EPJ Web of Conferences **214**, 07019 (2019) https://doi.org/10.1051/epjconf/201921407019
CHEP 2018

Lightweight WLCG Sites

The SIMPLE Grid Framework

Mayank Sharma[1,*], *Maarten* Litmaath[1,**], and *Eraldo* Silva Junior[2,***] *Renato* Santana[2,****]

[1]IT-DI-LCG, CERN, Geneva, Switzerland
[2]COHEP, CBPF, Rio de Janeiro, Brazil

Abstract. This article describes a new framework, called SIMPLE, for setting up and maintaining classic WLCG sites with minimal operational efforts and insights needed into the WLCG middleware. The framework provides a single common interface to install and configure any of its supported grid services, such as Compute Elements, Batch Systems, Worker Nodes and miscellaneous middleware packages. It leverages modern container orchestration tools like Kubernetes, Docker Swarm, and configuration management tools like Puppet, Ansible, to automate deployment of the WLCG services on behalf of a site admin. The framework is modular and extensible by design. Therefore, it is easy to add support for more grid services as well as infrastructure automation tools to accommodate diverse scenarios at different sites. We provide insight into the design of the framework and our efforts towards development, release and deployment of its first implementation featuring CREAM CE, TORQUE Batch System and TORQUE based Worker Nodes.

1 Introduction

The Worldwide LHC Computing Grid (WLCG)[1] project is a global collaboration whose mission is to provide an international computing infrastructure to store and analyze the data collected by the 4 main experiments at the Large Hadron Collider (LHC) at CERN: ALICE, ATLAS, CMS and LHCb. Each experiment uses computing resources primarily from its own member institutes. The institutes forming the WLCG infrastructure are grouped into tiers with corresponding responsibilities. Tier 0 is at the center and consists just of CERN. Tier 1 is formed by 15 institutes with large computing centers that share responsibilities with Tier 0 in the long-term curation of experiment data. Tier 2 is formed by more than 150 institutes that typically have much smaller computing infrastructures and much fewer people to operate WLCG resources than the big sites have available. There also exists a Tier 3 consisting of more than 50 sites that typically are small and support a single LHC experiment each, while having no formal obligations to the WLCG project.

One of the goals of WLCG Operations Coordination[2] activities is to help simplify what the majority of the sites, i.e. the smaller ones, need to do to be able to contribute resources in a useful manner, i.e. with large benefits compared to efforts invested. Classic grid sites

*e-mail: mayank.sharma@cern.ch
**e-mail: maarten.litmaath@cern.ch
***e-mail: esilvaju@cern.ch
****e-mail: renato.santana@cern.ch

may profit from simpler mechanisms to deploy and manage services. Moreover, some service types may no longer be needed in the end. Cloud computing setups may also become viable in some cases. There may be different options also depending on the experiment(s) that the site supports. There is no one-size-fits-all solution. There will rather be a matrix of possible approaches, allowing any site to check which ones could work in its situation, and then pick the best.

For the work described in this article we first have defined boundaries. Here we are not directly concerned with storage and data access, areas that are covered by the DOMA [3] project in WLCG. Instead, we focus on the provision of computing resources at sites. We note, however, that storage service deployment may also profit from the work we describe here. We also focus on sites participating in WLCG through the European Grid Infrastructure (EGI), where there is a lot more middleware diversity and a lot less influence from LHC experiments compared to the situation on the Open Science Grid (OSG) in the USA.

Several services currently are or may be needed to enable computing at a classic grid site: Computing Element; Batch system; Worker Nodes; Authorization system; Info system; Accounting; CVMFS Squid; Monitoring. In this work, we consider the first three service types for a start. In general, we can look into various approaches to help sites spend less effort there. For example:

1. Reduce the catalog of required services.
2. Replace the classic, complex portfolio with alternative, more widespread technologies.
3. Simplify deployment, maintenance and operation of what needs to remain.

A survey [4] launched in the autumn of 2016 showed that most of the EGI sites in WLCG need to continue providing classic grid services for the foreseeable future and would appreciate mechanisms to deal with those more easily. In particular, there was a strong interest in shared repositories for OpenStack images, Docker containers and Puppet modules.

