager configuration file, perform the following actions:

- Specify the repository database name using the `repo.name` parameter:

  ```
  repo:
    name: "ppem"
  ```

- Specify the DBMS user name and password using the `repo.user` and `repo.password` parameters:

  ```
  repo:
    user: "ppem"
    password: "<password_of_the_DBMS_user >"
  ```

- Specify the URL for connecting the manager to the repository database using the `repo.url` parameter:

  ```
  repo:
    url: "postgres://ppem:<DBMS_user_password>@localhost/ppem"
  ```

  For more information about the URL format, refer to the official Postgres Pro documentation on connection strings.

7. Start the manager service and add it to the server startup:

   ```
   # systemctl start ppem
   # systemctl enable ppem
   ```

The web application will be installed on the server.

### Configure an Agent #

To configure an agent, follow the steps below:

1. Install the agent:

   ```
   # apt install ppem-agent
   ```

   The `ppem-agent.yml` agent configuration file will be downloaded to your local device.

2. Create the DBMS user on behalf of which the agent will connect to the repository database:

   ```
   # sudo -u postgres createuser -s --pwprompt ppem_agent
   ```

   When executing this command, specify the DBMS user password.

3. Ensure that the DBMS user can connect to the repository database:

   ```
   # psql -h localhost -U ppem_agent -d ppem
   ```

   In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file. For more information, refer to the official Postgres Pro documentation on the `pg_hba.conf` configuration file.

4. Get an API key for configuring the agent by performing the following actions:

   1. Log in to the web application .
   2. Copy the API key from the displayed agent installation instruction and save this key.

5. Specify the required agent parameters in the `ppem-agent.yml` agent configuration file:

   - `agent.name`: The unique name of the agent.
   - `agent.manager.url`: The URL for connecting the agent to the manager in the `<scheme>://<manager_web_address>/<path_to_API_version>` format.
   - `agent.manager.api_key`: The previously obtained API key for connecting the agent to the manager.
   - `agent.instance.connection_defaults.user` and `agent.instance.connection_defaults.password`: The name and password of the DBMS user.
   - `http.server.address` and `http.server.port`: The network address and the port number for incoming network connections. To enable listening of all network addresses and ports, do not specify any values for these parameters.

   Example of the `ppem-agent.yml` agent configuration file:

   ```
   agent:
     name: "local"
     manager:
       url: "https://ppem.example.org/v1"
       api_key: "741c5c39-3ac9-402f-aeba-e2fa05bd3037"
     instance:
       connection_defaults:
         user: "ppem_agent"
         password: "<password of the DBMS user >"
   http:
     server:
       address: "192.0.2.1"
       port: "8081"
   ```

6. Start the agent service and add it to the server startup:

   ```
   # systemctl start ppem-agent
   # systemctl enable ppem-agent
   ```

## Installation and Setup #

This section explains how to initially install and configure PPEM. It includes the following instructions:

- Installation in Secure Environments
- Installation in the High Availability Mode
- Installation and Configuration of Backup and Restore Tools
- Installation and Configuration of Logging and Monitoring Tools
- Integration with External Data Sources

It is recommended to see the Quick Start section first.

### Installation in Secure Environments #

In secure environments, the manager and agent services run on servers under separately created operating system users, not under the superuser. In this case, operating system users must be granted additional privileges so that the manager and agent services can function correctly.

When executing the commands provided in this section, consider the following:

- The Debian Linux operating system commands are specified. For other operating system versions, use corresponding commands.
- Standard object names are specified in the commands, for example, `ppem` for the repository database. If required, you can specify different names.

The installation process includes the following steps:

1. Ensure that prerequisites are met.
2. Create operating system users.
3. Configure the manager.
4. Configure agents.

PPEM will be installed. You can update the browser page with the web application and start working.

## Prerequisites #

Before installation, read the following information and perform the required actions:

1. Prepare the servers for installing PPEM according to hardware and software requirements. You will need at least one server.
2. Install a Postgres Pro DBMS instance at least on one of the servers. For more information about installation, refer to the official Postgres Pro documentation.

## Create Operating System Users #

Create separate operating system users on all servers:

```
# useradd ppem
```

The manager and agent services will start under the created operating system users.

## Configure the Manager #

To configure the manager, follow the steps below on the Postgres Pro DBMS instance server:

1. Install the repository:

   ```
   # wget -O pgpro-repo-add.sh https://repo.postgrespro.ru/ppem/ppem/keys/pgpro-repo-add.sh
   # sh pgpro-repo-add.sh
   ```

2. Install the manager:

   ```
   # apt install ppem ppem-gui
   ```

   The manager configuration file `ppem-manager.yml` will be downloaded to your local device.

3. Create the DBMS user on behalf of which the manager will connect to the repository database:

   ```
   # sudo -u postgres createuser --pwprompt ppem
   ```

   When executing this command, specify the DBMS user password.

4. Create the repository database:

   ```
   # sudo -u postgres createdb -O ppem ppem
   ```

5. Ensure that the DBMS user can connect to the database:

   ```
   # psql -h localhost -U ppem -d ppem
   ```

   In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file. For more information, refer to the official Postgres Pro documentation on the `pg_hba.conf` configuration file.

6. In the `ppem-manager.yml` manager configuration file, perform the following actions:

   - Specify the repository database name using the `repo.name` parameter:

     ```
     repo:
       name: "ppem"
     ```

   - Specify the DBMS user name and password using the `repo.user` and `repo.password` parameters:

     ```
     repo:
       user: "ppem"
       password: "<password_of_the_DBMS_user >"
     ```

   - Specify the URL for connecting the manager to the repository database using the `repo.url` parameter:

     ```
     repo:
       url: "postgres://ppem:<DBMS_user_password>@localhost/ppem"
     ```

     For more information about the URL format, refer to the official Postgres Pro documentation on connection strings.

7. Configure the manager service to start under the created operating system user by performing the following actions:

   1. Start editing the `systemd` unit:

      ```
      # systemctl edit ppem
      ```

   2. In the `[Service]` section, specify the name of the operating system user:

   ```
   [Service]
   User=ppem
   ```

   1. Ensure that the operating system user is granted the privilege to read the `ppem-manager.yml` manager configuration file. If the privilege is not granted, execute the following commands:

   ```
    # chown ppem:ppem /etc/ppem-manager.yml
    # chmod 400 /etc/ppem-manager.yml
   ```

   1. Save the `systemd` unit parameters, then reload it:

   ```
    # systemctl daemon-reload
   ```

8. Start the manager service and add it to the server startup:

   ```
   # systemctl start ppem
   # systemctl enable ppem
   ```

The web application will be installed on the server.

## Configure Agents #

To configure agents, follow the steps below on all servers:

1. Install the agent:

   ```
   # apt install ppem-agent
   ```

   The `ppem-agent.yml` agent configuration file will be downloaded to your local device.

2. Create the DBMS user on behalf of which the agent will connect to the repository database:

   ```
   # sudo -u postgres createuser --pwprompt ppem_agent
   ```

   When executing this command, specify the DBMS user password.

3. Grant the DBMS user the privilege to read the system catalog and run functions. It is recommended to grant the following privileges:

   ```
   GRANT pg_monitor, pg_maintain, pg_signal_backend, pg_read_all_settings TO ppem_agent;
   ```

   The following privileges must be granted for each database in the instance:

   ```
   GRANT EXECUTE ON FUNCTION pg_catalog.pg_stat_file(TEXT) TO ppem_agent;
   ```

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stat_file(TEXT, BOOLEAN) TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_statistic TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_config TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_config() TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_file_settings TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_show_all_file_settings() TO ppem_agent;
GRANT SELECT ON pg_catalog.pg_authid TO ppem_agent;
```

Creating backups with `pg_probackup` requires `REPLICATION` privilege along with other privileges:

```
ALTER ROLE ppem_agent WITH REPLICATION;

GRANT USAGE ON SCHEMA pg_catalog TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.current_setting(text) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.set_config(text, text, boolean) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_is_in_recovery() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_start(text, boolean) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_backup_stop(boolean) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_create_restore_point(text) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_switch_wal() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_last_wal_replay_lsn() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_current_snapshot() TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.txid_snapshot_xmax(txid_snapshot) TO ppem_agent;
GRANT EXECUTE ON FUNCTION pg_catalog.pg_control_checkpoint() TO ppem_agent;
```

You can grant privileges only for the database that will be used for connecting the user to the instance. For more information about privileges, refer to the official Postgres Pro documentation on `pg_probackup`.

4. Ensure that the DBMS user can connect to the repository database:

```
# psql -h localhost -U ppem_agent -d ppem
```

In case of connection issues, ensure that the HBA rule allowing the DBMS user to connect to the database, as well as the authorization method, are specified in the `pg_hba.conf` configuration file. For more information, refer to the official Postgres Pro documentation on the `pg_hba.conf` configuration file.

5. Get an API key for configuring the agent by performing the following actions:

   - Log in to the Web Application
   1. Copy the API key from the displayed instruction on installing agents and save it.

6. Specify the required agent parameters in the `ppem-agent.yml` agent configuration file:

   - `agent.name`: The unique name of the agent.
   - `agent.manager.url`: The URL for connecting the agent to the manager in the `<scheme>://<manager_web_address>/<path_to_API_version>` format.
   - `agent.manager.api_key`: The previously obtained API key for connecting the agent to the manager.
   - `agent.instance.connection_defaults.user` and `agent.instance.connection_defaults.password`: The name and password of the DBMS user.
   - `http.server.address` and `http.server.port`: The network address of the server and the port number for incoming network connections. To enable listening of all network addresses and ports, do not specify any values for these parameters.

Example of the `ppem-agent.yml` agent configuration file:

```
agent:
  name: "local"
  manager:
    url: "https://ppem.example.org/v1"
    api_key: "741c5c39-3ac9-402f-aeba-e2fa05bd3037"
  instance:
    connection_defaults:
      user: "ppem_agent"
      password: "<password of the DBMS user >"
http:
  server:
    address: "192.0.2.1"
    port: "8081"
```

7. Grant the created operating system user the superuser privileges to work with target directories using the `chown` and `chmod` commands. Running, stopping, and reloading instances also require the operating system user to have the following privileges:

```
Cmnd_Alias PG_SYS = \
    /usr/bin/systemctl status postgresql*.service, \
    /usr/bin/systemctl stop postgresql*.service, \
    /usr/bin/systemctl start postgresql*.service, \
    /usr/bin/systemctl restart postgresql*.service, \
    /usr/bin/systemctl reload postgresql*.service

Cmnd_Alias PG_CTL = \
    /usr/lib/postgresql/17/bin/pg_ctl, \
    /usr/lib/postgresql/16/bin/pg_ctl

ppem ALL = (root) NOPASSWD: PG_SYS
ppem ALL = (postgres) NOPASSWD: PG_CTL
```

8. Configure the agent service to start on behalf of the operating system user by performing the following actions:

   1. Start editing the `systemd` unit:

      ```
      # systemctl edit ppem-agent
      ```

   2. In the `Service` section, specify the operating system user:

```
[Service]
User=ppem
```

   1. Ensure that the operating system user is granted the privilege to read the agent configuration file `ppem-agent.yml`. If the privilege is not granted, execute the following commands:

```
 # chown ppem:ppem /etc/ppem-agent.yml
 # chmod 400 /etc/ppem-agent.yml
```

   1. Save the `systemd` unit parameters, then reload it:

      ```
      # systemctl daemon-reload
      ```

9. Start the agent service and add it to the server startup:

```
# systemctl start ppem-agent
# systemctl enable ppem-agent
```

## Installation in the High Availability Mode #

This section provides an example of installing and setting up PPEM in high availability mode.

The following software is used in the example:

- Debian Linux 12 OS
- HAProxy 2.6.12 (included in the Debian repository)
- keepalived 2.2.7 (included in the Debian repository)
- Postgres Pro Enterprise 17.2.2 + BiHA (included in the `postgrespro-ent-17-contrib` package)

### Architecture Description #

The recommended high availability PPEM cluster architecture includes the following components:

- **The high availability cluster based on the BiHA solution** available in Postgres Pro Enterprise 16 and higher. BiHA cluster includes three or more servers. One of the servers acts as the leader at a time. PPEM managers automatically connect to the database server with this role. If the leader fails, one of the remaining servers automatically becomes the leader.
- **The cluster of HAProxy servers + keepalived**. A virtual IP address is automatically activated on one of the servers using the keepalived service. Users and PPEM agents interact with PPEM managers using this virtual IP address. If the server with the virtual IP fails, the virtual IP is activated on one of the remaining servers. HAProxy also balances client HTTP requests between available PPEM managers. For sending requests of a specific client to the same manager, the HAProxy “IP-based stickiness” or “cookie session stickiness” functionality is used. You can deploy this component on separate servers and PPEM managers.
- **PPEM manager** (two or more servers). All servers are in the active mode.

### Example of the Recommended Architecture Implementation #

The following components are required for implementation of the recommended architecture:

| Component | Amount | Minimum Requirements |
|---|---|---|
| PPEM Manager server + HAProxy + keepalived | 2 | 2 CPU, 4 GB RAM, 20 GB HDD |
| Postgres Pro BiHA server | 3 | 2 CPU, 4 GB RAM, 20 GB HDD |
| Virtual static IP address | 1 | The address must be excluded from the DHCP pool. You can also create a DNS A record for this IP address. |

### Installation Procedure #

1. Install Postgres Pro Enterprise with the BiHA solution using the instruction.

2. To provide all PPEM servers with access to DBMS, edit the `pg_hba.conf` file.

   Example of the `pg_hba.conf` configuration:

   ```
   # cat /var/lib/pgpro/ent-17/data/pg_hba.conf

   host    all     all     192.168.1.0/24      scram-sha-256
   ```

3. Configure the PPEM manager.

   Required manager configuration parameters:

   ```
   # cat /etc/ppem-manager.yml
   http:
     server:
       address: ''
       port: 8080
   repo:
     url: postgres://ppem:<DBMS_user_password>@biha-server-1/ppem
     fallback_addresses:
       - biha-server-2
       - biha-server-3
     target_session_attrs: read-write
   ```

   Where:

   - `http.server.port` is the PPEM manager port. In terms of HAProxy — `backend`.
   - `repo.url` is the address of the first BiHA cluster node.
   - `fallback_addresses` are the addresses of the remaining nodes.

   The PPEM manager will automatically connect to the BiHA cluster leader once the `read-write` session attribute condition is met.

4. On HAProxy servers, install `haproxy`, `keepalived`, and the required tools:

   ```
   sudo apt-get install haproxy keepalived psmisc
   ```

5. Configure HAProxy according to the configuration file example below.

   The cookie-based persistence method is used in this example. This method assigns a cookie with the PPEM server name to users. This is required for sending all queries within the HTTP session to a single PPEM manager server.

   Example HAProxy configuration in the `/etc/haproxy/haproxy.cfg` file:

   ```
   global
     log /dev/log  local0
     log /dev/log  local1 notice
     stats socket /var/lib/haproxy/stats level admin
     chroot /var/lib/haproxy
     user haproxy
     group haproxy
     daemon

   defaults
     log global
     mode  http
     option  httplog
     option  dontlognull
     timeout connect 5000
     timeout client 50000
     timeout server 50000
     errorfile 400 /etc/haproxy/errors/400.http
     errorfile 403 /etc/haproxy/errors/403.http
     errorfile 408 /etc/haproxy/errors/408.http
     errorfile 500 /etc/haproxy/errors/500.http
     errorfile 502 /etc/haproxy/errors/502.http
     errorfile 503 /etc/haproxy/errors/503.http
     errorfile 504 /etc/haproxy/errors/504.http

   frontend hafrontend
     bind *:80
     mode http
     default_backend habackend

   backend habackend
     mode http
     balance roundrobin
     option forwardfor
     option httpchk
     http-check send meth HEAD uri /
     cookie SERVERID insert indirect
     server ppem-server-1 <PPEM_server_address-1>:8080 cookie ppem-server-1 check
     server ppem-server-2 <PPEM_server_address-2>:8080 cookie ppem-server-2 check
   ```

6. Configure keepalived according to the example below.

   Example of the HAProxy-1 server configuration in the `/etc/keepalived/keepalived.conf` file:

   ```
   global_defs {
     enable_script_security
   }

   vrrp_script chk_haproxy {
     script "/usr/bin/killall -0 haproxy"
     interval 3
     fall 2
     rise 3
     timeout 3
   ```

```
    user root
  }

  vrrp_instance internal {
    interface <interface>
    state MASTER
    virtual_router_id 124
    priority 100

    unicast_src_ip <HAproxy-1_server_IP_address>
    unicast_peer {
      <HAproxy-2_server_IP_address>
    }
    virtual_ipaddress {
      <virtual_IP_address>/<subnet_class (for example, 16 or 24)> dev <interface>
    }
    track_script {
      chk_haproxy
    }
  }
```

The configuration is the same on the HAProxy-2 server, but the IP addresses in `unicast_src_ip` and `unicast_peer` are switched.

7. On all BiHA servers, configure PPEM agents.

   **Note**

   It is not required to configure agents on PPEM servers.

   The connection to the PPEM manager must be configured via a virtual IP.

   Required agent configuration parameters:

   ```
   cat /etc/ppem-agent.yml

   agent:
     manager:
       url: http://<virtual_IP_address>:80/v1
   ```

### Healthcheck #

Ensure that:

- The PPEM database of the BiHA cluster is available on PPEM servers.
- The systemd service `ppem` is running on all PPEM servers.

  ```
  systemctl status ppem
  ```

- The PPEM web application is available on all PPEM servers on port 8080:

  ```
  curl http://<PPEM_server_IP_address>:8080
  ```

- The haproxy and keepalived services are running on all PPEM servers:

  ```
  systemctl status haproxy
  systemctl status keepalived
  ```

- The PPEM web application is available on all PPEM servers on port 80 (using haproxy):

  ```
  curl http://<PPEM_server_IP_address>:80
  ```

- The PPEM web application is available via a virtual IP address:

  ```
  curl http://<virtual_IP_address>:80
  ```

## Installation and Configuration of Backup and Restore Tools #

Backup is performed in PPEM using `pg_probackup`. You must install `pg_probackup` manually on all servers. The version of `pg_probackup` must correspond to the version of DBMS instances.

The backup features available in PPEM depend on the `pg_probackup` edition installed. For more information about installation, refer to the official Postgres Pro documentation on `pg_probackup`.

It is also recommended to view the example of the `pg_probackup` installation using APT (for Debian-based operating systems).

After installing `pg_probackup`, the agent automatically detects it and notifies the manager. Then, backup will become available on the server.

It is recommended to configure the installed `pg_probackup`. The configuration process includes the following steps:

1. Configure DBMS users.
2. Configure a STREAM backup.
3. Configure continuous WAL Archiving.

### Configure DBMS Users #

Create dedicated DBMS users on all servers and grant them backup privileges using `pg_probackup`. For more information, refer to the official Postgres Pro documentation on configuring the database cluster.

### Configure a STREAM Backup #

Configure a STREAM backup on all servers. For more information, refer to the official Postgres Pro documentation on setting up STREAM backups.

### Configure Continuous WAL Archiving #

Configure continuous WAL archiving on all servers to provide point-in-time recovery (PITR). This can be done in one of the following ways:

- When configuring an instance on a server. For more information, refer to the official Postgres Pro documentation on setting up continuous WAL archiving.
- When configuring an instance in the web application .

### Example of the pg_probackup installation using APT #

This section includes an example of `pg_probackup` installation on a server with a DBMS instance and PPEM. It is recommended to see the Installation and Configuration of Backup and Restore Tools section first.

The installation process includes the following steps:

1. Start a new session under the superuser:

   ```
   $ sudo -s
   ```

2. Add the GPG key for the `pg_probackup` repository. To do this, you may require additional utilities:

   ```
   # apt install gpg wget
   # wget -qO - https://repo.postgrespro.ru/pg_probackup/keys/GPG-KEY-PG-PROBACKUP | \
   ```

```
# tee /etc/apt/trusted.gpg.d/pg_probackup.asc
```

3. Configure the package repository:

   ```
   # . /etc/os-release
   # echo "deb [arch=amd64] https://repo.postgrespro.ru/pg_probackup/deb $VERSION_CODENAME  main-$VERSION_CODENAME " | tee /etc/apt/sources.list.d/pg_probackup.list
   ```

4. Update the package manager metadata so that `pg_probackup` packages are available for viewing and installing:

   ```
   # apt update
   # apt search pg_probackup
   ```

5. Install `pg_probackup`:

   ```
   # apt install pg-probackup-16
   ```

   The version of `pg_probackup` must correspond to the version of the instance. Version 16 is used in this example.

After installing `pg_probackup`, the agent automatically detects it and notifies the manager. Then, backup will become available on the server.

## Installation and Configuration of Logging and Monitoring Tools #

Logging of DBMS instances and working with their metrics is done in PPEM using `pgpro-otel-collector`. You must manually install `pgpro-otel-collector` on all servers. For more information, refer to the official Postgres Pro documentation on installing pgpro-otel-collector.

You must configure the installed `pgpro-otel-collector`. The configuration process includes the following steps:

1. Configure instance logging.
2. Configure Log and Metrics Collection in pgpro-otel-collector .
3. Configure logs and metrics sending to PPEM or an external storage .

### Configure Instance Logging #

Configure instance logging. For more information, refer to the official Postgres Pro documentation on server configuration. You must ensure that values are specified for the following parameters:

- `logging_collector`
- `log_destination`
- `log_directory`
- `log_filename`

Instance logging must be done in the `CSV` or `JSON` format.

### Configure Log and Metrics Collection in `pgpro-otel-collector` #

Configure `pgpro-otel-collector` to collect instance logs and metrics. For more information, refer to the official Postgres Pro documentation on configuring logs and metrics for `pgpro-otel-collector`.

### Configure Logs and Metrics Sending to PPEM or an External Storage #

Configure `pgpro-otel-collector` to send instance logs and metrics to PPEM or an external storage. For more information, refer to the official Postgres Pro documentation on integrating `pgpro-otel-collector` with PPEM.

> **Note**
>
> Local storages are indended for introductory use with PPEM. For production environments with a large number of instances and metrics, it is recommended to use external storages.

## Integration with External Data Sources #

This section describes different ways of integrating with the following external data sources:

- Integration with Prometheus
- Integration with Elasticsearch and APM

For more information about the monitoring architecture, refer to the Monitoring Architecture section.

> **Note**
>
> Fictional service names are used in the configuration examples:
>
> - example.org is the main domain
> - prometheus.example.org is the Prometheus monitoring system service
> - elasticsearch.example.org is the Elasticsearch monitoring system service
> - elasticsearch-apm.example.org is the APM service of the Elasticsearch monitoring system
> - postgresql-01.example.org is the PostgreSQL DBMS service

### Integration with Prometheus #

Integration with external Prometheus data sources is used for reading metrics written there by the `pgpro-otel-collector` utility.

> **Note**
>
> You can use VictoriaMetrics instead of Prometheus, since these solutions use similar interfaces for working with metrics (for reading and writing metrics).

#### Architecture #

Integration includes the following components:

- **pgpro-otel-collector** is the monitoring agent. It performs the following functions:
  - collecting statistics from Postgres Pro DBMS instances and converting them to metrics;
  - publishing metrics for further collection by the Prometheus monitoring system.
- **Prometheus** is the monitoring system. It performs the following functions:
  - collects metrics from pgpro-otel-collector monitoring agents;
  - stores metrics from monitoring agents (according to the internal configuration parameters);
  - provides the HTTP interface for receiving metrics.
- **PPEM** is the Postgres Pro Enterprise Manager system. It performs the following functions:
  - accesses the Prometheus monitoring system for receiving DBMS instance metrics;
  - provides the user with the monitoring interface (in the form of graphs).

#### Configuring pgpro-otel-collector #

1. Enable and configure the `postgrespro` and `hostmetrics` receivers:

   ```
   receivers:
     hostmetrics:
       initial_delay: 1s
   ```

```
    collection_interval: 60s
    scrapers:
      cpu:
        metrics:
          system.cpu.utilization:
            enabled: true
      disk: null
      filesystem: null
      load: null
      memory: null
      network: null
      paging: null
      processes: null
  postgrespro:
    max_threads: 3
    initial_delay: 1s
    collection_interval: 60s
    transport: tcp
    endpoint: localhost:5432
    database: postgres
    username: postgres
    password: ${env:POSTGRESQL_PASSWORD}
    plugins:
      activity:
        enabled: true
      archiver:
        enabled: true
      bgwriter:
        enabled: true
      cache:
        enabled: true
      databases:
        enabled: true
      io:
        enabled: true
      locks:
        enabled: true
      version:
        enabled: true
      wal:
        enabled: true
```

2. Configure metric publishing using `prometheusexporter` and the pipeline.

   PPEM expects the metrics sent by pgpro-otel-collector to have the `instance` label with the node FQDN and DBMS instance port number separated by a colon, for example, `postgresql_activity_connections{instance="postgresql-01.example.org:5432"}`

   Configuration example:

   ```
   exporters:
     prometheus:
       const_labels:
         instance: postgresql-01.example.org:5432
       endpoint: :8889
       send_timestamps: true

   service:
     extensions: []
     pipelines:
       metrics:
         exporters:
         - prometheus
         receivers:
         - postgrespro
         - hostmetrics
   ```

3. Start the collector and ensure that metrics are published on its side:

   ```
   # systemctl status pgpro-otel-collector

   # systemctl status pgpro-otel-collector
   ● pgpro-otel-collector.service - PostgresPro OpenTelemetry Collector
        Loaded: loaded (/lib/systemd/system/pgpro-otel-collector.service; enabled; preset: enabled)
        Active: active (running) since Thu 2025-03-20 01:18:08 MSK; 4h 13min ago
      Main PID: 6991 (pgpro-otel-coll)
         Tasks: 8 (limit: 3512)
        Memory: 119.3M
           CPU: 2min 49.311s
        CGroup: /system.slice/pgpro-otel-collector.service
                └─6991 /usr/bin/pgpro-otel-collector --config /etc/pgpro-otel-collector/basic.yml

   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.366656,"msg":"Setting up own telemetry..."}
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.367178,"msg":"Skipped telemetry setup."}
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.3679142,"msg":"Development component. May change in the future.","kind":"rec
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"warn","ts":1742422688.3494158,"caller":"envprovider@v1.16.0/provider.go:59","msg":"Configuration r
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.4481084,"msg":"Starting pgpro-otel-collector...","Version":"v0.3.1","NumCPU"
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.4481149,"msg":"Starting extensions..."}
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"warn","ts":1742422688.4483361,"msg":"Using the 0.0.0.0 address exposes this server to every networ
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.4515307,"msg":"Starting stanza receiver","kind":"receiver","name":"filelog",
   Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.451749,"msg":"Everything is ready. Begin running and processing data."}

   curl -s 127.0.0.1:8889/metrics |grep -c postgres
   4254
   ```

## Configuring Prometheus #

You can configure collection of metrics using pgpro-otel-collector on the side of Prometheus in different ways. One of the options is collecting using the static configuration:

```
- job_name: pgpro-otel-collector
  static_configs:
    targets:
    - postgresql-01.example.org:8889/metrics
```

For more information about other collection methods, refer to the official Prometheus documentation.

PPEM does not require additional Prometheus configuration.

## Configuring a PPEM Agent #

Additional configuration of the PPEM agent is not required.

## Checking for Metrics in Prometheus #

After configuring metrics collection using pgpro-otel-collector, ensure that metrics are received by the monitoring system. For this check, you can use the built-in Expression Browser graphical tool or the `promtool` utility from Prometheus.

Example check using the promtool utility:

```
promtool query instant https://prometheus.example.org 'postgresql_activity_connections{instance="postgresql-01.example.org:5432"}'
```

Where

- https://prometheus.example.org is the URL of the monitoring service
- postgresql_activity_connections{instance="postgresql-01.example.org:5432"} is the name of the metric

Response example:

```
postgresql_activity_connections{database="postgres", instance="postgresql-01.example.org:5432", job="pgpro-otel-collector", state="active", user="postgres"} 5
postgresql_activity_connections{database="postgres", instance="postgresql-01.example.org:5432", job="pgpro-otel-collector", state="idle", user="postgres"} 10
```

#### Configuring a Metrics Data Source in the Web Application #

To create a metrics storage, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure → Data sources → Metrics storages** page.

   The table of metrics storages will be displayed.

2. In the top-right corner of the page, click **CREATE STORAGE**.

   The metrics storage creation window will open.

3. Specify the metrics storage parameters (parameters marked with an asterisk are required):

   - **Name**: The unique name of the metrics storage, for example, Prometheus.
   - **URL**: The network address for connecting to the metrics storage, for example, `https://prometheus.example.org/select/0/prometheus`.
   - **User**: The unique name of the user, if authorization is used.
   - **Password**: The password of the user, if the authorization is enabled.
   - **Description**: The description of the metrics storage.
   - **Make default datasource**: Whether the metrics storage is used by default for all metric queries. This checkbox is disabled by default.

4. Click **Save**.

The metric storage will be created and displayed in the table of metrics storages.

#### Checking the Operation of a Metrics Storage #

To check the operation of a metrics storage, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Monitoring → Metrics** page.

2. In the top-right corner of the page, select the instance for which there are metrics in the storage from the drop-down list.

   The graphs will be displayed on the page.

3. Change the default data source to the internal one and ensure that graphs are displayed without errors.

### Integration with Elasticsearch and APM #

Integration with external Elasticsearch data sources is used for reading logs written there by the `pgpro-otel-collector` utility.

#### Architecture #

Integration includes the following components:

- **pgpro-otel-collector** is the monitoring agent. It performs the following functions:
  - collecting activity logs from Postgres Pro DBMS instances;
  - sending activity logs to the APM Elasticsearch system.
- **APM Elasticsearch** is the application performance monitoring system based on Elastic Stack. It performs the following functions:
  - receives data from the monitoring agent and converts it to the ES document format;
  - sends converted data to Elasticsearch.
- **Elasticsearch** is the activity log storage system. It performs the following functions:
  - receives activity logs from the application performance monitoring system;
  - stores activity logs according to internal storage parameters;
  - provides the interface for receiving activity logs.
- **PPEM** is the Postgres Pro Enterprise Manager system. It performs the following functions:
  - accesses the Elasticsearch system for receiving DBMS instance activity logs;
  - provides the user with the monitoring interface based on activity logs in the form of text data.

#### Configuring Elasticsearch #

To send logs to Elasticsearch, follow the steps below:

1. Install the Elasticsearch APM server using the standard documentation.

2. Integrate the Elasticsearch APM server with Elasticsearch using the standard documentation.

3. Configure the pgpro-otel-collector ingest pipeline.

   This is required for compatibility between document (log) fields and the Elasticsearch Common Schema (ECS) field naming schema.

   Pipeline configuration example (both queries must be executed sequentially in Kibana Developer Tools):

