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 **I**nstallation **M**anagement and **P**rovisioning of **L**ightweight **E**lements.

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.

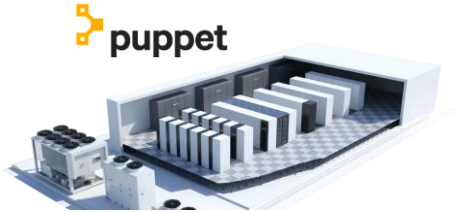
Why SIMPLE?

- 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.

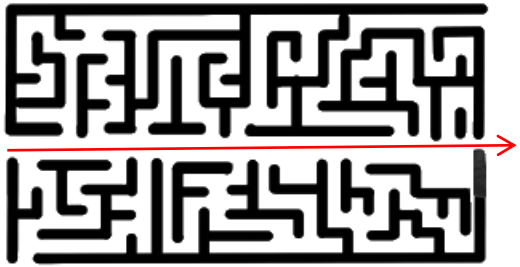
SIMPLE Workflow

site level uration file

Site Admin



Data Center

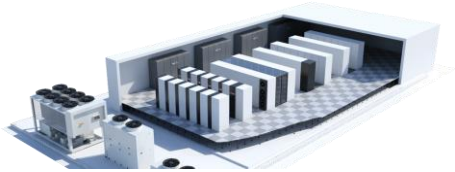


SIMPLE Core

SIMPLE Component repositories

Grid Service Experts

SIMPLE Framework: Example



Data Center

Config Master(CM)

Centrally manage installation and configuration of grid services on the LC nodes.

simple-condor-cm

188.184.91.176

Lightweight Component(LC)

The nodes on which grid services are deployed by the framework.

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:

declare variables

```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

```
154 supported_virtual_organizations:
155   - *default_vo_alice
```

Pick from several default variables

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

```
152 supported_virtual_organizations:
153   - *default_vo_alice
154   - *default_vo_dteam
155   - *default_vo_ops
```

Pick from several default variables

SIMPLE Framework: Example

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

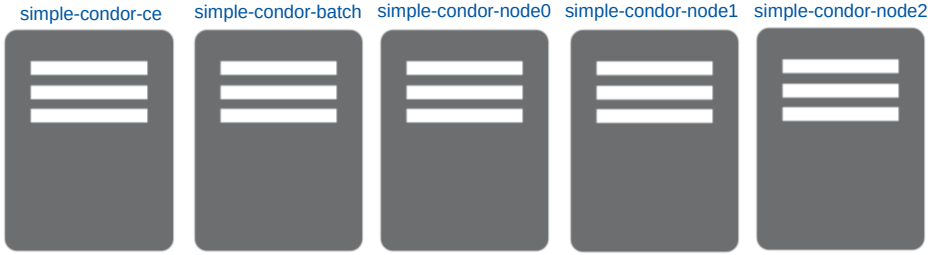
SIMPLE Framework: Example

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 maany-simple_grid
```

