Extending Rucio with modern cloud storage support

Mario.Lassnig@cern.ch

with content from *Martin Barisits, Bob Barnsley, Fernando Barreiro Megino, Johannes Elmsheuser, Mihai Patrascoiu, James Perry, Cedric Serfon, Alba Vendrell Moya, Tobias Wegner*

CHEP'23

8 - 12 May 2023, Norfolk VA

https://indico.jlab.org/event/459/

Cloud storage?

In recent years there has been **significant work** done **integrating Rucio with cloud storage**

Two major angles to consider when discussing clouds

Technical	Access tools, transfer protocols, monitoring, authn/z, accounting, billing, storage, …
Organisational	Deployed on-site or off-site Centralised or distributed Open or closed source software Public (institute, laboratory, …) or commercial In-kind contribution or paid service

It can get **complicated quickly,** e.g. …

Self-hosted MinIO S3 server on a CERN data centre VM using a centrally managed CephFS volume
WebDAV portal to self-hosted Nextcloud on a commercial hoster which points to free-tier AWS S3 storage
Experiment collaborates with commercial cloud provider and gets free storage with S3v4 protocol support

From a Rucio point of view, cloud storage is **storage that requires URL-based signatures**

Putting CephFS on top of RADOS -> requires some sort of storage system on top (*grid-style* storage)
Putting Ceph Object Gateway S3 API on top of RADOS -> cloud storage

Rucio credential mechanism

For namespace (*listing replicas*) and storage operations (*rucio upload/download*)

Generate URL signatures **at the time of execution** of the command

URL signatures are **generated server-side** by the Rucio server

No deployment of secrets necessary to clients

The account must have **schema permission** (`perm_get_signed_url`) and **account attribute** (`sign_url`)

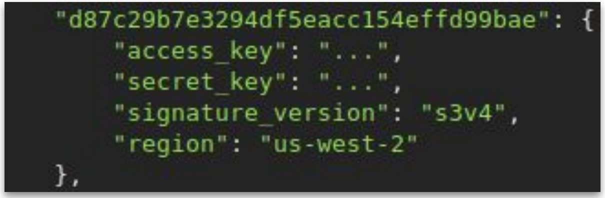
The Rucio Storage Element (RSE) must have several configurations applied

```
scheme      https
impl        rucio.rse.protocols.gfal.NoRename
attributes  sign_url: s3 | gcs | swift     verify_checksum: False        s3_url_style: path
            skip_upload_stat: True         strict_copy: True
```

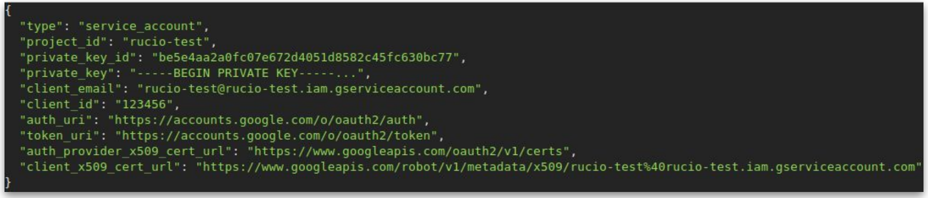
Credential secrets configuration

For S3 and SWIFT compatible interfaces (e.g. MinIO, Amazon, Ceph S3 Gateway), requires an entry in `rse-account.cfg`

For Google Cloud Storage requires the JSON credential file from Google Cloud Console



```
  "d87c29b7e3294df5eacc154effd99bae": {
      "access_key": "...",
      "secret_key": "...",
      "signature_version": "s3v4",
      "region": "us-west-2"
  },
```

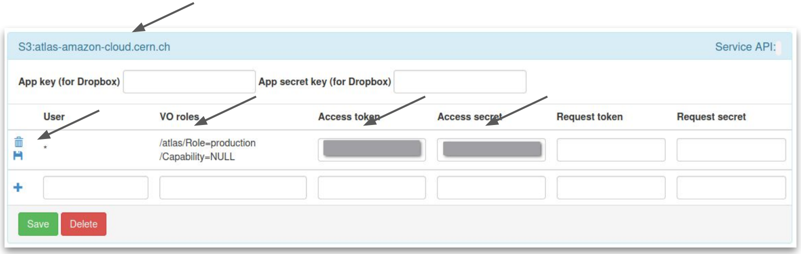


```
{
  "type": "service_account",
  "project_id": "rucio-test",
  "private_key_id": "be5e4aa2a0fc07e672d4051d8582c45fc630bc77",
  "private_key": "-----BEGIN PRIVATE KEY-----...",
  "client_email": "rucio-test@rucio-test.iam.gserviceaccount.com",
  "client_id": "123456",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://accounts.google.com/o/oauth2/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/rucio-test%40rucio-test.iam.gserviceaccount.com"
}
```