```
PUT _ingest/pipeline/postgrespro-otelcol-enrich-logs
{
  "description": "Enrich PostgresPro Otel collector logs",
  "processors": [
    {
      "rename": {
        "if": "ctx?.labels?.message != null",
        "field": "labels.message",
        "target_field": "message",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.numeric_labels?.pid != null",
        "field": "numeric_labels.pid",
        "target_field": "process.pid",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.error_severity != null",
        "field": "labels.error_severity",
        "target_field": "log.level",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.user != null",
        "field": "labels.user",
        "target_field": "user.name",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.session_start != null",
        "field": "labels.session_start",
        "target_field": "session.start_time",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.session_id != null",
        "field": "labels.session_id",
        "target_field": "session.id",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.numeric_labels?.tx_id != null",
        "field": "numeric_labels.tx_id",
        "target_field": "transaction.id",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.log_file_name != null",
        "field": "labels.log_file_name",
        "target_field": "log.file.path",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "rename": {
        "if": "ctx?.labels?.dbname != null",
        "field": "labels.dbname",
        "target_field": "db.name",
        "ignore_failure": true,
        "ignore_missing": false,
        "override": true
      }
    },
    {
      "gsub": {
        "if": "ctx?.service?.node?.name != null",
        "field": "service.node.name",
        "target_field": "host.name",
        "pattern": ":.+$",
        "replacement": "",
        "ignore_failure": true,
        "ignore_missing": false
      }
    },
    {
      "remove": {
        "field": [
          "observer.version",
          "observer.hostname",
          "service.language.name"
        ],
        "ignore_failure": true
      }
    },
    {
      "remove": {
        "field": "agent.version",
        "if": "ctx?.agent?.version == \"unknown\"",
        "ignore_failure": true
      }
    }
  ]
}

PUT _ingest/pipeline/logs-apm.app@custom
{
  "processors": [
    {
      "pipeline": {
        "name": "postgrespro-otelcol-enrich-logs"
      }
    }
  ]
}
```

## Configuring pgpro-otel-collector #

1. Enable and configure the `filelog` receiver.

Receiver configuration example for the scenario when PostgreSQL logs are generated in the JSON format:

```
receivers:
  filelog:
    include:
    - /var/log/postgresql/*.json
    operators:
    - parse_ints: true
      timestamp:
        layout: '%Y-%m-%d %H:%M:%S.%L %Z'
        layout_type: strptime
        parse_from: attributes.timestamp
      type: json_parser
    - field: attributes.timestamp
      type: remove
    retry_on_failure:
      enabled: true
      initial_interval: 1s
      max_elapsed_time: 5m
      max_interval: 30s
    start_at: end
```

1. Configure processors:

```
processors:
  attributes/convert:
    actions:
    action: convert
      converted_type: string
      - key: query_id
  resource:
    attributes:
    action: upsert
      key: service.name
      value: postgresql
    action: upsert
      key: service.instance.id
      value: postgresql-01.example.org:5432
```

Where:

    - the required `service.name` field is the key for naming the data stream (`data stream`) and, consequently, indexes;
    - the required `service.instance.id` field is the key for identifying the instance;
    - for logs in the JSON format, converting the `query_id` field to a string is required because integers are displayed incorrectly in ES.

      **Note**

      It is important to mention the data structures within Elasticsearch where logs will be stored. [Data streams](#)) are used for storing data. The target stream is selected automatically and has the `logs-apm.app.<service.name>-<namespace>` format. The `service.name` value is specified in the collector configuration, in the `processors.resource.attributes` list, by the `key: service.name` element. The `namespace` value is defined by the element with the `service.environment` key. It is not sent in this configuration so the `default` value is entered by default. If this configuration is used, activity logs will be stored in the stream named `logs-apm.app.postgresql-default`.

2. Configure logs sending using `otlphttpexporter` and the pipeline:

```
exporters:
  otlphttp/elastic_logs:
    compression: gzip
    endpoint: https://elasticsearch-apm.example.org
    tls:
      insecure_skip_verify: false

service:
  extensions: []
  pipelines:
    logs:
      receivers:
      - filelog
      processors:
      - resource
      - attributes/convert
      exporters:
      - otlphttp/elastic_logs
```

3. Start the collector and ensure that metrics are published on its side:

```
# systemctl status pgpro-otel-collector

# systemctl status pgpro-otel-collector
● pgpro-otel-collector.service - PostgresPro OpenTelemetry Collector
     Loaded: loaded (/lib/systemd/system/pgpro-otel-collector.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-03-20 01:18:08 MSK; 4h 13min ago
   Main PID: 6991 (pgpro-otel-coll)
      Tasks: 8 (limit: 3512)
     Memory: 119.3M
        CPU: 2min 49.311s
     CGroup: /system.slice/pgpro-otel-collector.service
             └─6991 /usr/bin/pgpro-otel-collector --config /etc/pgpro-otel-collector/basic.yml

Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.366656,"msg":"Setting up own telemetry..."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.367178,"msg":"Skipped telemetry setup."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.3679142,"msg":"Development component. May change in the future.","kind":"rec
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"warn","ts":1742422688.3494158,"caller":"envprovider@v1.16.0/provider.go:59","msg":"Configuration r
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.4481084,"msg":"Starting pgpro-otel-collector...","Version":"v0.3.1","NumCPU"
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.4481149,"msg":"Starting extensions..."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"warn","ts":1742422688.4483361,"msg":"Using the 0.0.0.0 address exposes this server to every networ
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.4515307,"msg":"Starting stanza receiver","kind":"receiver","name":"filelog",
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.451749,"msg":"Everything is ready. Begin running and processing data."}
Mar 20 01:18:08 postgresql-01.example.org pgpro-otel-collector[6991]: {"level":"info","ts":1742422688.6523068,"msg":"Started watching file","kind":"receiver","name":"filelog","da
```

### Configuring a PPEM Agent [#](#)

Additional configuration of the PPEM agent is not required.

### Checking Logs in Elasticsearch [#](#)

After configuring logs sending from pgpro-otel-collector in Elasticsearch, ensure that metrics are received by the log storage system. For this check, you can execute a query to the storage using the `curl` utility. Query example:

```
curl -s -XGET "https://elasticsearch.example.org:9200/logs-apm.app.postgresql-default/_search?size=10" -H 'Content-Type: application/json' -d'
{
  "_source": ["message","service.node.name","@timestamp"],
"sort": [
    { "@timestamp": "desc" }
  ],
  "query": {
        "bool": {
            "filter": [
                { "term":{"service.node.name":"postgresql-01.example.org:5432" }}]
        }
    }
}'
```

Where:

- `https://elasticsearch.example.org:9200` is the URL of the log storage system;
- `logs-apm.app.postgresql-default` is the name of the data stream for the search;
- `size=10` is the size of the sample;
- `"_source": ["message","service.node.name","@timestamp"]` are requested fields.

Response example:

```
{
  "took": 18,
  "timed_out": false,
  "_shards": {
    "total": 11,
    "successful": 11,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 10000,
      "relation": "gte"
    },
    "max_score": null,
    "hits": [
      {
        "_index": ".ds-logs-apm.app.postgresql-default-2025.03.19-000379",
        "_id": "qmuArJUB2PKtie47RffA",
        "_score": null,
        "_source": {
          "message": "checkpoint complete: wrote 2038 buffers (16.6%); 0 WAL file(s) added, 0 removed, 10 recycled; write=269.563 s, sync=1.192 s, total=270.962 s; sync files=246, lon
          "@timestamp": "2025-03-19T03:44:01.336Z",
          "service": {
            "node": {
              "name": "postgresql-01.example.org:5432"
            }
          }
        },
        "sort": [
          1742355841336
        ]
      }
    ]
  }
}
```

#### Configuring a Log Data Source in the Web Application #

To create a log storage, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Data sources** → **Message storages** page.

   The table of log storages will be displayed.

2. In the top-right corner of the page, click **CREATE STORAGE**.

   The log storage creation window will open.

3. Specify the log storage parameters (parameters marked with an asterisk are required):

   - **Name**: The unique name of the log storage, for example, Elasticsearch.
   - **URL**: The network address for connecting to the log storage, for example, `https://elasticsearch.example.org`.
   - **Elasticsearch index**: The name of the index (stream) that will be used for search queries. You must specify `logs-apm.app.postgresql-default`.
   - **User**: The unique name of the user, if authorization is used.
   - **Password**: The password of the user, if the authorization is enabled.
   - **Description**: The description of the log storage.
   - **Make default datasource**: Whether the log storage is used by default for all metric queries. This checkbox is disabled by default.

The log storage will be created and displayed in the table of log storages.

#### Checking the Operation of the Log Storage #

To check the operation of the log storage, in the **Management** section of the navigation panel, go to the **Monitoring** → **Message journal** page.

The DBMS instance logs will be displayed on the page.

## Integration with OpenLDAP and Active Directory #

PPEM supports authentication using the directory services OpenLDAP and Active Directory. Within the integration, roles are assigned to LDAP users in the web application using directory service user groups. For more information about authentication and authorization, refer to the Authentication and Authorization Operation Scheme section.

The integration process includes the following steps:

1. Create a user group in a directory service.
2. Configure integration with a directory service.
3. Configure authorization in PPEM.
4. Check authentication and authorization in PPEM and troubleshoot authentication problems if required.

### Creating a User Group in a Directory Service #

Create a user group in OpenLDAP or Active Directory. For Active Directory, two ways of creating a user group are supported:

- Creating a User Group in Active Directory using ADUC
- Creating a User Group in Active Directory Using PowerShell

  **Warning**

  Users added to groups must be created in OpenLDAP or Active Directory with the following parameters required for PPEM:

  - `first_name`: The first name of the user.
  - `last_name`: The last name of the user.
  - `email`: The email address of the user.
  - `login`: The login of the user.
  - `password`: The password of the user.

  You must specify a value in the `string` format for each parameter.

For more information about creating user groups, refer to the official OpenLDAP documentation or to the official Microsoft documentation on Active Directory.

#### Creating a User Group in OpenLDAP #

To create a user group in OpenLDAP, follow the steps below:

1. Create a user group configuration file in the `LDIF` format and specify the following parameters in it:

   - `dn`: The distinguished name (DN) of the user group. You can specify the following attributes:

     - `cn`: The unique name of the user group.
     - `ou`: The organizational unit where the user group will be placed.
     - `dc`: The components of the domain that will be associated with the user group, for example, example.com.

   - `cn`: The unique name of the user group.

   - `gidNumber`: The unique ID of the user group.

   - `memberUid`: The unique IDs of users that will be added to the group.

   Example of the user group configuration file:

   ```
   dn: cn=examplegroup,ou=groups,dc=example,dc=com
   objectClass: posixGroup
   cn: examplegroup
   gidNumber: 1001
   memberUid: user1
   memberUid: user2
   ```

2. Create a user group in OpenLDAP:

   ```
   ldapadd -x -D "cn=admin,dc=example,dc=com" -W -f example-group.ldif
   ```

   Where:

   - `-D`: The distinguished name of the OpenLDAP administrator.
   - `-W`: The password request.
   - `-f`: The name of the user group configuration file.

3. Ensure that the user group was created successfully:

   ```
   ldapsearch -x -b "ou=groups,dc=example,dc=com" "(cn=examplegroup)"
   ```

#### Creating a User Group in Active Directory using ADUC #

To create a user group in Active Directory using the ADUC (Active Directory Users and Computers) snap-in, follow the steps below:

1. In the Active Directory GUI, go to the organizational unit (OU) where the user group will be placed.

2. Click **Create → Group**.

3. In the window that opens, specify the required user group parameters.

4. Click **OK**.

   The user group will be created.

5. Add users to the group by following the steps below:

   1. Go to the user group properties by double-clicking it.
   2. Select the **Member Of** tab and add users to the group.

#### Creating a User Group in Active Directory Using PowerShell #

To create a user group in Active Directory using PowerShell, follow the steps below:

1. Create a user group:

   ```
   New-ADGroup -Path "OU=Groups,OU=Example,DC=example,DC=com" -Name "GROUPMSAD" -GroupScope Global -GroupCategory Distribution
   ```

   Where:

   - `-Path`: The distinguished name (DN) of the user group. You can specify the following attributes:
     - `OU`: The organizational unit where the user group will be placed.
     - `DC`: The components of the domain that will be associated with the user group.
   - `-Name`: The unique name of the user group.

2. Add users to the group:

   ```
   Add -ADGroupMember GROUPMSAD -Members user1,user2,user3
   ```

   Where:

   - `-ADGroupMember`: The unique name of the group to which users will be added.
   - `-Members`: The unique IDs of users that will be added to the group.

### Configuring Integration with a Directory Service #

To configure integration with a directory service, follow the steps below:

1. In the `ppem-manager.yml` manager configuration file, add the `ldap` section and specify integration parameters in it:

   - `type`: The type of the directory service. Possible values:
     - `openldap`
     - `ms_active_directory`
   - `url`: The network address of the directory service.
   - `bind_username`: The name of the directory service user for integration with PPEM. The value format depends on the directory service:
     - For OpenLDAP, a complete distinguished name (DN) is usually specified, for example, `cn=admin,ou=users,dc=example,dc=com`.
     - For Active Directory, a value in the `<username>@<domain>` format is usually specified, for example, `admin@example.com`.
   - `bind_password`: The password of the directory service user for integrating with PPEM.
   - `base_dn`: The base distinguished name of the directory service.
   - `prefix_user_dn`: The distinguished name prefix for users. Optional parameter. If this parameter is specified, users are searched using the `<prefix_user_dn>,<base_dn>` distinguished name. To search for users in the entire directory, specify the `""` value.
   - `prefix_group_dn`: The distinguished name prefix for user groups. Optional parameter. If this parameter is specified, user groups are searched using the `<prefix_group_dn>,<base_dn>` distinguished name. To search for user groups in the entire directory, specify the `""` value.
   - `user_class`: The name of the user object class. Optional parameter for Active Directory.
   - `user_name_attr`: The name of the user login attribute. Default value: `cn` for OpenLDAP, `sAMAccountName` for Active Directory. Optional parameter for Active Directory.
   - `user_first_name_attr`: The name of the user first name attribute. Default value: `givenName`. Optional parameter.
   - `user_last_name_attr`: The name of the user last name attribute. Default value: `sn`. Optional parameter.
   - `user_display_name_attr`: The name of the user display name attribute. Default value: `displayName`. Optional parameter.
   - `user_email_attr`: The name of the user email address attribute. Default value: `mail`. Optional parameter.
   - `user_phone_attr`: The name of the user phone number attribute. Default value: `telephoneNumber`. Optional parameter.
   - `user_job_title_attr`: The name of the user job title attribute. Default value: `title`. Optional parameter.
   - `user_membership_attr`: The name of the user group membership attribute. Default value for Active Directory: `memberOf`. Optional parameter.
   - `group_class`: The name of the user group object class. Default value for Active Directory: `group`.
   - `group_name_attr`: The name of the user group name attribute. The default value: `cn`. Optional parameter.
   - `group_members_attr`: The name of the group member attribute. Optional parameter.
   - `group_filter`: The filter for searching user groups, for example, `(&(objectClass=group)(cn=*PPEM*))`. Optional parameter.
   - `group_membership_filter`: The filter for searching groups of which the specified user is a member, for example, `(&(objectClass=group)(uniqueMember=%USER_DN%))`. Optional parameter.
   - `group_list_size_limit`: The maximum number of user groups that can be received from the directory service. Optional parameter.
   - `user_sync_interval`: Synchronization time between the manager and directory service. Default value: `5m`. Optional parameter.
   - `ssl_cert_skip_verify`: Whether verification of the directory service server certificate is skipped. Possible values:
     - `true`
     - `false`

     Optional parameter.
   - `ssl_root_ca`: The path to the file in the `PEM` format with the CA certificate on the directory service server. Optional parameter.

   Example of the `ldap` section in the `ppem-manager.yml` manager configuration file when integrating with OpenLDAP:

   ```
   ldap:
   ```

```
type: openldap
url: ldap://ldapserver.example.com
use_ssl: false
base_dn: dc=example,dc=com
bind_username: cn=admin,ou=users,dc=example,dc=com
bind_password: password
group_class: groupOfUniqueNames
group_members_attr: uniqueMember
group_name_attr: cn
prefix_group_dn: ou=groups
prefix_user_dn: ou=users
user_class: inetOrgPerson
user_display_name_attr: displayName
user_email_attr: mail
user_name_attr: cn
user_first_name_attr: givenName
user_last_name_attr: sn
user_job_title_attr: title
user_membership_attr: memberOf
user_phone_attr: telephoneNumber
user_sync_interval: 5m
```

Example of the `ldap` section in the `ppem-manager.yml` manager configuration file when integrating with Active Directory:

```
ldap:
type: ms_active_directory
url: ldap://ldapserver.example.com
base_dn: dc=example,dc=com
bind_username: administrator@example.com
bind_password: password
user_sync_interval: 5m
```

2. Restart the PPEM service:

```
systemctl restart ppem.service
```

### Configuring Authorization in PPEM #

To configure authorization in PPEM, follow the steps below:

1. Log in to the web application under a user with the `System administrator role` role. For more information about roles, refer to the Roles and Privileges section.
2. Create a user group. In doing so, select the LDAP group for which you want to configure authorization from the **LDAP group** drop-down list. If the LDAP group is not displayed in the drop-down list, ensure that you configured integration with the directory service correctly.

Roles assigned to a user group at its creation will be automatically assigned to users from the specified LDAP group when they log in to the web application.

### Checking Authentication and Authorization in PPEM #

To check authentication and authorization in PPEM, follow the steps below:

1. Log in to the web application under a directory service user. The user must be a member of the LDAP group specified when configuring authorization in PPEM. The login format depends on the directory service:
   - For OpenLDAP, a short user login is usually specified, for example, `j.doe`.
   - For Active Directory, a login in the `<username>@<domain>` format is usually specified, for example, `j.doe@example.com`
2. If authentication is successful, check authorization by following the steps below:
   - In the top-right corner of the page, click the username.

     The personal account will open. It displays the email address, job title, and assigned roles of the user.
   - If you logged in to the web application under a user with the `System administrator role` role, in the **System** section of the navigation panel, go to the **Users** page.

     The table of users will be displayed. Distinguished names (DNs) of LDAP users are displayed in the **Login** column.

### Troubleshooting Authentication #

To troubleshoot authentication, you must view the manager log. You can specify logging parameters in the manager configuration file `ppem-manager.yml`. Logging can be done in a separate file or in the system journal (`journalctl`).

To troubleshoot authentication, follow the steps below:

1. Connect to the server where the manager is installed.
2. View the manager log by executing one of the following commands:
   - If logging is done in a separate file:

     ```
     -- Output recent PPEM errors --
     tail -n 1000 "<path_to_the_PPEM_log_file>" | grep ERROR
     ```
   - If logging is done in the system journal:

     ```
     -- Output PPEM errors for the last 5 minutes --
     journalctl --since "5m ago" -u ppem.service -g ERROR
     ```
3. Perform the actions required for troubleshooting authentication.

## Using PPEM #

### Logging in to the Web Application #

You can log in to the web application after installing PPEM.

To log in to the web application, follow the steps below:

1. In the browser address bar, enter the network address of the server where the manager is installed. You may need to specify the port number on which the server listens for incoming connections. If a DNS service is configured on the network, you can enter the network name of the server.

   The server network address and port number can be specified in the `ppem-manager.yml` manager configuration file using the `http.server.address` and `http.server.port` parameters.
2. On the authorization page, enter your login and password, then click **Log in**. When logging in to the web application for the first time, use the default user account with the login `admin` and password `admin`.

The **Summary** page in the **Management** section of the navigation panel will open.

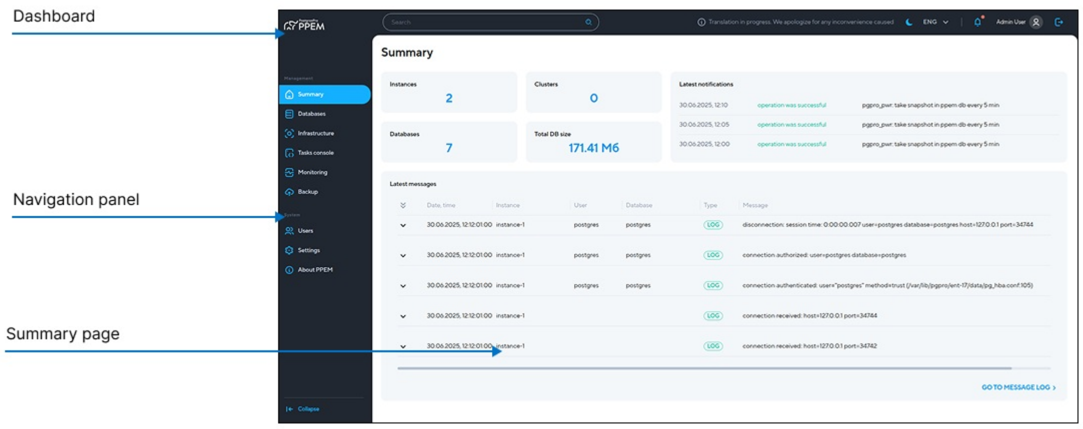
It is recommended that you take the following steps after logging in to the web application for the first time:

- Edit the default user with the `admin` login and change its password.
- Create a separate user for yourself.

### Web Application Interface #

This section describes the following elements of the web application interface:

- [Summary page](#)
- [Navigation panel](#)
- [Dashboard](#)



Summary

Instances 2 | Clusters 0 | Databases 7 | Total DB size 171.41 M6

Latest notifications

| | | |
|---|---|---|
| 30.06.2025, 12:10 | operation was successful | pgpro_pwr: take snapshot in ppem db every 5 min |
| 30.06.2025, 12:05 | operation was successful | pgpro_pwr: take snapshot in ppem db every 5 min |
| 30.06.2025, 12:00 | operation was successful | pgpro_pwr: take snapshot in ppem db every 5 min |

Latest messages

| Date, time | Instance | User | Database | Type | Message |
|---|---|---|---|---|---|
| 30.06.2025, 12:12:01.00 | instance-1 | postgres | postgres | LOG | disconnection: session time: 0:00:00.007 user=postgres database=postgres host=127.0.0.1 port=34744 |
| 30.06.2025, 12:12:01.00 | instance-1 | postgres | postgres | LOG | connection authorized: user=postgres database=postgres |
| 30.06.2025, 12:12:01.00 | instance-1 | postgres | postgres | LOG | connection authenticated: user="postgres" method=trust (/var/lib/pgpro/ent-17/data/pg_hba.conf:105) |
| 30.06.2025, 12:12:01.00 | instance-1 | | | LOG | connection received: host=127.0.0.1 port=34744 |
| 30.06.2025, 12:12:01.00 | instance-1 | | | LOG | connection received: host=127.0.0.1 port=34742 |

## Summary Page [#](#)

After you log in to the web application, the **Summary** page with the main PPEM information is automatically displayed. The following blocks are displayed on the summary page:

- **Instances**: Allows you to go to managing [instances](#).
- **Clusters**: Allows you to go to managing [clusters](#).
- **Databases**: Allows you to go to managing [databases](#).
- **Total DB size**: Displays the total amount of memory occupied by all databases.
- **Recent Notifications**: displays the three latest notifications received from PPEM.
- **Latest messages**: Displays the latest [log](#) messages. You can view the full message log by clicking **Go to message log**.

## Navigation Panel [#](#)

Navigation through the web application pages is done using the navigation panel. When you go to a page, its contents are displayed on the right.

Certain pages have nested pages. If you go to a page containing nested pages, they are also displayed in the navigation panel.

To collapse or expand the navigation panel, at the bottom of the navigation panel, click **Collapse** or the expand icon →|.

You can go to the **Summary** page from any page by clicking the PPEM logo at the top of the navigation panel.

## Dashboard [#](#)

When going to any web application page, the dashboard is displayed at the top of the page. You can perform the following actions using the dashboard:

- Find a PPEM component by entering its name in the search field.
- Switch between web application themes:
  - To switch to the dark theme, click the moon icon ☾.
  - To switch to the light theme, click the sun icon ☼.
- Change the web application language:
  - To switch the language to English, select **English** from the drop-down list.
  - To switch the language to Russian, select **Русский** from the drop-down list.
- View notifications received from PPEM by clicking the bell icon 🔔. You can perform the following actions in the window that opens:
  - To mark all notifications as read, click **Read all**.
  - To view [tasks](#), click **Read all**.
- View [roles](#) assigned to your account by hovering over the username.
- Log out of the web application by clicking the logout icon ⇥.

## System Configuration [#](#)

This section explains how to [view PPEM summary information](#), as well as how to manage [agents](#) and [tags](#).

## Viewing PPEM Summary Information [#](#)

To view PPEM summary information, in the **System** section of the navigation panel, go to the **About PPEM** page.

The following blocks will be displayed:

- **Platform**: Information about the server where the manager is installed. The following parameters are displayed:
  - **Platform**: The architecture of the server CPU.
  - **Kernel**: The Linux kernel version.
  - **System**: The version of the operating system installed on the server.
  - **Nodename**: The FQDN of the server.
- **Manager**: Information about the manager. The following parameters are displayed:
  - **Version**: The manager version.
  - **Configuration**: The path to the server catalog where the `ppem-manager.yml` configuration file is placed.
- **Repository**: Information about the repository. The following parameters are displayed:

    - **Version**: The repository version.
    - **Connection**: The connection string for connecting to the repository database.

- **Useful links**: Links to the Postgres Pro official documentation.

## Agents #

This section explains how to manage agents. It includes the following instructions:

- Adding an Agent
- Viewing Agents
- Editing an Agent
- Deleting an Agent

For more information about agents, refer to the Architecture section.

### Adding an Agent #

Once installed on the server, the agent is automatically created in the web application. If this does not happen, you can add the installed agent manually.

To add an agent, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Agents** page.

   The table of agents will be displayed.

2. In the top-right corner of the page, click **Add agent**.

   A window for adding an agent will open.

3. Enter details of the new agent (parameters marked with an asterisk are required):

   - **Name**: The unique name of the agent.

   - **Host**: The network address of the server where the agent is installed. You can find the server network address by entering it in the search field.

   - **Port**: The port number that the server uses to listen for incoming connections.

   - **Protocol**: The protocol used by the agent. Possible values:

     - **http**. This value is selected by default.
     - **https**.

   **Note**

   The URL for connecting the agent to the manager is generated automatically based on the **Host**, **Port**, and **Protocol** parameters.

4. Click **Save**.

The agent will be added and displayed in the table of agents.

### Viewing Agents #

To view agents, in the **Management** section of the navigation panel, go to the **Infrastructure** → **Agents** page.

The table of agents with the following columns will be displayed:

- **Agent**: The unique name of the agent. If the agent is automatically created in the web application, the name is generated based on the host name of the server where the agent is installed.

  The agent status indicator is displayed to the left of the agent name. You can view the status by hovering over the agent name. Possible statuses:

  - `online`: The agent is working normally.
  - `stopped`: The agent service was stopped.
  - `unknown`: The agent status is unknown.
  - `not_responding`: Failed to access the agent service.

  You can filter and sort values in this column using the corresponding icons.

- **URL**: The URL for connecting the agent to the manager. You can filter values in this column using the corresponding icon.

- **Authentication key**: The key that the agent uses to authenticate when connecting to the manager. You can copy or view the authentication key by clicking **Copy the key** or the view icon next to it.

- **Version**: The agent version. If the agent version differs from the manager version, the corresponding warning is displayed.

- **Actions**: Actions with the agent. For more information about the actions that you can perform with an agent, refer to other instructions in this section.

You can perform the following actions with the table of agents:

- To update the table, in the top-right corner of the page, click **Refresh data**.
- To reset all filters, in the top-right corner of the page, click **Reset**.

### Editing an Agent #

By default, agents automatically update their information, so the changes you make get overwritten over time. You can disable the automatic update. To do this, set `collectors.instances.disabled` to `true` in the `ppem-agent.yml` configuration file.

To edit an agent, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Agents** page.

   The table of agents will be displayed.

2. Click **Edit** next to the agent that you want to edit.

   The agent editing window will open.

3. Edit the required agent parameters.

4. Click **Save**.

The agent will be edited, and the updates will be displayed in the table of agents.

### Deleting an Agent #

**Warning**

After deleting the agent from the web application, it must also be deleted on the server. Otherwise, the agent will be automatically created in the web application again.

To delete an agent, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Agents** page.

   The table of agents will be displayed.

2. Click **Delete** next to the agent that you want to delete.

   The agent deletion window will open.

3. Click **Delete*.

The agent will be deleted and no longer be displayed in the table of agents.

## Tags #

*Tags* are unique labels that can be assigned to instances when creating and editing instances. You can filter instances by assigned tags.

This section explains how to manage tags. It includes the following instructions:

- Creating a Tag
- Viewing Tags
- Editing a Tag
- Deleting a Tag

### Creating a Tag #

To create a tag, follow the steps below:

1. In the **System** section of the navigation panel, go to the **Settings** page.

   The table of tags will be displayed.

2. In the top-right corner of the page, click **Add tag**.

   The tag creation window will open.

3. Enter details of the new tag (parameters marked with an asterisk are required):

   - **Description**: A short description of the tag.
   - **Display name**: The unique name of the tag.

4. Click **Add**.

The tag will be created and displayed in the table of tags.

### Viewing Tags #

To view tags, in the **System** section of the navigation panel, go to the **Settings** page.

The table of tags with the following columns will be displayed:

- **Description**: A short description of the tag. You can sort values in this column using the corresponding icon.
- **Display name**: The unique name of the tag. You can sort values in this column using the corresponding icon.
- **Actions**: Actions with the tag. For more information about the actions that you can perform with a tag, refer to other instructions in this section.

You can perform the following actions with the table of tags:

- To change the column width, drag its border with the mouse.
- To update the table, in the top-right corner of the page, click **Refresh data**.

### Editing a Tag #

To edit a tag, follow the steps below:

1. In the **System** section of the navigation panel, go to the **Settings** page.

   The table of tags will be displayed.

2. Click **Edit** next to the tag that you want to edit.

   The tag editing window will open.

3. Edit the required tag parameters.

4. Click **Save**.

The tag will be edited, and the updates will be displayed in the table of tags.

### Deleting a Tag #

> **Warning**
>
> Deleted tags cannot be restored.

To delete a tag, follow the steps below:

1. In the **System** section of the navigation panel, go to the **Settings** page.

   The table of tags will be displayed.

2. Click **Delete** next to the tag that you want to delete.

   The tag deletion window will open.

3. Click **Delete*.

The tag will be deleted and will stop being displayed in the table of tags.

## PPEM Users #

Users manage PPEM through the web application. If required, they can be grouped for centralized management.

Users access levels to different operations are determined by roles. Each role has a predefined set of privileges. Roles can be assigned to both individual users and groups.

For more information about roles and privileges, refer to the RBAC Model section.

This section explains how to manage users and user groups. It also includes information about existing roles and privileges.

## Managing Users #

This section explains how to manage users. It includes the following instructions:

### Creating a User #

To create a user, follow the steps below:

1. In the **System** section of the navigation panel, go to the **Users** page.

   The table of users will be displayed.

2. In the top-right corner of the page, click **Add user**.

   The user creation window will open.

3. Enter details of the new user (parameters marked with an asterisk are required):

   - **Name**: The name of the user.
   - **Last name**: The last name of the user.
   - **Email**: The email address of the user.
   - **Login**: The user login for logging in to the web application.
   - **Password**: The user password for logging in to the web application. The minimum password length is 8 characters.
   - **Position**: The user position in your organization.
   - **Telephone**: The phone number of the user.
   - **Group**: The groups to which the user will be added. You can deselect the group by clicking the cross icon × next to its name.

     You can also add a user to a group when creating or editing the group.

   - **Access privileges**: roles that will be assigned to the user. To assign a role to a user, click **Add role +** and select a role from the drop-down list. You can remove a role by clicking 🗑 next to it.

     For certain roles, select the objects to which these roles will grant access from the drop-down list. You can remove the object by clicking the cross icon × next to its name.

   - **Status**: The status of the user access to the web application after the user is created. Possible values:
     - *Access granted**: The user can log in to the web application.
     - **Access blocked**: The user is denied access to the web application. To grant a user access to the web application, edit the user and select **Access granted** from the **Status** drop-down list.

4. Click **Save**.

The user will be created and displayed in the table of users.

### Viewing Users #

To view users, in the **System** section of the navigation panel, go to the **Users** page.

The table of users with the following columns will be displayed:

- **User name**: The name and last name of the user. You can filter and sort values in this column using the corresponding icons.
- **Login**: The user login for logging in to the web application. You can filter and sort values in this column using the corresponding icons.
- **Email**: The email address of the user. You can filter and sort values in this column using the corresponding icons.
- **Individual roles**: The roles assigned to the user. You can filter values in this column using the corresponding icons.
- **Group**: PPEM groups and LDAP groups to which the user is added.

You can perform the following actions with the table of users:

- To update the table, in the top-right corner of the page, click **Refresh data** ⟳.
- To reset all filters, in the top-right corner of the page, click **Reset**.

### Editing a User #

To edit a user, follow the steps below:

1. In the **System** section of the navigation panel, go to the **Users** page.

   The table of users will be displayed.

2. Click **Edit** ✎ next to the user that you want to edit.

   The user editing window will open.

3. Edit the required user parameters.

4. Click **Save**.

The user will be edited, and the changes will be displayed in the table of users.

### Deleting a User #

**Warning**

Deleted users cannot be restored.

To delete a user, follow the steps below:

1. In the **System** section of the navigation panel, go to the **Users** page.

   The table of users will be displayed.

2. Click **Delete** 🗑 next to the user you want to delete.

   The user deletion window will open.

3. Click **Yes, delete**.

The user will be deleted and no longer be displayed in the table of users.

## Roles and Privileges #

This section describes roles and privileges. It also includes the instruction on how to view roles and privileges. For more information about roles and privileges, refer to the Role-Based Access Control (RBAC) Model section.

### Roles Description #

The following roles are available in PPEM:

- `System administrator role`: Has a full set of privileges.
- `Guest role`: Is allowed to view a limited number of objects.
- `Instance objects administrator role`: Is allowed to manage instance objects.
- `Instance objects viewer role`: Is allowed to view instance objects.
- `Instance administrator role`: Is allowed to manage an instance.
- `Instance PSQL user role`: Is allowed to run psql within an instance.
- `Access administrator role`: Is allowed to manage user roles and group roles.
- `Repositories and packages administrator`: Is allowed to manage repositories and packages.

You cannot create new roles or edit the existing ones in the web application. However, it can be done using API.

### Privileges Description #

Roles can have the following privileges:

- `privilege_view`: Viewing privileges and their descriptions.
- `role_create`: Creating roles.
- `role_view`: Viewing roles and their parameters.
- `role_edit`: Editing roles.
- `role_delete`: Deleting roles.
- `user_create`: Creating users.
- `user_view_all`: Viewing any users and their parameters.
- `user_edit_all`: Editing any users.
- `user_delete`: Deleting users.
- `project_create`: Creating projects.
- `project_view`: Viewing projects and their parameters.
- `project_edit`: Editing projects.
- `project_delete`: Deleting projects.
- `notification_create`: Creating notifications.
- `notification_view`: Viewing notifications and their parameters.
- `notification_edit`: Editing notifications.
- `notification_delete`: Deleting notifications.
- `group_create`: Creating user groups.
- `group_view`: Viewing user groups and their parameters.
- `group_edit`: Editing user groups.
- `group_delete`: Deleting user groups.
- `host_create`: Creating servers.
- `host_view`: Viewing servers and their parameters.
- `host_edit`: Editing servers.
- `host_delete`: Deleting servers.
- `agent_create`: Creating agents.
- `agent_view`: Viewing agents and their parameters.
- `agent_edit`: Editing agents.
- `agent_delete`: Deleting agents.
- `instance_create`: Creating instances.
- `instance_view`: Viewing instances.
- `instance_edit`: Editing instances.
- `instance_delete`: Deleting instances.
- `session_view_all`: Viewing any user sessions.
- `session_delete_all`: Editing any user sessions.
- `session_update`: Updating user sessions.
- `command_create`: Creating commands.
- `command_view_all`: Viewing any commands.
- `command_edit_all`: Editing any commands.
- `command_delete_all`: Canceling any commands.
- `instance_object_view`: Viewing instance objects and their parameters.
- `metrics_view`: Viewing metrics.
- `job_create`: Creating jobs.
- `job_view_all`: Viewing any jobs.
- `job_edit_all`: Editing any jobs.
- `job_delete_all`: Deleting any jobs.
- `backup_create`: Creating backups.
- `backup_view`: Creating backups.
- `backup_edit`: Editing backups.
- `backup_delete`: Deleting backups.
- `datasource_create`: Creating data storages.
- `datasource_view`: Viewing data storages.
- `datasource_edit`: Editing data storages.
- `datasource_delete`: Deleting data storages.
- `maintenance_create`: Executing maintenance commands.
- `instance_service_control`: Executing utility commands.
- `instance_settings_create`: Creating instance parameters. This service privilege allows agents to add instance parameters to the repository database.
- `instance_settings_view`: Viewing instance parameters.
- `instance_settings_edit`: Editing instance parameters.
- `query_state_read`: Executing the `pg_query_state` command.
- `logs_view`: Viewing logs.
- `chart_create`: Creating graphs.
- `chart_view`: Viewing graphs.
- `chart_edit`: Editing graphs.
- `chart_delete`: Deleting graphs.
- `chart_group_create`: Creating graph groups.
- `chart_group_view`: Viewing graph groups.
- `chart_group_edit`: Editing graph groups.
- `chart_group_delete`: Deleting graph groups.

- `stat_activity_view`: Viewing the `pg_stat_activity` view statistics.
- `stat_statements_view`: Viewing any SQL statements executed by the server.
- `overview_view`: Viewing the system overview.
- `tag_create`: Creating tags.
- `tag_view`: Viewing tags and their parameters.
- `tag_edit`: Editing tags.
- `tag_delete`: Deleting tags.
- `progress_stats_view`: Viewing `pg_stat_progress_*` views statistics.
- `about_view`: Viewing the system information.
- `pgpro_pwr_databases_view`: Viewing `pgpro_pwr` extensions.
- `pgpro_pwr_servers_delete`: Deleting`pgpro_pwr` servers.
- `pgpro_pwr_servers_view`: Viewing `pgpro_pwr` servers.
- `replication_node_create`: Creating replication nodes.
- `pgpro_pwr_servers_add`: Adding`pgpro_pwr` servers.
- `pgpro_pwr_servers_patch`: Installing patches for `pgpro_pwr`extensions.
- `stat_locktree_view`: Viewing a locktree.
- `pgpro_pwr_samples_create`: Creating `pgpro_pwr` samples.
- `pgpro_pwr_samples_get`: Viewing `pgpro_pwr` samples.
- `pgpro_pwr_samples_delete`: Deleting `pgpro_pwr` samples.
- `pgpro_pwr_report_create`: Creating`pgpro_pwr` reports.
- `pgpro_pwr_report_delete`: Deleting `pgpro_pwr` reports.
- `replication_node_view`: Viewing replication nodes.
- `pgpro_pwr_report_view`: Viewing `pgpro_pwr` reports.
- `settings_preset_view`: Viewing presets.
- `pgpro_pwr_overview`: Viewing the contents of `pgpro_pwr`reports.
- `user_roles_edit`: Assigning and removing user roles.
- `group_roles_edit`: Assigning and removing user group roles.
- `user_groups_edit`: Adding and deleting users from groups.
- `job_run_all`: Running any jobs.

#### Viewing Roles and Privileges #

To view roles and privileges, in the **System** section of the navigation panel, go to the **Users** → **Roles** page.

Roles will be displayed. For each role, the class of objects it provides access to is displayed, as well as the table of privileges with the following columns:

- **Privilege**: The unique name of the privilege.
- **Name**: The description of the privilege.
- **Parametric**: Whether the privilege applies to specific objects.
- **Objects**: The objects to which the privilege applies.

To display the table of privileges of a specific role, in the top-left corner of the page, select the role from the drop-down list.

## Replication Clusters #

PPEM supports both standard replication clusters and BiHA clusters (Built-in High Availability). If you install an agent on a server where a cluster is created, the cluster is automatically created in the web application. Standard clusters can also be created manually.

For more information, refer to the official Postgres Pro documentation on replication and built-in high availability.

This section explains how to manage clusters. It includes the following instructions:

- Creating a Cluster
- Viewing Clusters
- Editing a Cluster

### Creating a Cluster #

> **Note**
>
> PPEM does not support the creation of BiHA clusters in the web application.

To create a cluster, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Clusters** page.

   The table of clusters will be displayed.

2. In the top-right corner of the page, click **Create cluster**.

   The cluster creation window will open.

3. At the **Mode** step, select **From instance** and click **Next**.

4. At the **Parameters** step, enter details of the new cluster (parameters marked with an asterisk are required):

   - **Primary node**: The primary node of the cluster. When the primary node is selected, the following information about the instance is displayed:

     - **Version and edition**: The Postgres Pro version and edition installed on the instance server.
     - **Data catalog**: The path to the server directory where the main instance directories and files are placed.
     - **Network address** and **Port**: The network address and port number that the instance uses to accept client connections.
     - **Database**: The database for connecting to the instance.
     - **User**: The name of the DBMS superuser on behalf of which the agent connects to the instance.

   - **Standby server nodes**: Allows adding nodes to a cluster. To ensure a quorum, it is recommended to add an odd number of nodes to the cluster.

     To add nodes to a cluster, follow the steps below:

     1. Click **Add servers**.

     2. Select the node from the drop-down list. When the node is selected, the following information about the instance is displayed:

        - **Data catalog**: The path to the server directory where the main instance directories and files are placed.
        - **Network address** and **Port**: The network address and port number that the instance uses to accept client connections.

     3. Click **Add server**.

        You can add multiple nodes. The added node can be removed by clicking the bin icon 🗑 next to it.

     4. Click **Add to cluster**.

     Nodes will be added to the cluster and displayed in the **Standby server nodes** section. The list of nodes can be edited by clicking **Edit**.

5. Click **Create cluster**.

The cluster will be created and displayed in the table of clusters.

### Viewing Clusters #

To view clusters, in the **Management** section of the navigation panel, go to the **Infrastructure** → **Clusters** page.

The table of clusters with the following columns will be displayed:

- **Cluster**: The unique name and identifier of the cluster. You can filter and sort values in this column using the corresponding icons.
- **Failover manager**: The failover manager of the cluster. Possible values:
  - `Missing`
  - `Patroni`
  - `Biha`
  - `Shardman`

  You can filter values in this column using the corresponding icon.
- **Nodes, pcs.**: The number of nodes in the cluster. You can sort values in this column using the corresponding icon.
- **Version**: The Postgres Pro version and edition on the cluster nodes. You can filter and sort values in this column using the corresponding icons.
- **State**: The state of the cluster. Possible values:
  - `Active`: All cluster nodes are started.
  - `Stale`: Replication is stopped in the cluster.
  - `Promotion`: The primary node of the cluster is being changed.

  You can filter values in this column using the corresponding icon.
- **Network address**: The network address of the cluster primary node.
- **Last update**: The date and time of the last update of the cluster state information.

  **Note**

  The manager receives cluster state information from agents with a delay. Due to the delay, the web application may display outdated information.

  You can sort values in this column using the corresponding icon.
- **Actions**: Actions with the cluster. For more information about the actions that you can perform with a cluster, refer to other instructions in this section.

You can perform the following actions with the table of clusters:

- To change the display order of the columns, in the top-right corner of the table, click **Show/hide columns**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Show/hide columns**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-left corner of the table, click **Reset**.

### Editing a Cluster #

To edit a cluster, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Clusters** page.

   The table of clusters will be displayed.
2. Click **Edit** next to the cluster that you want to edit.

   The cluster editing window will open.
3. Edit the required cluster parameters.
4. Click **Save**.

The cluster will be edited, and the updates will be displayed in the table of clusters.

## Instances #

This section explains how to manage and configure DBMS instances. It contains the following instructions:

- Managing Instances
- Tablespaces
- Databases
- Metrics
- Activity
- SQL statistics
- Instance Parameters
- Extensions
- Profiler
- Authentication
- Roles

### Managing Instances #

This section explains how to manage instances. It includes the following instructions:

- Creating an Instance
- Viewing Instances
- Checking the Consistency of an Instance Catalog
- Stopping and Starting an Instance
- Restarting an Instance
- Creating a Backup
- Editing an Instance Configuration Preset
- Editing an Instance
- Selecting a Repository Instance
- Deleting an Instance

#### Creating an Instance #

Agents automatically discover hosts, instances, instance objects, and other components installed on the server and create them in the web application. You can configure this in the `ppem-agent.yml` agent configuration file.

**Note**

The automatic discovery and a number of other features are not supported when installing PPEM in secure environments.

There are two ways of creating instances that were not automatically discovered:

- Creating a New Instance
- Adding an Existing Instance
- Creating an Instance from a Backup

### Creating a New Instance #

Creating a new instance includes creating a new main data catalog and starting the instance service on the server.

To create a new instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. In the top-right corner of the page, click **ADD INSTANCE**.

   The instance creation window will open.

3. In the **General** step, select **Create a new instance**, and then click **Next**.

4. In the **Instance parameters** step, enter details of the new instance (parameters marked with an asterisk are required):

   - **Name**: The unique name of the instance.

   - **Server**: The server on which the instance is installed.

   - **System user**: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user. It is recommended to ensure that the specified user exists in the OS.

   - **Main data directory**: The path to the server catalog where the main instance catalogs and files will be placed.

   - **Connection address** and **Connection port**: The network address and port number that the instance will use for receiving client connections. The default network address is `localhost`.

   - **Authentication method**: The authentication method that the instance will use for verifying users when receiving client connections. Possible values:

     - **scram-sha-256**. This value is selected by default.
     - **md5**.
     - **trust**: Do not perform authentication. It is recommended to select this value only for test environments.

   - **DB super user** and **Super user password**: The name and password of the DBMS user that will be created and on behalf of which the agent will connect to the instance. The default DBMS user name is `postgres`.

   - **Tags**: The tags that will be assigned to the instance. You can unassign a tag by clicking the cross icon × next to its name.

   - **Configuration preset**: The configuration preset that will be applied to the instance. Possible values:

     - **—**: Do not apply a configuration preset to the instance. This value is selected by default.
     - **Settings for 1C**: Apply the 1C configuration preset to the instance.
     - **Settings for OLTP**: Apply the OLTP (OnLine Transaction Processing) configuration preset to the instance.

     You can edit the configuration preset later.

   - **Start instance after creation**: Whether the instance service starts after instance creation. This checkbox is disabled by default.

5. Click **Execute**.

The new instance will be created and displayed in the table of instances.

### Adding an Existing Instance #

You may need to add an existing instance if the automatic discovery of instances installed on the server is disabled.

Before performing this procedure, ensure that the instance is running on the server and is ready to receive client connections.

To add an existing instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. In the top-right corner of the page, click **ADD INSTANCE**.

   The instance creation window will open.

3. In the **General** step, select **Add an existing instance**, and then click **Next**.

4. In the **Instance parameters** step, enter details of the instance (parameters marked with an asterisk are required):

   - **Name**: The unique name of the instance.
   - **Server**: The server where the instance is installed.
   - **Main data directory**: The path to the server catalog where the main instance catalogs and files will be placed.
   - **Connection address** and **Connection port**: The network address and port number that the instance uses for receiving client connections. The default network address is `localhost`.
   - **DB super user** and **Super user password**: The name and password of the DBMS user on behalf of which the agent will connect to the instance. The default DBMS user name is `postgres`.
   - **Tags**: The tags that will be assigned to the instance. You can unassign a tag by clicking the cross icon × next to its name.

5. Click **Execute**.

The existing instance will be added and displayed in the table of instances.

### Creating an Instance from a Backup #

For more information about backups, refer to the Backup section.

Before performing this procedure, create a backup.

To create an instance from a backup, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. In the top-right corner of the page, click **ADD INSTANCE**.

   The instance creation window will open.

3. In the **General** step, select **Create from backup**, and then click **Next**.

4. In the **Select backup** step, enter details of the backup from which the instance will be created (parameters marked with an asterisk are required):

   - **Catalog**: The storage catalog where the backup is placed.
   - **Instance**: The instance for which the backup is created.
   - **Period**: The time period when the backup was created.
   - **Backup**: The backup from which the instance will be created.

5. Click **Next**, and then in the **Instance parameters** step, enter details of the new instance (parameters marked with an asterisk are required):

   - **Name**: The unique name of the instance.

   - **Server**: The server on which the instance is installed.

   - **System user**: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user. It is recommended to ensure that the specified user exists in the OS.

   - **Main data directory**: The path to the server catalog where the main instance catalogs and files will be placed.

   - **Connection address** and **Connection port**: The network address and port number that the instance will use for receiving client connections. The default network address is `localhost`.

   - **Tags**: The tags that will be assigned to the instance. You can unassign a tag by clicking the cross icon ✕ next to its name.

   - **Backup**: The backup from which the instance will be created. The value is entered automatically.

   - **Backup size**: The size of the backup from which the instance will be created. The value is entered automatically.

   - **Restore point**: The state to which the instance must be restored. Possible values:

     - **—**: Restore the instance to the last state covered by the backup. This value is selected by default.
     - **Time**: Restore the instance state to a specific date and time covered by the backup. For this value, in the **Time** field, specify the required date and time.
     - **LSN**: Restore the instance to the state corresponding to a specific WAL LSN. For this value, in the **LSN** field, enter the required WAL LSN .
     - **Transaction**: Restore the instance state to a specific transaction number. For this value, in the **Transaction** field, enter the transaction number.

     For the **Time**, **LSN**, and **Transaction** values, you must also specify the following parameters:

     - **Restore including the specified value**: Whether the instance state is restored including the specified value. This checkbox is enabled by default.

       For example, if you entered `123456` in the **Transaction** field and enabled the **Restore including the specified value** checkbox, the instance state will be restored to the transaction 123456. If you did not enable the checkbox, the instance state will be restored to the 123455 transaction.

     - **Action after restore**: The action to perform on the server after restoring the instance to the required state. Possible values:

       - **Pause after restore**: Pause the creation of the instance from the backup. It allows ensuring that the correct state was restored for the instance before creating it. This value is selected by default.
       - **Promote after restore**: Create the instance from the backup and start receiving client connections.
       - **Shutdown instance after restore**: Create the instance from the backup, and then stop the server.

   - **Partial recovery**: Specifies the instance databases that will be restored or excluded from the restoration process. Possible values:

     - **Do not use**: Restore all instance databases. This value is selected by default.
     - **Exclude some databases**: Exclude specific databases from restoration.
     - **Restore some databases**: Restore the specific instance databases.

     For the **Exclude some databases** and **Restore some database** values, specify the unique name of the database using the **Databases** parameter, and then click **Add database**. You can add multiple names of databases. A database name can be removed by clicking the bin icon icon-delete next to it.

   - **Checking available space**: Allows checking whether there is enough disk space on the server for creating the instance from the backup. To start the check, click **Check**.

6. Click **Execute**.

The instance will be created from the backup and displayed in the table of instances.

## Viewing Instances #

To view instances, in the **Management** section of the navigation panel, go to the **Infrastructure → Instances** page.

The table of instances with the following columns will be displayed:

- **Name**:

  - The unique name of the instance.
  - `PostgreSQL version`: The version and edition of Postgres Pro on the instance server.
  - **Data directory**: The path to the server catalog where the main instance catalogs and files are placed.
  - `Port`: The port number that the instance uses for receiving client connections.

  You can filter and sort values in this column using the corresponding icons.

- **Server**:

  - The FQDM of the instance server.

  - The IP address of the instance server.

  - The status of the instance. Possible values:

    - `Unknown`: The status of the instance is unknown.
    - `Initialization scheduled**: The instance initialization was scheduled.
    - `Initializing`: The instance is being initialized.
    - `Initialized`: The instance is initialized.
    - `Restoring`: The instance is being created from a backup.
    - `Restored`: The instance is created from the backup.
    - `Restore canceled`: The creation of an instance from a backup was canceled.
    - `Starting`: The instance is being started.
    - `Started`: The instance is started.
    - `Stopping`: The instance is being stopped.
    - `Stopped`: The instance is stopped.
    - `Restarting`: The instance is being restarted.
    - `Reloading`: The cluster is being reloaded.
    - `Failed`: There is an error with the instance.
    - `Scheduled for removal`: The instance is being deleted.

        - `Base backup`: A backup is being created for the instance.
        - `Agent not responding`: The agent installed on the instance server is not responding.

    You can filter values in this column using the corresponding icon.

- **Role**: The role of the instance. Possible values:

    - `primary`: The instance is the main node of a replication cluster. It provides data replication to standby nodes.
    - `standby`: The instance is a standby node of a replication cluster and receives replicated data from the primary node.
    - `cascade`: The instance is a standby node of a streaming replication cluster. It receives replicated data from the main node and provides data replication to other standby nodes simultaneously.
    - `maintenance`: The instance is a separate node and is not added to a replication cluster.

    For more information about data replication, refer to the official Postgres Pro documentation.

    You can filter values in this column using the corresponding icon.

- **DB**:

    - `Database`: The unique names of databases.
    - `Transactions per second`: The number of transactions per second in the database.
    - `Connections`: The number of client database connections.

- **Tags**: The tags assigned to the instance. You can filter values in this column using the corresponding icons.

You can perform the following actions with the table of instances:

- To update the table, in the top-right corner of the page, click **Refresh data**.
- To reset all filters, in the top-right corner of the page, click **Reset**.

### Checking the Consistency of an Instance Catalog #

The catalog consistency check ensures that the main instance files were not damaged during storage.

To check the consistency of an instance catalog, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

    The table of instances will be displayed.

2. Click **Check directory integrity** next to the instance for whose catalog you want to check the consistency.

    The window for starting the consistency check will open.

3. Click **Run**.

The consistency check of the instance catalog will start. To view the result, in the **Management** section of the navigation panel, go to the **Tasks console** page and click **Show log** next to the task that was automatically created for checking the consistency of the instance catalog.

### Stopping and Starting an Instance #

To stop or start an instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

    The table of instances will be displayed.

2. Click **Stop** or **Run** next to the instance that you want to stop or start.
3. To stop an instance, in the window that opens, click **Stop**.

The instance will be stopped or started.

### Restarting an Instance #

To restart an instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

    The table of instances will be displayed.

2. Click **Restart** next to the instance that you want to restart.

    The instance restarting window will open.

3. Click **Restart**.

The instance will be restarted.

## Creating a Backup #

For more information about backups, refer to the Backup section.

To create a backup, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

    The table of instances will be displayed.

2. Click **Create backup** next to the instance for which you want to create a backup.

    The backup creation window will open.

3. In the **General** step, enter details of the new backup (parameters marked with an asterisk are required):

    - **Instance**: The instance for which the backup will be created. The value is entered automatically.

    - **Copy storage**: The storage where the backup will be placed. You can select a local or S3 storage. A local storage must be located on the same server as the instance for which you are creating the backup.

    - **User** and **Password**: The name and password of the DBMS user under which the backup will be performed.

    - **Database**: The database for connecting to the instance.

    - **Backup mode**: The backup creation mode. Possible values:

        - **full**. This value is selected by default.
        - **page**.

        - **ptrack**.
        - **delta**.

      For more information about backup modes, refer to the official Postgres Pro documentation on `pg_probackup`.

    - **Threads count**: The number of parallel threads that will be started at backup creation.

    - **Waiting time (sec)**: The waiting timeout in seconds for WAL segment archiving and streaming. Default value: `300`.

    - **Create a stand-alone backup**: Whether to create a streaming backup that includes the WAL records required for restoring the instance later. This checkbox is disabled by default.

    - **Replication slot**: The replication slot that will be used for transferring WAL records.

    - **Create temporary replication slot**: Whether to create a temporary replication slot for transferring WAL records of the instance for which you are creating the backup. This checkbox is disabled by default. If this checkbox is enabled, WAL segments are available even if they are switched at backup creation.

4. Click **Next**, and then in the **Advanced** step, specify additional information if required:

    - **External catalogs**: The path to the instance catalog that will also be included in the backup.

    - **include log catalog**: Whether the backup includes the catalog with the instance activity logs. This checkbox is disabled by default.

    - **Don't check copy**: Whether to skip the automatic verification of the created backup. This checkbox is disabled by default. If this checkbox is enabled, the backup is created faster.

    - **Smooth execution of the checkpoint**: Whether backup creation starts only after the scheduled checkpoint. This checkbox is disabled by default.

    - **Disable block-level verification**: Whether to disable the block-level checksum verification for faster consistency checking at backup creation. This checkbox is disabled by default.

    - **Compression level**: The file compression level at backup creation. You can enter a value from 0 to 9, where 0 — to disable the file compression, and 9 — to use the highest file compression. Default value: `0`.

    - **Compression algorithm**: The algorithm used for compressing files. This parameter is available only if you entered a value greater than 0 in the **Compression level** field. Possible values:

        - **zlib**
        - **lz4**
        - **zstd**
        - **pglz**

    - **Pinning**: The pinning parameters of the backup. Possible values:

        - **Do not pin**: Do not pin the backup. If you select this value, the parameters specified in the **Parameters of storing** section are used. This value is selected by default.
        - **ttl**: After the backup is created, it cannot be deleted from a storage during a specific number of days. For this value, in the **Retention period, days** field, enter the required number of days.
        - **expire-time**: The backup cannot be deleted from a storage until a specific date and time. For this value, in the **Retention period until** field, specify the required date and time.

    - **Storage parameters**: The backup storage parameters of the storage catalog created for the instance. Available parameters:

        - **Retention redundancy**: The maximum number of full backups. For example, if you specify `3`, the catalog can contain a maximum of three full backups.

          To disable this limitation, specify `0`. In this case, the number of backups in the catalog is not limited.

        - **Retention window**: The number of days (24 hours) covered by backups. For example, if you specify `7`, the catalog must always contain backups required for restoring the data for the last seven days, including today.

          To disable this limitation, specify `0`. In this case, backups can be deleted from the catalog at any moment.

        - **WAL depth**: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR). For example, if you specify `3`, the catalog must always contain at least three backups on each timeline.

          To disable this limitation, specify `0`. In this case, point-in-time recovery is not possible.

        - **Expired copies**: The management policy for expired backups. Possible values:

            - **Merge**: Merge expired backups with new ones if possible.
            - **Delete**: Delete expired backups from the catalog.
            - **Remove expired WAL**: Delete WAL of expired backups from the catalog.

          You can enable all checkboxes simultaneously.

      The **Retention redundancy**, **Retention window**, and **WAL depth** are applied only if you enabled the **Merge** and/or **Delete** checkbox for the **Expired copies** parameter.

      The values of the **Retention redundancy** and **Retention window** parameters are considered simultaneously when deleting obsolete backups from the catalog. For example, if you entered `3` in the **Retention redundancy** field and `7` in the **Retention window** field, a maximum of three full backups will be saved, as well as all backups required for restoring the data for the past 7 days, including today.

      You can also configure storage parameters for an instance, as well as for a storage when creating or editing it. The following priority is applied:

        - Backup parameters are applied first.
        - Instance parameters are applied second.
        - Storage parameters are applied third.

      For more information about storage parameters, refer to the official Postgres Pro documentation on `pg_probackup`.

5. Click **Execute backup**.

The backup will be created.

## Editing a Configuration Preset of an Instance #

You can apply another configuration preset to the instance.

To edit a configuration preset of an instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click **Configuration presets** ⚙ next to the instance to which you want to apply another configuration preset.

   The window for editing the configuration preset will open.

3. Select a new configuration preset from the drop-down list.

4. Click **Apply**.

#### Editing an Instance #

You can select the repository cluster when editing a cluster.

To edit an instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click **Edit** next to the instance that you want to edit.

   The instance editing window will open.

3. Edit the required instance parameters.

4. Click **Save**.

The instance will be edited, and the updates will be displayed in the table of instances.

#### Selecting a Repository Instance #

The repository database is located in an instance. Similarly to other instances, you can create this instance in the web application. You must manually specify in the web application the instance where the repository database is located.

To select the repository instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click **Edit** next to the instance where the repository database is placed.

   The instance editing window will open.

3. Enable the **Instance of EM repository** checkbox. This checkbox is disabled by default.

4. Click **Save**.

#### Deleting an Instance #

> **Warning**
>
> After deleting the instance from the web application, it must also be deleted on the server. Otherwise, the instance will be automatically created in the web application again.
>
> When PPEM is installed in a secure environment, if you delete an automatically created instance, it will not be automatically created anymore. In this case, to restore the instance, you must add it manually.

To delete an instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click **Remove from EM** next to the instance that you want to delete.

   The instance deletion window will open.

3. (Optional) To delete the main instance data catalog on the server, enable the **Delete with data directory** checkbox. This checkbox is disabled by default.

4. Confirm the deletion and click **Delete **.

The instance will be deleted and no longer be displayed in the table of instances.

## Tablespaces #

Tablespaces allow you to manage how database object files are placed within the file system. A tablespace can store individual tables, indexes, and entire databases.

For more information about tablespaces, refer to the official Postgres Pro documentation.

This section explains how to manage tablespaces. It includes the following instructions:

- Creating a Tablespace
- Viewing Tablespaces
- Editing a Tablespace
- Updating the List of Tablespaces

#### Creating a Tablespace #

To create a tablespace, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where the tablespace will be created.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Tablespaces** page.

   The table of tablespaces will be displayed.

4. In the right-top corner of the page, click **Create space**.

   The tablespace creation window will open.

5. Enter details of the new tablespace (parameters marked with an asterisk are required):

   - **Instance**: The instance to which the tablespace will be assigned. The value is filled in automatically.
   - **Name**: The unique name of the tablespace.
   - **Catalog**: The path to the directory where the tablespace will be placed.
   - **Owner**: The DBMS user who will own the tablespace.
   - **seq_page_cost**: The value of the `seq_page_cost` parameter that will be assigned to the tablespace.
   - **random_page_cost**: The value of the `random_page_cost` parameter that will be assigned to the tablespace.
   - **effective_io_concurrency**: The value of the `effective_io_concurrency` parameter that will be assigned to the tablespace.

   - **maintenance_io_concurrency**:The value of the `maintenance_io_concurrency` parameter that will be assigned to the tablespace.
   - **compression**: The use of the Compressed File System. Possible compression algorithms: zstd, lz4, pglz, and zlib. This functionality is available only in Postgres Pro Enterprise.

   When creating a tablespace, a directory is created in the file system, and the tablespace is then created using the `CREATE TABLESPACE` command. You can also create a tablespace in an existing directory, provided the directory is empty.

6. Click **Save**.

The tablespace will be created and displayed in the table of tablespaces.

#### Viewing Tablespaces #

To view the list of tablespaces, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to view the tablespaces.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Tablespaces** page.

The table of tablespaces with the following columns will be displayed:

- **Name**: The unique name of the tablespace. You can filter and sort values in this column using the corresponding icons.
- **Instance**: The instance where the tablespace is created.
- **Owner**: The DBMS user who owns the tablespace.
- **Path**: The directory of the tablespace in the file system.
- **Configuration parameters**: Additional configuration parameters, specific to tablespaces.
- **Size**: The total size of objects within the tablespace.
- **Actions**: Actions with the tablespace. For more information about the actions that you can perform with a tablespace, refer to other instructions in this section.

You can perform the following actions with the table of tablespaces:

- To update the table, in the top-right corner of the page, click **Refresh data**.
- To reset all filters, in the top-right corner of the page, click **Reset filters**.

#### Editing a Tablespace #

Tablespaces can be renamed. When editing a tablespace, the ‘ALTER TABLESPACE’ SQL command is executed.

> **Note**
>
> System tablespaces cannot be edited.

To edit a tablespace, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where the tablespace is created.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Tablespaces** page.

   The table of tablespaces will be displayed.

4. Click **Edit** next to the tablespace that you want to edit.

   The tablespace editing window will open.

5. Edit the name of the tablespace.

6. Click **Save**.

The tablespace will be edited, and the updates will be displayed in the table of tablespaces.

#### Deleting a Tablespace #

> **Warning**
>
> Deleted tablespaces cannot be restored.

When deleting a tablespace, the `DROP TABLESPACE` SQL command is executed.

> **Notes**
>
> - You can only delete empty tablespace.
> - Before deleting a tablespace, move or delete all objects it contains.
> - You cannot delete system tablespaces.

To delete a tablespace, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where the tablespace is created.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Tablespaces** page.

   The table of tablespaces will be displayed.

4. Click**Delete** next to the tablespace that you want to delete.

   The tablespace deletion window will open.

5. Click **Delete*.

The tablespace will be deleted and no longer be displayed in the table of tablespaces.

### Databases #

PPEM allows working with databases, schemas, tables, indexes, functions, and sequences.

## Managing Databases #

This section explains how to manage databases. It contains the following instructions:

### Creating a Database #

To create a database, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. In the top-right corner of the page, click **Create database**.

   The database creation window will open.

3. Enter details of the new database (parameters marked with an asterisk are required):

   - **Instance**: The instance where the new database will be created.
   - **Database name**: The unique name of the database.
   - **Database owner**: The owner of the database.
   - **Encoding (ENCODING)**: The character encoding in the new database.
   - **Sorting category (LC_COLLATE)**: Specifies the `LC_COLLATE` value in the operating system environment of the database server.
   - **Category of symbol types (LC_CTYPE)**: Specifies the `LC_CTYPE` value in the operating system environment of the database server.
   - **Tablespace**: The tablespace where the database will be created.
   - **Allow connections**: Whether the database will be available for connections.
   - **Limiting the number of connections**: The maximum number of simultaneous database sessions.
   - **Is template**: Whether the created database will be a template.
   - **Template**: The template of the database.

4. Click **Save**.

The database will be created and displayed in the table of databases.

### Viewing Databases #

To view databases, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

The table of databases includes the following columns:

- **DB name**: The unique name of the database. You can filter and sort values in this column using the corresponding icons.
- **Total size**: The size of the database. The horizontal indicator shows the size of tables, bloat, and indexes. You can sort values in this column using the corresponding icon.
- **Wraparound**: The number of released transaction IDs in the database (percentage).
- **Instance**: The instance where the database is created.
- **Data collecting**: The amount of the instance service information collected by agents.
- **Tables**: General table information:
  - The number of tables
  - The size of tables
  - The table bloat percentage
- **Indexes**: General information about indexes:
  - The number of indexes
  - The size of indexes
  - The index bloat percentage
- **Actions**: Actions with the database. For more information about the actions that you can perform with a database, refer to other instructions in this section.

You can perform the following actions with the table of databases:

- To update the table, in the top-right corner of the page, click **Refresh data** ⟳.
- To reset all filters, in the top-right corner of the page, click **Reset**.

The database size is calculated as the total sum of all its relational objects. This allows avoiding redundant calls of the `pg_database_size` function that uses the recursive traversal of directories and files and can negatively impact the performance of the main instance workload.

> **Warning**
>
> If a database is created outside PPEM, the information about it can be displayed in the web application with a delay. The delay duration depends on the agent configuration. By default, it is one minute.

### Starting the PSQL Terminal #

The PSQL terminal provides low-level access to the DBMS instance and can be used when PPEM features are not sufficient. Although providing most of the standard psql functionality, it has certain limitations. For example, you cannot use the `\!` meta-command to run shell commands.

The DBMS instance connection parameters are used when the terminal is started.

To start the PSQL terminal, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click **PSQL** next to the database for which you want to start the terminal.

The PSQL terminal window will open. In the top-right corner of the window, you can expand the terminal to full screen, open it in a separate window, or close and exit it.

### Collecting the Planner Statistics #

The `ANALYZE` SQL command is executed when collecting the planner statistics.

To collect the planner statistics, perform the following actions:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the three vertical dots icon ⋮ → **Analyze** next to the database where you want to collect statistics.

   The statistics collection window will open.

4. Specify the required parameters (parameters marked with an asterisk are required):
   - **Set cron-string execution**: Allows specifying the schedule for performing the operation in the `crontab` format.
   - **Task planning**: Allows planning the task execution.
5. Click **Apply**.
6. Confirm the operation.

The statistics collection will start in the asynchronous mode.

> **Note**
>
> The `ANALYZE` SQL command is executed for collecting the planner statistics. Usually significant resources are not required for the statistics collection process. However, its duration depends on the number of tables and columns and can take a long time.

To track the planner statistics collection, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The **Activity** page will be displayed.