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

Click to view Terminal captures

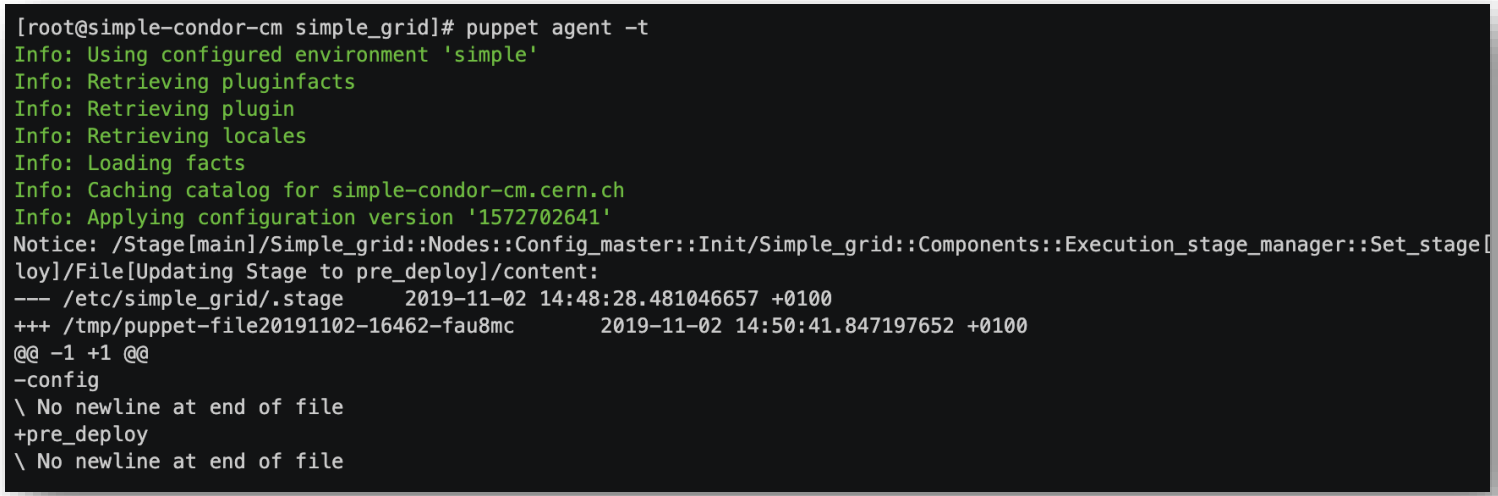
```
[root@simple-condor-cm ~]# puppet apply -e "class {'simple_grid::install::config_master::simple_installer':}"
```

```
[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

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



* Click on the image to see the terminal capture

SIMPLE Framework: Example

- The HTCondor pool is ready!

HTCondorCE

```
-- Schedd: simple-condor-ce.cern.ch : <10.0.0.10:8767> @ 11/02/19 15:39:45
OWNER BATCH_NAME      SUBMITTED   DONE   RUN    IDLE   HOLD  TOTAL JOB_IDS

Total for query: 0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
Total for all users: 0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended

sh-4.2# condor_ce_q


-- Schedd: simple-condor-ce.cern.ch : <10.0.0.10:8767> @ 11/02/19 15:39:57
OWNER  BATCH_NAME    SUBMITTED   DONE   RUN    IDLE  TOTAL JOB_IDS
simple ID: 7       11/2  15:39      _      _      1      1 7.0

Total for query: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
Total for all users: 1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

HTCondor Submit Node

```
universe                = grid
executable              = sleep.sh
log                     = sleep.log
output                  = outfile.txt
error                   = errors.txt
should_transfer_files   = Yes
when_to_transfer_output = ON_EXIT
use_x509userproxy = true
+WantJobRouter = true
+TransferOutput = ""
grid_resource = condor simple-condor-ce.cern.ch simple-condor-ce.cern.ch:9619
queue
[condor_user@simple-lc02 sleep_job]$ condor_submit sleep_simple_condor_ce.sub
```

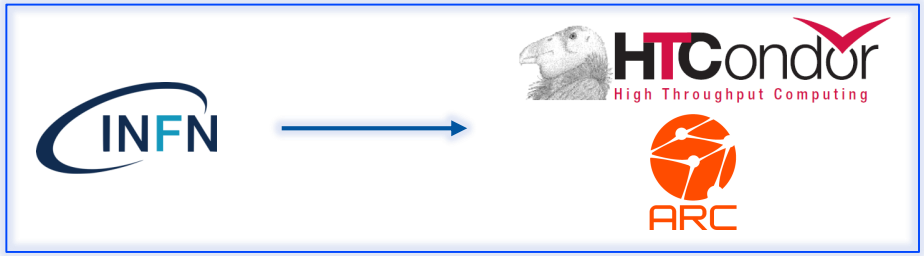
* Click on the images to see the terminal captures

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.