FTS credential mechanism

When adding rules for third-party-copy, the URL signatures are generated by FTS when needed

- We don't know how long transfer jobs will be in the **queue of FTS**
- URL signatures are **time-limited**
- **No universal TPC method** for **cloud storage to cloud storage**



S3:atlas-amazon-cloud.cern.ch

Service API:

App key (for Dropbox)

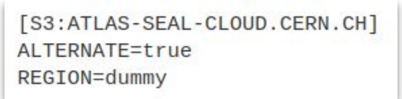
App secret key (for Dropbox)

User	VO roles	Access token	Access secret	Request token	Request secret
*	/atlas/Role=production /Capability=NULL	[redacted]	[redacted]		

Credentials need to be inserted in FTS configuration

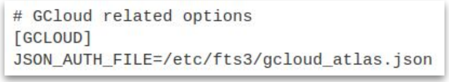
Secrets *:8446/config/cloud_storage*
Insert entry in specific format

GFAL Configuration *:8449/fts3/ftsmon/#/config/gfal2*
Cannot be edited directly, has to be set by FTS admin



```
[S3:ATLAS-SEAL-CLOUD.CERN.CH]
ALTERNATE=true
REGION=dummy
```

HTTP Configuration *:8449/fts3/ftsmon/#/config/http_plugin.so*
Cannot be edited directly, has to be set by FTS admin

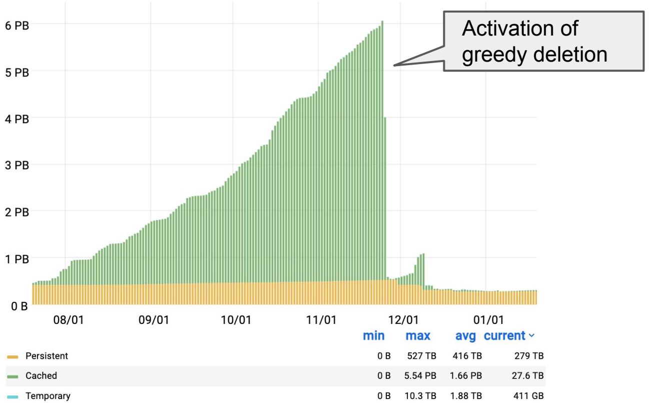


```
# GCloud related options
[GCLOUD]
JSON_AUTH_FILE=/etc/fts3/gcloud_atlas.json
```

Commercial clouds :: Google

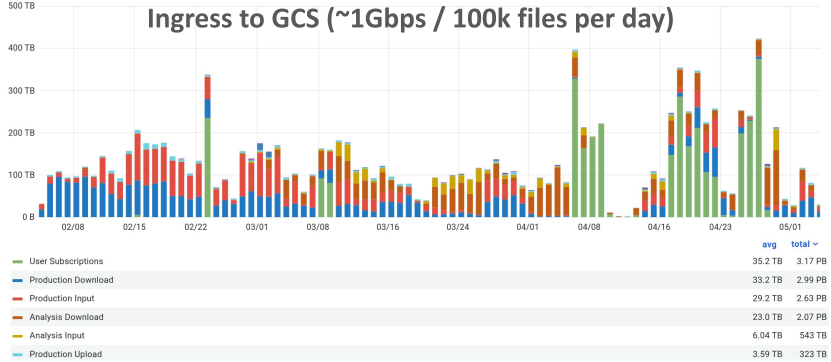
Google Cloud Storage

- Long-term ATLAS R&D project to evaluate a grid site in the cloud
  - Shoehorning X.509 certificates into commercial clouds
  - Friendly administrators at sites were required
  - CERN-provided certificate injected into new Google loadbalancer
- Custom proxy rules to accommodate our typical Tier-1 storage
  - Didn't properly work out, had to return to legacy Google loadbalancer
  - Running stable since then with jobs on Google Compute Engine
- Space occupancy model moved to greedy deletion