4. Select the **Analyze** tab.

The planner statistics will be displayed.

### Reindexing #

The `REINDEX` SQL command is executed when reindexing.

To start database reindexing, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the three vertical dots icon ⋮ → **Reindex** next to the database that you want to reindex.

   The reindexing window will open.

4. Specify the required parameters (parameters marked with an asterisk are required):
   - **Concurrently**: Whether the operation will be performed in the non-locking mode. This is the safest mode that allows avoiding blocking other sessions by slowing down the process of reindexing. This mode is used by default.
   - **Tablespace**: The tablespace where new indexes will be created.
   - **Set cron-string execution**: Allows specifying the schedule for performing the operation in the `crontab` format.
   - **Task planning**: Allows planning the task execution.
5. Click **Apply**.
6. Confirm the operation.

The reindexing task will start.

> **Note**
>
> The `REINDEX` SQL command is executed when reindexing. Depending on the number and size of indexes, significant resources can be required for performing this operation so it is recommended to track it.

To track reindexing, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The **Activity** page will be displayed.

4. Select the **Indexes** tab.

The information about database reindexing will be displayed.

### Vacuuming #

You can vacuum expired versions of rows in all database tables. In this case, the `VACUUM` SQL command is executed.

To vacuum outdated row versions, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the three vertical dots icon ⋮ → **Vacuum** next to the database that you want to vacuum.

   The vacuum window will open.

4. Specify the required parameters (parameters marked with an asterisk are required):

   - **Modes**: The vacuum modes. Possible values:

     - **Analyze**: Updates the statistics that the planner uses for selecting the most efficient way of query execution.
     - **Full**: Performs a full vacuum via reindexing all table files. Note that while the vacuum is being performed in this mode, all sessions working with tables that are being reindexed will be blocked.
     - **Freeze**: Performs an aggressive tuple freeze to expand freezing and release transaction IDs for further reuse. The aggressive freeze is always performed when overwriting the table, so do not select the **Freeze** mode if the **Full** mode is already selected.

   - **Set cron-string execution**: Allows specifying the schedule for performing the operation in the `crontab` format.

   - **Task planning**: Allows planning the task execution.

5. Click **Apply**.

6. Confirm the operation.

The vacuum task will start.

> **Note**
>
> The `VACUUM` SQL command is executed when vacuuming tables. Depending on the number and size of tables, significant resources can be required for performing this operation so it is recommended to track it.

To track reindexing, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The **Activity** page will be displayed.

4. Select the **Vacuum** or **Cluster** tab if the **Full** vacuum mode is selected.

The information about the table vacuum will be displayed.

### Renaming a Database #

The `ALTER DATABASE` SQL command is executed with the `RENAME TO` clause when renaming a database.

To rename a database, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the three vertical dots icon ⋮ → **Rename** next to the database that you want to rename.

   The database renaming window will open.

4. Specify the new database name.

5. Click **Save**.

The database will be renamed, and the update will be displayed in the table of databases.

Deleting a Database #

> **Warning**
>
> Deleted databases cannot be restored.

The `DROP DATABASE` SQL command is executed when deleting a database.

To delete a database, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the three vertical dots icon ⋮ → **Drop** next to the database that you want to delete.

   The database deletion window will open.

4. (Optional) To force delete the database, enable the **Forced deletion** checkbox. This checkbox is disabled by default. If it is enabled, all database connections will be terminated before deleting the database. The database will not be deleted if there are prepared transactions, logical replication slots, or subscriptions.

5. Click **Delete*.

6. Confirm the operation.

The database will be deleted and no longer be displayed in the table of databases.

> **Warning**

The `DROP DATABASE` SQL command is executed for deleting the database. After confirming the operation, the database and all its data will be permanently deleted. You cannot cancel or roll back this operation.

## Schemas #

This section explains how to manage schemas. It contains the following instructions:

- Creating a Schema
- Viewing Schemas
- Editing a Schema
- Deleting a Schema

### Creating a Schema #

To create a schema, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the database where you want to create a schema.

   The table of schemas will be displayed.

4. In the top-right corner of the page, click **Create schema**.

   The schema creation window will open.

5. Enter details of the new schema (parameters marked with an asterisk are required):

   - **Name**: The unique name of the schema.
   - **Owner**: The owner of the schema.
   - **lock_timeout, s**: Abort any statement that waits longer than the specified amount of time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt. For a more detailed description of this parameter, refer to the official Postgres Pro documentation.

6. Click **Create**.

The schema will be created and displayed in the table of schemas.

### Viewing Schemas #

To view database schemas, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the database where you want to view schemas.

The table of schemas with the following columns will be displayed:

- **Schemas**: The unique name of the schema. You can filter and sort values in this column using the corresponding icons.
- **Total size**: The total size of all schema objects. The horizontal indicator shows the size of tables, bloat, and indexes.

- **Owner**: The owner of the schema.

- **Tables**:

  - The number of tables
  - The size of tables
  - The table bloat percentage

- **Indexes**:

  - The number of indexes
  - The size of indexes
  - The index bloat percentage

- **Actions**: Actions with the schema. For more information about the actions that you can perform with a schema, refer to other instructions in this section.

You can perform the following actions with the table of schemas:

- To update the table, in the top-right corner of the page, click **Refresh data**.
- To reset all filters, in the top-right corner of the page, click **Reset**.

The size of the schema objects is calculated as the total sum of all its relational objects.

**Warning**

If a schema is created outside PPEM, the information about it can be displayed in the web application with a delay. The delay duration depends on the agent configuration. By default, it is one minute.

### Editing a Schema #

To edit a database schema, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the database where the schema is created.

   The table of schemas will be displayed.

4. Click **Edit** next to the schema that you want to edit.

   The schema editing window will open.

5. In the window, edit the required schema parameters.

6. Click **Save**.

The schema will be edited, and the updates will be displayed in the table of schemas.

## Deleting a Schema #

**Warning**

Deleted schemas cannot be restored.

To delete a schema, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the name of the database where the schema is created.

   The table of schemas will be displayed.

4. Click **Delete** 🗑 next to the schema that you want to delete.

   The schema deletion window will open.

5. (Optional) To delete all dependent objects of the schema, turn on the **Cascade deletion** toggle. This toggle is turned off by default.
6. Click **Delete*.
7. Confirm the operation.

The schema will be deleted and no longer be displayed in the table of schemas.

**Warning**

The `DROP SCHEMA` SQL command is executed when deleting a schema. After confirming the operation, the schema will be permanently deleted. You cannot cancel or roll back this operation.

### Tables #

This section explains how to manage tables. It contains the following instructions:

- Creating a Table
- Viewing Tables
- Viewing Information about a Specific Table
- Editing a Table
- Deleting a Table

#### Creating a Table #

To create a table, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where you want to create a table.

   The schema page with the selected **Tables** tab will be displayed.

5. In the top-right corner of the page, click **Create table**.

   The table creation window will open.

6. Enter details of the new table (parameters marked with an asterisk are required):
   - **Name**: The unique name of the table.
   - **Tablespace**: The tablespace where the table will be created.
   - **Columns**: Columns of the created table. To add a table column, follow the steps below:
     1. Click **Add +** and specify the column parameters (parameters marked with an asterisk are required):
        - **Name**: The unique name of the column.
        - **Type**: The type of the column data.
        - **NOT NULL**: Whether the column accepts `NULL` values. This toggle is turned off by default.
        - **Default value**: The default value of the column.
        - **Primary key**: Whether the table column(s) can contain only unique (non-repeating) values different from `NULL`. This toggle is turned off by default. To specify the primary key storage parameters, click **Add +**:
          - `fillfactor`
          - `deduplicate_items`
        - **Unique key**: Whether the group of one or multiple table columns can contain only unique values. To specify the unique key storage parameters, click **Add +**:
          - `fillfactor`
          - `deduplicate_items`
        - **Check expression**: The constraint of the column.
        - **Compression method**: The compression method for the column. Possible values:
          - **pglz**
          - **lz4**
     2. Click **Save**.
   - **Storage parameters**: Storage parameters of the table. To specify a parameter, click **Add +**, select the parameter from the drop-down list on the left, and then specify the required value in the field on the right.
   - **UNLOGGED**: Whether the created table is unlogged.

    - **lock_timeout, s**: Abort any statement that waits longer than the specified amount of time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt. For a more detailed description of this parameter, refer to the official Postgres Pro documentation.

7. Click **Save**.

The table will be created and displayed on the **Tables** tab.

Viewing Tables #

To view schema tables, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where you want to view tables.

The schema page with the selected **Tables** tab will be displayed. Information is displayed in the following table columns:

- **Tables**: The unique name of the table within the schema. You can filter and sort values in this column using the corresponding icons.

- **Total size**: The size of the table in bytes. The horizontal indicator shows the following:

    - The size of the table user data and bloating.
    - The total size of all table indexes, including the index bloat.
    - The size of the service TOAST storage without bloating.

  You can sort values in this column using the corresponding icon.

- **CFS**: Whether the CFS compression is used for the table. Applicable only for the Postgres Pro Enterprise edition.

- **Data**: The size of data in bytes, with the bloat percentage indicated separately.

- **Indexes**: The total number of indexes, size of indexes in bytes, and bloating percentage.

- **TOAST**: The size of the service TOAST storage without bloating.

- **Actions**: Actions with the table:

    - **Analyze**: Update the planner statistics. This action is available when you click the three vertical dots icon ⋮ .
    - **Reindex**: Reindex the table. This action is available when you click the three vertical dots icon ⋮ .
    - **Vacuum**: Vacuum obsolete versions of rows in the table. You can perform this action by clicking the three vertical dots icon ⋮ .
    - **Edit**: Edit the schema parameters.
    - **Delete**: Delete the schema.

You can perform the following actions with the table:

- To update the table, in the top-right corner of the page, click **Refresh data** ↻.
- To reset all filters, in the top-right corner of the page, click **Reset**.

  **Note**

  After completing the table maintenance operations, the size information is updated asynchronously.

One page displays 50 tables. To go to the next page, use the page navigation buttons in the bottom-right corner of the page.

Viewing Information about a Specific Table #

To display information about a specific table when viewing schema tables, on the **Tables** tab, click the name of this table.

The selected table page with the following tabs will be displayed:

- **Structure**: Information about the logical structure of the table and its fields. The table on the tab includes the following columns:

    - **Number**: The ordinal number of the column in the table.

    - **Name**: The unique name of the column.

    - **Type**: The type of the column data.

    - **Nullable**: Whether the column can contain `NULL` values.

    - **Default value**: The default column value.

    - **Description**: The description of the column.

    - **Actions**: Actions with the column:

        - **Edit**: Edit the column. The following parameters can be edited:

            - **Type**: The type of data.
            - **NOT NULL**: Whether the column can store `NULL` values.
            - **Default value**: The default column value.

        - **Delete**: Delete the column.

- **Indexes**: Information about the table indexes. The table on the tab includes the following columns:

    - **Indexes**: The unique name of the index. You can filter and sort values in this column using the corresponding icons.

    - **Total size**: The total size of the index. - **CFS**: Whether the CFS compression is used for the index.

    - **Status**: The status of the index. Possible values:

    - `valid`: The standard state in which the index can be used in queries.

    - `invalid`: The index cannot be used in queries. This status indicates that the index is being created or was damaged.

  You can filter values in this column using the corresponding icon.

    - **Actions**: Actions with the index:

        - ⓘ: View the index creation SQL command. It is displayed when hovering over the tooltip icon.
        - **Reindex**: Start reindexing.
        - **Edit**: Edit the index. The following parameters can be edited:
            - **Name**: The unique name of the index.
            - **Tablespace**: The tablespace where the index will be placed.
            - **lock_timeout, s**: Abort any statement that waits longer than the specified amount of time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt. For a more detailed description of this parameter, refer to the official Postgres Pro documentation.
        - **Delete**: Delete the index.

- **Constraints**: Constraints of the table. The table on the tab includes the following columns:
    - **Name**: The unique name of the restriction.
    - **Definition**: The definition of the constraint.

- **Storage**: The table storage structure on disk. The following layers are used for storing the main and service table data:
    - **Main data**: The layer for storing the main user data.
    - **Free space map**: The service layer for storing information about free space segments of the main storage layer.
    - **Visibility map**: The service layer for storing information about visible rows in the main storage layer.
    - **CFM**: The service layer for the CFS compression.
    - **TOAST**: The service layer for storing large values exceeding the standard page data storage restrictions.

    The table includes the following columns for each layer:

    - **Size**: The size of the layer file.
    - **Tablespace**: The tablespace where the layer is placed.
    - **File path**: The path to the layer file.

### Editing a Table #

To edit a table, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where the table is created.

   The schema page with the selected **Tables** tab will be displayed.

5. Click **Edit** ✎ next to the table that you want to edit.

   The table editing window will open.

6. In the window, edit the required table parameters. Available storage parameters are listed in the Table Storage Parameters section.

7. Click **Save**.

The table will be edited, and the updates will be displayed on the **Tables** tab.

### Deleting a Table #

**Warning**

Deleted tables cannot be restored.

To delete a table, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where the table is created.

   The schema page with the selected **Tables** tab will be displayed.

5. Click **Delete** 🗑 next to the table that you want to delete.

   The table deletion window will open.

6. Click **Delete*.

The table will be deleted and no longer be displayed on the **Tables** tab.

### Table Storage Parameters #

You can specify the following storage parameters when creating or editing tables:

- `fillfactor` (integer)
- `toast_tuple_target` (integer)
- `parallel_workers` (integer)
- `autovacuum_enabled` (boolean)
- `toast.autovacuum_enabled` (boolean)
- `vacuum_index_cleanup` (enum)
- `toast.vacuum_index_cleanup` (enum)
- `vacuum_truncate` (boolean)
- `toast.vacuum_truncate` (boolean)
- `autovacuum_vacuum_threshold` (integer)
- `toast.autovacuum_vacuum_threshold` (integer)
- `autovacuum_vacuum_scale_factor` (floating point)
- `toast.autovacuum_vacuum_scale_factor` (floating point)
- `autovacuum_analyze_threshold` (integer)
- `autovacuum_analyze_scale_factor` (floating point)
- `autovacuum_vacuum_cost_delay` (floating point)

- `toast.autovacuum_vacuum_cost_delay` (floating point)
- `autovacuum_vacuum_cost_limit` (integer)
- `toast.autovacuum_vacuum_cost_limit` (integer)
- `autovacuum_freeze_min_age` (integer)
- `toast.autovacuum_freeze_min_age` (integer)
- `autovacuum_freeze_max_age` (integer)
- `toast.autovacuum_freeze_max_age` (integer)
- `autovacuum_freeze_table_age` (integer)
- `toast.autovacuum_freeze_table_age` (integer)
- `autovacuum_multixact_freeze_min_age` (integer)
- `toast.autovacuum_multixact_freeze_min_age` (integer)
- `autovacuum_multixact_freeze_max_age` (integer)
- `toast.autovacuum_multixact_freeze_max_age` (integer)
- `autovacuum_multixact_freeze_table_age` (integer)
- `toast.autovacuum_multixact_freeze_table_age` (integer)
- `log_autovacuum_min_duration` (integer)
- `toast.log_autovacuum_min_duration` (integer)
- `user_catalog_table` (boolean)

## Indexes #

This section explains how to manage indexes. It contains the following instructions:

- Creating an Index
- Viewing Indexes
- Editing an Index
- Reindexing
- Deleting an Index

### Creating an Index #

To create an index, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the required schema.

   The schema page with the selected **Tables** tab will be displayed.

5. Click the name of the table where you want to create an index.

   The table page with the selected **Structure** tab will be displayed.

6. In the top-right corner of the page, click **Create index**.

   The index creation window will open.

7. Enter details of the new index (parameters marked with an asterisk are required):

   - **Name**: The unique name of the index.

   - **Concurrently building**: Whether the index will be created in a special mode that minimizes the number of locks and reduces the risk of their occurrence when there are concurrent workloads.

   - **Unique index**: Whether to control duplicate table values when creating the index and adding new values to it.

   - **Columns**: Columns that will be included in the index. Available actions:

     - To add a column, click **Add column +**, and then select the column from the list.
     - To delete a column, click the bin icon 🗑.

8. (Optional) In the top-right corner of the window, turn on the **Extended settings** toggle to specify additional parameters:

   - **Tablespace**: The tablespace where the index will be placed.

   - **Using method**: The access method. Possible values:

   - `btree` (this value is selected by default)

   - `gin`

   - `gist`

   - `brin`

   - `hash`

   - **Where predicate**: The condition of the index.

   - **lock_timeout, s**: Abort any statement that waits longer than the specified amount of time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt. For a more detailed description of this parameter, refer to the official Postgres Pro documentation.

   - **Columns**:

     - **Column**: The column from the list.

     - **Operator class**

     - **Collation**

     - **Order**: The sorting order. Possible values:

       - **Default**
       - **Ascending**
       - **Descending**

     - **NULL's order**: The sorting order of `NULL` values. Possible values:

       - **Default**

        - **NULL first**
        - **NULL last**

9. Click **Create**.

The index will be created and displayed in the table of indexes on the **Indexes** tab of the corresponding table or schema page where the table is created.

Viewing Indexes #

To view indexes, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the required schema.

   The schema page with the selected **Tables** tab will be displayed.

5. Click the name of the table where you want to view indexes.

   The table page with the selected **Structure** tab will be displayed.

6. Select the **Indexes** tab.

The table of indexes with the following columns will be displayed:

- **Indexes**: The unique name of the index. You can filter and sort values in this column using the corresponding icons.
- **Total size**: The total size of the index.
- **CFS**: Whether the CFS compression is used for the index.
- **Status**: The status of the index. Possible values:
    - `valid`: The standard state in which the index can be used in queries.
    - `invalid`: The index cannot be used in queries. This status indicates that the index is being created or was damaged.

  You can filter values in this column using the corresponding icon.

- **Actions**: Actions with the index. For more information about the actions that you can perform with an index, refer to other instructions in this section.

  The exclamation sign icon ⓘ is displayed next to this column. Hovering over this icon displays the SQL query for creating the index.

You can perform the following actions with the table of indexes:

- To update the table, in the top-right corner of the page, click **Refresh data** ⟳.
- To reset all filters, in the top-right corner of the page, click **Reset**.

  **Warning**

  If an index is created outside PPEM, the information about it can be displayed in the web application with a delay. The delay duration depends on the agent configuration. By default, it is one minute.

## Editing an Index #

To edit an index, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the required schema.

   The schema page with the selected **Tables** tab will be displayed.

5. Click the name of the table where the index is created.

   The table page with the selected **Structure** tab will be displayed.

6. Select the **Indexes** tab.

   The table of indexes will be displayed.

7. Click **Edit** ✎ next to the index that you want to edit.

   The index editing window will open.

8. In the window, edit the required index parameters.
9. Click **Save**.

The index will be edited, and the updates will be displayed in the table of indexes.

Reindexing #

To reindex, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the required schema.

   The schema page with the selected **Tables** tab will be displayed.

5. Click the name of the table where the index is created.

   The table page with the selected **Structure** tab will be displayed.

6. Select the **Indexes** tab.

   The table of indexes will be displayed.

7. Click **Reindex** next to the index that you want to reindex.

   The confirmation window with the following parameters will open:

   - **Concurrently**: Whether the operation will be performed in the non-locking mode. This is the safest mode that allows avoiding locking other sessions by slowing down the process of reindexing. This mode is selected by default. Possible values:

     - **On**
     - **Off**

   - **Tablespace**: The tablespace where the index will be reindexed.

8. Click **Execute**.

The reindexing task will start.

> **Note**
>
> The `REINDEX` SQL command is executed for reindexing. Significant resources can be required for performing this operation so it is recommended to track it.

To track reindexing, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The **Activity** page will be displayed.

4. Select the **Indexes** tab.

Reindexing information will be displayed.

### Deleting an Index #

> **Warning**
>
> Deleted indexes cannot be restored.

To delete an index, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the required schema.

   The schema page with the selected **Tables** tab will be displayed.

5. Click the name of the table where the index is created.

   The table page with the selected **Structure** tab will be displayed.

6. Select the **Indexes** tab.

   The table of indexes will be displayed.

7. Click **Delete** icon-delete next to the index that you want to delete.

   The index deletion window will open.

8. Click *Delete*.

The index will be deleted and no longer be displayed in the table of indexes.

## Functions #

This section explains how to manage functions. It contains the following instructions:

- Creating a Function
- Viewing Functions
- Editing a Function
- Deleting a Function

### Creating a Function #

To create a function, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where you want to create a function.

   The schema page with the selected **Tables** tab will be displayed.

5. Select the **Functions** tab.

   The table of functions will be displayed.

6. In the top-right corner of the page, click **Create function**.

   The function creation window will open.

7. Enter details of the new function (parameters marked with an asterisk are required):

   - **Name**: The unique name of the function.
   - **Language**: The procedural language of the function.
   - **Arguments**: Function arguments. Possible values:
     - **Name**: The unique name of the argument.
     - **Argument mode**: The argument mode. Possible values:
       - **IN**, this value is selected by default
       - **OUT**
       - **INOUT**
       - **VARIADIC**
     - **Type**: The argument data type.
     - **Default value**: The default value of the argument.

     Click **ADD ARGUMENT +** to add an argument.
   - **Function return table**: Whether the function returns a table. If this toggle is turned on, specify the parameters of the returned table:
     - **Name**: The name of the returned value.
     - **Type**: The type of the returned value.

     Click **Add return value +** to add a return value.
   - **Return value type**: The data type of the value that the function will return. This parameter cannot be edited if the **Function return table** toggle is turned on.
   - **Function body**: The body of the function in the selected procedural language.
   - **Window**: Whether the function is a window function.
   - **Optimizer**: Specifies the attribute that informs the optimizer about the behavior of the function. Possible values:
     - **Default**: Default value — `Volatile`.
     - **Immutable**: Whether the function is immutable — it cannot modify the database and always returns the same result for specific arguments. It means that the function does not interact with the database and does not use the information that was not passed to the argument list. If a function is immutable, any function call with constant arguments can be immediately replaced with the function value.
     - **Stable**: Whether the function is stable — it cannot modify the database and always returns the same result for specific arguments within one table scan, but the result can differ for different SQL operators. This is applicable to functions whose results depend on the database contents and parameters, such as the time zone. However, this is not applicable to `AFTER` triggers that are trying to read the rows changed by the current command. Note that the `current_timestamp` functions are also considered stable, since their results do not change in the transaction.
     - **Volatile**: Whether the function is volatile — its result can change even within one table scan, so the function calls cannot be optimized. Only a limited number of functions are volatile, for example, `random()`, `currval()`, and `timeofday()`. Note that any function that has side effects must be considered volatile even if its results are predictable so that its calls are not optimized. An example of such a function is `setval()`.
   - **Strict**: Whether the function will always return `NULL` if `NULL` is passed in one of the arguments.
   - **Leakproof**: Whether the function does not have any side effects — it does not reveal any information about its arguments and only returns the result.
   - **Security**: Specifies the privileges that will be used for calling and executing the function. Possible values:
     - **Default**: Default value — `Invoker`.
     - **Invoker**: Whether the function will be executed with the privileges of the user who called it.
     - **Definer**: Whether the function will be executed with the privileges of the user who called it.
   - **Parallel**: Specifies parameters for calling the function in the parallel mode. Possible values:
     - **Default**: Default value — `Unsafe`.
     - **Unsafe**: Specifies that the function cannot be executed in the parallel mode and that having such a function in an SQL operator will lead to selecting a sequential query plan.
     - **Restricted**: Whether the function can be executed in the parallel mode, but only in the parallel group leader process.
     - **Safe**: Whether the function can be safely executed in the parallel mode with no restrictions, including parallel worker processes.
   - **lock_timeout, s**: Abort any statement that waits longer than the specified amount of time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt. For a more detailed description of this parameter, refer to the official Postgres Pro documentation.
   - **Show SQL**: Displays the SQL query for creating the function with the specified parameters.

8. Click **Create**.

The function will be created and displayed in the table of functions.

#### Viewing Functions #

To view functions, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.

3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where you want to view functions.

   The schema page with the selected **Tables** tab will be displayed.

5. Select the **Functions** tab.

The table of functions with the following columns will be displayed:

- **Name**: The unique name of the function or procedure. You can filter and sort values in this column using the corresponding icons.
- **Arguments**: Arguments with their data types passed to the function.
- **Return values**: Values with their data types returned by the function.
- **Access control list**: Function access privileges.
- **Actions**: Actions with the function. For more information about the actions that you can perform with a function, refer to other instructions in this section.

You can perform the following actions with the table of functions:

- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-left corner of the table, click **Reset filters**.

#### Editing a Function #

To edit a function, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where the function is created.

   The schema page with the selected **Tables** tab will be displayed.

5. Select the **Functions** tab.

   The table of functions will be displayed.

6. Click **Edit** next to the function that you want to edit.

   The function editing window will open.

7. Edit the required function parameters.
8. Click **Save**.

The function will be edited, and the updates will be displayed in the table of functions.

#### Deleting a Function #

> **Warning**
>
> Deleted functions cannot be restored.

To delete a function, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. (Optional) Turn on the **Show system databases** toggle to also display system databases. This toggle is turned off by default.
3. Click the name of the required database.

   The table of schemas will be displayed.

4. Click the name of the schema where the function is created.

   The schema page with the selected **Tables** tab will be displayed.

5. Select the **Functions** tab.

   The table of functions will be displayed.

6. Cluck **Delete** next to the function that you want to delete.

   The function deletion window will open.

7. Click **Delete*.

The function will be deleted and no longer be displayed in the table of functions.

### Sequences #

This section explains how to manage sequences. It contains the following instructions:

- Creating a Sequence
- Viewing Sequences
- Editing a Sequence
- Deleting a Sequence

#### Creating a Sequence #

To create a sequence, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.

2. Click the name of the required database.

   The table of schemas will be displayed.

3. Click the name of the schema where you want to create a sequence.

   The schema page with the selected **Tables** tab will be displayed.

4. Select the **Sequences** tab.

   The table of sequences will be displayed.

5. In the top-right corner of the page, click **Create sequence**.

The sequence creation window will open.

6. Enter details of the new sequence (parameters marked with an asterisk are required):

    1. In the **Name** field, enter the unique name of the sequence.

    2. (Optional) In the top-right corner of the window, turn on the **Extended settings** toggle to specify additional parameters:

        - **No logging**: Whether the changes of the created sequence will not be saved to WAL. This toggle is turned off by default.

        - **Cycle**: Whether the created sequence is cycled. This toggle is turned off by default.

        - **Data type**: The type of the sequence data. Possible values:

            - **Default** (**Bigint**)
            - **Smallint**
            - **Integer**
            - **Bigint**

        - **Min. value**: The minimum value that will be generated by the sequence.

        - **Max. value**: The maximum value that will be generated by the sequence.

        - **Start**: The start value of the sequence.

        - **Step**: The number that will be added to the current sequence value to receive a new value.

        - **Cache**: The number of sequence values that will be allocated and stored in memory for faster access.

        - **lock_timeout, s**: Abort any statement that waits longer than the specified amount of time while attempting to acquire a lock on a table, index, row, or other database object. The time limit applies separately to each lock acquisition attempt. For a more detailed description of this parameter, refer to the official Postgres Pro documentation.

7. Click **Create**.

The sequence will be created and displayed in the table of sequences.

#### Viewing Sequences #

To view sequences, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

    The table of databases will be displayed.

2. Click the name of the required database.

    The table of schemas will be displayed.

3. Click the name of the schema where you want to view sequences.

    The schema page with the selected **Tables** tab will be displayed.

4. Select the **Sequences** tab.

The table of sequences with the following columns will be displayed:

- **Name**: The unique name of the sequence. You can filter and sort values in this column using the corresponding icons.
- **Owner**: The owner of the sequence. You can filter and sort values in this column using the corresponding icons.
- **Data type**: The data type of the sequence values.
- **Last_value**: The last value of the sequence. You can sort values in this column using the corresponding icon.
- **Min. value** and **Max. value**: The minimum and maximum possible sequence values. You can sort values in this column using the corresponding icon.
- **Step**: The step for changing the sequence values. You can sort values in this column using the corresponding icon.
- **Cycle**: Whether the sequence is cycled and restarts upon reaching its limit.
- **Cache**: The number of sequence values that will be allocated and stored in memory for faster access. You can sort values in this column using the corresponding icon.
- **Last_value**: The last value of the sequence. You can sort values in this column using the corresponding icon.
- **Actions**: Actions with the sequence. For more information about the actions that you can perform with a sequence, refer to other instructions in this section.

You can perform the following actions with the table of sequences:

- To update the table, in the top-right corner of the page, click **Refresh data** 🔄.
- To reset all filters, in the top-right corner of the page, click **Reset**.

#### Editing a Sequence #

To edit the sequence, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

    The table of databases will be displayed.

2. Click the name of the required database.

    The table of schemas will be displayed.

3. Click the name of the schema where the sequence is created.

    The schema page with the selected **Tables** tab will be displayed.

4. Select the **Sequences** tab.

    The table of sequences will be displayed.

5. Click **Edit** ✏ next to the sequence that you want to edit.

    The sequence editing window will open.

6. In the window, edit the required sequence parameters.

7. Click **Save**.

The sequence will be edited, and the updates will be displayed in the table of sequences.

#### Deleting a Sequence #

> **Warning**
>
> Deleted sequences cannot be restored.

To delete a sequence, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Databases** page.

   The table of databases will be displayed.
2. Click the name of the required database.

   The table of schemas will be displayed.
3. Click the name of the schema where the sequence is created.

   The schema page with the selected **Tables** tab will be displayed.
4. Select the **Sequences** tab.

   The table of sequences will be displayed.
5. Click **Delete** next to the sequence that you want to delete.

   The sequence deletion window will open.
6. (Optional) To delete all sequence objects, turn on the **Cascade deletion** toggle. This toggle is turned off by default.
7. Click *Delete.

The sequence will be deleted and no longer be displayed in the table of sequences.

## Control Center #

PPEM supports the **control center** that you can use for monitoring the state of instances and troubleshooting them if required.

To view the state of instances using control center, in the **Management** section of the navigation panel, go to the **Monitoring** → **Operation control center** page.

The control center with the following elements of the web application interface will be displayed:

- List of all instances (the left block)
- Instance tiles (upper block)
- Instance performance indicators (lower block)

### List of All Instances #

The list of installed instances is displayed in the left block. The color of the dot located next to each instance name indicates the instance status:

- Green ( ● ): The instance is operating normally.
- Red ( ● ): An error occurred during instance operation.
- Gray ( ● ): Failed to determine the instance state.

You can find the required instance by entering its name in the search field at the top of the block and sort instances using the corresponding icon.

### Instance Tiles #

Installed instances are displayed as tiles in the upper block. Each tile displays the instance uptime, name, and server where it is installed.

The tile color indicates the instance status. The correlation between the instance status and tile color is the same as for the dots in the list of all instances. You can click one of the following buttons to filter instances by status:

- **All**
- **Ok**
- **Error**
- **Not defined**

You can perform the following actions using an instance tile:

- Go to instance management by clicking the three vertical dots icon ⋮ → **Go to instance**.
- Start the PSQL terminal of an instance by clicking the three vertical dots icon ⋮ → **Open PSQL terminal**.
- Start a stopped instance by clicking the three vertical dots icon ⋮ → **Start instance**.

You can configure the update rate of instance information. To do this, in the top-right corner of the block, select one of the following values from the drop-down list:

- **1 minute**. This value is selected by default.
- **5 minutes**.
- **15 minutes**.
- **30 minutes**.
- **1 hour**.
- **2 hours**.
- **4 hours**.
- **Disable**.

To manually update instance information, click **Refresh data** .

### Instance Performance Indicators #

The lower block allows monitoring key instance performance indicators using graphs based on metrics collected by `pgpro-otel-collector`.

For the graphs to work properly, you must first install and configure logging and monitoring tools .

To view a graph, in the upper block click the instance tile, and then in the top-right corner of the lower block, select the graph from the drop-down list. For a description of the graphs, refer to the Metrics section.

You can display a graph for a specific time period. To do this, in the top-right corner of the block, select one of the following values from the drop-down list:

- **Last 5 minutes**.
- **Last 15 minutes**. This value is selected by default.
- **Last hour**.
- **Last 3 hours**.
- **Last 6 hours**.
- **Last 12 hours**.
- **Last 24 hours**.
- **Select period**. For this value, specify the start and end date and time in the opened window, and then click **Apply**.

You can perform the following actions using the icons in the top-right corner of graphs:

- To display a specific time period covered by a graph, click **Select interval**, and then select the time period on the graph.
- To reset the time period, click **Reset**.
- To download a graph in the `PNG` or `CSV` format, click **Download .png** or **Download .csv**.

## Metrics #

Key performance indicators of the system can be monitored using graphs based on metrics collected by `pgpro-otel-collector`.

For the graphs to work properly, you must first install and configure logging and monitoring tools .

Also, you can access SQL metrics that are provided based on planning statistics and SQL statement execution statistics collected by the `pgpro_stats` extension.

> **Warning**
>
> When creating SQL metrics use queries with aggregate functions, such as `COUNT`, `SUM`, and `AVG`. The instance can fail when using metrics based on queries returning more than one row or separate values, for example, `SELECT 1`.

### Main Metrics #

#### Viewing Main Metrics #

To view main metrics, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to view metrics.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Main metrics** page.

Available graphs will be displayed. You can filter graphs for main metrics by time using the **Period** drop-down list in the top-right corner of the page.

#### Available Graphs #

To hide or show graphs, click **Graphs** and select the required ones. The following graphs are available:

- **WAL Archiver**

  Number of archived WAL segments.

- **Vacuum workers**

  The number of vacuum operations.

- **Background Writes: Buffers**

  The data volume written on the background from shared cache to disk.

- **Background Writes: Maxwritten/Fsync**

  This graph shows the dynamics of two metrics:

  - **maxwritten**: The number of times the background writer had to stop writing because the limit was reached.
  - **fsync**: The number of forced fsync calls.

- **Background Writes: Checkpoints**

  The number of checkpoints.

- **Background Writes: Checkpoints Write/Sync**

  The time spent writing and synchronizing blocks during execution of checkpoints.

- **Instance: Connections**

  Connections established with the DBMS instance and their state.

- **Instance: Blocks rate**

  Usage of the DBMS instance shared cache: the number of cache hits and misses that resulted in the need to read data from disk.

- **Instance: Transactions rate**

  Transactional activity in a DBMS instance.

- **Instance: Events**

  Events in the DBMS instance: deadlocks, replication conflicts, checksum validation errors.

- **Instance: Tuples**

  The workload expressed in row processing: the number of rows read, inserted, updated, and deleted.

- **Instance: Cache hit ratio**

  The shared cache workload: reflects the proportion of cache hits relative to all shared cache accesses.

- **Instance: Temp bytes written**

  The amount of data written to temporary files.

- **Instance: Temp files**

  The number of temporary files written by the DBMS instance.

- **Instance: Locks**

  Lock dynamics.

- **WAL: Written bytes**

  Bytes written to WAL.

- **System: Load Average**

  The average server load.

- **System: Memory Usage**

  Usage of server memory.

- **System: Swap Usage**

Usage of swap space in the system.

- **System: Processes**

  The state of processes in the system.

## SQL Metrics [#](#)

### Creating an SQL Metric [#](#)

To create an SQL metric, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to create an SQL metric.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Main metrics** → **SQL metrics** page.

   The table of SQL metrics will be displayed.

4. In the top-right corner of the page, click **Add SQL metric**.

   The SQL metric creation window will open.

5. Enter details of the new SQL metric (parameters marked with an asterisk are required):

   - **Name**: The unique name of the metric.
   - **Database**: The database where the query will be executed.
   - **User**: The system role on behalf of which the query will be executed.
   - **Collection interval**: The frequency of data collection.
   - **Query**: The SQL query based on which the metric is computed.

     **Warning**

     Use queries with aggregate functions, such as `COUNT`, `SUM`, and `AVG`. The instance can fail when using metrics based on queries returning more than one row or separate values, for example, `SELECT 1`.

   - **Instance restart is required**: Whether to restart the instance. This checkbox is disabled by default.

6. Click **Add**.

The SQL metric will be created and displayed in the table of SQL metrics.

### Viewing SQL Metrics [#](#)

To view SQL metrics, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to view SQL metrics.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Main metrics** → **SQL metrics** page.

   The table of SQL metrics with the following columns will be displayed:

   - **ID**: The unique metric ID.
   - **Name**: The unique name of the metric.
   - **Database**: The database where the query is executed.
   - **User**: The system role on behalf of which the query will be executed.
   - **Interval**: The frequency of data collection.
   - **Last value**: The result of the last query in the JSON format.
   - **Updated at**: The time the data was last updated.
   - **Query**: The SQL query based on which the metric is computed.
   - **Actions**: Actions with the SQL metric. For more information about the actions that you can perform with SQL metrics, refer to other instructions in this section.

   To update the table, in the top-right corner of the page, click **Refresh data** .

4. To view the history of collected data, click on the metric name.

   The table with historical values will be displayed with data collection time specified for each value.

### Editing an SQL Metric [#](#)

To edit an SQL metric, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where the SQL metric is created.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Main metrics** → **SQL metrics** page.

   The table of SQL metrics will be displayed.

4. Click **Edit** next to the SQL metric that you want to edit.

   The SQL metric editing window will open.

5. Change the SQL metric parameters.

6. Click **Save**.

The SQL metric will be edited, and the updates will appear in the table of SQL metrics.

### Deleting an SQL Metric [#](#)

**Warning**

Deleted SQL metrics cannot be restored.

To delete an SQL metric, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where the SQL metric is created.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Main metrics** → **SQL metrics** page.

   The table of SQL metrics will be displayed.

4. Click **Delete** 🗑 next to the SQL metric that you want to delete.

   The SQL metric deletion window will open.

5. Click **Delete*.

The SQL metric will be deleted and no longer be displayed in the table of SQL metrics.

## Activity #

PPEM supports monitoring of the following instance activities:

- User sessions
- Vacuum processes
- Statistics collection processes
- Clustering processes
- Reindexing or index creation processes
- Base backup processes
- Copying processes

### User Sessions #

You can view information about user sessions and background worker processes of an instance. The information is based on the `pg_stat_activity` view.

This section explains how to manage user sessions. It includes the following instructions:

- Viewing User Sessions
- Viewing Statistics on Wait Events
- Viewing the Current Backend Query Locks
- Viewing the Current Backend Query Plan
- Canceling the Current Backend Query
- Terminating a User Session

#### Viewing User Sessions #

To view user sessions, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to view user sessions.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

The table of user sessions with the following columns will be displayed:

- **pid**: The backend ID. You can sort values in this column using the corresponding icon.
- **leader_pid**: The backend ID for the process in the parallel query execution group. You can sort values in this column using the corresponding icon.
- **State**: The type of the backend. For more information about possible values, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `backend_type` column).

  You can sort values in this column using the corresponding icon.
- **backend_start**: The backend start date and time. You can sort values in this column using the corresponding icon.
- **client_hostname**, **client_addr** and **client_port**: The network name, address, and port number of the client that initiated the user session. You can sort values in these columns using the corresponding icon.
- **usesysid**: The ID of the DBMS user under which the session is initiated. You can sort values in this column using the corresponding icon.
- **username**: The name of the DBMS user under which the session is initiated. You can sort values in this column using the corresponding icon.
- **datid**: The ID of the database associated with the user session. You can sort values in this column using the corresponding icon.
- **database**: The name of the database associated with the user session. You can filter and sort values in this column using the corresponding icons.
- **application_name**: The name of the source application of the user session. You can sort values in this column using the corresponding icon.
- **State**: The state of the backend. For more information about possible values, refer to the official Postgres Pro documentation on the pg_stat_activity view (see the `state` column).

  You can filter and sort values in this column using the corresponding icons.
- **wait_event_type**: The type of the wait event for the backend. For more information, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `wait_event_type` and `wait_event` columns). You can sort values in this column using the corresponding icon.
- **wait_event**: The name of the wait event for the backend. You can sort values in this column using the corresponding icon.
- **transaction_duration_seconds**: The duration of the current backend transaction, in seconds. You can sort values in this column using the corresponding icon.
- **xact_start**: The start date and time of the current backend transaction. You can sort values in this column using the corresponding icon.
- **query_duration_seconds**: The duration of the current backend query in seconds. You can sort values in this column using the corresponding icon.
- **query_start**: The start date and time of the current backend query. You can sort values in this column using the corresponding icon.

- **state_change**: The date and time of the last backend state update (see the **state** column). You can sort values in this column using the corresponding icon.
- **backend_xid**: The ID of the backend top-level transaction.
- **backend_xmin**: The current `xmin` bound of the backend.
- **query_id**: The ID of the current or last backend query. You can sort values in this column using the corresponding icon.
- **query**: The text of the current or last backend query. You can copy the query text. To do this, click the exclamation mark icon next to the query text, and then in the opened window, click **Copy** .

  You can sort values in this column using the corresponding icon.
- **Actions**: Actions with the user session. For more information about the actions that you can perform with a user session, refer to other instructions in this section.

You can perform the following actions with the table of user sessions:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To disable or enable the automatic update of the table, in the top-right corner of the page, click the start icon or the stop icon .
- To update the table, in the top-right corner of the page, click **Refresh data** . This action is available only if you disabled the automatic update of the table.
- To reset all filters, in the top-left corner of the table, click **Reset all**.

## Viewing Statistics on Wait Events #

PPEM supports integration with the `pg_wait_sampling` extension for viewing the wait history and profile in user sessions.

Before performing this procedure, install the `pg_wait_sampling` extension.

To view statistics on wait events, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.
4. In the **pid** column, click the backend ID of the user session where you want to view statistics on wait events.

The **Wait events profile** page with the following web application interface elements will be displayed:

- The **Session** block with the following parameters of the selected user session:
  - **Duration**: The duration of the user session.
  - **Active transaction**: The duration of the current transaction in the user session.
  - **Active request**: The duration of the current query in the user session.
  - **Total time spent waiting**: The total duration of wait events in the user session.
  - **Observations**: The number of sampled wait events in the user session.
  - **Unique events**: The number of unique wait events sampled in the user session.
- The **Sampling settings** block with the following wait profile configuration parameters:
  - **profile_period** (corresponds to `pg_wait_sampling.profile_period`)
  - **profile_pid** (corresponds to `pg_wait_sampling.profile_pid`)
  - **profile_queries** (corresponds to `pg_wait_sampling.profile_queries`)
  - **sample_cpu** (corresponds to `pg_wait_sampling.sample_cpu`)

  For the description of the configuration parameters, refer to the official Postgres Pro documentation. If required, you can change the values of these parameters.
- The graph with statistics on wait events. You can display statistics for the last 5, 10, and 20 wait events using the corresponding buttons at the top of the graph. To scale up or down, use the slider at the bottom of the graph.
- The **History** block with the following wait history parameters:
  - **Duration**: The duration of the wait history.
  - **Start** and **End**: The start and end date and time of the wait history.
- The **History settings** block with the following wait history configuration parameters:
  - **history_size** (corresponds to `pg_wait_sampling.history_size`)
  - **history_period** (corresponds to `pg_wait_sampling.history_period`)

  For the description of the configuration parameters, refer to the official Postgres Pro documentation. If required, you can change the values of these parameters.
- The table of events with the following columns:
  - **Time**: The date and time of the event. You can filter and sort values in this column using the corresponding icons.
  - **Standby event type**: The type of the wait event associated with the event. For more information about possible values, refer to the official Postgres Pro documentation. You can filter values in this column using the corresponding icon.
  - **Standby event**: The wait event associated with the event. You can filter values in this column using the corresponding icon.
  - **Query id**: The unique ID of the query associated with the event. You can filter values in this column using the corresponding icon.
  - **Query**: The text of the query associated with the event. To view the query text separately, click **Query** next to it. You can copy the query text by clicking **Copy** in the opened window.

  You can perform the following actions with the table of events:
  - To change the column width, drag its border with the mouse.
  - To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
  - To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.

## Viewing the Current Backend Query Locks #

To view the current backend query locks, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.
4. Click **Open lock tree** next to the user session for which you want to view the current backend query locks.

The window with the current backend query lock tree will open.

#### Viewing the Current Backend Query Plan #

You can view the current backend query plan if the `pg_query_state` module is installed.

To view the current backend query plan, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.
4. Click **Active query plan** next to the user session for which you want to view the current backend query plan.

The window with the query plan will open. If required, you can view a query plan visualization.

To update the query plan, click **Update execution statistics**.

#### Canceling the Current Backend Query #

Canceling a current backend query does not terminate a user session.

To cancel the current backend query, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.
4. Click **Cancel current request** next to the user session for which you want to cancel the current backend query.

   The confirmation window will open.
5. Click **Execute**.

#### Terminating a User Session #

When the user session is terminated, the current backend query is canceled automatically.

To terminate a user session, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.
4. Click **Cancel session** next to the user session that you want to terminate.

   The confirmation window will open.
5. Click **Execute**.

### Vacuum Processes #

You can view information about vacuum (`VACUUM`) and autovacuum (`autovacuum`) processes. The information is based on the `pg_stat_progress_vacuum` view.

This section explains how to manage vacuum processes. It includes the following instructions:

- Viewing Vacuum Processes
- Canceling a Vacuum Process
- Terminating the User Session for a Vacuum Process

#### Viewing Vacuum Processes #

To view vacuum processes, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

The table of user sessions will be displayed.

4. Select the **Vacuum** tab, and then in the top-right corner of the page, select the database where you want to view vacuum processes.

The table of vacuum processes with the following columns will be displayed:

- **PID**: The ID of the backend. You can sort values in this column using the corresponding icon.
- **State**: The state of the backend. For more information about possible values, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `state` column).

  You can filter and sort values in this column using the corresponding icons.
- **Wait event**: The name and type of the wait event for the backend. For more information, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `wait_event_type` and `wait_event` columns). You can sort values in this column using the corresponding icon.
- **Phase**: The vacuum phase. For more information about possible values, refer to the official Postgres Pro documentation.
- **Database**: The name of the database being vacuumed. You can filter and sort values in this column using the corresponding icons. **User**: The name of the DBMS user on behalf of which vacuum is being performed. You can sort values in this column using the corresponding icon.
- **Table**: The name or ID of the table being vacuumed.
- **Query**: The text of the current or last backend query. You can copy the query text. To do this, click the exclamation mark icon next to the query text, and then in the window that opens, click **Copy**.

  You can sort values in this column using the corresponding icon.
- **Query duration**: The duration of the current or last backend query. You can sort values in this column using the corresponding icon.
- **Heap size**: The size of the table for which vacuum is being performed.
- **Total size**: The total size of the table, including indexes, for which vacuum is being performed.
- **Scanned, %**: The percentage of data scanned in the table being vacuumed. This column includes additional information:
  - **Size**: The size of scanned table data.
- **Vacuumed, %**: The percentage of vacuumed table data. This column includes additional information:
  - **Size**: The size of vacuumed table data.

**Index vacuum**: The number of vacuumed table indexes.

- **Memory usage, %**: The usage of the memory that stores pointers to obsolete versions of table rows. For more information, refer to the official Postgres Pro documentation on the `autovacuum_work_mem` and `maintenance_work_mem` parameters.
- **Actions**: Actions with the vacuum process. For more information about the actions that you can perform with a vacuum process, refer to other instructions in this section.

You can perform the following actions with the table of vacuum processes:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To disable or enable the automatic update of the table, in the top-right corner of the page, click the start icon or the stop icon.
- To update the table, in the top-right corner of the page, click **Refresh data**. This action is available only if you disabled the automatic update of the table.
- To reset all filters, in the top-left corner of the table, click **Reset all**.

## Canceling a Vacuum Process #

Canceling a vacuum process does not terminate a user session.

To cancel a vacuum process, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.
4. Select the **Vacuum** tab, and then in the top-right corner of the page, select the database where the vacuum process is running.
5. Click **Cancel current request** next to the vacuum process that you want to cancel.

   The confirmation window will open.
6. Click **Execute**.

## Terminating the User Session for a Vacuum Process #

When the user session is terminated, the vacuum process is canceled automatically.

To terminate the user session for a vacuum process, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.
4. Select the **Vacuum** tab, and then in the top-right corner of the page, select the database where the vacuum process is running.
5. Click **Cancel session** next to the vacuum process for which you want to terminate the user session.

   The confirmation window will open.

6. Click **Execute**.

## Clustering Processes #

You can view information about clustering processes — reindexing tables (`CLUSTER`) and about full vacuum processes (`VACUUM FULL`). The information is based on the `pg_stat_progress_cluster` view.

This section explains how to manage clustering processes. It includes the following instructions:

- Viewing Clustering Processes
- Canceling a Clustering Process
- Terminating the User Session for a Clustering Process

### Viewing Clustering Processes #

To view clustering processes, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.

4. Select the **Cluster** tab, and then in the top-right corner of the page, select the database where you want to view clustering processes.

The table of clustering processes with the following columns will be displayed:

- **PID**: The ID of the backend. You can sort values in this column using the corresponding icon.