In the past 2 years, Docker-compatible container orchestration systems have become ever more popular and are also more portable and lightweight than VM images. On the configuration side, there are alternatives to Puppet on the rise, in particular Ansible. Taking such considerations into account, we have devised a framework that will allow sites to set up classic grid services through Docker-compatible container orchestration systems and with popular configuration systems. The framework allows the site administrator to work with such common technologies and only spend a minimum of effort on grid service peculiarities.

2 SIMPLE Grid Framework

The WLCG is a diverse ecosystem in terms of:

- The available grid services sites can run to manage WLCG workloads: There are several compute element flavors, batch systems and other middleware technologies available in the WLCG ecosystem.
- The infrastructure automation and management technologies preferred by site admins: There are a wide variety of configuration management tools (Puppet, Ansible, Chef, SaltStack, ...), container orchestration technologies (Docker Swarm, Kubernetes, Apache Mesos, ...) etc. that admins could use to automate their IT infrastructure.

SIMPLE stands for Solution for Installation, Management and Provisioning of Lightweight Elements. The Lightweight Elements in the acronym, SIMPLE, refer to the grid services a site needs to operate in order to run WLCG workloads. The SIMPLE Grid

Framework offers a common interface to site admins for configuring any of the supported grid services using the infrastructure management tools they prefer to use and may already be using at their site. The framework does not expect a site admin to be familiar with the intricate architecture and configuration workflows involved with the grid services they need to set up at their site. The framework takes over the responsibility of managing such details and ensures, on behalf of the site admin, that all the essential middleware packages and grid services have been correctly installed and configured.

3 Design Principles

We have adopted some of the popular and well-established software design practices to ensure that the framework is scalable, modular and technology agnostic and that it can adapt to the requirements of WLCG sites, which are diverse and may change in the future. The following design principles ensure separation of concerns, reusability and extensibility in the design of the framework:

- Abstraction: The framework assumes that just basic knowledge of grid services should be sufficient for setting up a functional grid site.
- DRY: The Don't Repeat Yourself principle is stated as "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system".
- Modularity: The framework's architecture splits functionality into independent, interchangeable components. Each component is expected to provide strong encapsulation, well-defined and stable interfaces/APIs and explicit dependencies on other components.
- Simplicity of deployment through a single configuration master node: The framework leverages configuration management tools (Puppet, Ansible) and container orchestration tools (Docker Swarm, Kubernetes) to reduce effort on the site admin's part for setting up their site.
- Extensibility: The project is an open source and community driven effort to enable experts from the community to add support for their grid services or deployment technologies to the framework.

4 Project Architecture

The SIMPLE grid framework aims to be generic enough to accommodate the diversity in the WLCG and offer a standard recipe to set up lightweight WLCG sites that can be implemented using logically sound combinations of grid services and infrastructure technologies. All such possible implementations are governed by a SIMPLE Grid Project Specification Document[5]. This specification is inspired by the RFC standard documents published by the IETF.

Each version of the specification can have several valid implementations depending on the choice of grid services and infrastructure technologies that result in a functional WLCG site. As long as the framework components abide by the project specification document, the implementation of each framework component should not affect the overall functionality of the framework. A few possible implementations are shown in Figure 1.

5 SIMPLE Grid Specification

This section provides an overview of the first version of the project specification document [5] for the SIMPLE Grid Project.

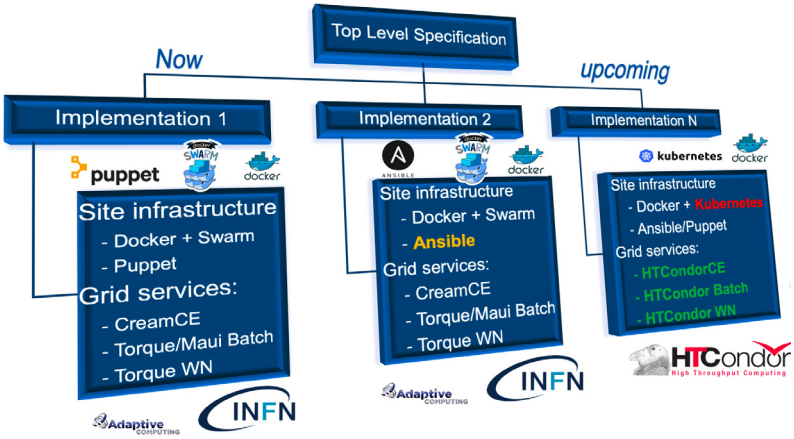

Figure 1: The various implementations of the specification that can be achieved using the SIMPLE Grid Framework.