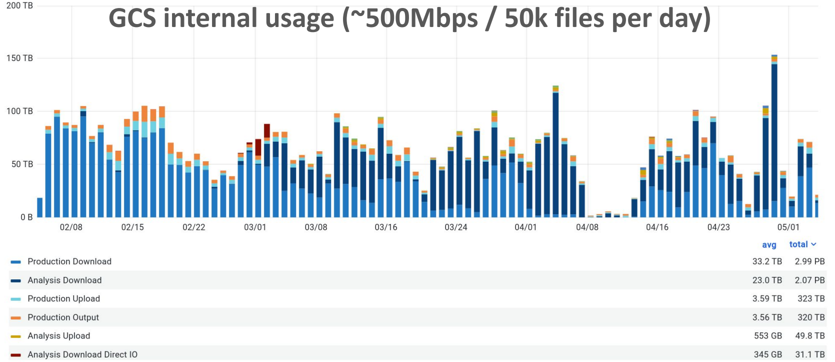


Activation of greedy deletion

	min	max	avg	current ⌄
Persistent	0 B	527 TB	416 TB	279 TB
Cached	0 B	5.54 PB	1.66 PB	27.6 TB
Temporary	0 B	10.3 TB	1.88 TB	411 GB


Ingress to GCS (~1Gbps / 100k files per day)

	avg	total ⌄
User Subscriptions	35.2 TB	3.17 PB
Production Download	33.2 TB	2.99 PB
Production Input	29.2 TB	2.63 PB
Analysis Download	23.0 TB	2.07 PB
Analysis Input	6.04 TB	543 TB
Production Upload	3.59 TB	323 TB


GCS internal usage (~500Mbps / 50k files per day)

	avg	total ⌄
Production Download	33.2 TB	2.99 PB
Analysis Download	23.0 TB	2.07 PB
Production Upload	3.59 TB	323 TB
Production Output	3.56 TB	320 TB
Analysis Upload	553 GB	49.8 TB
Analysis Download Direct IO	345 GB	31.1 TB

Commercial clouds :: SEAL

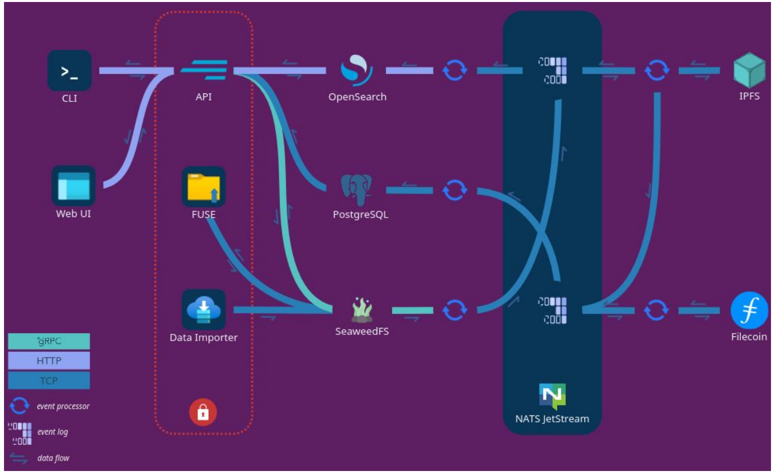
SEAL Storage Technology

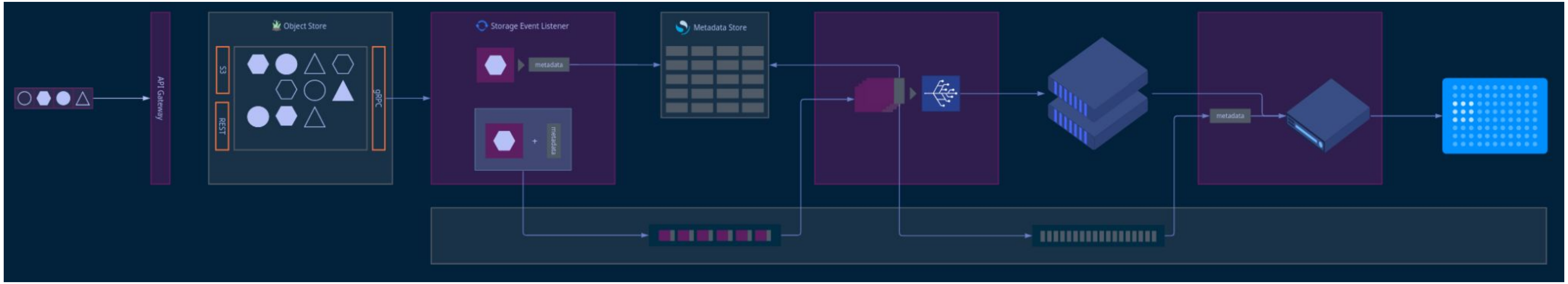
- Distributed cloud storage backed by Interplanetary File system (IPFS) and Filecoin (FIS)
- Offered 10PB of storage to ATLAS for a long-term R&D project
- Sealing process of data for long-term archival and safe-keeping