- **State**: The state of the backend. For more information about possible values, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `state` column).

  You can filter and sort values in this column using the corresponding icons.
- **Wait event**: The name and type of the wait event for the backend. For more information, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `wait_event_type` and `wait_event` columns). You can sort values in this column using the corresponding icon.
- **Phase**: The clustering phase. For more information about possible values, refer to the official Postgres Pro documentation.
- **Database**: The name of the database where clustering is being performed. You can filter and sort values in this column using the corresponding icons.
- **User**: The name of the DBMS user on behalf of which clustering is being performed. You can sort values in this column using the corresponding icon.
- **Table**: The name or ID of the table for which clustering is being performed.
- **Query**: The text of the current or last backend query. You can copy the query text. To do this, click the exclamation mark icon ⓘ next to the query text, and then in the window that opens, click **Copy** ⧉.

  You can sort values in this column using the corresponding icon.
- **Query duration**: The duration of the current or last backend query. You can sort values in this column using the corresponding icon.
- **Blocked totals**: The number of user sessions blocked by the clustering process.
- **Heap scanned, %**: The percentage of data scanned in the table for which clustering is being performed.
- **Heap total**: The total size of the table for which clustering is being performed.
- **Scanned**: The size of data scanned in the table for which clustering is being performed.
- **Tuples scanned**: The number of rows scanned in the table for which clustering is being performed.
- **Tuples written**: The number of rows written to the table for which clustering is being performed.
- **Actions**: Actions with the clustering process. For more information about the actions that you can perform with a clustering process, refer to other instructions in this section.

You can perform the following actions with the table of clustering processes:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To disable or enable the automatic update of the table, in the top-right corner of the page, click the start icon ▷ or the stop icon ⏸.
- To update the table, in the top-right corner of the page, click **Refresh data** ⟳. This action is available only if you disabled the automatic update of the table.
- To reset all filters, in the top-left corner of the table, click **Reset all**.

### Canceling a Clustering Process #

Canceling a clustering process does not terminate a user session.

To cancel a clustering process, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.

4. Select the **Cluster** tab, and then in the top-right corner of the page, select the database where the clustering process is running.

5. Click **Cancel current request** next to the clustering process that you want to cancel.

   The confirmation window will open.

6. Click **Execute**.

#### Terminating the User Session for a Clustering Process #

When the user session is terminated, the clustering process is canceled automatically.

To terminate the user session for a clustering process, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.

4. Select the **Cluster** tab, and then in the top-right corner of the page, select the database where the clustering process is running.

5. Click **Cancel session** ⊗ next to the clustering process for which you want to terminate the user session.

   The confirmation window will open.

6. Click **Execute**.

### Reindexing or Index Creation Processes #

You can view information about reindexing (`REINDEX`) or index creation (`CREATE INDEX`) processes. The information is based on the `pg_stat_progress_create_index` view.

This section explains how to manage reindexing or index creation processes. It includes the following instructions:

- Viewing Reindexing or Index Creation Processes
- Canceling a Reindexing or Index Creation Process
- [Terminating the User Session for a Reindexing or Index Creation Process](#terminating-the-user-session-for-a-reindexing-or-index-creation process)

#### Viewing Reindexing or Index Creation Processes #

To view reindexing or index creation processes, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.

4. Select the **Reindex** tab, and then in the top-right corner of the page, select the database where you want to view reindexing or index creation processes.

The table of reindexing or index creation processes with the following columns will be displayed:

- **PID**: The ID of the backend. You can sort values in this column using the corresponding icon.
- **State**: The state of the backend. For more information about possible values, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `state` column).

  You can filter and sort values in this column using the corresponding icons.

- **Wait event**: The name and type of the wait event for the backend. For more information, refer to the official Postgres Pro documentation on the `pg_stat_activity` module (see the `wait_event_type` and `wait_event` columns). You can sort values in this column using the corresponding icon.
- **Phase**: The reindexing or index creation phase. For more information about possible values, refer to the official Postgres Pro documentation.
- **Database**: The name of the database where reindexing or index creation is being performed. You can filter and sort values in this column using the corresponding icons.
- **User**: The name of the DBMS user under which reindexation or index creation is being performed. You can sort values in this column using the corresponding icon.
- **Table**: The name or ID of the table for which reindexing or index creation is being performed.
- **Index**: The name or ID of the index being reindexed or created.
- **Query**: The text of the current or last backend query. You can copy the query text. To do this, click the exclamation mark icon ⓘ next to the query text, and then in the window that opens, click **Copy** ⧉.

  You can sort values in this column using the corresponding icon.

- **Query duration**: The duration of the current or last backend query. You can sort values in this column using the corresponding icon.
- **Done, %**: The percentage of table data processed in the current phase of reindexing or index creation. This column includes additional information:
  - **Total size**: The total size of the table data to be processed.
  - **Done size**: The size of the processed table data.
- **Tuples done, %**: The percentage of table rows processed in the current phase of reindexing or index creation. This column includes additional information:
  - **Total size**: The total number of table rows to be processed.
  - **Done size**: The number of processed table rows.
- **Lockers**: The processes that were locked when performing reindexing or index creation. This column includes additional information:
  - **Done**: The number of locked processes for which the waiting is finished.
  - **Total**: The total number of locked processes.
  - **PID**: The ID of the currently locked process.
- **Partitions done, %**: The percentage of partitioned tables processed when performing reindexing or index creation. This column includes additional information:
  - **Total**: The total number of partitioned tables that must be processed.
  - **Done**: The number of processed partitioned tables.
- **Actions**: Actions with the reindexing or index creation process. For more information about the actions that you can perform with a reindexing or index creation process, refer to other instructions in this section.

You can perform the following actions with the table of reindexation or index creation processes:

- To change the column width, drag its border with the mouse.

- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To disable or enable the automatic update of the table, in the top-right corner of the page, click the start icon ▷ or the stop icon ⏸.
- To update the table, in the top-right corner of the page, click **Refresh data** ↻. This action is available only if you disabled the automatic update of the table.
- To reset all filters, in the top-left corner of the table, click **Reset all**.

#### Canceling a Reindexing or Index Creation Process #

Canceling a reindexing or index creation process does not terminate a user session.

To cancel a reindexing or index creation process, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.

4. Select the **Reindex** tab, and then in the top-right corner of the page, select the database where the reindexing or index creation process is running.

5. Click **Cancel current request** next to the reindexing or index creation process that you want to cancel.

   The confirmation window will open.

6. Click **Execute**.

#### Canceling the User Session for a Reindexing or Index Creation Process #

When the user session is terminated, the reindexing or index creation process is canceled automatically.

To terminate the user session for a reindexing or index creation process, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Activity** page.

   The table of user sessions will be displayed.

4. Select the **Reindex** tab, and then in the top-right corner of the page, select the database where the reindexing or index creation process is running.

5. Click **Cancel session** ⊗ next to the reindexing or index creation process for which you want to terminate the user session.

   The confirmation window will open.

6. Click **Execute**.

### SQL Statistics #

PPEM allows viewing execution statistics of SQL statements. This information is provided by the `pg_stat_statements` and `pgpro_stats` extensions. For SQL statistics collection to work correctly, one of these extensions must be installed and configured in the DBMS instance.

To view SQL statistics, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to view SQL statistics.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **SQL statistics** page.

The table with the following columns will be displayed:

- **Query ID**: The non-unique hash code of the statement. It may be the same as that of other statements executed in other databases or on behalf of other users. This column includes additional information:

  - **Plan ID**: The non-unique hash code of the plan. It may be the same as that of other plans for statements executed in other databases or on behalf of other users. The **Plan ID** is displayed only for the Postgres Pro Enterprise edition.

  - **Top level**: Specifies the statement execution level. To display this information, set the `pg_stat_statements.track` or `pgpro_stats.track` parameter to `top`. Possible values:

  - `true`: The statement is executed at the top level.

  - `false`: The statement is nested in a procedure or function.

    **Note**

    In the **Query ID** column, you can view the query and the statement plan. To do this, click **Expand** ⌄. To collapse the pop-up window, click **Collapse** ⌃. The statement plan is displayed only for the Postgres Pro Enterprise edition.

  You can filter and sort values in this column using the corresponding icons.

- **Database**: The instance database where the statement was executed. You can filter and sort values in this column using the corresponding icons.

- **User**: The name of the user who executed the statement. You can filter and sort values in this column using the corresponding icons.

- **Calls**: The total number of times the statement was executed. This column includes additional information:

  - **Rows**: The total number of rows retrieved or affected by the statement.

  You can sort values in this column using the corresponding icon.

- **Total execution time, ms**: The total time spent executing the statement (in milliseconds). This column includes additional information:

- **Max**: The maximum time spent executing the statement.
    - **Min**: The minimum time spent executing the statement.
    - **Mean**: The mean time spent executing the statement.
    - **Stddv**: The standard deviation of time spent executing the statement.

  You can sort values in this column using the corresponding icons.

- **Total planning time, ms**: The total time spent planning the statement (in milliseconds). To display this information, set the `pg_stat_statements.track_planning` or `pgpro_stats.track_planning` parameter to `on`. Otherwise, `0` is displayed. This column includes additional information:
    - **Max**: The maximum time spent planning the statement.
    - **Min**: The minimum time spent planning the statement.
    - **Mean**: The mean time spent planning the statement.
    - **Stddv**: The standard deviation of time spent planning the statement.

  You can sort values in this column using the corresponding icons.

- **Blocks time, ms**: The total time the statement spent reading and writing data file blocks (in milliseconds). To display this information, enable the `track_io_timing` configuration parameter. Otherwise, `0` is displayed. This column includes additional information:
    - **Write**: The time spent for writing blocks.
    - **Read**: The time spent for reading blocks.

  You can sort values in this column using the corresponding icon.

- **Temp blocks, pc.**: The total number of blocks affected by the statement when working with temporary files. This column includes additional information:
    - **Written**: The number of written blocks.
    - **Read**: The number of read blocks.

  You can sort values in this column using the corresponding icons.

- **WAL bytes, B**: The total amount of WAL bytes generated during the statement execution. This column includes additional information:
    - **Records, pc.**: The total number of WAL records generated during the statement execution.
    - **FPI, pc.**: The total number of WAL full page images generated during the statement execution.

  You can sort values in this column using the corresponding icons.

- **Shared blocks • Hits, pc.**: The total number of shared block cache hits by the statement. This column includes additional information:
    - **Read**: The total number of shared blocks read by the statement.
    - **Dirtied**: The total number of shared blocks dirtied by the statement.
    - **Written**: The total number of shared blocks written by the statement.

  You can sort values in this column using the corresponding icons.

- **Local blocks • Hits, pc.**: The total number of local block cache hits by the statement. This column includes additional information:
    - **Read**: The total number of local blocks read by the statement.
    - **Dirtied**: The total number of local blocks dirtied by the statement.
    - **Written**: The total number of local blocks written by the statement.

  You can sort values in this column using the corresponding icons.

To view information about a specific statement, click **Detailed** next to it.

By default, the displayed statistics are sorted by the total statement execution time. One page displays 50 rows.

The displayed statistics are requested via the PPEM agent working with the instance. For this reason, the speed at which statistics are retrieved depends on two factors:

- Network connectivity between the PPEM manager and agent that works with the instance.
- The volume of transferred data that can also indirectly affect the instance performance.

Given the cumulative nature of statistics, the resulting statistics snapshot may differ over time from the actual instance statistics.

You can perform the following actions with the table:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-right corner of the page, click **Reset filters**.

## Instance Parameters #

PPEM allows viewing and editing parameters of the DBMS instance.

### Viewing Parameters #

To view the instance parameters, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance whose parameters you want to view.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Parameters** page.

The parameters will be displayed, grouped into tables by different semantic categories. The tables include the following columns:

- **Parameter**: The parameter name with a short description. If the parameter is specified in a specific configuration file, the path to that file and the row number will be displayed next to it. This column also displays the parameter application status and errors, if they occurred. A hint may appear to the left of the parameter name, indicating that you need to restart an instance to apply the parameter. To open a full description of the parameter in the official Postgres Pro documentation, click on its name.

- **Value**: The parameter value. Two types of values are available:
    - The toggle that enables or disables the parameter. When the toggle is turned on, the icon for restoring the saved value appears to the right of the toggle.
    - A field for entering a specific parameter value. When a value is entered, an icon for restoring the saved value appears to the right of the field.

- **Default value**: The default parameter value.

- **Source**: The source of the parameter. Possible values:
  - `default': The default value.
  - `configuration file`: The value is specified in the configuration file.
  - `override`: The value is overridden at the instance initialization stage.

For convenient search, the page of parameters provides filtering by categories, parameter name search, and displaying either all parameters or only non-default ones, whose source value is `default`:

- To display parameters of a specific category, select the category from the drop-down list in the top-left corner of the page.
- To search for parameters by name, start typing the parameter name in the search bar in the top-right corner of the page.
- To display only non-default parameters, in the top-right corner of the page, set the toggle below the search bar to **Show non-default**. To display all parameters, switch the toggle back to the **Show all** default position.

#### Editing Parameters #

To edit instance parameters, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance whose parameters you want to edit.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Parameters** page.

   The parameters will be displayed, grouped into tables by different semantic categories.

4. In the **Value** column, edit one of two parameter value types:

   - The toggle that enables or disables the parameter. When the toggle is turned on, the icon for restoring the saved value ↺ appears to the right of the toggle.
   - A field for entering a specific parameter value. When a value is entered, an icon for restoring the saved value ↺ appears to the right of the field.

5. (Optional) To reset all edited but not yet saved parameters, in the bottom-right corner of the page, click **Reset**. The reset button will display the number of edited parameters.

6. Click **Save** in the bottom-right corner of the page. The save button will display the number of edited parameters.

   The action selection window will open if parameters, which require an instance restart, were edited.

7. In the window, select one of the actions:

   - **Apply changes and restart the instance**: the parameter changes will be applied, the instance will restart automatically.
   - **Apply changes, but restart instance later manually**: Parameter changes will be applied, but you will need to restart the instance manually.

8. Click **Apply**.

9. Confirm the instance restart.

The parameter changes will be applied.

> **Warning**
>
> Restarting the DBMS instance terminates all its active sessions. The restart process may take a significant amount of time due to the checkpoint execution. The restart duration depends on the configuration and the instance load. It is recommended to pay special attention to restarts, especially in production environments with strict DBMS downtime requirements.

#### Implementation Details #

The `ALTER SYSTEM` command is used to manage configuration parameters. It modifies the `postgresql.auto.conf` file. If a parameter was initially specified in the main configuration file, such as `postgresql.conf`, but later changed via PPEM and written to `postgresql.auto.conf`, the PPEM interface will display this parameter in both files. This is normal because values from `postgresql.auto.conf` take precedence over other configuration files.

PPEM uses the `pg_reload_conf()` function to apply parameters without restarting the instance.

### Extensions #

This section explains how to manage instance extensions. It includes the following instructions:

- Installing Extensions
- Editing an Extension
- Deleting an Extension

#### Installing Extensions #

To install an extension, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to install the extension.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Extensions** page.

   The table of extensions will be displayed.

4. In the top-right corner of the page, click **Install extension**.

   The extension installation window will open.

5. Enter details of the new extension (parameters marked with an asterisk are required):

   - **Database**: The database for which the extension will be installed.
   - **Extension**: The extension that will be installed.
   - **Version**: The version of the extension.
   - **Schema**: The schema into which the extension objects will be installed. If no schema is specified, the current schema is used.
   - **Install extensions that the selected one depends on**: Whether to install all extensions that the selected one depends on if they are not already installed.

6. Click **Install**.

The extension will be installed and displayed in the table of extensions.

#### Editing an Extension #

You can change the version of the installed extension and the schema where it is installed.

To edit an extension, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where the extension is installed.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Extensions** page.

   The table of extensions will be displayed.

4. Click **Edit** next to the extension that you want to edit.

   The extension editing window will open.

5. Edit the required extension parameters.

6. Click **Save**.

#### Deleting an Extension #

**Warning**

Deleted extensions cannot be restored.

To delete an extension, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where the extension is installed.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Extensions** page.

   The table of extensions will be displayed.

4. Click **Delete** next to the extension that you want to delete.

   The extension deletion window will open.

5. (Optional) To delete all related extensions, enable the **Delete related extensions** checkbox. This checkbox is disabled by default.

6. Click **Delete*.

The extension will be deleted and no longer be displayed in the table of extensions.

## Profiler #

*Profiler* is an interface to the `pgpro_pwr` module. With profiler, you can build detailed reports on database performance. Reports are based on samples of database data and cover specific periods of time.

To build a report, take at least two samples. The samples are placed in the directory of the profiler server. Before taking samples, prepare profiler servers and create them in the web application.

You can create schedules that allow taking samples automatically at a specific time interval. Such schedules also allow taking a sample for a specific date and time.

This section contains the following subsections that explain how to manage the profiler:

- Profiler Servers
- Samples
- Viewing Profiler Graphs
- Sampling Schedules
- Reports

### Profiler Servers #

This section explains how to manage profiler servers. It includes the following instructions:

- Creating a Profiler Server
- Viewing Profiler Servers
- Editing a Profiler Server
- Deleting a profiler server

#### Creating a Profiler Server #

Before performing this procedure:

- Create an instance.
- Create a database.
- Prepare a profiler server.

To create a profiler server, follow the steps below:

1. Go to profiler servers in one of the following ways:

   - Using the monitoring section:

     1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Servers** page.
     2. Select the instance where you want to create a profiler server from the **Instance** drop-down list.

   - Using the configuration section of a specific instance:

     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance where you want to create a profiler server.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Servers** page.

2. In the top-right corner of the page, click **Create server**.

   The profiler server creation window will open.

3. Enter details of the new profiler server (parameters marked with an asterisk are required):

   - **Database**: The database for which the profiler server will take samples.
   - **Server name*: The unique name of the profiler server.
   - **Description**: A short description of the profiler server.
   - **Connection string**: The connection string for the profiler server.
   - **Enabled**: Whether the profiler server is enabled. Possible values:
     - ** Yes**
     - **no**
   - **Retention period (days)**: The number of days during which samples can be stored on the profiler server. Samples stored longer than the specified period will be deleted.

4. Click **Save**.

The profiler server will be created and displayed in the table of profiler servers.

#### Viewing Profiler Servers #

To view the profiler servers, follow the steps below:

1. Go to profiler servers in one of the following ways:

   - Using the monitoring section:
     1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Servers** page.
     2. Select the instance where you want to view profiler servers from the **Instance** drop-down list.
   - Using the configuration section of a specific instance:
     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance where you want to view profiler servers.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Servers** page.

2. Select the database for which profiler servers take samples from the **Database** drop-down list.

3. Click **Select**.

The table of profiler servers with the following columns will be displayed:

- **Name**: The unique name of the profiler server. You can filter and sort values in this column using the corresponding icons.
- **Description**: A short description of the profiler server.
- **Connection string**: The connection string for the profiler server.
- **Enabled**: Whether the profiler server is enabled. Possible values:
  - `yes`
  - `no`
- **Retention period**: The number of days during which samples are stored. Samples stored longer than the specified period are deleted.
- **Actions**: Actions with the profiler server. For more information about the actions that you can perform with a profiler server, refer to other instructions in this section.

You can perform the following actions with the table of profiler servers:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-left corner of the page, click **Reset**.

#### Editing a Profiler Server #

To edit a profiler server, follow the steps below:

1. Go to profiler servers in one of the following ways:

   - Using the monitoring section:
     1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Servers** page.
     2. Select the instance where the profiler server is created from the **Instance** drop-down list.
   - Using the configuration section of a specific instance:
     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance where the profiler server is created.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Servers** page.

2. Select the database for which the profiler server takes samples from the **Database** drop-down list.

3. Click **Select**.

   A table of profiler servers will be displayed.

4. Click **Edit** next to the profiler server that you want to edit.

   The profiler server editing window will open.

5. Edit the required profiler server parameters.
6. Click **Save**.

The profiler server will be edited, and the updates will be displayed in the table of profiler servers.

#### Deleting a Profiler Server #

> **Warning**
>
> Deleted profiler servers cannot be restored.

Deleting a profiler server also removes all associated samples and disables sampling schedules that place samples to this server directory.

To delete a profiler server, follow the steps below:

1. Go to profiler servers in one of the following ways:
    - Using the monitoring section:
        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Servers** page.
        2. Select the instance where the profiler server is created from the **Instance** drop-down list.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

            The table of instances will be displayed.
        2. Click the name of the instance where the profiler server is created.

            The **Overview** page will be displayed.
        3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Servers** page.
2. Select the database for which the profiler server takes samples from the **Database** drop-down list.
3. Click **Select**.

    A table of profiler servers will be displayed.
4. Click **Delete** 🗑 next to the profiler server that you want to delete.

    The profiler server deletion window will open.
5. Click **Delete*.

The profiler server will be deleted and no longer be displayed in the table of profiler servers.

### Samples #

This section explains how to take and view samples.

#### Taking a sample #

Before performing this procedure:

- Create an instance.
- Create a database.
- Create a profiler server.

To take a sample, follow the steps below:

1. Go to samples in one of the following ways:
    - Using the monitoring section:
        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Samples** page.
        2. Select the instance where you want to take a sample from the **Instance** drop-down list.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

            The table of instances will be displayed.
        2. Click the name of the instance where you want to take a sample.

            The **Overview** page will be displayed.
        3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Samples** page.
2. In the top-right corner of the page, click **Take snapshot**.

    The sampling window will open.
3. Enter details of the new sample (parameters marked with an asterisk are required):
    - **Database**: The database for which the sample will be taken.
    - **Server**: The profiler server whose directory will be used to store the sample.
    - **skip_sizes**: Whether to skip the collection of relation sizes when taking the sample. Possible values:
        - **true**: The collection of relation sizes will be skipped when taking the sample.
        - **false**: The collection of relation sizes will not be skipped when taking the sample.
4. Click **Save**.

The sample will be taken and displayed in the table of samples.

#### Viewing Samples #

To view samples, follow the steps below:

1. Go to samples in one of the following ways:
    - Using the monitoring section:
        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Samples** page.
        2. Select the instance where you want to view samples from the drop-down list.

   - Using the configuration section of a specific instance:

     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance where you want to view samples.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Samples** page.

2. Select the database for which samples are taken from the **Database** drop-down list.

3. From the **Server** drop-down list, select the profiler server whose directory contains the samples.

4. Click **Select**.

The table of samples with the following columns will be displayed:

- **ID**: The unique identifier (ordinal number) of the sample.

- **Time**: The date and time when the sample was created.

- **Sizes were collected**: Whether relation sizes were collected when taking the sample. Possible values:

  - `Yes`
  - `No`

- **Database stat reset**: Whether database statistics were reset when taking the sample. Possible values:

  - `Yes`
  - `No`

- **Bgwriter stat reset**: Whether `bgwriter` statistics were reset. Possible values:

  - `Yes`
  - `No`

- **Archiver stat reset**: Whether archiver statistics were reset. Possible values:

  - `Yes`
  - `No`

You can perform the following actions with the table of samples:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** .

## Viewing Profiler Graphs #

Before performing this procedure:

- Create an instance.
- Create a database.
- Create a profiler server.
- Take at least two samples.

To view profiler graphs, follow the steps below:

1. Go to profiler graphs in one of the following ways:

   - Using the monitoring section:

     1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** page.
     2. Select the instance for which you want to view the profiler graphs from the **Instance** drop-down list.

   - Using the configuration section of a specific instance:

     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance where you want to view the profiler graphs.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Profiler** page.

2. Select the database for which you want to display the profiler graphs from the **Database** drop-down list.

3. From the **Server** drop-down list, select the profiler server whose directory contains the samples.

4. (Optional) In the **Period** field, specify the time period for which the profiler graphs should be displayed. If you do not specify a value, the profiler graphs are displayed for the entire time range.

5. Click **Select**.

The following profiler graphs will be displayed:

- **PostgreSql Instance: tuples**: Operations with rows per second in the database. Available metrics:

  - **Tuples returned**: The number of live rows fetched by sequential scans and index entries returned by index scans. This metric corresponds to the `tup_returned` column of the `pg_stat_database` view.
  - **Tuples fetched**: The number of live rows fetched by index scans. This metric corresponds to the `tup_fetched` column of the `pg_stat_database` view.
  - **Tuples inserted**: The number of rows inserted by queries. This metric corresponds to the `tup_inserted` column of the `pg_stat_database` view.
  - **Tuples updated**: The number of rows updated by queries. This metric corresponds to the `tup_updated` column of the `pg_stat_database` view.
  - **Tuples deleted**: The number of rows deleted by queries. This metric corresponds to the `tup_deleted` column of the `pg_stat_database` view.

  For more information about these metrics, refer to the official Postgres Pro documentation on the `pg_stat_database` view.

- **PostgreSql bgwriter buffers**: Operations with buffers per second in the database. Available metrics:

  - **Checkpoints buffers written**: The number of buffers written during checkpoints and restartpoints. This metric corresponds to the `buffers_written` column of the `pg_stat_checkpointer` view.
  - **Background buffers written**: The number of buffers written by the background writer. This metric corresponds to the `buffers_clean` column of the `pg_stat_bgwriter`

view.
  - **Backend buffers written**: The number of buffers written directly by the backend. In Postgres Pro 16, this metric corresponds to the `buffers_backend` column of the `pg_stat_bgwriter` view. In Postgres Pro 17, this metric is collected by the `pg_stat_io` view.
  - **Number of buffers allocated**: The number of allocated buffers. This metric corresponds to the `buffers_alloc` column of the `pg_stat_bgwriter` view.

  For more information about these metrics, refer to the official Postgres Pro documentation on the `pg_stat_bgwriter`, `pg_stat_io`, and `pg_stat_checkpointer` views.

- **PostgreSql bgwriter write/sync**: Synchronization and writing operations with buffers per second in the database. Available metrics:

  - **Bgwriter interrupts**: The number of times the background writer stopped a cleaning scan due to writing too many buffers. This metric corresponds to the `maxwritten_clean` column of the `pg_stat_bgwriter` view.
  - **Backend fsync count**: The number of times the backend executed its own `fsync` call. Normally, these calls are executed by the background writer, even when the backend performs its own writes. In Postgres Pro 16 or lower, this metric corresponds to the `buffers_backend_fsync` column of the `pg_stat_bgwriter` view. In Postgres Pro 17, this metric is collected by the `pg_stat_io` view.

  For more information about these metrics, refer to the official Postgres Pro documentation on `pg_stat_bgwriter` and `pg_stat_io`.

- **PostgreSql checkpoints count**: Operations with checkpoints per second in the database. Available metrics:

  - **Scheduled checkpoints**: The number of scheduled checkpoints that were completed due to timeout. Scheduled checkpoints can be skipped if the server has been idle since the last checkpoint. Both completed and skipped scheduled checkpoints are counted. This metric corresponds to the `num_timed` column of the `pg_stat_checkpointer` view.
  - **Requested checkpoints**: The number of requested checkpoints that were completed. This metric corresponds to the `num_requested` column of the `pg_stat_checkpointer` view.

  For more information about these metrics, refer to the official Postgres Pro documentation on the `pg_stat_checkpointer` view.

- **PostgreSql checkpoints write/sync**: The time per second spent writing and synchronizing files during checkpoints in the database. Available metrics:

  - **Checkpoint write time (s)**: The time in seconds spent writing files to disk while completing checkpoints and restartpoints. This metric corresponds to the `write_time` column of the `pg_stat_checkpointer` view.
  - **Checkpoint sync time (s)**: The time in seconds spent synchronizing files to disk while completing checkpoints and restartpoints. This metric corresponds to the `sync_time` column of the `pg_stat_checkpointer` view.

  For more information about these metrics, refer to the official Postgres Pro documentation on the `pg_stat_checkpointer` view.

- **PostgreSql Instance: events**: Operations with events per second in the database. Available metrics:

  - **Conflicts**: The number of queries canceled due to conflicts with recovery. Conflicts can occur only on standby servers. This metric corresponds to the `conflicts` column of the `pg_stat_database` view. For more information about conflicts, refer to the official Postgres Pro documentation on the pg_stat_database_conflicts view.
  - **Deadlocks**: The number of deadlocks. This metric corresponds to the `deadlocks` column of the `pg_stat_database` view.
  - **Rollbacks**: The number of rolled back transactions. This metric corresponds to the `xact_rollback` column of the `pg_stat_database` view.
  - **Commits**: The number of committed transactions. This metric corresponds to the `xact_commit` column of the `pg_stat_database` view.

  For more information about these metrics, refer to the official Postgres Pro documentation on the `pg_stat_database` view.

- **PostgreSql: cache hit ratio**: This graph displays the **Cache hit ratio** metric that provides the percentage of the data received from the buffer cache per second in the database. This metric is based on the `blks_hit` and `blks_read` columns of the `pg_stat_database` view and is calculated as follows:

  `blks_hit` / (`blks_hit` + `blks_read`)

  For more information about the `blks_hit` and `blks_read` columns, refer to the official Postgres Pro documentation on the `pg_stat_database` view.

- **PostgreSql temp: bytes written**: This graph displays the **Bytes written** metric that provides the amount of data written to temporary files by queries per second in the database. All temporary files are counted, regardless of why they were created and of the `log_temp_files` parameter value. This metric corresponds to the `temp_bytes` column of the `pg_stat_database` view.

  For more information about this metric, refer to the official Postgres Pro documentation on the `pg_stat_database` view.

- **PostgreSql temp: files created**: This graph displays the **Number of files** metric that provides the number of temporary files created by queries per second in the database. All temporary files are counted, regardless of why they were created (sorting, hashing) and of the `log_temp_files` parameter value. This metric corresponds to the `temp_files` column of the `pg_stat_database` view.

  For more information about this metric, refer to the official Postgres Pro documentation on the `pg_stat_database` view.

- **PostgreSql archive command**: Archiving operations with WAL files per second in the database. Available metrics:

  - **WAL segments archived**: The number of successfully archived WAL files. This metric corresponds to the `archived_count` column of the `pg_stat_archiver` view.
  - **WAL segments archive failed**: The number of failed attempts to archive WAL files. This metric corresponds to the `failed_count` column of the `pg_stat_archiver` view.

  For more information about these metrics, refer to the official Postgres Pro documentation on the `pg_stat_archiver` view.

- **PostgreSQL WAL write speed**: This graph displays the **WAL generated** metric that provides the amount of WAL in bytes generated per second in the database. This metric corresponds to the `wal_bytes` column of the `pg_stat_wal` view.

  For more information about this metric, refer to the official Postgres Pro documentation on the `pg_stat_wal` view.

You can perform the following actions using the icons in the top-right corner of the planner graphs:

- To build a report using a graph, click **Select the period for the report**. For more information about building a report, refer to the Reports section.
- To reset the selected time period when building a report, click **Reset**.
- To download the graph in `PNG` format, click **Save as Image**.

## Sampling Schedules #

This section explains how to manage sampling schedules. It includes the following instructions:

- Creating a Sampling Schedule
- Viewing Sampling Schedules
- Editing a Sampling Schedule
- Executing a Sampling Schedule
- Activating and Deactivating a Sampling Schedule
- Deleting a Sampling Schedule

### Creating a Sampling Schedule #

Before performing this procedure:

- Create an instance.
- Create a database.
- Create a profiler server.

To create a sampling schedule, follow the steps below:

1. Go to sampling schedules in one of the following ways:

   - Using the monitoring section:

        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Schedule** page.
        2. Select the instance where you want to create a sampling schedule from the **Instance** drop-down list.

    - Using the configuration section of a specific instance:

        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

           The table of instances will be displayed.

        2. Click the name of the instance where you want to create a sampling schedule.

           The **Overview** page will be displayed.

        3. In the **Instance details** of the navigation panel, go to the **Profiler** → **Schedule** page.

    The table of sampling schedules will be displayed.

2. In the top-right corner of the page, click **Create task**.

   The sampling schedule creation window will open.

3. Enter details of the new sampling schedule (parameters marked with an asterisk are required):

    - **Name**: The unique name of the sampling schedule.

    - **Set cron-string execution**: Allows specifying the time interval for taking samples in the `crontab` format. This toggle is turned off by default. If it is turned on, in the **Execution** field, enter a string in the `crontab` format.

    - **Task planning**: The type of the sampling schedule. This parameter is available only if you turned the **Set crone-string execution** toggle off. Possible values:

        - **Time-delayed**: One sample will be created at a specific date and time. For this value, specify the required date and time in the **Time** field.

        - **On schedule**: Samples will be taken within a specific time interval. For this value, specify the following parameters:

            - **Interval**: The unit of measurement of the time interval. Possible values:

                - **Minutes**. This value is selected by default.
                - **Hours**.
                - **Days**.

            - **Repeat every**: The time interval for taking samples by minutes or hours. This parameter is available only if you selected **Minutes** or **Hours** for the **Task planning** parameter.

            - **Execution days**: Specifies the days for taking samples.

    - **Start** and **Repeat until**: The start and end date and time for taking samples on schedule. These parameters are available only if you turned on the **Set cron-string execution** toggle or selected **On schedule** for the **Task planning** parameter.

    - **Cron total line**: The string in the `crontab` format that specifies the time interval for taking samples. The value is entered automatically. This parameter is available only if you selected **On schedule** for the **Task planning** parameter.

    - **Database**: The database for which the sample will be taken.

    - **Server**: The profiler server whose directory will be used to store the sample.

    - **skip_sizes**: Whether to skip the collection of relation sizes when taking the sample. Possible values:

        - **true**: The collection of relation sizes will be skipped when taking the sample.
        - **false**: The collection of relation sizes will not be skipped when taking the sample.

4. Click **Save**.

The sampling schedule will be created and displayed in the table of sampling schedules.

#### Viewing Sampling Schedules #

To view sampling schedules, follow the steps below:

1. Go to sampling schedules in one of the following ways:

    - Using the monitoring section:

        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Schedule** page.
        2. Select the instance where you want to view sampling schedules from the **Instance** drop-down list.

    - Using the configuration section of a specific instance:

        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

           The table of instances will be displayed.

        2. Click the name of the instance where you want to view sampling schedules.

           The **Overview** page will be displayed.

        3. In the **Instance details** of the navigation panel, go to the **Profiler** → **Schedule** page.

The table of sampling schedules with the following columns will be displayed:

- **Task**: The unique name of the sampling schedule. You can filter and sort values in this column using the corresponding icons.
- **Instance**: The instance where the sampling schedule is created.
- **Database**: The database for which samples are taken. You can filter and sort values in this column using the corresponding icons.
- **Schedule**: The string in the `crontab` format that specifies the time interval for taking samples.
- **User**: The user that created the sampling schedule. You can filter and sort values in this column using the corresponding icons.
- **Actions**: Actions with the sampling schedule. For more information about the actions that you can perform with a sampling schedule, refer to other instructions in this section.

You can perform the following actions with the table of sampling schedules:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** 🔄.
- To reset all filters, in the top-right corner of the page, click **Reset**.

#### Editing a Sampling Schedule #

To edit a sampling schedule, follow the steps below:

1. Go to sampling schedules in one of the following ways:
    - Using the monitoring section:
        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Schedule** page.
        2. Select the instance where the sampling schedule is created from the **Instance** drop-down list.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

            The table of instances will be displayed.
        2. Click the name of the instance where the sampling schedule is created.

            The **Overview** page will be displayed.
        3. In the **Instance details** of the navigation panel, go to the **Profiler** → **Schedule** page.

    The table of sampling schedules will be displayed.
2. Click the three vertical dots icon ⋮ → **Edit** next to the sampling schedule that you want to edit.

    The sampling schedule editing window will open.
3. Edit the required sampling schedule parameters.
4. Click **Save**.

The sampling schedule will be edited, and the updates will be displayed in the table of sampling schedules.

###### Executing a Sampling Schedule #

You can manually execute a sampling schedule to instantly start taking samples.

To execute a sampling schedule, follow the steps below:

1. Go to sampling schedules in one of the following ways:
    - Using the monitoring section:
        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Schedule** page.
        2. Select the instance where the sampling schedule is created from the **Instance** drop-down list.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

            The table of instances will be displayed.
        2. Click the name of the instance where the sampling schedule is created.

            The **Overview** page will be displayed.
        3. In the **Instance details** of the navigation panel, go to the **Profiler** → **Schedule** page.

    The table of sampling schedules will be displayed.
2. Click the three vertical dots icon ⋮ → **Execute** next to the sampling schedule that you want to execute.

The sampling schedule execution will start. The sample will be taken and displayed in the table of samples.

###### Activating and Deactivating a Sampling Schedule #

You can deactivate a sampling schedule to temporarily stop taking samples. Sampling schedules are activated by default.

To deactivate or activate a sampling schedule, follow the steps below:

1. Go to sampling schedules in one of the following ways:
    - Using the monitoring section:
        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Schedule** page.
        2. Select the instance where the sampling schedule is created from the **Instance** drop-down list.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

            The table of instances will be displayed.
        2. Click the name of the instance where the sampling schedule is created.

            The **Overview** page will be displayed.
        3. In the **Instance details** of the navigation panel, go to the **Profiler** → **Schedule** page.

    The table of sampling schedules will be displayed.
2. Click the three vertical dots icon ⋮ → **Deactivate** or **Activate** next to the sampling schedule that you want to deactivate or activate.

###### Deleting a Sampling Schedule #

> **Warning**
>
> Deleted sampling schedules cannot be restored.

When you delete a sampling schedule, the associated samples are not deleted.

To delete a sampling schedule, follow the steps below:

1. Go to sampling schedules in one of the following ways:
    - Using the monitoring section:
        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Schedule** page.
        2. Select the instance where the sampling schedule is created from the **Instance** drop-down list.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

    2. Click the name of the instance where the sampling schedule is created.

        The **Overview** page will be displayed.

    3. In the **Instance details** of the navigation panel, go to the **Profiler** → **Schedule** page.

   The table of sampling schedules will be displayed.

2. Click the three vertical dots icon ⋮ → **Delete** next to the sampling schedule that you want to delete.

   The sampling schedules deletion window will open.

3. Click *Delete.

The sampling schedule will be deleted and no longer be displayed in the table.

## Reports #

This section explains how to manage reports. It includes the following instructions:

- Reports
    - Building a Report
    - Building a Report Using a Profiler Graph
    - Viewing All Built Reports
    - Viewing and Downloading a Report
    - Deleting a Report

It is recommended to see available profiler graphs first.

### Building a Report #

Before performing this procedure:

- Create an instance.
- Create a database.
- Create a profiler server.
- Take at least two samples.

To build a report, follow the steps below:

1. Proceed to building a report in one of the following ways:

    - Using the monitoring section:

        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Reports generating** page.
        2. Select the instance where you want to build a report from the **Instance** drop-down list.

    - Using the configuration section of a specific instance:

        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

            The table of instances will be displayed.

        2. Click the name of the instance where you want to build a report.

            The **Overview** page will be displayed.

        3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Reports generating** page.

2. Enter details of the new report (parameters marked with an asterisk are required):

    - **Database**: The database for which the report will be built.

    - **Server**: The profiler server whose directory contains samples.

    - **Report**: The format of the time period that the report will cover. Possible values:

        - **By snapshot**: The report will cover the time period between two samples. This value is selected by default.
        - **By time**: The report will cover the time period between two dates.

    - **Type**: The report type. Possible values:

        - **Standard**: Provides the database load statistics for a specified time period. This value is selected by default.
        - ** Differential**: Provides comparative database load statistics for two specified time periods.

    - **Interval** or **Interval 1** and **Interval 2**: The time period covered by the report. The period format depends on the value selected in the **Report** section.

        - If you selected **By snapshot**, specify the first and last samples.
        - If you selected **By time**, specify the start and end date and time.

3. Click **Generate report**.

The report will be built and displayed in the table of reports.

### Building a Report Using a Profiler Graph #

You can also build a report when viewing profiler graphs. Note that in this case, the time period between two samples cannot be used as the time period that the report will cover.

Before performing this procedure:

- Create an instance.
- Create a database.
- Create a profiler server.
- Take at least two samples.

To build a report using a graph profiler, follow the steps below:

1. Go to profiler graphs in one of the following ways:

    - Using the monitoring section:

        1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** page.
        2. Select the instance where you want to build a report from the **Instance** drop-down list.

    - Using the configuration section of a specific instance:

        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

   2. Click the name of the instance where you want to build a report.

      The **Overview** page will be displayed.

   3. In the **Instance details** section of the navigation panel, go to the **Profiler** page.

2. Select the database for which you want to build a report from the **Database** drop-down list.
3. From the **Server** drop-down list, select the profiler server whose directory contains the samples.
4. (Optional) In the **Period** field, specify the time period for which the profiler graphs should be displayed. If you do not specify a value, the profiler graphs are displayed for the entire time range.
5. Click **Select**.

   Profiler graphs will be displayed.

6. In the top-right corner of the profiler graph, **Select the period for the report**.
7. Click and drag on the profiler graph to select the time period for your report.

   The report building window will open.

8. (Optional) To include comparative database load statistics for two specified time periods in the report, click **Select another period** and repeat step 7.
9. Click **Generate report**.

The report will be built and displayed in the table of reports.

#### Viewing All Built Reports #

To view all built reports, follow the steps below:

1. Go to reports in one of the following ways:
   - Using the monitoring section:

     In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Reports** page.

   - Using the configuration section of a specific instance:
     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance where you want to view all built reports.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Reports** page.

The table of reports with the following columns will be displayed:

- **Instance**: The instance where the report is built.
- **Server**: The profiler server whose directory contains samples. You can sort values in this column using the corresponding icon.
- **Database**: The database for which the report is generated.
- **Status**: The status of the report. Possible values:
  - `done`: The report is built.
  - `pending`: The report is being built.
  - `error`: An error occurred while building a report.

  You can sort values in this column using the corresponding icon.

- **Period**: The time period covered by the report.
- **Execution start** and **Execution end**: The start and end date and time of report building. You can sort values in this column using the corresponding icon.
- **User**: The user that built the report. You can sort values in this column using the corresponding icon.
- **Actions**: Actions with the report. For more information about the actions that you can perform with a report, refer to other instructions in this section.

You can perform the following actions with the table of reports:

- To change the column width, drag its border with the mouse.
- To update the table, in the top-right corner of the page, click **Refresh data**.

#### Viewing and Downloading a Report #

To view or download a report, follow the steps below:

1. Go to reports in one of the following ways:
   - Using the monitoring section:

     In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Reports** page.

   - Using the configuration section of a specific instance:
     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance where the report is built.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Reports** page.

   The table of reports will be displayed.

2. Perform one of the following actions:
   - To view the report, click **View** next to it.

     The report page will be displayed. For more information about the report contents, refer to the official Postgres Pro documentation.

- To download a report, in the top-right corner of the page, click **Download report**.

  The report in the `HTML` format will be downloaded to your local device.

#### Deleting a Report #

> **Warning**
>
> Deleted reports cannot be restored.

To delete a report, follow the steps below:

1. Go to reports in one of the following ways:
   - Using the monitoring section:

     In the **Management** section of the navigation panel, go to the **Monitoring** → **Profiler** → **Reports** page.
   - Using the configuration section of a specific instance:
     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.
     2. Click the name of the instance where the report is built.

        The **Overview** page will be displayed.
     3. In the **Instance details** section of the navigation panel, go to the **Profiler** → **Reports** page.

   The table of reports will be displayed.
2. Click **Delete** 🗑 next to the report that you want to delete.

   The report deletion window will open.
3. Click **Delete*.

The report will be deleted and no longer be displayed in the table of reports.

## Authentication #

PPEM allows viewing authentication rules based on the host name of the instance (host-based authentication; HBA) and editing them. The rules are specified in the `pg_hba.conf` configuration file.

### Viewing Authentication Rules #

To view authentication rules, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the required instance.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Authentication** page.

The table with the `pg_hba.conf` configuration file information and the following columns will be displayed:

- **Type**: The type of the connection. Possible values:
  - `local`
  - `host`
  - `hostssl`
  - `hostnossl`
  - `hostgssenc`
  - `hostnogssenc`
- **Database**: The list of databases to which the authentication rule applies. The `all` value means that the rule applies to all databases.
- **User**: The list of users and groups to which the authentication rule is applied. The `all` value means that the rule applies to all users.
- **Address**: The network address(es) of the client machine to which the authentication rule applies. This field can include the name of the computer, IP range, or one of the keywords. The `all` value means that the rule applies to all IP addresses. This column is not used for rules with the `local` type.
- **IP-mask**: The mask of the IP address. This parameter is displayed if only an IP address is specified in the **Address** column. Specifying a mask in a separate column is an alternative for the <IP address/mask length> record. This column is not used for rules with the `local` type .
- **Method**: The authentication method. Possible values:
  - `trust`
  - `reject`
  - `scram-sha-256`
  - `md5`
  - `password`

  For the full list of possible authentication methods, refer to the Authentication Methods section of the official Postgres Pro documentation.
- **Options**: Authentication method parameters in the <name=value> format. For more information about available authentication method parameters, refer to the Authentication Methods section of the official Postgres Pro documentation.

The rules in the table follow the same order as in the main `pg_hba.conf` configuration file. If the `include`, `include_if_exists`, and `include_dir` directives are used in the file, the rules listed in the included files are not displayed.

To view the full `pg_hba.conf` configuration file, in the top-right corner of the page, click **View full file**.

### Editing Authentication Rules #

There are two ways to edit authentication rules:

- Add a line to the `pg_hba.conf` configuration file
- Go to edit mode of the `pg_hba.conf` configuration file

#### Adding a Line to the Configuration File #

To add a line to the `pg_hba.conf` configuration file, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Authentication** page.

   The table with the `pg_hba.conf` configuration file information will be displayed.

4. In the top-right corner of the page, click **Add line**.

   The window for adding a line will open.

5. Enter details of the new line in the `pg_hba.conf` configuration file (parameters marked with an asterisk are required):

   - **Type**: The type of the connection.
   - **Users (search)**: The names of users and groups to which the authentication rule that you are adding will apply. You can select users from the drop-down list.
   - **Users (will be saved to file)**: The names of users and groups to which the authentication rule that you are adding will be applied. You can manually specify a list of comma-separated users.
   - **Databases (search)**: The names of databases to which the authentication rule that you are adding will apply. You can select databases from the drop-down list.
   - **Databases (will be saved to file)**: The names of databases to which the authentication rule that you are adding will apply. You can manually specify a list of comma-separated databases.
   - **Address**: The network address(es) of the client machine to which the authentication rule that you are adding will apply. This parameter is displayed only if any value except `local` is selected in the **Type** field.
   - **IP-mask**: The mask of the IP address. This parameter is displayed only if any value except `local` is selected in the **Type** field.
   - **Method**: The authentication method.
   - **Options**: The authentication method parameters in the <name=value> format.

6. Click **Add**.

   The line will be added and displayed in the table of the `pg_hba.conf` configuration file.

7. (Optional) To reset all added but not yet saved lines, in the bottom-right corner of the page, click **Reset**. The number of added lines will be displayed on the reset button.

8. In the bottom-right corner of the page, click **Save**. The number of added lines will be displayed on the save button.

The line will be added to the end of the `pg_hba.conf` configuration file, and the updates will be displayed in the table.

