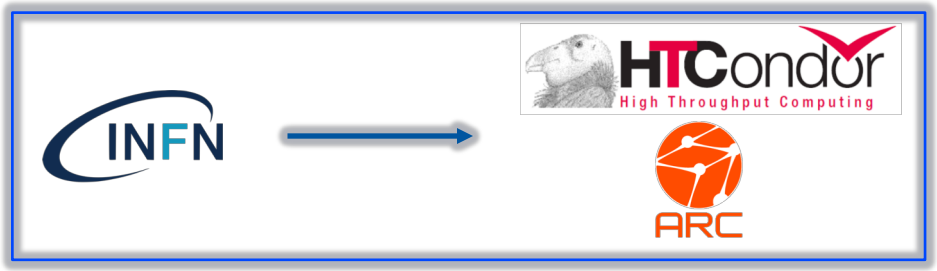
SIMPLE Framework (Easy deployment)

Mayank Sharma (CERN, speaker)
Maarten Litmaath (CERN)

Use case

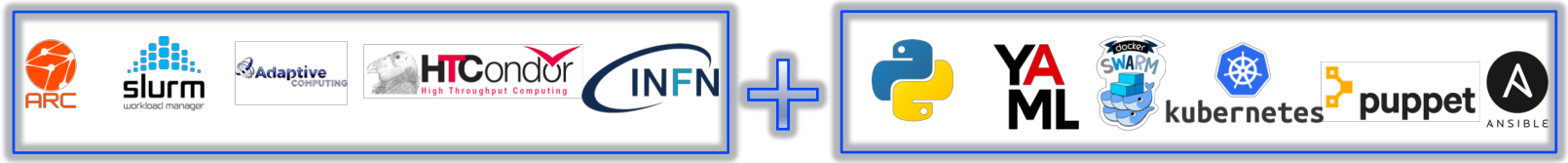
- A first natural use case for the framework is migration from CREAM-CE.



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

SIMPLE Framework

- Package sensible default configurations for grid services into **Docker containers**.
- Enable **hassle-free deployment** of these containers **across the site** using popular technologies under the hood:
  - container orchestration tools (**Docker Swarm/ Kubernetes**)
  - configuration management tools (**Puppet/Ansible**)

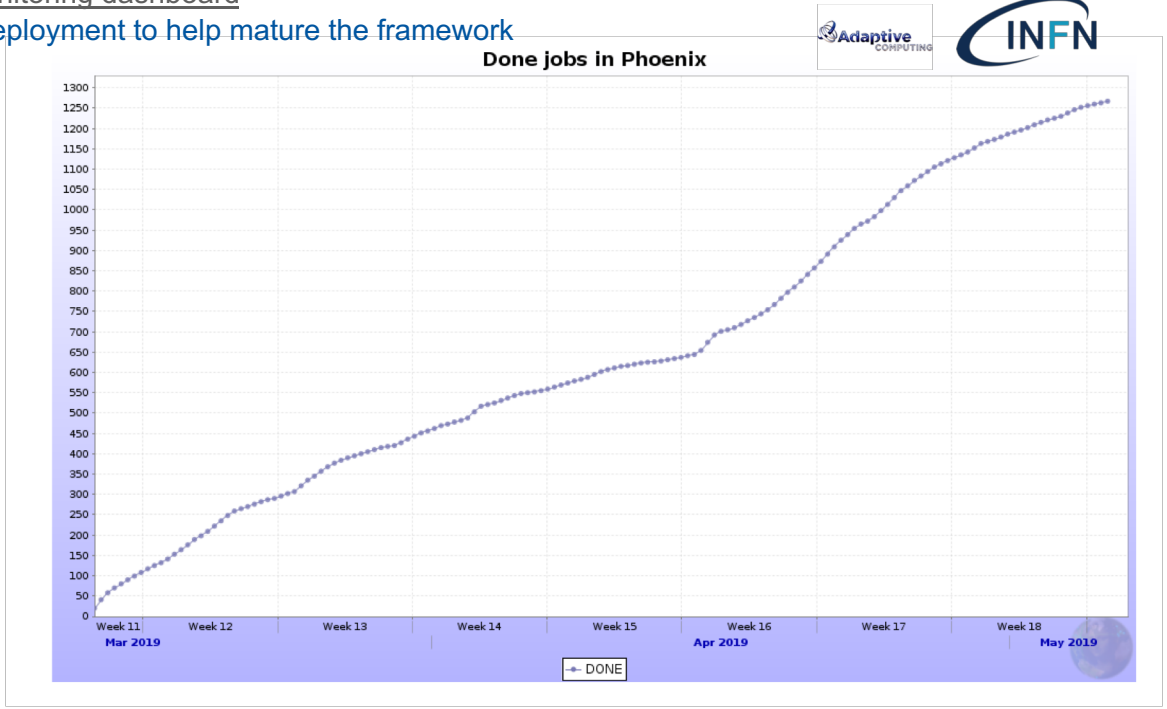


SIMPLE Framework

- **Updating services:** change version number in your site level simple configuration file.
- **Installing new services**: Add a few lines in your site level simple configuration and re-run the configuration.
- If you want, you can look under the hood to tweak and enhance the system.

