Lightweight Sites – SIMPLE framework

Mayank Sharma (CERN)
Maarten Litmaath (CERN)
Eraldo Silva Junior (CBPF)
Diversity in WLCG

Types of CE/Batch/WN/Middleware packages

Technologies preferred by site admins for managing their infrastructure
The Vision

- **Reduce operational efforts and oversight required to set up and maintain grid services at sites.**
- **Leverage modern infrastructure automation, configuration management and containerization tools to install, configure, deploy and maintain grid services.**
Site Admin’s Perspective

• Lightweight Sites Survey: [http://cern.ch/go/rhV9](http://cern.ch/go/rhV9)
• 51 Sites responded to the questionnaire that shows potential benefits of shared repositories

**Conclusion:**
• Most sites still require classic grid services which can be complicated to configure/deploy
• Simpler mechanisms for orchestration of sites utilizing modern infrastructure tools will be beneficial
• Strong support for Docker, Puppet, OpenStack images
SIMPLE

• Solution for Installation, Management and Provisioning of Lightweight Elements

• Support diversity in WLCG sites with minimal oversight and operation efforts

• Keep functionality the same, but easier for site admins to set up and maintain
SIMPLE: Usage Overview

• Create site-level-configuration-file.yaml
  • Describe infrastructure and grid services that will be deployed at the site.
• Execute the SIMPLE Grid Framework
  • The framework will configure all the hosts and deploy appropriate containers that run the required grid services.
  • The framework combines:
    • configuration management tools (Puppet/Ansible)
    • container orchestrators (Docker Swarm/ Kubernetes)
    • containerization technologies (Docker)
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-Infrastructure

Nodes on which grid services will be deployed:

```python
site_infrastructure:
- fqdn: ce01.mysite.domain
  ip_address: 192.168.10.2
- fqdn: wn01.mysite.domain
  ip_address: 192.168.10.70
...
Section: lightweight_components

lightweight_components

- name: WN-Pbs
  type: worker_node
  repository_url: "https://github.com/WLCG-Lightweight-Sites/wlcg_lightweight_site_wn_pbs"
  repository_revision: "master"
  execution_id: 1
  lifecycle_hooks:
    pre_config:
      - /etc/simple_grid/lifecycle/wn_pre_config.sh
    pre_init:
      - /etc/simple_grid/lifecycle/wn_pre_inst1.sh
    post_init:
      - /etc/simple_grid/lifecycle/wn_post_inst1.sh
  deploy:
    - node: wn01.mysite.domain
      container_count: 2
    preferred_tech_stack:
      level_2_configuration: yaim
  config:
    cc_host: #pbs_runtime_var_ce_host
    batch_server: pbs
  supplemental_config:
    some_additional_parameter: some_value
Section: lightweight_components

lightweight_components (advanced features)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>-</td>
<td>name: WN-Pbs</td>
</tr>
<tr>
<td>76</td>
<td></td>
<td>type: worker_node</td>
</tr>
<tr>
<td>77</td>
<td></td>
<td>repository_url: &quot;<a href="https://github.com/WLCG-Lightweight-Sites/wlcg_lightweight_site_wn_pbs">https://github.com/WLCG-Lightweight-Sites/wlcg_lightweight_site_wn_pbs</a>&quot;</td>
</tr>
<tr>
<td>78</td>
<td></td>
<td>repository_revision: &quot;master&quot;</td>
</tr>
<tr>
<td>79</td>
<td></td>
<td>execution_id: 1</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>lifecycle_hooks:</td>
</tr>
<tr>
<td>81</td>
<td></td>
<td>pre_config:</td>
</tr>
<tr>
<td>82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td></td>
<td>pre_init:</td>
</tr>
<tr>
<td>84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td></td>
<td>post_init:</td>
</tr>
<tr>
<td>86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td></td>
<td>deploy:</td>
</tr>
<tr>
<td>88</td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td>preferred_tech_stack:</td>
</tr>
<tr>
<td>91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td></td>
<td>config:</td>
</tr>
<tr>
<td>93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td></td>
<td>supplemental_config:</td>
</tr>
<tr>
<td>96</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Custom scripts to fine tune configuration of hosts and containers, if required.

Additional config params
Advanced features

• **Variables**
  • Declare YAML anchors and reuse them anywhere in the site-level-configuration file

```yaml
### Variable declaration:
vars:
- &lightweight_component01_ip_address 192.168.0.4
- &lightweight_component01_fqdn lightweight_component01.cern.ch
- &lightweight_component02_ip_address 192.168.0.5
- &lightweight_component02_fqdn lightweight_component02.cern.ch
```

• **Default Values**
  • Several **sensible default variables** already exist in the framework to make configuring a site more efficient.

```yaml
supported_virtual_organizations:
- *default.vo.alice
- *default.vo.dteam
- *default.vo.ops
```
Advanced features

• **Override default values**
  • Override default values based on your configuration requirements.