#### Configuration File Editing Mode #

To edit the configuration file in editing mode, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the required instance.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Authentication** page.

   The table with the `pg_hba.conf` configuration file information will be displayed.

4. In the top-right corner of the page, turn on the **Edit mode** toggle. This toggle is turned off by default.

   The contents of the `pg_hba.conf` configuration file will be displayed.

5. Perform one of the following actions with the line:

   - To move the line one position up, click **Move up** ^ next to it.
   - To move the line one position down, click **Move down** icon-chevron-down-blue next to it.
   - To delete the line, click **Delete** 🗑 next to it.

6. (Optional) To reset all edited but not yet saved lines, in the bottom-right corner of the page, click **Reset**. The number of edited lines will be displayed on the reset button.

7. In the bottom-right corner of the page, click **Save**. The number of edited lines will be displayed on the save button.

The `pg_hba.conf` configuration file will be edited, and the updates will be displayed in the table.

> **Note**
>
> When rules are saved, the instance receives a configuration reload signal. In this case, all configuration files of the instance are reread.

### Roles #

To view active roles, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.

2. Click the name of the instance where you want to view roles.

   The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Roles** page.

   The table of roles will be displayed.

4. (Optional) Turn on the **Show system roles** toggle to also display system roles. This toggle is turned off by default.

The table of roles includes the following columns:

- **Role**: The unique name of the role. You can filter values in this column using the corresponding icon.
- **Superuser**: Whether the role has the `SUPERUSER` privilege. You can filter values in this column using the corresponding icon.
- **Create role**: Whether the role has the `CREATEROLE` privilege. You can filter values in this column using the corresponding icon.
- **Create DB**: Whether the role has the `CREATEDB` privilege. You can filter values in this column using the corresponding icon.
- **Login**: Whether the role has the `LOGIN` privilege. You can filter values in this column using the corresponding icon.
- **Inheritance of privileges**: Whether the role automatically inherits privileges of the roles it is a member of.
- **Replication**: Whether the role is a replication role. Such roles can initiate replication connections and create and drop replication slots.
- **Bypass RLS**: Whether the role bypasses row level security policies. For more information, refer to the official Postgres Pro documentation on row security policies.
- **Connection limit**: The maximum number of concurrent connections this role can make. Applies to roles that can connect to the server.

- **Valid until**: The password expiry time. Only used for password authentication.
- **Member_of**: The roles that include this role as a member.
- **Include roles**: The roles that are members of this role.

You can filter values in all the columns using the corresponding icon.

You can perform the following actions with the table of roles:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-right corner of the page, click **Reset**.

## Message Log #

PPEM logs DBMS instances. To provide logging, you must first install and configure logging tools.

To view the message log, follow the steps below:

- To view the message log of all instances, in the **Management** section of the navigation panel, go to the **Monitoring** → **Message journal** page.
- To view the message log of a specific instance, follow the steps below:
  1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

     The table of instances will be displayed.
  2. Click the name of the instance whose message log you want to view.

     The **Overview** page will be displayed.
  3. In the **Instance details** section of the navigation panel, go to the **Message journal** page.

The table of messages with the following columns will be displayed:

- **Date, time**: The date and time when the message was created.
- **Instance**: The instance associated with the message.
- **Session**: The ID of the user session where the message was created.
- **Application**: The source application of the message.
- **User**: The user on behalf of which the session was initiated.
- **Database**: The database associated with the user session. You can filter values in this column using the corresponding icon.
- **Type**: The type of the message. Possible values:
  - `DEBUG1`
  - `DEBUG2`
  - `DEBUG3`
  - `DEBUG4`
  - `DEBUG5`
  - `INFO`
  - `NOTICE`
  - `WARNING`
  - `ERROR`
  - `LOG`
  - `FATAL`
  - `PANIC`

  For more information about message types, refer to the official Postgres Pro documentation.

  You can filter values in this column using the corresponding icon.
- **SQLSTATE code**: The SQLSTATE code of the message.
- **Message**: The text of the message.
- **Details**: The detailed text of the message.
- **Hint**: The error hint. Only values of the `ERROR`, `FATAL`, and `PANIC` messages are displayed in this column.
- **Query ID**: The ID of the query associated with the message. You can use this ID to find statistics in the `pg_stat_activity`, `pg_stat_statements`, and `pgpro_stats` views.
- **Query**: The text of the query associated with the message. The length of the displayed text is limited. You can expand or collapse queries in one of the following ways:
  - To expand or collapse a specific query, click **Expand** or **Collapse** next to it.
  - To expand or collapse all queries, in the top-left corner of the table of messages, click **Expand all** or **Collapse all** .

You can perform the following actions with the table of messages:

- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To find messages by keywords, in the top-right corner of the page, enter keywords in the search field.
- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, at the top of the page, click **Reset**.

## Backup #

PPEM supports creating backups for instances. You can use a backup to create a new instance if the previous instance fails. To work with backups, you must first install and configure the backup and restore tools.

Before creating backups, you must create storages where they will be placed. You can create local and S3 storages.

> **Note**
>
> Local storages are indended for introductory use with PPEM. For production environments with a large number of instances and created backups, it is recommended to use S3 storages.

When creating a storage, you must bind an instance to it. When an instance is bound to a storage, a catalog for this instance's backups is created in the storage. You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage. You can manage instance binds separately.

Backup schedules can also be created for automatically creating instance backups within a specified time interval. Backup schedules also allow for scheduling a single backup at a specific date and time.

This section explains how to manage storages, instance binds, backups, and backup schedules.

## Storages #

This section explains how to manage storages. It includes the following instructions:

- Creating a Storage
- Viewing Storages
- Editing a Storage
- Deleting a Storage

### Creating a Storage #

You can create local storages and S3 storages.

#### Creating a Local Storage #

To create a local storage, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Backup storages** page.

   The table of storages will be displayed.

2. In the top-right corner of the page, click **Create backup storage**.

   The storage creation window will open.

3. In the **Select type** step, select **Local storage**, and then click **Next**.

4. In the **Specify parameters** step, enter details of the new local storage (parameters marked with an asterisk are required):

   - **Storage name**: The unique name of the storage.

   - **Catalog of copies**: The path to the local storage catalog where backups will be placed. You must specify the path to an empty catalog.

   - **System user**: The OS user who will own the local storage catalog. It is recommended to specify the user on behalf of which the instance is installed, because this user must have read and write privileges in the local storage catalog.

   - **Instance**: The instance that will be bound to the local storage. When an instance is bound to a storage, a catalog for this instance's backups is created in the storage. You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage.

   - **Storage parameters**: The backup storage parameters of the storage catalog created for the instance. Available parameters:

     - **Retention redundancy, pcs.**: The maximum number of full backups. For example, if you specify `3`, the catalog can contain a maximum of three full backups.

       To disable this limitation, specify `0`. In this case, the number of backups in the catalog is not limited.

     - **Retention window, days**: The number of days (24 hours) covered by backups. For example, if you specify `7`, the catalog must always contain backups required for restoring the data for the last seven days, including today.

       To disable this limitation, specify `0`. In this case, backups can be deleted from the catalog at any moment.

     - **WAL depth, pcs.**: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR). For example, if you specify `3`, the catalog must always contain at least three backups on each timeline.

       To disable this limitation, specify `0`. In this case, point-in-time recovery is not possible.

     - **Expired copies**: The management policy for expired backups. Possible values:

       - **Merge**: Merge expired backups with new ones if possible.
       - **Delete**: Delete expired backups from the catalog.

       You can enable all checkboxes simultaneously.

     The **Retention redundancy, pcs**, **Retention window, days**, and **WAL depth, pcs** are applied only if you enabled the **Merge** and/or **Delete** checkbox for the **Expired copies** parameter.

     The values of the **Retention redundancy, pcs.** and **Retention window, days** parameters are considered simultaneously when deleting expired backups from the catalog. For example, if you entered `3` in the **Retention redundancy, pcs.** field and `7` in the **Retention window, days** field, a maximum of three full backups will be saved, as well as all backups required for restoring the data for the past 7 days, including today.

     You can also configure storage parameters for an instance, as well as for a backup when creating it. The following priority is used:

     - Backup parameters are applied first.
     - Instance parameters are applied second.
     - Storage parameters are applied third.

     For more information about storage parameters, refer to the official Postgres Pro documentation on `pg_probackup`.

5. Click **Save**.

The local storage will be created and displayed in the table of storages.

#### Creating an S3 Storage #

To create an S3 storage, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Backup storages** page, and then select the **S3 storages** tab.

   The table of storages will be displayed.

2. In the top-right corner of the page, click **Create backup storage**.

   The storage creation window will open.

3. In the **Select type** step, select **S3 storage**, and then click **Next**.

4. In the **Specify parameters** step, enter details of the new S3 storage (parameters marked with an asterisk are required):

   - **Storage name**: The unique name of the S3 storage.

   - **Type**: The provider of the S3 storage. Possible values:

     - **AWS**
     - **Minio**
     - **VK**

- **Host**: The server where the commands for interactions between the manager and S3 storage will be executed.
- **Server name**: The network address of the S3 storage server.
- **Protocol https**: Whether the HTTPS is used for interactions between the manager and S3 storage. This toggle is disabled by default.
- **Port**: The number of the port for connecting the manager to the S3 storage.
- **Access Key ID** and **Secret access key**: The secure keys for connecting the manager to the S3 storage.
- **Bucket**: The name of the bucket on the S3 storage server where backups will be placed.
- **Region**: The region where the S3 storage server is located.
- **Catalog of copies**: The path to the bucket catalog where backups will be placed.
- **Instance**: The instance that will be bound to the S3 storage. When an instance is bound to a storage, a catalog for this instance's backups is created in the storage. You cannot bind an instance to the same storage twice. If an instance is bound for the second time, it is unbound from the previous storage.
- **Storage parameters**: The backup storage parameters of the storage catalog created for the instance. Available parameters:
    - **Retention redundancy, pcs.**: The maximum number of full backups. For example, if you specify `3`, the catalog can contain a maximum of three full backups.

      To disable this limitation, specify `0`. In this case, the number of backups in the catalog is not limited.
    - **Retention window, days**: The number of days (24 hours) covered by backups. For example, if you specify `7`, the catalog must always contain backups required for restoring the data for the last seven days, including today.

      To disable this limitation, specify `0`. In this case, backups can be deleted from the catalog at any moment.
    - **WAL depth, pcs.**: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR). For example, if you specify `3`, the catalog must always contain at least three backups on each timeline.

      To disable this limitation, specify `0`. In this case, point-in-time recovery is not possible.
    - **Expired copies**: The management policy for expired backups. Possible values:
        - **Merge**: Merge expired backups with new ones if possible.
        - **Delete**: Delete expired backups from the catalog.

      You can enable all checkboxes simultaneously.

  The **Retention redundancy, pcs**, **Retention window, days**, and **WAL depth, pcs** are applied only if you enabled the **Merge** and/or **Delete** checkbox for the **Expired copies** parameter.

  The values of the **Retention redundancy, pcs.** and **Retention window, days** parameters are considered simultaneously when deleting expired backups from the catalog. For example, if you entered `3` in the **Retention redundancy, pcs.** field and `7` in the **Retention window, days** field, a maximum of three full backups will be saved, as well as all backups required for restoring the data for the past 7 days, including today.

  You can also configure storage parameters for an instance, as well as for a backup when creating it. The following priority is used:

    - Backup parameters are applied first.
    - Instance parameters are applied second.
    - Storage parameters are applied third.

  For more information about storage parameters, refer to the official Postgres Pro documentation on `pg_probackup`.

5. Click **Save**.

The S3 storage will be created and displayed in the table of storages.

## Viewing Storages #

To view storages, in the **Management** section of the navigation panel, go to the **Backup → Backup storages** page. To view S3 storages, select the **S3 storages** tab.

The table of storages with the following columns will be displayed:

- **Name**: The unique name of the storage. You can filter values in this column using the corresponding icon.
- **Instance**: The instance where the catalog of the local storage is located. This column is only displayed on the **Local** tab.
- **Catalog of copies**: The path to the catalog where backups are placed. This column is only displayed on the **Local** tab.
- **Parameters**:
    - **Type**: The provider of the S3 storage. Possible values:
        - `AWS`
        - `Minio`
        - `VK`
    - **Host**: The server where the commands for interactions between the manager and S3 storage are executed.
    - **Port**: The number of the port for connecting the manager to the S3 storage.
    - **Bucket**: The name of the bucket on the S3 storage server where backups are placed.
    - **Region**: The region where the S3 storage server is located.

  This column is only displayed on the **S3 storages** tab.
- **Retention redundancy, pcs.**: The maximum number of full backups in a storage.
- **Retention window, days**: The number of days (24 hours) covered by backups in the storage.
- **WAL depth, pcs.**: The minimum number of backups on each timeline in a storage. Having backups on all timelines is required for point-in-time recovery (PITR).
- **Expired copies**: The management policy for obsolete backups. Possible values:
    - `Disabled`: Do not perform any actions with expired backups.
    - `Merge`: Merge expired backups with new ones if possible.
    - **Delete**: Delete expired backups from the catalog.

  All values except `Disabled` can be displayed simultaneously.

You can perform the following actions with the table of storages:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit table**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.

- To hide or show columns, in the top-right corner of the page, click **Edit table**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-right corner of the page, click **Reset filters**.

#### Editing a Storage #

To edit a storage, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Backup storages** page. To edit an S3 storage, select the **S3 storages** tab.

   The table of storages will be displayed.

2. Click **Edit** next to the storage that you want to edit.

   The storage editing window will open.

3. Edit the required storage parameters.

4. Click **Save**.

The storage will be edited, and the updates will be displayed in the table of storages.

#### Deleting a Storage #

To delete a storage, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Backup storages** page. To delete an S3 storage, select the **S3 storages** tab.

   The table of storages will be displayed.

2. Click **Delete** next to the storage that you want to delete.

   The storage deletion window will open.

3. Select one of the following values:

   - **Remove from repository only**: Delete the storage from the repository and web application. When the agent is restarted on the server, the storage is automatically recreated in the web application. This value is selected by default.

   - **Recursive removing of all backups and storage**: Delete the storage and its catalog from the repository, web application, and server.

     **Warning**

     If you select this value, the storage cannot be restored after deletion.

4. Confirm the deletion and click **Delete **.

The storage will be deleted and no longer be displayed in the table of storages.

## Backups #

This section explains how to manage backups. It includes the following instructions:

- Creating a Backup
- Viewing Backups
- Checking the Consistency of a Backup
- Viewing the Log of a Backup
- Editing Pinning Parameters of a Backup
- Creating an Instance from a Backup
- Deleting a Backup

### Creating a Backup #

To create a backup, follow the steps below:

1. Go to backups in one of the following ways:

   - Using the backup section:

     In the **Management** section of the navigation panel, go to the **Backup** page.

   - Using the configuration section of a specific instance:

     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.

     2. Click the name of the instance for which you want to create a backup.

        The **Overview** page will be displayed.

     3. In the **Instance details** section of the navigation panel, go to the **Backup** page.

   The table of backups will be displayed.

2. In the top-right corner of the page, click **Create backup**.

   The backup creation window will open.

3. In the **General** step, enter details of the new backup (parameters marked with an asterisk are required):

   - **Instance**: The instance for which the backup will be created. The value is entered automatically when creating a backup using the table of backups of a specific instance.

   - **Copy storage**: The storage where the backup will be placed. You can select a local or S3 storage. A local storage must be located on the same server as the instance for which you are creating the backup.

   - **User** and **Password**: The name and password of the DBMS user under which the backup will be performed.

   - **Database**: The database for connecting to the instance.

   - **Backup mode**: The backup creation mode. Possible values:

     - **full**. This value is selected by default.
     - **page**.
     - **ptrack**.
     - **delta**.

For more information about backup modes, refer to the official Postgres Pro documentation on `pg_probackup`.

- **Threads count**: The number of parallel threads that will be started at backup creation.
- **Waiting time (sec)**: The waiting timeout in seconds for WAL segment archiving and streaming. Default value: `300`.
- **Create a stand-alone backup**: Whether to create a streaming backup that includes the WAL records required for restoring the instance later. This checkbox is disabled by default.
- **Replication slot**: The replication slot that will be used for transferring WAL records.
- **Create temporary replication slot**: Whether to create a temporary replication slot for transferring WAL records of the instance for which you are creating the backup. This checkbox is disabled by default. If this checkbox is enabled, WAL segments are available even if they are switched at backup creation.

4. Click **Next**, and then in the **Advanced** step, specify additional information if required:

   - **External catalogs**: The path to the instance catalog that will also be included in the backup.
   - **include log catalog**: Whether the backup includes the catalog with the instance activity logs. This checkbox is disabled by default.
   - **Don’t check copy**: Whether to skip the automatic verification of the created backup. This checkbox is disabled by default. If this checkbox is enabled, the backup is created faster.
   - **Smooth execution of the checkpoint**: Whether backup creation starts only after the scheduled checkpoint. This checkbox is disabled by default.
   - **Disable block-level verification**: Whether to disable the block-level checksum verification for faster consistency checking at backup creation. This checkbox is disabled by default.
   - **Compression level**: The file compression level at backup creation. You can enter a value from 0 to 9, where 0 — to disable the file compression, and 9 — to use the highest file compression. Default value: `0`.
   - **Compression algorithm**: The algorithm used for compressing files. This parameter is available only if you entered a value greater than 0 in the **Compression level** field. Possible values:
     - **zlib**
     - **lz4**
     - **zstd**
     - **pglz**
   - **Pinning**: The pinning parameters of the backup. Possible values:
     - **Do not pin**: Do not pin the backup. If you select this value, the parameters specified in the **Parameters of storing** section are used. This value is selected by default.
     - **ttl**: After the backup is created, it cannot be deleted from a storage during a specific number of days. For this value, in the **Retention period, days** field, enter the required number of days.
     - **expire-time**: The backup cannot be deleted from a storage until a specific date and time. For this value, in the **Retention period until** field, specify the required date and time.
   - **Storage parameters**: The backup storage parameters of the storage catalog created for the instance. Available parameters:
     - **Retention redundancy**: The maximum number of full backups. For example, if you specify `3`, the catalog can contain a maximum of three full backups.

       To disable this limitation, specify `0`. In this case, the number of backups in the catalog is not limited.
     - **Retention window**: The number of days (24 hours) covered by backups. For example, if you specify `7`, the catalog must always contain backups required for restoring the data for the last seven days, including today.

       To disable this limitation, specify `0`. In this case, backups can be deleted from the catalog at any moment.
     - **WAL depth**: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR). For example, if you specify `3`, the catalog must always contain at least three backups on each timeline.

       To disable this limitation, specify `0`. In this case, point-in-time recovery is not possible.
     - **Expired copies**: The management policy for expired backups. Possible values:
       - **Merge**: Merge expired backups with new ones if possible.
       - **Delete**: Delete expired backups from the catalog.
       - **Remove expired WAL**: Delete WAL of expired backups from the catalog.

       You can enable all checkboxes simultaneously.

     The **Retention redundancy**, **Retention window**, and **WAL depth** are applied only if you enabled the **Merge** and/or **Delete** checkbox for the **Expired copies** parameter.

     The values of the **Retention redundancy** and **Retention window** parameters are considered simultaneously when deleting obsolete backups from the catalog. For example, if you entered `3` in the **Retention redundancy** field and `7` in the **Retention window** field, a maximum of three full backups will be saved, as well as all backups required for restoring the data for the past 7 days, including today.

     You can also configure storage parameters for an instance, as well as for a storage when creating or editing it. The following priority is applied:
     - Backup parameters are applied first.
     - Instance parameters are applied second.
     - Storage parameters are applied third.

     For more information about storage parameters, refer to the official Postgres Pro documentation on `pg_probackup`.

5. Click **Execute backup**.

The backup will be created and displayed in the table of backups.

## Viewing Backups #

To view backups, go to them in one of the following ways:

- Using the backup section:

  In the **Management** section of the navigation panel, go to the **Backup** page.

- Using the configuration section of a specific instance:

  1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

     The table of instances will be displayed.

  2. Click the name of the instance whose backups you want to view.

     The **Overview** page will be displayed.

3. In the **Instance details** section of the navigation panel, go to the **Backup** page.

The table of backups with the following columns will be displayed:

- **Size**: The size of the backup. This column includes additional information:
  - **ID**: The ID of the backup in `pg_probackup`.

  You can filter and sort values in this column using the corresponding icons.
- **Mode**: The backup creation mode. Possible values:
  - `FULL`
  - `DELTA`
  - `PAGE`
  - `PTRACK`

  For more information about backup modes, refer to the official Postgres Pro documentation on `pg_probackup`.

  You can filter values in this column using the corresponding icon.
- **Storage**: The storage where the backup is placed. You can filter and sort values in this column using the corresponding icons.
- **Instance**: The instance for which the backup is created. You can filter and sort values in this column using the corresponding icons.
- **Status**: The status of the backup. Possible values:
  - `REQUESTED`: Backup creation was requested.
  - `SCHEDULED`: The backup creation was scheduled.
  - `RUNNING`: The backup is being created.
  - `DONE`: The backup was created successfully.
  - `DELETING_CANCEL`: Backup deletion was canceled.
  - `DELETING`: The backup is being deleted.
  - `VALIDATING`: The backup consistency check is being performed.
  - `VALIDATED`: The consistency check demonstrated that the backup was not damaged during storage.
  - `MERGING`: An expired backup is being merged with a new one.
  - `MERGED`: An expired backup was successfully merged with a new one.
  - `ERROR`: An error occurred during backup. Note that this status is also displayed if the consistency check showed that the backup was damaged during storage.

  You can filter values in this column using the corresponding icon.
- **Started at** and **Finished at**: The start and end date of backup creation. You can filter and sort values in this column using the corresponding icons.
- **User**: The user that created the backup. You can filter values in this column using the corresponding icon.
- **Validation status**: The status of the backup consistency check. Possible values:
  - `validated`: The consistency check of the backup was performed successfully.
  - `validating`: The backup consistency check is being performed.
- **Start of validation** and **End of validation**: The start and end date and time of the backup consistency check.
- **Actions**: Actions with the backup. For more information about the actions that you can perform with a backup, refer to other instructions in this section.

You can perform the following actions with the table of backups:

- To change the column width, drag its border with the mouse.
- To change the display order of the columns, in the top-right corner of the table, click **Edit**, and then in the menu drag the column names using the mouse. The column with the name displayed at the top of the menu is the first in the table.
- To hide or show columns, in the top-right corner of the page, click **Edit**, and then in the menu, turn off or turn on the toggles next to the column names.
- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-right corner of the page, click **Reset filters**.

### Checking the Consistency of a Backup #

The consistency check ensures that a backup was not damaged during storage.

To check the consistency of a backup, follow the steps below:

1. Go to backups in one of the following ways:
   - Using the backup section:

     In the **Management** section of the navigation panel, go to the **Backup** page.
   - Using the configuration section of a specific instance:
     1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

        The table of instances will be displayed.
     2. Click the name of the instance for which the backup is created.

        The **Overview** page will be displayed.
     3. In the **Instance details** section of the navigation panel, go to the **Backup** page.

   The table of backups will be displayed.
2. Click **Validate backup** next to the backup whose consistency you want to check.

The consistency check of the backup will start. You can view the result in the **Status** column of the table of backups:

- If the backup was damaged during storage, the `ERROR` value will be displayed.
- If the backup was not damaged during storage, the `VALIDATED` value will be displayed.

### Viewing the Log of a Backup #

The backup log provides information about the backup creation progress.

To view the log of a backup, follow the steps below:

1. Go to backups in one of the following ways:
   - Using the backup section:

     In the **Management** section of the navigation panel, go to the **Backup** page.

    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

           The table of instances will be displayed.
        2. Click the name of the instance for which the backup is created.

           The **Overview** page will be displayed.
        3. In the **Instance details** section of the navigation panel, go to the **Backup** page.

   The table of backups will be displayed.
2. Click **Backup log** next to the backup whose log you want to view.

The backup log will open.

### Editing Pinning Parameters of a Backup #

To edit pinning parameters of a backup, follow the steps below:

1. Go to backups in one of the following ways:
    - Using the backup section:

      In the **Management** section of the navigation panel, go to the **Backup** page.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

           The table of instances will be displayed.
        2. Click the name of the instance for which the backup is created.

           The **Overview** page will be displayed.
        3. In the **Instance details** section of the navigation panel, go to the **Backup** page.

   The table of backups will be displayed.
2. Click the three vertical dots icon ⋮ → **Edit pinning** next to the backup whose pinning parameters you want to edit.

   The window for editing backup pinning parameters will open.
3. Select one of the following values:
    - **Do not pin**: Apply backup storage parameters. This value is selected by default.
    - **ttl**: After the backup storage parameters are edited, the backup cannot be deleted for a specific number of days. For this value, in the **Retention period, days** field, enter the required number of days.
    - **expire-time**: The backup cannot be deleted until a specific date and time. For this value, in the **Retention period until** field, specify the required date and time.
4. Click **Save**.

Pinning parameters of the backup will be edited, and the updates will be displayed in the table of backups.

### Creating an Instance from a Backup #

Before performing this procedure, create a backup.

To create an instance from a backup, follow the steps below:

1. Go to backups in one of the following ways:
    - Using the backup section:

      In the **Management** section of the navigation panel, go to the **Backup** page.
    - Using the configuration section of a specific instance:
        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

           The table of instances will be displayed.
        2. Click the name of the instance for which the backup is created.

           The **Overview** page will be displayed.
        3. In the **Instance details** section of the navigation panel, go to the **Backup** page.

   The table of backups will be displayed.
2. Click the three vertical dots icon ⋮ → **Restore** next to the backup from which you want to restore an instance.

   The instance creation window will open.
3. Enter details of the new instance (parameters marked with an asterisk are required):
    - **Name**: The unique name of the instance.
    - **Server**: The server on which the instance is installed.
    - **System user**: The OS user who will own the instance files and catalogs and on behalf of which the instance service will run. In most cases, it is the `postgres` user. It is recommended to ensure that the specified user exists in the OS.
    - **Main data directory**: The path to the server catalog where the main instance catalogs and files will be placed.
    - **Connection address** and **Connection port**: The network address and port number that the instance will use for receiving client connections. The default network address is `localhost`.
    - **Tags**: The tags that will be assigned to the instance. You can unassign a tag by clicking the cross icon × next to its name.
    - **Backup**: The backup from which the instance will be created. The value is entered automatically.
    - **Backup size**: The size of the backup from which the instance will be created. The value is entered automatically.
    - **Restore point**: The state to which the instance must be restored. Possible values:
        - **—**: Restore the instance to the last state covered by the backup. This value is selected by default.

        - **Time**: Restore the instance state to a specific date and time covered by the backup. For this value, in the **Time** field, specify the required date and time.
        - **LSN**: Restore the instance to the state corresponding to a specific WAL LSN. For this value, in the **LSN** field, enter the required WAL LSN .
        - **Transaction**: Restore the instance state to a specific transaction number. For this value, in the **Transaction** field, enter the transaction number.

      For the **Time**, **LSN**, and **Transaction** values, you must also specify the following parameters:

        - **Restore including the specified value**: Whether the instance state is restored including the specified value. This checkbox is enabled by default.

          For example, if you entered `123456` in the **Transaction** field and enabled the **Restore including the specified value** checkbox, the instance state will be restored to the transaction 123456. If you did not enable the checkbox, the instance state will be restored to the 123455 transaction.

        - **Action after restore**: The action to perform on the server after restoring the instance to the required state. Possible values:

            - **Pause after restore**: Pause the creation of the instance from the backup. It allows ensuring that the correct state was restored for the instance before creating it. This value is selected by default.
            - **Promote after restore**: Create the instance from the backup and start receiving client connections.
            - **Shutdown instance after restore**: Create the instance from the backup, and then stop the server.

    - **Partial recovery**: Specifies the instance databases that will be restored or excluded from the restoration process. Possible values:

        - **Do not use**: Restore all instance databases. This value is selected by default.
        - **Exclude some databases**: Exclude specific databases from restoration.
        - **Restore some databases**: Restore the specific instance databases.

      For the **Exclude some databases** and **Restore some database** values, specify the unique name of the database using the **Databases** parameter, and then click **Add database**. You can add multiple names of databases. A database name can be removed by clicking the bin icon icon-delete next to it.

    - **Checking available space**: Allows checking whether there is enough disk space on the server for creating the instance from the backup. To start the check, click **Check**.

4. Click **Execute**.

The instance will be created from the backup and displayed in the table of instances.

## Deleting a Backup #

**Warning**

Deleted backups cannot be restored after deletion.

To delete a backup, follow the steps below:

1. Go to backups in one of the following ways:

    - Using the backup section:

      In the **Management** section of the navigation panel, go to the **Backup** page.

    - Using the configuration section of a specific instance:

        1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

           The table of instances will be displayed.

        2. Click the name of the instance for which the backup is created.

           The **Overview** page will be displayed.

        3. In the **Instance details** section of the navigation panel, go to the **Backup** page.

   The table of backups will be displayed.

2. Click the three vertical dots icon ⋮ → **Delete** next to the backup that you want to delete.

   The backup deletion window will open.

3. Click **Delete*.

The backup will be deleted and no longer be displayed in the table of backups.

## Backup Schedules #

This section explains how to manage backup schedules. It includes the following instructions:

- Creating a Backup Schedule
- Viewing Backup Schedules
- Executing a Backup Schedule
- Deactivating and Activating a Backup Schedule
- Deleting a Backup Schedule

### Creating a Backup Schedule #

To create a backup schedule, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Schedule** page.

   The table of backup schedules will be displayed.

2. In the top-right corner of the page, click **Create task**.

   The backup schedule creation window will open.

3. In the **General** step, enter details of the new backup schedule (parameters marked with an asterisk are required):

    - **Name**: The unique name of the backup schedule.

    - **Set cron-string execution**: Allows specifying the time interval for creating backups in the `crontab` format. This toggle is turned off by default. If it is turned on, in the **Execution** field, enter a string in the `crontab` format.

    - **Task planning**: The type of the backup schedule. This parameter is available only if you turned off the **Set cron-string execution** toggle. Possible values:

        - **Time-delayed**: One backup will be created at a specific date and time. For this value, in the **Time** field, specify the required date and time.

        - **On schedule**: Backups will be created within a specific time interval. For this value, specify the following parameters:

            - **Interval**: The unit of measurement of the time interval. Possible values:

                - **Minutes**. This value is selected by default.
                - **Hours**.

            - **Days**.
        - **Repeat every**: The time interval for creating backups by minutes or hours. This parameter is available only if you selected **Minutes** or **Hours** for the **Task planning** parameter.
        - **Execution days**: Specifies the days for creating backups.
    - **Start** and **Repeat until**: The start and end date and time for creating backups on schedule. These parameters are available only if you turned on the **Set cron-string execution** toggle or selected **On schedule** for the **Task planning** parameter.
    - **Cron total line**: The string in the `crontab` format that specifies the time interval for creating backups. The value is entered automatically. This parameter is available only if you selected **On schedule** for the **Task planning** parameter.
    - **Instance**: The instance for which the backup will be created.
    - **Copy storage**: The storage where the backup will be placed. You can select a local or S3 storage. A local storage must be located on the same server as the instance for which you are creating the backup.
    - **User** and **Password**: The name and password of the DBMS user under which the backup will be performed.
    - **Database**: The database for connecting to the instance.
    - **Backup mode**: The backup creation mode. Possible values:
        - **full**. This value is selected by default.
        - **page**.
        - **ptrack**.
        - **delta**.

      For more information about backup modes, refer to the official Postgres Pro documentation on `pg_probackup`.
    - **Threads count**: The number of parallel threads that will be started at backup creation.
    - **Waiting time (sec)**: The waiting timeout in seconds for WAL segment archiving and streaming. Default value: `300`.
    - **Create a stand-alone backup**: Whether to create a streaming backup that includes the WAL records required for restoring the instance later. This checkbox is disabled by default.
    - **Replication slot**: The replication slot that will be used for transferring WAL records.
    - **Create temporary replication slot**: Whether to create a temporary replication slot for transferring WAL records of the instance for which you are creating the backup. This checkbox is disabled by default. If this checkbox is enabled, WAL segments are available even if they are switched at backup creation.
4. Click **Next**, and then in the **Advanced** step, specify additional information if required:
    - **External catalogs**: The path to the instance catalog that will also be included in the backup.
    - **include log catalog**: Whether the backup includes the catalog with the instance activity logs. This checkbox is disabled by default.
    - **Don’t check copy**: Whether to skip the automatic verification of the created backup. This checkbox is disabled by default. If this checkbox is enabled, the backup is created faster.
    - **Smooth execution of the checkpoint**: Whether backup creation starts only after the scheduled checkpoint. This checkbox is disabled by default.
    - **Disable block-level verification**: Whether to disable the block-level checksum verification for faster consistency checking at backup creation. This checkbox is disabled by default.
    - **Compression level**: The file compression level at backup creation. You can enter a value from 0 to 9, where 0 — to disable the file compression, and 9 — to use the highest file compression. Default value: `0`.
    - **Compression algorithm**: The algorithm used for compressing files. This parameter is available only if you entered a value greater than 0 in the **Compression level** field. Possible values:
        - **zlib**
        - **lz4**
        - **zstd**
        - **pglz**
    - **Pinning**: The pinning parameters of the backup. Possible values:
        - **Do not pin**: Do not pin the backup. If you select this value, the parameters specified in the **Parameters of storing** section are used. This value is selected by default.
        - **ttl**: After the backup is created, it cannot be deleted from a storage during a specific number of days. For this value, in the **Retention period, days** field, enter the required number of days.
        - **expire-time**: The backup cannot be deleted from a storage until a specific date and time. For this value, in the **Retention period until** field, specify the required date and time.
    - **Storage parameters**: The backup storage parameters of the storage catalog created for the instance. Available parameters:
        - **Retention redundancy**: The maximum number of full backups. For example, if you specify `3`, the catalog can contain a maximum of three full backups.

          To disable this limitation, specify `0`. In this case, the number of backups in the catalog is not limited.
        - **Retention window**: The number of days (24 hours) covered by backups. For example, if you specify `7`, the catalog must always contain backups required for restoring the data for the last seven days, including today.

          To disable this limitation, specify `0`. In this case, backups can be deleted from the catalog at any moment.
        - **WAL depth**: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR). For example, if you specify `3`, the catalog must always contain at least three backups on each timeline.

          To disable this limitation, specify `0`. In this case, point-in-time recovery is not possible.
        - **Expired copies**: The management policy for expired backups. Possible values:
            - **Merge**: Merge expired backups with new ones if possible.
            - **Delete**: Delete expired backups from the catalog.
            - **Remove expired WAL**: Delete WAL of expired backups from the catalog.

          You can enable all checkboxes simultaneously.

      The **Retention redundancy**, **Retention window**, and **WAL depth** are applied only if you enabled the **Merge** and/or **Delete** checkbox for the **Expired copies** parameter.

      The values of the **Retention redundancy** and **Retention window** parameters are considered simultaneously when deleting obsolete backups from the catalog. For example, if you entered `3` in the **Retention redundancy** field and `7` in the **Retention window** field, a maximum of three full backups will be saved, as well as all backups required for restoring the data for the past 7 days, including today.

      You can also configure storage parameters for an instance, as well as for a storage when creating or editing it. The following priority is applied:

    - Backup parameters are applied first.
    - Instance parameters are applied second.
    - Storage parameters are applied third.

   For more information about storage parameters, refer to the official Postgres Pro documentation on `pg_probackup`.

5. Click **Save**.

The backup schedule will be created and displayed in the table of backup schedules.

#### Viewing Backup Schedules #

To view backup schedules, in the **Management** section of the navigation panel, go to the **Backup** → **Schedule** page.

The table of backup schedules with the following columns will be displayed:

- **Task**: The unique name of the backup schedule. You can sort values in this column using the corresponding icon.
- **Agent**: The agent that creates backups. This column includes additional information:
    - **Instance**: The instance for which backups will be created.

  You can filter values in this column using the corresponding icon.
- **Storage**: The storage where backups are placed. You can filter values in this column using the corresponding icon.
- **Compression**: The file compression level at backup creation. You can enter a value from 0 to 9, where 0 — to disable the file compression, and 9 — to use the highest file compression.
- **Last execution**: The date and time of the last backup creation.
- **Schedule**: The string in the `crontab` format that specifies the time interval for creating backups.
- **User**: The user that created the backup schedule. You can filter values in this column using the corresponding icon.

You can perform the following actions with the table of backup schedules:

- To update the table, in the top-right corner of the page, click **Refresh data** .
- To reset all filters, in the top-right corner of the page, click **Reset filters**.

### Executing a Backup Schedule #

You can manually execute a backup schedule to instantly start the backup creation.

To execute a backup schedule, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Schedule** page.

   The table of backup schedules will be displayed.
2. Click the three vertical dots icon → **Run** next to the backup schedule that you want to execute.

The backup schedule execution will start. The backup will be created and displayed in the table of backups.

### Deactivating and Activating a Backup Schedule #

You can deactivate a backup schedule to temporarily stop backup creation. Backup schedules are activated by default.

To deactivate or activate a backup schedule, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Schedule** page.

   The table of backup schedules will be displayed.
2. Click the three vertical dots icon → **Deactivate** or **Activate** next to the backup schedule that you want to deactivate or activate.

#### Deleting a Backup Schedule #

**Warning**

Deleted backup schedules cannot be restored.

When you delete a backup schedule, the associated backups are not deleted.

To delete a backup schedule, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Backup** → **Schedule** page.

   The table of backup schedules will be displayed.
2. Click the three vertical dots icon → **Delete** next to the backup schedule that you want to delete.

   The backup schedule deletion window will open.
3. Click **Delete*.

The backup schedule will be deleted and no longer be displayed in the table of backup schedules.

### Configuring Backup Storage Parameters for an Instance #

To configure backup storage parameters for an instance, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Infrastructure** → **Instances** page.

   The table of instances will be displayed.
2. Click the name of the instance where you want to configure backup storage parameters.

   The **Overview** page will be displayed.
3. In the **Instance details** section of the navigation panel, go to the **Backup** → **Retention policy** page.

   The table of storages will be displayed.
4. Click the pencil icon next to the storage.

   The backup storage parameters editing window will open.

5. Specify the backup storage parameters of the storage catalog created for the instance:
    - **Retention redundancy, pcs.**: The maximum number of full backups. For example, if you specify `3`, the catalog can contain a maximum of three full backups.

      To disable this limitation, specify `0`. In this case, the number of backups in the catalog is not limited.
    - **Retention window, days**: The number of days (24 hours) covered by backups. For example, if you specify `7`, the catalog must always contain backups required for restoring the data for the last seven days, including today.

      To disable this limitation, specify `0`. In this case, backups can be deleted from the catalog at any moment.
    - **WAL depth**: The minimum number of backups on each timeline. Having backups on all timelines is required for point-in-time recovery (PITR). For example, if you specify `3`, the catalog must always contain at least three backups on each timeline.

      To disable this limitation, specify `0`. In this case, point-in-time recovery is not possible.
    - **Expired copies**: The management policy for obsolete backups. Possible values:
        - **Merge**: Merge expired backups with new ones if possible.
        - **Delete**: Delete expired backups from the catalog.

      You can enable all checkboxes simultaneously.

   The **Retention redundancy, pcs**, **Retention window, days**, and **WAL depth, pcs** are applied only if you enabled the **Merge** and/or **Delete** checkbox for the **Expired copies** parameter.

   The values of the **Retention redundancy, pcs.** and **Retention window, days** parameters are considered simultaneously when deleting expired backups from the catalog. For example, if you entered `3` for the **Retention redundancy, pcs.** parameter and `7` for the **Retention window, days** parameter, a maximum of three full backups will be saved, as well as all backups required for restoring the data for the past 7 days, including today.

   You can also configure storage parameters for a backup when creating it, as well as for a storage when creating or editing it. The following priority is used:
    - Backup parameters are applied first.
    - Instance parameters are applied second.
    - Storage parameters are applied third.

   For more information about storage parameters, refer to the official Postgres Pro documentation on `pg_probackup`.
6. Click **Save**.

### Query Plan Visualization #

PPEM supports visualization of query plans that can be retrieved using the `EXPLAIN` command. For more information about the command, refer to the official Postgres Pro documentation.

The visualization contains a tree of query plan nodes and allows you to navigate between them. Each node has a unique name and corresponds to a specific stage of query execution. Clicking on a node name displays expanded information about this node.

This section explains how to manage query plan visualizations. It includes the following instructions:

- Creating a Query Plan Visualization
- Viewing a Query Plan Visualization
- Editing a Query Plan Visualization
- Deleting a Query Plan Visualization

#### Creating a Query Plan Visualization #

Before performing this procedure:

1. Get the query plan:

   ```
   EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON)
   ```
2. Copy the query plan and save it.

To create a query plan visualization, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Visualization** page.

   Parameters of the query plan visualization will be displayed.
2. Enter details of the new query plan visualization (parameters marked with an asterisk are required):
    - **Name**: The unique name of the query plan visualization. If you do not specify the name, it is generated automatically.
    - **Query plan**: The query plan that will be visualized.
    - **Query**: The text of the query whose plan will be visualized. If you specify query text, it is added to the query plan visualization.

   You can create a visualization based on a query plan example. To do this, in the **Examples** block located in the top-right corner of the page, click the pencil icon next to the required example. In this case, parameter values of the query plan visualization are filled in automatically.

   You can clear the parameter values of the query plan visualization by clicking **Clear**.
3. Click **Visualize**.

The query plan visualization will be created, displayed on the page, and appear in the **Saved plans** block.

#### Viewing a Query Plan Visualization #

To view a query plan visualization, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Visualization** page.

   Parameters of the query plan visualization will be displayed.
2. In the **Saved plans** block, click the name of the query plan visualization that you want to view.

The query plan visualization will be displayed on the page.

#### Editing a Query Plan Visualization #

To edit a query plan visualization, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Visualization** page.

   Parameters of the query plan visualization will be displayed.
2. In the **Saved plans** block, click the pencil icon next to the query plan visualization that you want to edit.

3. Change the parameters of the query plan visualization.
4. Click **Visualize**.

The query plan visualization will be edited and displayed on the page.

### Deleting a Query Plan Visualization #

> **Warning**
>
> Deleted query plan visualizations cannot be restored.

To delete a query plan visualization, follow the steps below:

1. In the **Management** section of the navigation panel, go to the **Monitoring** → **Visualization** page.

   Parameters of the query plan visualization will be displayed.

2. In the **Saved plans** block, click the bin icon 🗑 next to the query plan visualization that you want to delete.

The query plan visualization will be deleted and no longer be displayed in the **Saved plans** block.

## Upgrade and Migration #

It is recommended to upgrade PPEM frequently to ensure access to recent fixes and improvements. You can upgrade to any supported version of any supported release starting with version 2.0.

Information about PPEM version upgrades can be found in the What is New section.

For more information, refer to the upgrade guides:

- Upgrading to PPEM 2.2
- Upgrading to PPEM 2.1
- Upgrading to PPEM 2.0

### Upgrading to PPEM 2.2 #

To upgrade PPEM to version 2.2, you must upgrade the manager and agents. Before upgrading, it is recommended to see backwards compatibility features.

### Considerations of Backward Compatibility in PPEM 2.2 #

In the PPEM 2.2, packaged API commands can now be sent by the manager. These are typically API commands required for executing various operations. If the manager and agents operate on different PPEM versions, you cannot create instances and perform any actions with databases and their objects, such as schemas.

To avoid backward compatibility issues, select the date and time when no operations with databases and their objects are scheduled, and then upgrade both the manager and agents to version 2.2.

### Upgrading the Manager to Version 2.2 #

> **Note**
>
> Before upgrading the manager, wait for all routine tasks run in PPEM to complete (see active tasks in the Task Console).

The manager upgrade process includes the following steps:

1. Upgrade the repository and package manager metadata. The upgrade procedure depends on the version of the used distributive and package manager.
2. Stop the manager 2.1 service using the `systemctl stop ppem` command.
3. Install the manager 2.2. The installation procedure is performed using the package manager.
4. Upgrade the configuration files in the `/etc` catalog. Place the parameters into a new configuration file if required.
5. Start the manager 2.2 service using the `systemctl start ppem` command.
6. Ensure that the manager service started successfully and is available using the `systemctl status ppem` command.

### Upgrading Agents to Version 2.2 #

The agent upgrade process includes the following steps:

1. Upgrade the repository and package manager metadata. The upgrade procedure depends on the version of the used distributive and package manager.
2. Stop the agent 2.1 service using the `systemctl stop ppem-agent` command.
3. Install the version 2.2 agent. The installation procedure is performed using the package manager.
4. Upgrade the configuration files in the `/etc` catalog. Place the parameters into a new configuration file if required.
5. Start the agent 2.2 service using the `systemctl start ppem-agent` command.
6. Ensure that the agent service started successfully using the `systemctl status ppem-agent` command.

### Upgrading to PPEM 2.1 #

To upgrade PPEM to version 2.1, you must upgrade the manager and agents.

### Upgrading the Manager to Version 2.1 #

> **Note**
>
> Before upgrading the manager, wait for all routine tasks run in PPEM to complete (see active tasks in the Task Console).

The manager upgrade process includes the following steps:

1. Upgrade the repository and package manager metadata. The upgrade procedure depends on the version of the used distributive and package manager.
2. Stop the manager 2.0 service using the `systemctl stop ppem` command.
3. Install the manager 2.1. The installation procedure is performed using the package manager.
4. Upgrade the configuration files in the `/etc` catalog. Place the parameters into a new configuration file if required.
5. Start the manager 2.1 service using the `systemctl start ppem` command.
6. Ensure that the manager service started successfully and is available using the `systemctl status ppem` command.

### Upgrading Agents to Version 2.1 #

The agent upgrade process includes the following steps:

1. Upgrade the repository and package manager metadata. The upgrade procedure depends on the version of the used distributive and package manager.
2. Stop the agent 2.0 service using the `systemctl stop ppem-agent` command.
3. Install the agent 2.1. The installation procedure is performed using the package manager.
4. Upgrade the configuration files in the `/etc` catalog. Place the parameters into a new configuration file if required.
5. Start the agent 2.1 service using the `systemctl start ppem-agent` command.
6. Ensure that the agent service started successfully using the `systemctl status ppem-agent` command.

## Upgrading to PPEM 2.0 #

This section contains recommendations and instructions for migrating to PPEM 2.0.

> **Note**
>
> Version 2.0 is not backward compatible with version 1.0. No tools for smooth data migration are currently available.

### Migration Recommendations #

When migrating to PPEM 2.0, note the following changes implemented in this version:

- The manager and agent components are rewritten in Golang and implement the new API version. For this reason, some PPEM 2.0 components are incompatible with version 1.0 components:
  - the manager version 2.0 and agents version 1.0 are mutually incompatible; -The web application version 2.0 and the manager version 1.0 are mutually incompatible;
  - configuration files of the manager version 1.0 and the manager version 2.0 are mutually incompatible;
  - configuration files of agents version 1.0 and agents version 2.0 are mutually incompatible.
- Agents version 2.0 no longer regularly collect metrics and logs of DBMS instances. PPEM now uses `pgpro-otel-collector` to work with metrics and logs.

Therefore, the following migration recommendations for PPEM 2.0 are provided:

- It is recommended to install PPEM 2.0 on hardware separate from that used for PPEM 1.0. It is recommended to completely shut down all PPEM 1.0 components (including the manager and agents) before installing and starting the PPEM 2.0 manager and agents.

  > **Warning**
  >
  > Performing the same operations on the same DBMS instances simultaneously in two PPEM versions may lead to unpredictable consequences. Avoid using two PPEM versions simultaneously. Simultaneous operation of both versions was not tested and is not guaranteed.

- To use advanced DBMS instance health monitoring features in PPEM, it is recommended to install and configure pgpro-otel-collector. For more information, refer to the pgpro-otel-collector official documentation and the Integration with PPEM section.

### Migration Procedure #

Recommended procedure of migration to version 2.0:

1. Completely shut down all PPEM 1.0 components.
2. On a separate server, install the manager version 2.0.

   The manager 2.0 repository database can be placed to the same DBMS instance as the manager 1.0 repository database.

   > **Warning**
   >
   > Do not use the same repository database for both versions, otherwise the manager version 2.0 will not start.

3. To verify that the manager version 2.0 is successfully installed, log into the web application and get an API key for configuring the agent.
4. Install an agent version 2.0.
5. On all DBMS instance servers, delete agents first, and then delete the manager version 1.0.

# Troubleshooting #

This section explains how to troubleshoot issues.

- In case of issues, check the state of services.
- For more information, check message logs.
- If the log information is insufficient, change the verbosity of the log and repeat the search.

### Checking the State of Services #

The manager and agent services are controlled by the `systemd` system manager:

- `ppem` — the manager service.
- `ppem-agent` — the agent service.

Use the `systemctl` utility to view the state of services:

```
# systemctl status ppem
● ppem.service - PostgresPro Enterprise Manager
     Loaded: loaded (/lib/systemd/system/ppem.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-11-16 15:43:01 MSK; 48min ago
   Main PID: 53582 (ppem-manager)
      Tasks: 7 (limit: 3512)
     Memory: 226.9M
        CPU: 8.445s
     CGroup: /system.slice/ppem.service
             └─53582 /usr/sbin/ppem-manager -config /etc/ppem-manager.yml
```

The status output contains the following:

- `ppem.service` — the name and description of the service. The dot indicator ("●") uses colors to highlight the service state.

  White — the neutral "inactive" or "deactivating" state.

  Red — the "failed" or "error" state.

  Green — the normal "active", "reloading" or "activating" state.

- `Loaded` — the service configuration state indicating that the configuration is uploaded to memory and that this is the normal state.
- `Active` — the service running state, indicating that the service has successfully started and is currently operational. The start time and running duration are also displayed.
- `Main PID` — The ID and name of the main process in the list of OS processes.
- `Tasks` — the total number of processes and streams generated by the main process.
- `Memory` and `CPU` — the usage of system resources, memory, and CPU time.
- `CGroup` — the control group where service processes are placed.
- After the main section, the last service messages will also be displayed.

When operating correctly, the manager and agent services are in the `Active` state.

### Checking Message Logs #

The manager and agent services can send each other service messages during operation. By default, the manager and agent use the standard output (stdout) for sending service logs and messages. Messages are captured by the systemd tools and can be viewed using `journalctl`:

```
# journalctl -u ppem
...
```

You can use the `-f` option to output incoming messages:

```
# journalctl -fu ppem
...
```

### Configuring Verbosity for the Log #

Logging can have different verbosity:

- `error` — logging only errors.
- `warning` — logging errors and warnings.
- `info` — logging errors, warnings, and information messages. This level is used by default.
- `debug` — logging errors, warnings, information messages, and debug messages.

You can configure logging in the `/etc/ppem-manager.yml` manager and `/etc/ppem-agent.yml` agent configuration files using the `log.level` configuration parameter. Logging changes take effect after the service restart.

## Terms and Abbreviations #

### Key Terms #

**Web application** — a PPEM graphical user interface that supports in-browser control and management of resources and DBMS infrastructure.

**Manager** — a PPEM component that coordinates agent operation and serves requests from PPEM users.

**Agent** — a PPEM service component that receives manager instructions and manages DBMS instances.

### Resources and Objects #

**Host** — a physical server (or virtual machine) running an operating system. Hosts may be the place of deployment for agents, DBMS instances, and some data source types.

**DBMS instance** — the Postgres Pro DBMS operating on a server or virtual machine.

**Instance objects** — separate objects in the DBMS, such as tablespaces, databases, schemas, tables, indexes, etc.

**Instance parameters** — DBMS instance parameters, which are usually specified in the `postgresql.conf` main configuration file.

**Instance service** — a method of starting a DBMS instance, typically `systemd` though `pg_ctl` may also be used (this method is deprecated and not recommended).

**Data source** — a connectable source for storing resources of a certain type. PPEM classifies data sources by their deployment type (e.g., internal and external) and by the type of stored resources (e.g., metrics, logs, and backups).

### Access Management #

**Privilege** — a right to perform an action with a resource in PPEM.

**PPEM role** — a privilege group in PPEM.

**PPEM user** — an account associated with a user in PPEM.

**Group** — a user group in PPEM.