Rucio Integration

- Very smooth integration with standard URL signature mechanism
- Same trick used: SEAL administrators injected CERN-provided certificate in their loadbalancer
- Gradual selection and transfer of datasets

SEAL is funding a full-time development position to improve cloud support in Rucio





Commercial clouds :: SEAL

SEAL Storage Technology

Distributed cloud storage backed by Interplanetary File system (IPFS) and Filecoin (FIS)

Offered 10PB of storage to ATLAS for a long-term R&D project

Sealing process of data for long-term archival and safe-keeping

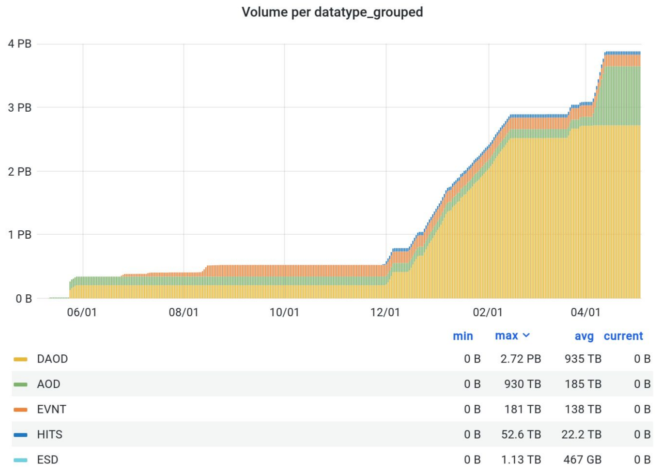
Rucio Integration

Very smooth integration with standard URL signature mechanism

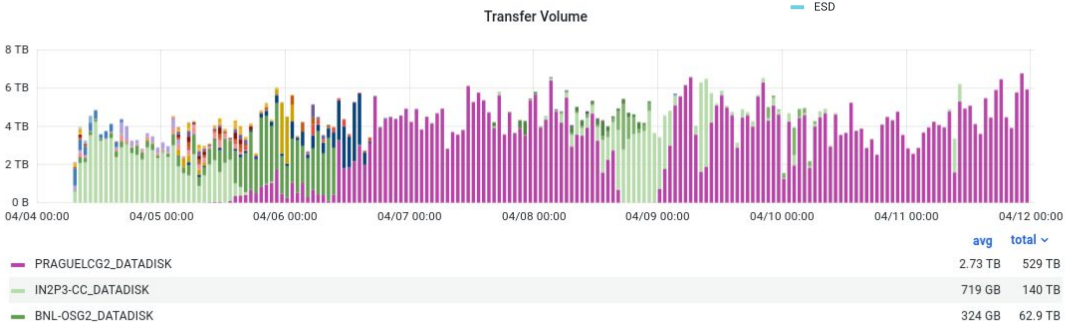
Same trick used: SEAL administrators injected CERN-provided certificate in their loadbalancer

Gradual selection and transfer of datasets

SEAL is funding a full-time development position to improve cloud support in Rucio


Volume per datatype_grouped

	min	max	avg	current
DAOD	0 B	2.72 PB	935 TB	0 B
AOD	0 B	930 TB	185 TB	0 B
EVNT	0 B	181 TB	138 TB	0 B
HITS	0 B	52.6 TB	22.2 TB	0 B
ESD	0 B	1.13 TB	467 GB	0 B