```yaml
supported_virtual_organizations:
  - *default.vo.alice
  - <<: default.se: 'my-se.mydomain'
  - *default.vo.dteam
  - *default.vo.ops
```

• **__include__ keyword**
  • Split site-level-config-file into smaller, logically related configuration files

```yaml
36   site_infrastructure:
37   | __include__: "./my-site-info.yaml"
```
SIMPLE: Practical Insights

- Initial Test Deployment:
  - **Centro Brasileiro de Pesquisas Físicas** (CBPF, Tier-2 in Brazil)
    - Site level configuration file is around 100-200 lines of YAML code.
    - Takes between 20-30 minutes to deploy CREAM-CE, Torque Batch system and Torque worker nodes on a mini test site.
  - Technologies: Puppet, Docker-Swarm, Docker and YAIM

- Upcoming test deployment:
  - **Institute of Physics of the Czech Academy of Sciences** (Tier-2, Prague, Czech Republic)
SIMPLE: Deployment steps

For the 1st implementation featuring Puppet, Docker-Swarm and Dockerized grid services:

- Install Puppet on all the nodes.
- Install SIMPLE Grid Puppet module on all the nodes.
- Write the site-level-configuration file
- Execute the SIMPLE Grid Framework.
SIMPLE: Note to Users

While the framework abstracts and automates low level configuration required by supported grid services, a **site admin must** still:

- Have basic understanding of the grid services they wish to configure using the framework. For instance,
  - **Queues** to create for the chosen batch systems.
  - **VOs** to be supported by their sites.
  - **Pool Accounts** that shall be created for the jobs.

- Ensure that the host machines have **sufficient resources** (compute, memory, storage) to run the grid service

- Ensure **availability of a healthy network** (physical/virtual) between the hosts.
SIMPLE: basic vs. advanced usage

- Basic usage is the first goal
  - An admin only needs to consider parameters etc. that cannot (easily) be determined by the SIMPLE framework
  - See examples on the previous page

- Support for advanced usage is a second goal
  - An admin can “look under the hood” to see how to enhance / extend the framework
  - Discover how popular technologies are being used
    - Container orchestration
    - Configuration management
  - Add or change some functionality in existing components
  - Add new (grid) components
SIMPLE – Specification

- Define **components** of the SIMPLE Grid Framework.
- Define **functions** of each framework component.
- Define the **execution pipeline** i.e. the sequence in which the functions are invoked in order to deploy a grid site.
SIMPLE – Core Components

- Site Level Configuration File
- Central Configuration Manager
- YAML Compiler
- Configuration Validation Engine

March 21, 2019
HOW2019, JLab
SIMPLE – Lightweight Elements

Component Repositories

- Cream-CE/ Torque Batch System
- HTCondor-CE
- ARC-CE
- Slurm
- Torque Worker Nodes

March 21, 2019

HOW2019, JLab
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, HTCondorCE, Torque, Slurm reside in separate repositories)
- Examples: CreamCE, TorqueWN
SIMPLE – Execution Pipeline

• Grid components are deployed via the following stages:
  • **Installation Stage**: install the various components of the framework.
  • **Configuration Stage**: configure the various components of the framework and compile/validate site level configuration file.
  • **Pre-Deployment Stage**: Prepare hosts, container orchestrators.
  • **Deployment Stage**: Deploy the containerized grid services
  • **Testing/Reporting Stage**: Fetch logs from hosts and containers about success/failure of the deployment.
SIMPLE – Execution Pipeline

1. Installation
- Installation of configuration management technology by the site admin
- Installation and configuration of the SIMPLE central configuration manager modules implemented in the chosen configuration management technology
- Description of the Site Level Configuration File

2. Configuration
- Process Site Level Configuration File
- Validate Site Level Configuration File
- Validate Site Infrastructure

3. Pre-Deployment
- Install Container Orchestrator
- Implement a networking strategy for the containers
- Aggregate lifecycle callback scripts for each component repository

4. Deployment
- Download component repositories on their respective nodes.
- Prepare Lightweight Component hosts for deploying containers(configure Firewalls, SELinux, CVFMS etc.)
- Deploy containers on the respective hosts and appropriately execute the lifecycle callbacks.

5. Testing
- Test state of the hosts that were configured by the CCM
- Submit test jobs to the compute element to validate the success/failure of the end to end configuration.

6. Reporting/Cleanup
- Remove temporary files and restore original state of the hosts.
- Generate reports and logs to summarize all stages and the overall configuration.

March 21, 2019

HOW2019, JLab
Flashback – Project Structure

**Top Level Specification**

**Now**

**Implementation 1**
- puppet
- 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

March 21, 2019

HOW2019, JLab
Try it out via SIMPLE Dev-Kit

• Simulate a grid site on your machine.
• Used to Develop/Test/Debug the framework
• Works locally until the pre-deployment stage of execution pipeline.

• https://github.com/maany/simple_grid_puppet_dev_kit/tree/master
The Community

Project Homepage
http://cern.ch/go/9IHd

GitHub Repositories
http://cern.ch/go/kr7p

Simple Grid Specification
http://cern.ch/go/X7cr

Technical Discussion List (E-Groups)
Name: WLCG-Lightweight-Sites-Dev
Link: http://cern.ch/go/I9wZ

Open Source Community
Name: WLCG Lightweight Sites
Link: http://cern.ch/go/Hz7S

Mattermost (IM):
Team: WLCG
Name: WLCG-Lightweight-Sites
Link: http://cern.ch/go/8HWP
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!!
Additional Slides
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
  • Possibility to inject custom validation rules
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
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
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
  - ...
- Central Configuration Management System
  - Puppet
  - Ansible
  - ...

Google Summer of Code 2018 Project + proposal for 2019