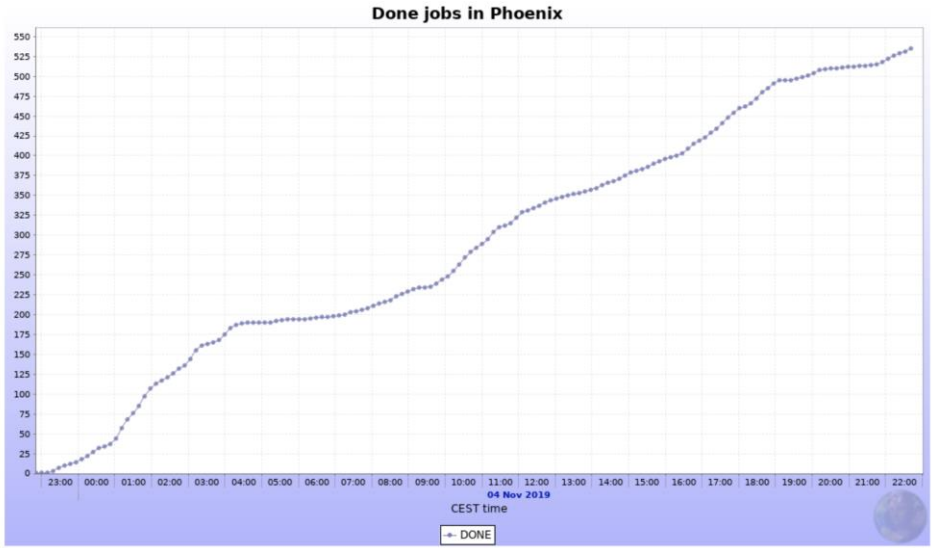
- Simplify **switching to HtCondorCE/HTCondor batch** powered site.

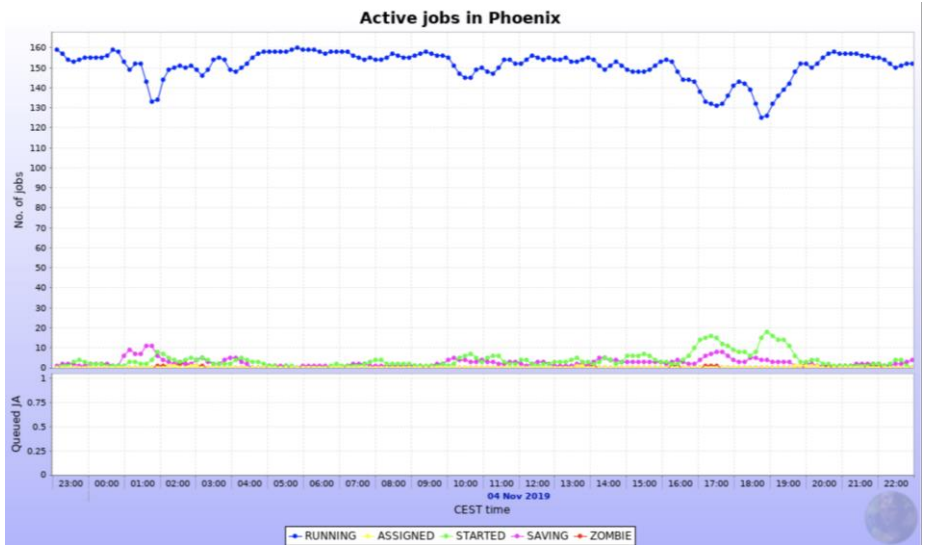
SIMPLE Framework: Use Case 1

Centro Brasileiro de Pesquisas Físicas (CBPF, Tier-2 in Brazil)

HTCondorCE, HTCondor Batch, HTCondor workers.