Transfer Volume

	avg	total
PRAGUELCG2_DATADISK	2.73 TB	529 TB
IN2P3-CC_DATADISK	719 GB	140 TB
BNL-OSG2_DATADISK	324 GB	62.9 TB

Commercial clouds :: AWS

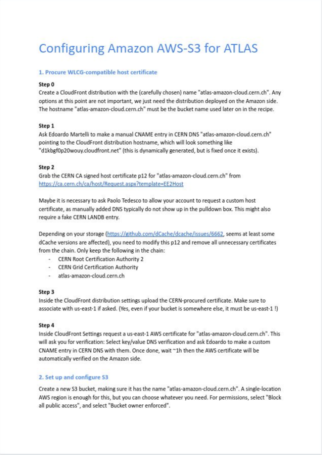
Now… Amazon was a different story

This is where it gets complicated

It worked out of the box for a year (thanks to DigiCert) until they changed to their own custom CA

In ATLAS there's a US Tier-3 (FRESNO) with a considerably sized investment

Setting this up was… challenging: 6+ months of trial & error lead to this short document



Configuring Amazon AWS-S3 for ATLAS

1. Procure WLCG-compatible host certificate

Step 0
Create a CloudFront distribution with the (carefully chosen) name "atlas-amazon-cloud.cern.ch". Any options at this point are not important, we just need the distribution deployed on the Amazon side. The hostname "atlas-amazon-cloud.cern.ch" must be the bucket name used later on in the recipe.

Step 1
Ask Edoardo Martelli to make a manual CNAME entry in CERN DNS "atlas-amazon-cloud.cern.ch" pointing to the CloudFront distribution hostname, which will look something like "d1kbgf0p20wouy.cloudfront.net" (this is dynamically generated, but is fixed once it exists).

Step 2
Grab the CERN CA signed host certificate p12 for "atlas-amazon-cloud.cern.ch" from https://ca.cern.ch/ca/host/Request.aspx?template=EE2Host

Maybe it is necessary to ask Paolo Tedesco to allow your account to request a custom host certificate, as manually added DNS typically do not show up in the pulldown box. This might also require a fake CERN LANDB entry.

Depending on your storage (https://github.com/dCache/dcache/issues/6662, seems at least some dCache versions are affected), you need to modify this p12 and remove all unnecessary certificates from the chain. Only keep the following in the chain:

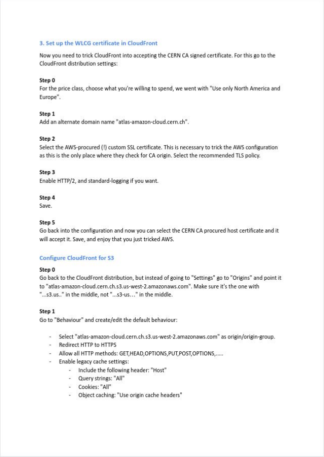
- CERN Root Certification Authority 2
- CERN Grid Certification Authority
- atlas-amazon-cloud.cern.ch

Step 3
Inside the CloudFront distribution settings upload the CERN-procured certificate. Make sure to associate with us-east-1 if asked. (Yes, even if your bucket is somewhere else, it must be us-east-1 !)

Step 4
Inside CloudFront Settings request a us-east-1 AWS certificate for "atlas-amazon-cloud.cern.ch". This will ask you for verification: Select key/value DNS verification and ask Edoardo to make a custom CNAME entry in CERN DNS with them. Once done, wait ~1h then the AWS certificate will be automatically verified on the Amazon side.

2. Set up and configure S3

Create a new S3 bucket, making sure it has the name "atlas-amazon-cloud.cern.ch". A single-location AWS region is enough for this, but you can choose whatever you need. For permissions, select "Block all public access", and select "Bucket owner enforced".



3. Set up the WLCG certificate in CloudFront

Now you need to trick CloudFront into accepting the CERN CA signed certificate. For this go to the CloudFront distribution settings:

Step 0
For the price class, choose what you're willing to spend, we went with "Use only North America and Europe".

Step 1
Add an alternate domain name "atlas-amazon-cloud.cern.ch".

Step 2
Select the AWS-procured (!) custom SSL certificate. This is necessary to trick the AWS configuration as this is the only place where they check for CA origin. Select the recommended TLS policy.

Step 3
Enable HTTP/2, and standard-logging if you want.

Step 4
Save.