5.1 Node Types

In the framework's ecosystem, there are two types of roles that can be associated with a node in the site's infrastructure. A 'Config Master' node is a single host at the site which is used by the framework to configure the entire grid infrastructure at that site. A 'Lightweight Component' node hosts one or more containers that run grid services. For a site with N nodes, there has to be exactly 1 config master node and a maximum of N lightweight component nodes.

5.2 Lightweight Components

The framework is composed of several components, each of which performs the functions defined for it by the project specification. This section briefly describes the components and their functions.

5.2.1 Site Admin

The Site Admin component steers sites administrators through the following sequence of operations to be performed by them:

1. Install the configuration management tool.

2. Configure the configuration management tool and download the central configuration manager installer, if required.

3. Fill out the site level configuration file.

4. Execute the central configuration manager to set up all grid services appropriately.

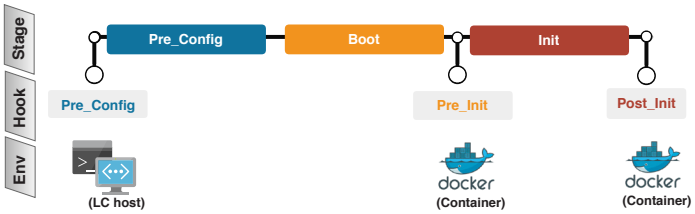

Figure 2: The component repository lifecycle indicating the different stages, callback hooks to which custom scripts can be attached and the execution environment for the hooks.

5.2.2 Site Level Configuration File, Schema and Defaults

The Site Level Configuration File contains all the information needed by the framework to set up and configure a functional WLCG site on a site's infrastructure. This file is created by the site admin. Each implementation of the framework is expected to provide its own Site Level Configuration Schema and Site Level Configuration Defaults files. The former describes the structure of a site level configuration file. The latter provides global default values for standard configuration parameters such as those related to Virtual Organizations to be supported, grid accounts etc. that can be used by the site admin. The data serialization technology for representing the aforementioned information is YAML 1.2.

5.2.3 Component Repositories

The component repositories are packages that represent a containerized grid service in the framework. In practice, they are standard git repositories hosted on GitHub that contain:

1. Dockerfiles for grid services like compute elements, batch systems, worker nodes, accounting services, etc.

2. Metadata to describe the repository to the framework and configure the containers using any one of the configuration management tools supported by the component repository.

Each grid service has its own component repository. For instance, computing element flavors CREAM CE, HTCondor CE and ARC CE, as well as batch system flavors TORQUE, Slurm etc. are expected to reside in their separate component repositories. The directory structure of a component repository must follow the guidelines described in the project specification document [5].

For each grid service, the site admin can fine-tune the configuration by providing additional parameters in the site level configuration file that were not defined in the component repository of that grid service. The site admin can also inject custom scripts that will be executed in the container or on the host machine during various stages of the lifecycle of a component repository. Figure 2 lists these callback events.

5.2.4 YAML Compiler and Configuration Validation Engine

The YAML compiler processes the site level configuration file prepared by the site admin. It generates an augmented version of the site level configuration file that contains a much more detailed description of the site. The augmented site level configuration file is used by other components of the framework when performing their functions. The YAML compiler builds on top of the YAML 1.2 specification and defines two new keywords:

1. *__from__*: Assign the value of another YAML anchor (variable) to the given YAML anchor. This keyword overcomes the fact that YAML anchors cannot be assigned values from other anchors in the YAML 1.2 specification. It enables *variable = variable* assignments.