*Test site running real production jobs



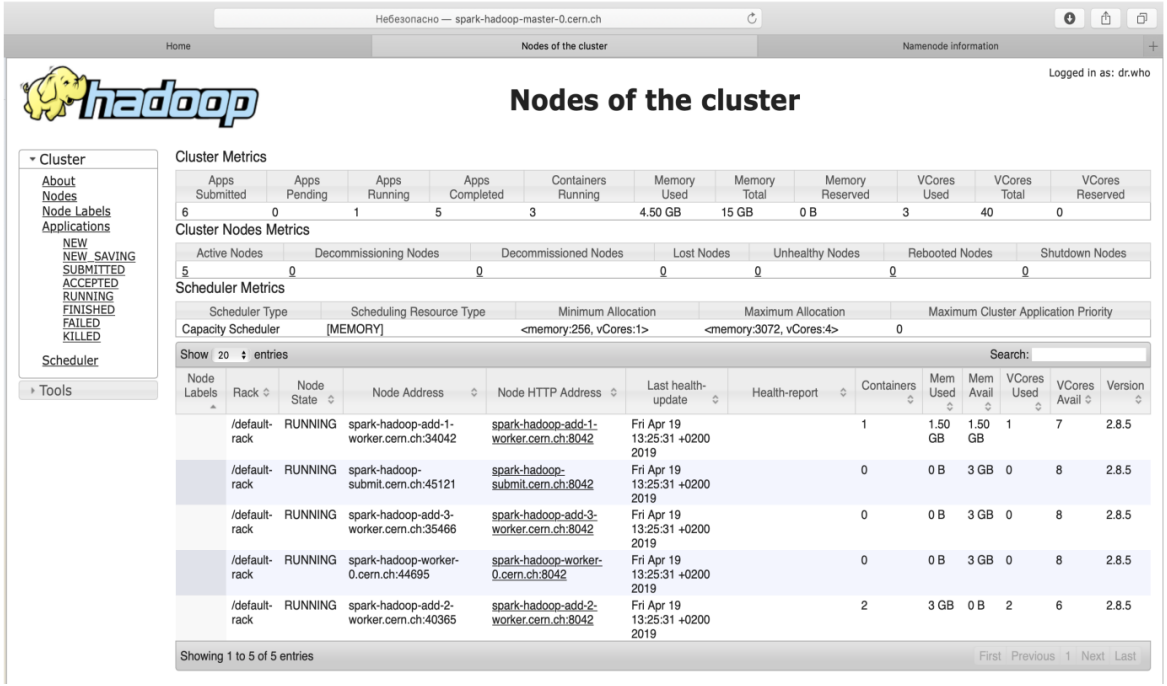


SIMPLE Framework: Use Case 2

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

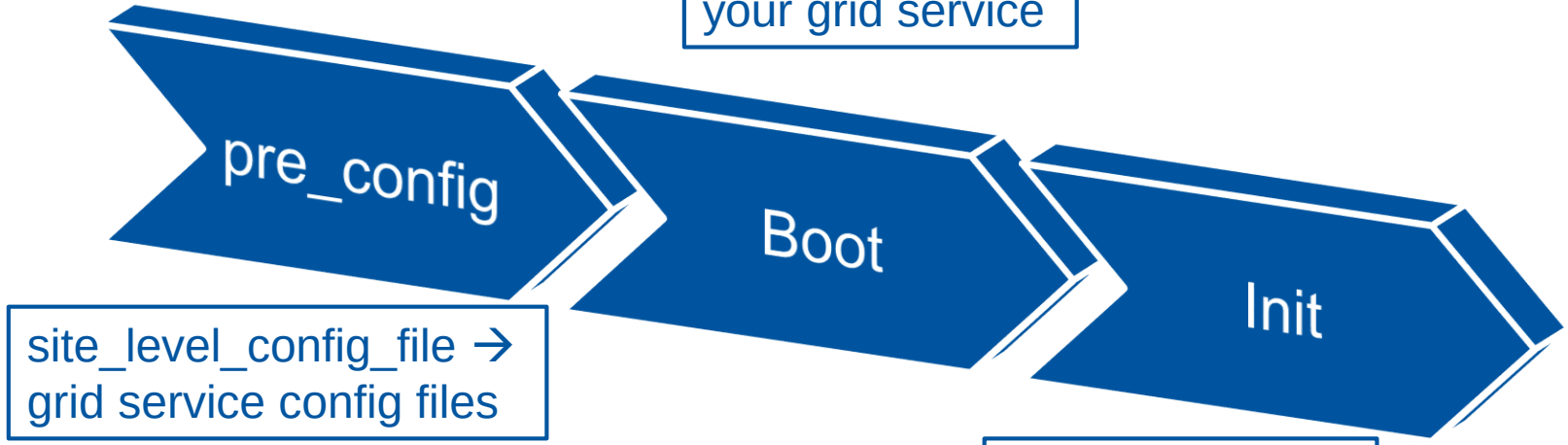
Technical Roadmap (WLCG): CERN TWiki

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



Dockerfile for your grid service

site_level_config_file → grid service config files

Entrypoint for your grid service container

Diversity in WLCG

Types of **CE/Batch/WN/Middleware** packages

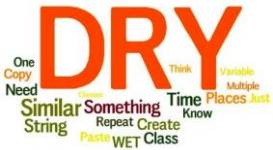
Technologies preferred by site admins for managing their infrastructure

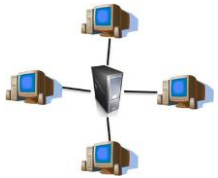
Principles

Abstraction

One node to configure the site

DRY (Don't Repeat Yourself)