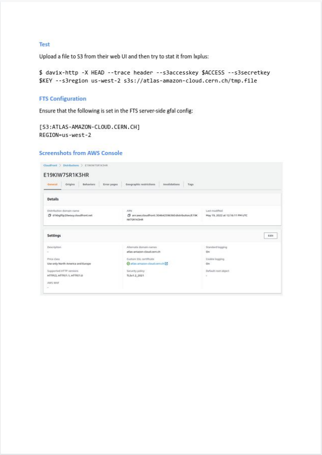
Step 5
Go back into the configuration and now you can select the CERN CA procured host certificate and it will accept it. Save, and enjoy that you just tricked AWS.

Configure CloudFront for S3

Step 0
Go back to the CloudFront distribution, but instead of going to "Settings" go to "Origins" and point it to "atlas-amazon-cloud.cern.ch.s3.us-west-2.amazonaws.com". Make sure it's the one with "...s3.us.." in the middle, not "...s3-us..." in the middle.

Step 1
Go to "Behaviour" and create/edit the default behaviour:

- Select "atlas-amazon-cloud.cern.ch.s3.us-west-2.amazonaws.com" as origin/origin-group.
- Redirect HTTP to HTTPS
- Allow all HTTP methods: GET,HEAD,OPTIONS,PUT,POST,OPTIONS,.....
- Enable legacy cache settings:
  - Include the following header: "Host"
  - Query strings: "All"
  - Cookies: "All"
  - Object caching: "Use origin cache headers"



Test

Upload a file to S3 from their web UI and then try to stat it from lxplus:

```
$ davix-http -X HEAD --trace header --s3accesskey $ACCESS --s3secretkey
$KEY --s3region us-west-2 s3s://atlas-amazon-cloud.cern.ch/tmp.file
```

FTS Configuration

Ensure that the following is set in the FTS server-side gfal config:

```
[S3:ATLAS-AMAZON-CLOUD.CERN.CH]
REGION=us-west-2
```

Screenshots from AWS Console

CloudFront > Distributions > E19KIW7SR1K3HR

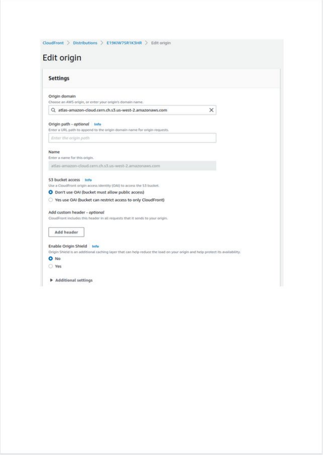
E19KIW7SR1K3HR

Details

Distribution domain name: d1kbgf0p20wouy.cloudfront.net
ARN: arn:aws:cloudfront::304642596360:distribution/E19KIW7SR1K3HR
Last modified: May 19, 2022 at 12:16:11 PM UTC

Settings

Description: -
Alternate domain names: atlas-amazon-cloud.cern.ch
Standard logging: On
Price class: Use only North America and Europe
Custom SSL certificate: atlas-amazon-cloud.cern.ch
Cookie logging: On
Supported HTTP versions: HTTP/2, HTTP/1.1, HTTP/1.0
Security policy: TLSv1.2_2021
Default root object: -
AWS WAF: -



CloudFront > Distributions > E19KIW7SR1K3HR > Edit origin

Edit origin

Settings

Origin domain
Choose an AWS origin, or enter your origin's domain name.
atlas-amazon-cloud.cern.ch.s3.us-west-2.amazonaws.com

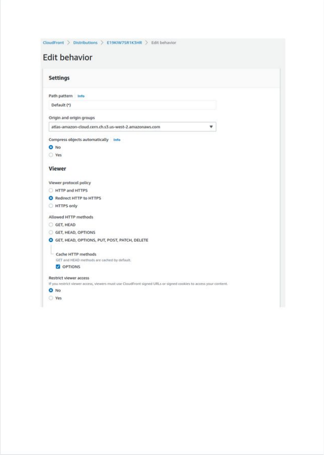
Origin path - *optional*
Enter a URL path to append to the origin domain name for origin requests.
Enter the origin path

Name
Enter a name for this origin.
atlas-amazon-cloud.cern.ch.s3.us-west-2.amazonaws.com

S3 bucket access
Use a CloudFront origin access identity (OAI) to access the S3 bucket.
● Don't use OAI (bucket must allow public access)
○ Yes use OAI (bucket can restrict access to only CloudFront)

Add custom header - *optional*
CloudFront includes this header in all requests that it sends to your origin.

