The SIMPLE Framework for deploying containerized grid services

Mayank Sharma (CERN)
Maarten Litmaath (CERN)
Eraldo Silva Junior (CBPF)
Renato Santana (CBPF)
SIMPLE Grid Project

Solution for Installation Management and Provisioning of Lightweight Elements.

A private PaaS that automates configuration and deployment of WLCG services, popular software frameworks like Hadoop, Spark etc.. on demand.

Setup and run services with minimal oversight and operational effort.

Under the hood, we leverage popular configuration management tools like Puppet /Ansible and container orchestrators such as Docker Swarm/Kubernetes.

Full autonomy to site admin to configure grid services through various framework hooks and easy access to containers running grid services.
Why SIMPLE?

• Based on **classic grid site model** used/preferred at most sites.
  • CEs, Batch, WNs, ...
  • Support non-LHC VOs with ease.
• Combines benefits of popular tools like **Puppet** with **Docker** and offers more...
  • Helps avoid common pitfalls associated with configuration of grid services.
  • Expects basic sys-admin know-how to exploit advantages of such technologies.
  • **Significant reduction** in amount of **config-info** and iterations needed to get a functional site.
  • **Validation of configuration** before deployment.
  • **Validation of infrastructure** and services after deployment.
• Easy to update or re-instantiate services
  • use **curated set of containers that provide stable grid services** from upstream.
  • **Rollback functionality** in case of re-deployments (for updates, config changes etc.)
• **Support** from the developers and team behind SIMPLE.
SIMPLE Framework: Example

Data Center

Config Master (CM)
Centrally manage installation and configuration of grid services on the LC nodes.

Lightweight Component (LC)
The nodes on which grid services are deployed by the framework.

188.184.91.176

simple-condor-ce  simple-condor-batch  simple-condor-node0  simple-condor-node1  simple-condor-node2
188.185.112.251  188.185.84.16  188.185.78.135  188.185.64.158  188.185.68.214
SIMPLE Framework: Example

• Write a **site-level-configuration.yaml** File:

```yaml
## Variable declaration:

```global_variables:
- &lightweight_component01_ip_address 188.185.112.251
- &lightweight_component01 fqdn simple-condor-ce.cern.ch
- &lightweight_component02_ip_address 188.185.84.16
- &lightweight_component02 fqdn simple-condor-batch.cern.ch
- &lightweight_component03_ip_address 188.185.78.135
- &lightweight_component03 fqdn simple-condor-node0.cern.ch
- &lightweight_component04_ip_address 188.185.64.158
- &lightweight_component04 fqdn simple-condor-node1.cern.ch
- &lightweight_component05_ip_address 188.185.68.214
- &lightweight_component05 fqdn simple-condor-node2.cern.ch
```
SIMPLE Framework: Example

Details about your site’s infrastructure

```
51    site_infrastructure:
52        - fqdn: *lightweight_component01_fqdn
53        ip_address: *lightweight_component01_ip_address
54        - fqdn: *lightweight_component02_fqdn
55        ip_address: *lightweight_component02_ip_address
56        - fqdn: *lightweight_component03_fqdn
57        ip_address: *lightweight_component03_ip_address
58        - fqdn: *lightweight_component04_fqdn
59        ip_address: *lightweight_component04_ip_address
60        - fqdn: *lightweight_component05_fqdn
61        ip_address: *lightweight_component05_ip_address
```

Use variables

```
51    supported_virtual_organizations:
52        - *default.vo.alice
```

Pick from several default variables
SIMPLE Framework: Example

```
site_infrastructure:
  - fqdn: lightweight_component01_fqdn
  - ip_address: lightweight_component01_ip_address
  - fqdn: lightweight_component02_fqdn
  - ip_address: lightweight_component02_ip_address
  - fqdn: lightweight_component03_fqdn
  - ip_address: lightweight_component03_ip_address
  - fqdn: lightweight_component04_fqdn
  - ip_address: lightweight_component04_ip_address
  - fqdn: lightweight_component05_fqdn
  - ip_address: lightweight_component05_ip_address
```

Use variables

Details about your site's infrastructure