SIMPLE

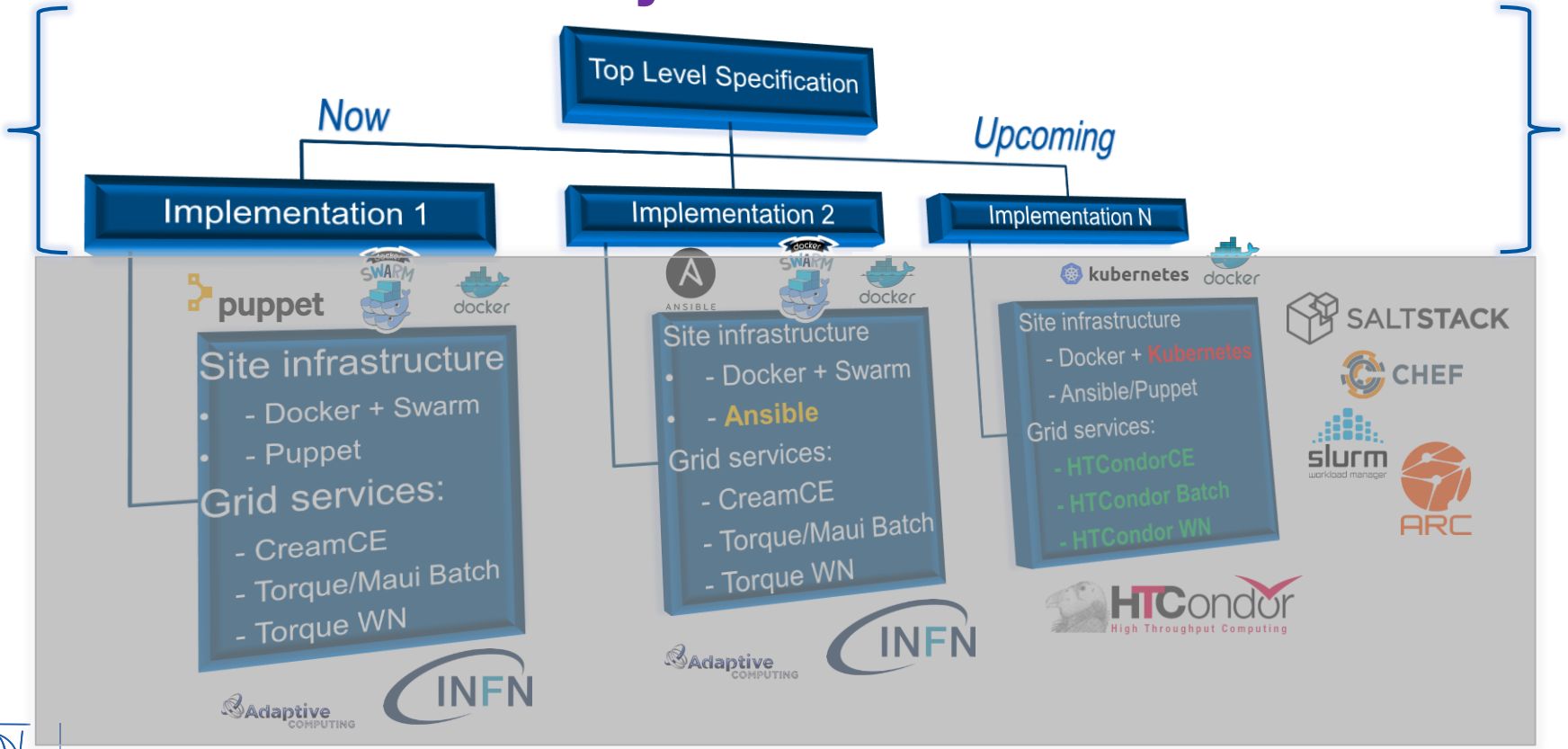
Extensibility

Community Effort

Modularity

Simple Deployment

SIMPLE – Project Structure



Site Level Configuration File