Enable Origin Shield
Origin Shield is an additional caching layer that can help reduce the load on your origin and help protect its availability.
● No
○ Yes



CloudFront > Distributions > E19KIW7SR1K3HR > Edit behavior

Edit behavior

Settings

Path pattern: Default (*)

Origin and origin groups: atlas-amazon-cloud.cern.ch.s3.us-west-2.amazonaws.com

Compress objects automatically
● No
○ Yes

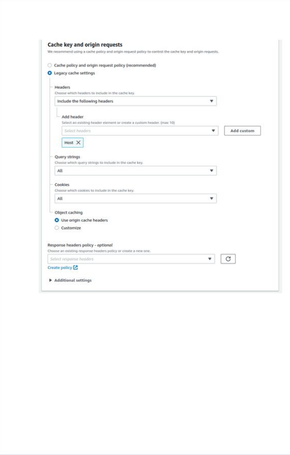
Viewer

Viewer protocol policy
○ HTTP and HTTPS
● Redirect HTTP to HTTPS
○ HTTPS only

Allowed HTTP methods
○ GET, HEAD
○ GET, HEAD, OPTIONS
● GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Cache HTTP methods
GET and HEAD methods are cached by default.
☑ OPTIONS

Restrict viewer access
If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.
● No
○ Yes



Cache key and origin requests
We recommend using a cache policy and origin request policy to control the cache key and origin requests.

○ Cache policy and origin request policy (recommended)
● Legacy cache settings

Headers
Choose which headers to include in the cache key.
Include the following headers

Add header
Select an existing header element or create a custom header. (max 10)
Select headers
Host

Query strings
Choose which query strings to include in the cache key.
All

Cookies
Choose which cookies to include in the cache key.
All

Object caching
● Use origin cache headers
○ Customize

Response headers policy - *optional*
Choose an existing response headers policy or create a new one.
Select response headers

Rucio ROOT Direct-IO mechanism

For interactive analysis and other stream processing cases: remote reads are used

The path returned from list-replicas usually can be fed straight into `TFile::Open()`

```
TFile::Open("https://mycloud:443/file.root?url_signature=1234");
```

S3 protocol **does not provide multi-range** byte requests
Amazon requires CloudFront CDN, which does **multi-range translation**
Others, e.g., Google Cloud Storage or MinIO, do not have this translation layer

Workaround is simply to disable multi-range requests through Davix

Have to append URL options to emulate: `#multirange=false&nconnections=30`

This is highly client dependent, *one size fits all* not really applicable
We will have to investigate if we should simply make Rucio reply with these options
Would require a potential hint to list-replicas (`--use-for-direct-io=30`) or similar solution

Future work

Configuration / Setup

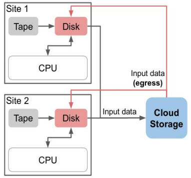
- Complicated, but grew organically from the ongoing Cloud R&D projects
- Needs a complete overhaul: esp. naming of attributes

Already identified features that we will need for production-level integration

- Access control right now is all-or-nothing, needs to be more fine grained
- Smarter peering mechanism
  - Static multihop distance config vs. dynamic cloud regions
  - The concept of cloud regions is missing completely
- Security considerations
  - Right now completely dependent on X.509 with DNS-injection trick
  - Clouds typically support OpenID/OAuth2 flows, should be helpful for token migration work
- Throughput and cost control not yet implemented, if you have the access rights you get the "full cloud power"
- Bucket-copy transfer tools, no need to go through FTS for this
- Cloud boosting option: Dynamically spend currency for extra throughput/storage
- Data lifetime considerations / different cloud QoS costs

Theoretical R&D studies: Simulation and evaluation of cloud storage caching (Tobias Wegner's PhD)

- Temporary cloud bursting to improve workflows needing tape recalls
- Demonstrates 15% improvement in job times



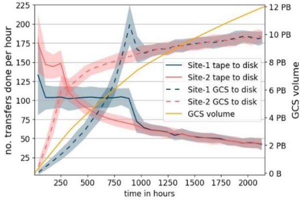


Fig. 8 The solid blue and red line show the number of transfers from tape to disk per hour for each site. The dashed lines show the number of transfers from GCS to disk per hour for each site. The orange line shows the GCS volume used.