```
supported_virtual_organizations:
  - *default.vo.alice
  - *default.vo.dteam
  - *default.vo.ops
```

Pick from several default variables

5 Nov 2019
Describe the component repositories should be deployed at the site

```yaml
lightweight_components:
  - type: compute_element
    name: HTCondor-CE
    repository_url: "https://github.com/WLCG-Lightweight-Sites/simple_htcondor_ce"
    repository_revision: "master"
    execution_id: 0
    lifecycle_hooks:
      pre_config: []
      pre_init: []
      post_init: []
    deploy:
      node: lightweight_component01_fqdn
      container_count: 1
    config:
      uid_domain: cern.ch
      condor_host_execution_id: 1
      preferred_tech_stack:
        level_2_configuration: sh
      supplemental_config:
        condor-ce:
          {condor_knob}: {value}
```

SIMPLE’s repository for HTCondorCE

Specify which node in the data center should run HTCondorCE

Specify configuration parameters requested by HTCondorCE repository developers

Additional Condor configuration knobs for your setup
SIMPLE Framework: Example

Config Master (CM)

Install puppetserver, puppet

Lightweight Component (LC)

Install puppet

Then, install `simple_grid_puppet_module` on all nodes. For instance,

```
[root@simple-condor-cm ~]# puppet module install many-simple_grid
```

Then, initialize the nodes using the puppet module. For instance,

```
[root@simple-condor-cm ~]# puppet apply -e "class {'simple_grid::install::config_master::simple_installer':}" 

[root@simple-condor-node0 ~]# puppet apply -e "class {'simple_grid::install::lightweight_component::simple_installer': puppet_master => 'simple-condor-cm.cern.ch'}"
```
SIMPLE Framework: Example

- Execute the framework