A single YAML file to describe:

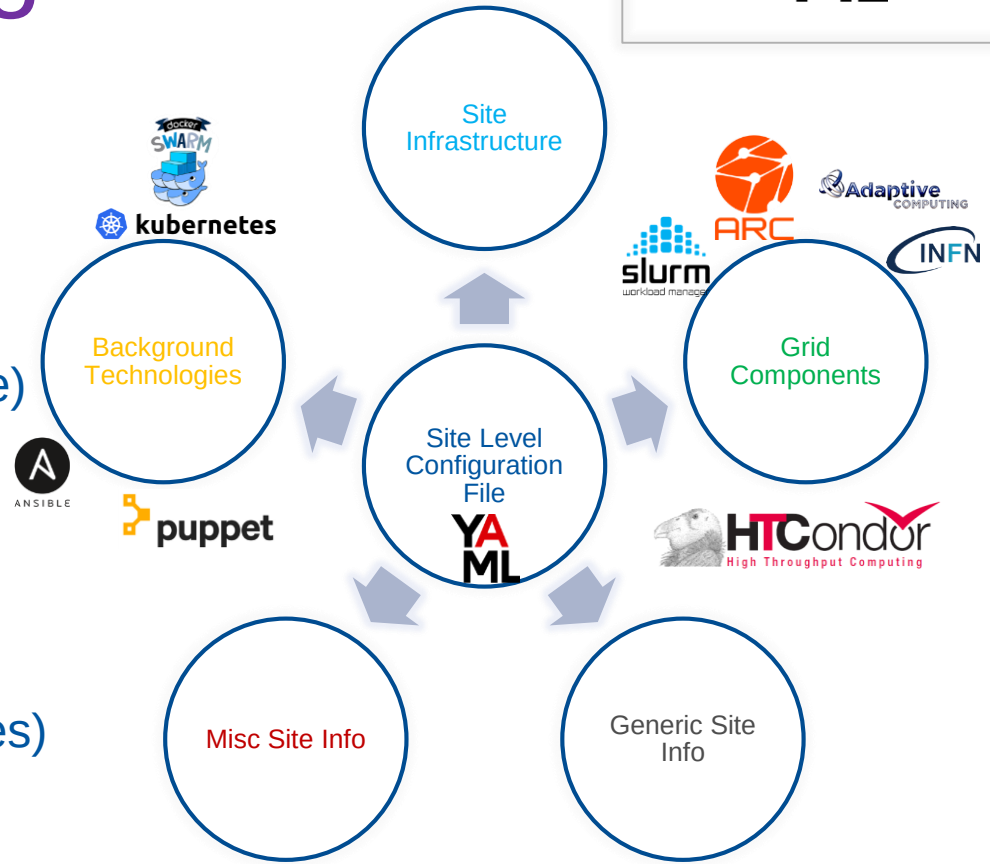
Site-Infrastructure (Hostnames, IP addresses, OS/Kernel, Disk/Memory)