2. *__include__*: Injects contents of another YAML file into the current YAML file. This allows configuration files to be split into multiple smaller, logically sound configuration files. This keyword can combine such files into the main site level configuration file.

The configuration validation engine is responsible for validating the contents of the site level configuration file, checking if the expected configuration can in fact be applied on the available infrastructure, and finally running tests to assess the success or failure of the configuration process.

The YAML Compiler and the Configuration Validation Engine significantly contribute to the Abstraction and DRY principles. They enable declaration of custom data types, validation of configuration files against their schema files, definition of default values for configuration parameters and defining conditions in which they should be used etc.

5.2.5 Central Configuration Manager

The Central Configuration Manager binds all the other framework components together. It controls the sequence of execution of the functions of the other components of the framework and ensures that their execution occurs at the right place and within the correct environment. The objectives of the central configuration manager are:

1. Install, configure and execute the YAML compiler, providing it with the site level configuration file as input.

2. Validate the information in the site level configuration file.

3. Check if the configuration can be applied on the available infrastructure.

4. Install the container orchestration tool (Docker Swarm, Kubernetes).

5. Configure the container orchestrator and implement a suitable networking strategy.

6. Aggregate and appropriately execute the custom shell scripts to be run during various lifecycle events of a component repository.

7. Configure firewall, CVMFS etc. on Lightweight Component nodes.

8. Validate that all the containers represented by the component repositories are in a stable state and submit tests to the lightweight site setup to ensure that end to end functionality has been attained.

9. Generate reports to summarize the results of each stage.

5.3 Configuration Levels

There are 2 configuration levels within the framework. They are defined as follows:

- Level-1: Any framework component that has access to the site level configuration file provided by the site admin in its original or compiled state is categorized as a level-1 configuration manipulator within the framework. Level-1 configurators are executed on Config Master or Lightweight Component nodes. They work together to generate input files for Level-2 configurators.

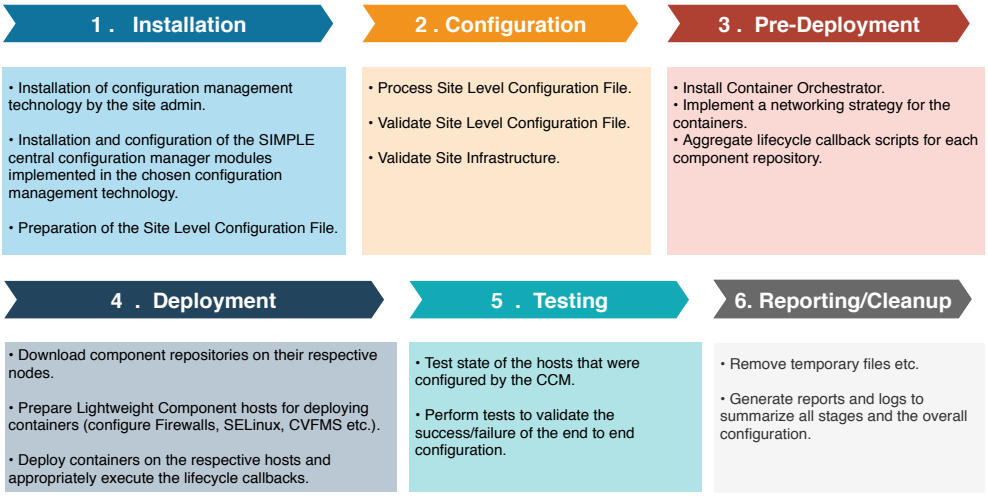

Figure 3: The stages of the execution pipeline and their main objectives.

- Level-2: Any lifecycle callback scripts and/or configuration management tools that are executed inside the containers are known as Level-2 configurators. They consume information processed by Level-1 configurators to configure the containers.

5.4 Execution Pipeline

The execution pipeline traverses five stages for the execution of functions in the framework, as shown in Figure 3. A sixth stage is devoted to reporting and cleanup.

Each stage executes functions of one or more framework components that have been discussed earlier. Each function in the pipeline prepares the framework's environment for subsequent functions.