```bash
[root@simple-cm ~]# puppet agent -t
```

* Click on the image to see the terminal capture
The HTCondor pool is ready!
SIMPLE Framework: Example

• Summing up:
  • Install puppet and *simple grid puppet module* on all nodes.
  • Write a *site-level-config-file.yaml*.
  • Execute the SIMPLE framework.
• **Getting Started Guide**
Use case

• A first natural use case for the SIMPLE framework is migration from CREAM-CE.

• Simplify **switching to** HtCondorCE/HTCondor batch powered site.
SIMPLE Framework: Use Case 1

Centro Brasileiro de Pesquisas Físicas (CBPF, Tier-2 in Brazil)
HTCondorCE, HTCondor Batch, HTCondor workers.
*Test site running real production jobs

**Done jobs in Phoenix**

**Active jobs in Phoenix**
SIMPLE Framework: Use Case 2

CERN
Dynamic Apache Spark Cluster for Economic Analysis
* Mini cluster that runs Apache Spark, Hadoop, Yarn, HDFS, Jupyter Notebook frontend.
Community Driven!

• Open Source community!

• Looking for:
  • **Site admins** who wish to try out and/or beta test creating HTCondorCE/HTCondor Batch sites with the SIMPLE framework.
  • **ARC/Slurm experts** to help support these grid services through SIMPLE.
What’s next?

• Accounting and explicit job priority configuration for HTCondor.
• Upcoming Component repositories:
  • Squid
  • Storage solutions like xCache/ …
  • ARC and SLURM
• RedHat Rundeck web interface for using the framework (real-time deployment monitoring, get email notifications)
• Support for Kubernetes in addition to Docker-Swarm
• Support for Ansible in addition to Puppet.
• Request support for grid services/ features/ bug report: GitHub Project
Communication channels

Website: https://simple-framework.github.io

Slack Channel: simple-framework.slack.com

Mailing List: Google Groups, E-Groups

GitHub Org: WLCG-Lightweight-Sites

Technical Roadmap (WLCG): CERN TWiki
Backup Slides
Grid Service Experts

• Easily **add support for grid services** by creating component repository

• **Add code + Dockerfile** to repository lifecycle events that instruct the core framework on how to deploy your grid service containers.

• **Get in touch** with us to learn more.
Component Repository

Pre-Config

Dockerfile for your grid service

Boot

Entrypoint for your grid service container

Init

Site-level_config_file → grid service config files

5 Nov 2019

SIMPLE Framework: CHEP 2019
Diversity in WLCG

Types of CE/Batch/WN/Middleware packages

Technologies preferred by site admins for managing their infrastructure
**Principles**

- **SIMPLE**
  - Abstraction
  - DRY (Don't Repeat Yourself)
  - Modularity
  - Extensibility
  - Community Effort
  - Simple Deployment
  - One node to configure the site

**Tools**

- kubernetes
- docker
SIMPLE – Project Structure

Now

Implementation 1
  - Site infrastructure
    - Docker + Swarm
    - Puppet
  - Grid services:
    - CreamCE
    - Torque/Maui Batch
    - Torque WN

Implementation 2
  - Site infrastructure
    - Docker + Swarm
    - Ansible
  - Grid services:
    - CreamCE
    - Torque/Maui Batch
    - Torque WN

Implementation N
  - Site infrastructure
    - Docker + Kubernetes
    - Ansible/Puppet
  - Grid services:
    - HTCondorCE
    - HTCondor Batch
    - HTCondor WN
Site Level Configuration File

A single YAML file to describe:

- **Site-Infrastructure** (Hostnames, IP addresses, OS/Kernel, Disk/Memory)
- **Grid Components** (What grid components to install and configure)
- **Generic Site Info** (Users, Groups, Supported VOs)
- **Misc. Site Info** (Security emails, location etc.)

**Background Technologies** (Puppet/Ansible, Docker/Kubernetes)
Site Level Configuration File

- **Minimize configuration** requirements via
  - Variables
  - Sensible **default values** for site-level configurations
  - Ability to **override values**
  - support additional parameters not defined in the system
  - Tested: **O(100) lines of YAML code** to set up the site
  - Split configuration into **multiple logically related YAML files** that can be shared
Component Repositories

- Publicly hosted repositories on GitHub that provide
  - Dockerized CE/WN/Batch/Squid etc.
  - Meta information for configuration of images using different configuration management tools
- 1 repository for every component (for instance, CreamCE, CondorCE, Torque, Slurm reside in separate repositories)
- Examples: HTCondorCE, HTCondorBatch, CreamCE, TorqueWN
Configuration Validation

- Configuration validation engine to ensure information supplied in site configuration file:
  - meets the configuration requirements of desired site component
  - is realizable on the available infrastructure using available background technologies
- [http://cern.ch/go/CvS8](http://cern.ch/go/CvS8)
- Possibility to inject custom validation rules
Central Configuration Manager

- The **main module** for centrally configuring everything at the site
- **Uses Validation Engine** to check site-configuration file
- Checks **status of available Site Infrastructure** that needs to be orchestrated
- Installs and **configures Grid components** from the repositories
Central Configuration Manager

- Implements a **Networking strategy** (overlay/dedicated)
- Ensures availability of **CVMFS** to the containers
- Runs **tests** to check for success or failure of site configuration
Specification: Putting it Together
Flashback – Project Structure

Top Level Specification

Implementation 1
- Site infrastructure
  - Docker + Swarm
  - Puppet
- Grid services:
  - CreamCE
  - Torque/Maui Batch
  - Torque WN

Implementation 2
- Site infrastructure
  - Docker + Swarm
  - Ansible
- Grid services:
  - CreamCE
  - Torque/Maui Batch
  - Torque WN

Implementation N
- Site infrastructure
  - Docker + Kubernetes
  - Ansible/Puppet
- Grid services:
  - HTCondorCE
  - HTCondor Batch
  - HTCondor WN
Implementations

- **Site Level Configuration File YAML Compiler**
  - Python command line utility

- **Configuration Validation Engine**
  - Python command line utility

- **Repositories for Grid Components**
  - Cream Compute Element + Torque Batch System
  - Torque Worker Node
  - ... (Other repositories mentioned)

- **Central Configuration Management System**
  - Puppet
  - Ansible
  - ... (Other tools mentioned)

---

Google Summer of Code 2019 Project

Google Summer of Code 2018 Project
Conclusions

• Set up a grid site with $O(100)$ lines of YAML
• Modular and easy to extend to support other grid services
• Community Driven: Open source and open discussion channels. Join Now!!