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
- Possibility to inject custom validation rules

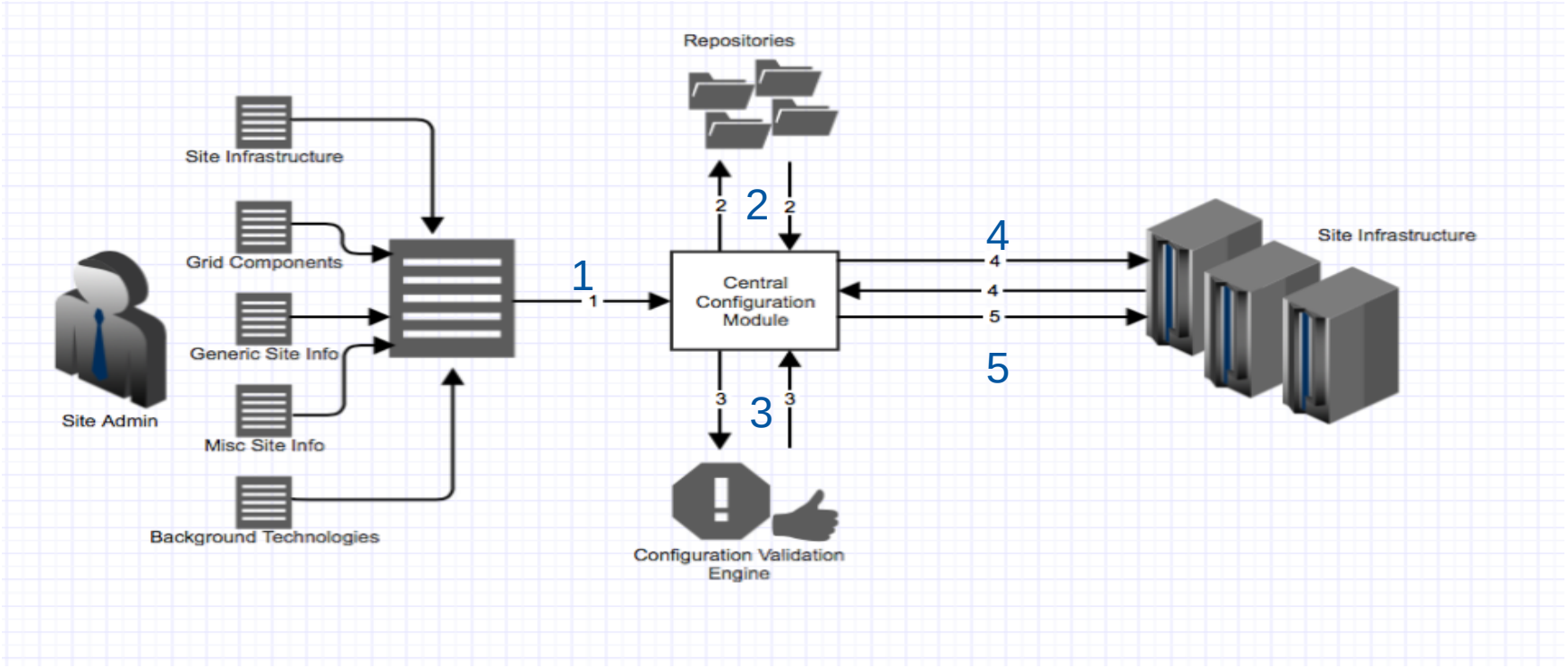
Central Configuration Manager

- 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

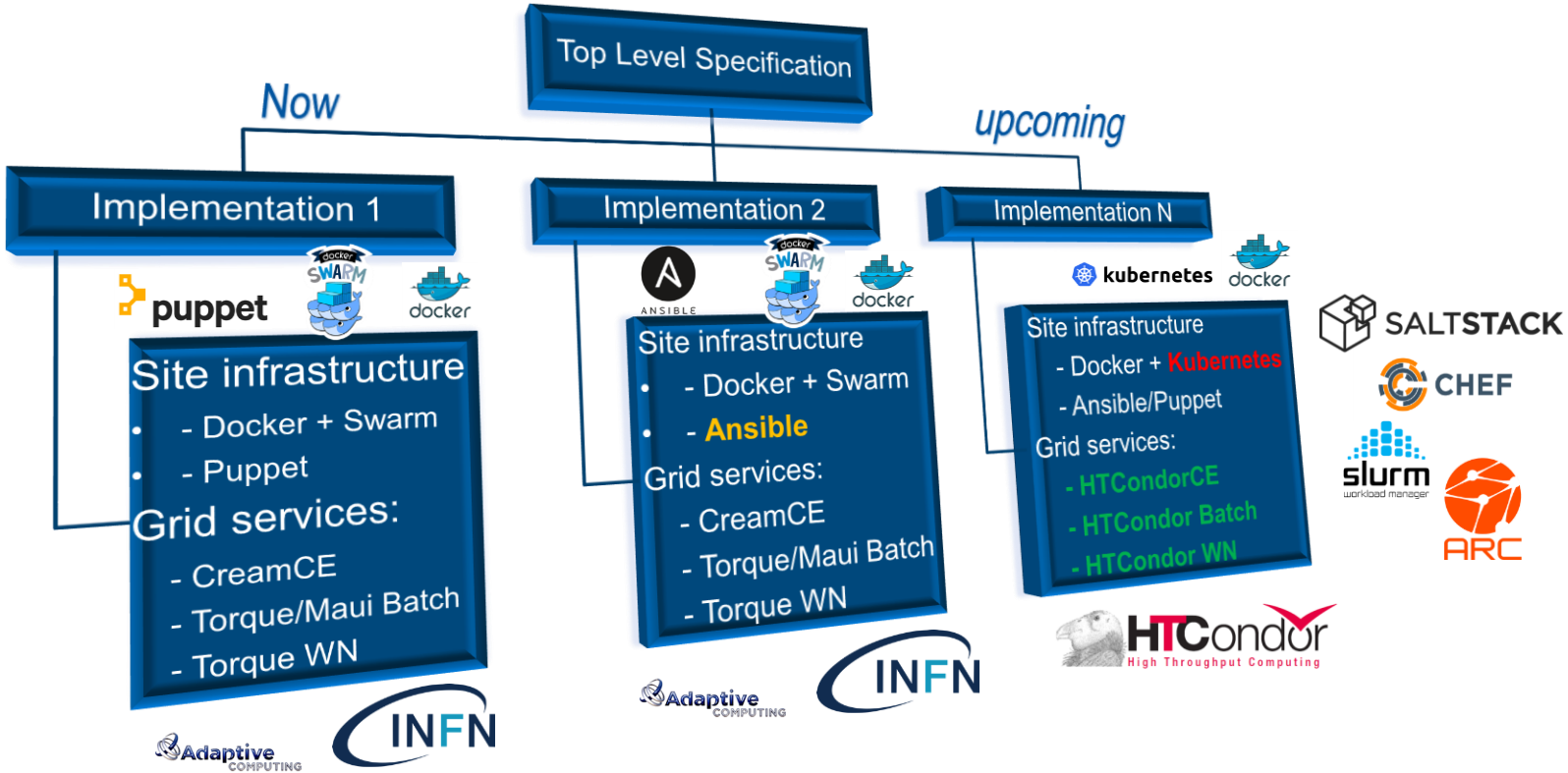
Central Configuration Manager

- 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



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 2019 Project Google

Google Summer of Code 2018 Project Google

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!!