6 Implementations

For the first version of the specification discussed, there can be several implementations depending on the choice of grid services and background technologies to configure them on the site's infrastructure. For the initial implementation, the choices are shown in Figure 4. It is being co-developed with an early adopter of the project: Centro Brasileiro de Pesquisas Físicas (CBPF), the Tier-2 in Rio de Janeiro, Brazil. All the implementations are maintained on GitHub Organization for the SIMPLE Grid Project. [6]

In addition to the first implementation, an alternative implementation of the central configuration manager was developed using Ansible by Tarang Mahapatra (University of British Columbia, Vancouver) in his work for the Google Summer of Code 2018. This helped us validate the modular design of the framework. At present, the software for integration of Kubernetes as a container orchestrator for the framework is being developed in collaboration with Julia Gavrilenko from the Plekhanov Russian University of Economics, Moscow. The Ansible based central configuration manager, Kubernetes integration and the component repositories for HTCondor CE, batch system and worker nodes will be part of upcoming releases, which are planned in the Technical Roadmap [7].

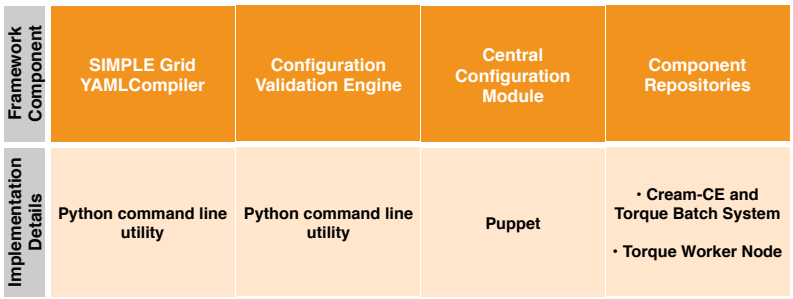

Figure 4: Background technologies and containerized grid services that are part of the first implementation.

7 Conclusions

The survey presented during CHEP 2016 revealed a continued need for classic grid sites that feature Compute Elements, Batch Systems and Worker Nodes. It also revealed a strong wish among site admins for reduction of overheads associated with setting up and maintaining grid sites, through use of modern infrastructure automation and management technologies like Puppet, Ansible, Kubernetes, Docker etc. We describe a high level specification document that underlies a modular, extensible and community driven framework, called SIMPLE, to accommodate the diverse deployment scenarios in the WLCG. The framework allows site admins to configure a WLCG site without assuming much knowledge of grid service details. The initial implementation features Docker Swarm and Puppet and provides support for the CREAM CE, the TORQUE batch system and the TORQUE worker node. Future implementations will also feature Kubernetes and Ansible, and support more grid services like the HTCondor CE, batch system and worker node. Experts of other grid services such as the ARC CE and Slurm are welcome to join the community driven effort and add support for their grid services to the framework.

References

[1] *Worldwide LHC Computing Grid*, accessed Nov. 2018, `http://wlcg.web.cern.ch/`

[2] *WLCG Operations TWiki*, accessed Nov. 2018, `https://twiki.cern.ch/twiki/bin/view/LCG/WLCGOperationsWeb`

[3] *DOMA Activities TWiki*, accessed Nov. 2018, `https://twiki.cern.ch/twiki/bin/view/LCG/DomaActivities`

[4] *Lightweight sites: Survey results* (2016), accessed Nov. 2018, `https://indico.cern.ch/event/540424/contributions/2194899/subcontributions/212150`

[5] M. Sharma, M. Litmaath, E. Silva Junior, *SIMPLE Framework: Specification Document - Google Docs* (2018), accessed Nov. 2018, `http://cern.ch/go/X7cr`

[6] M. Sharma, M. Litmaath, E. Silva Junior, T. Mahapatra, J. Gavrilenko, A. Arisal, *WLCG Lightweight Sites GitHub Organization*, accessed Nov. 2018, `https://github.com/WLCG-Lightweight-Sites`

[7] M. Sharma, M. Litmaath, E. Silva Junior, *SIMPLE Grid Framework: Technical Roadmap* (2018), accessed Nov. 2018, `https://twiki.cern.ch/twiki/bin/view/LCG/SimpleGridTechnicalRoadmap`