SIMPLE Framework: Deployments

Centro Brasileiro de Pesquisas Físicas (CBPF, Tier-2 in Brazil)
Cream-CE, PBS batch system and workers
Monalisa monitoring dashboard
*small test deployment to help mature the framework



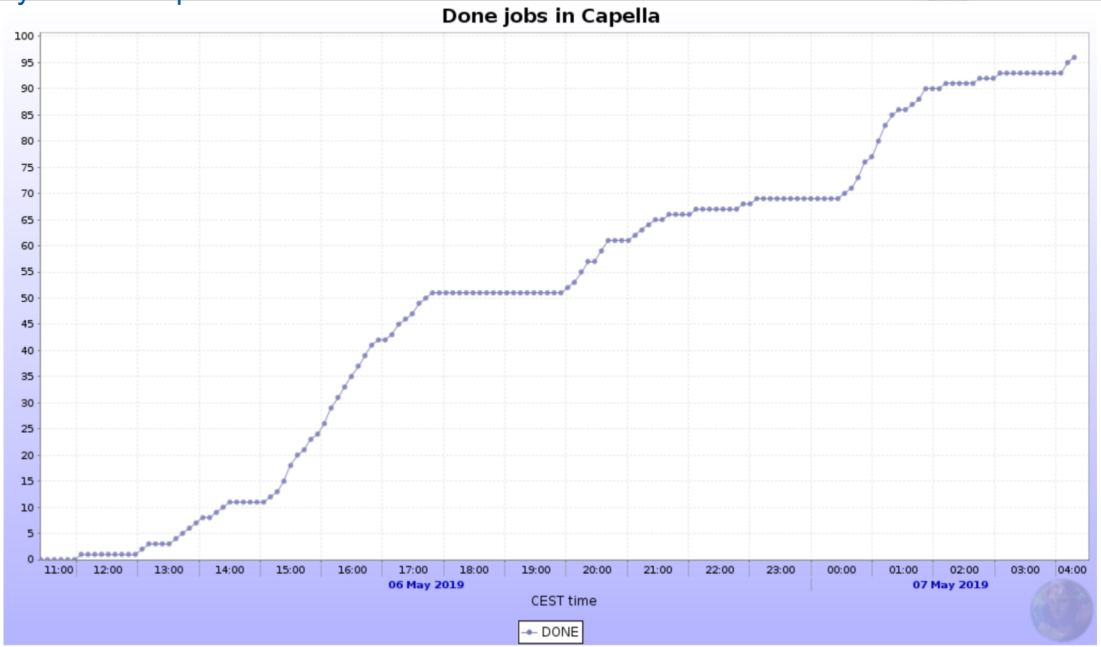
SIMPLE Framework: Deployments

CERN

(HTCondorCE, HTCondor batch system and workers

Monalisa monitoring dashboard

*small test deployment to help mature the framework

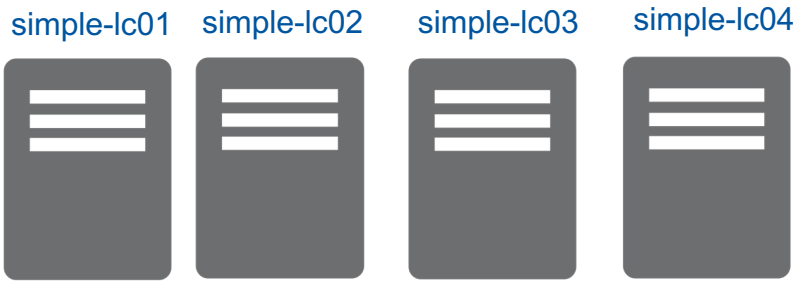


SIMPLE Framework: Example

Config Master(CM)

Install puppetserver, puppet

Lightweight Component(LC)



Install puppet and complete certificate signing process by the puppet master.

Then, install simple_grid_puppet_module on all nodes. For instance,

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

SIMPLE Framework: Example

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

declare variables

```yaml
global_variables:
  - &simple_cm_ip_address 188.184.91.176
  - &simple_cm_fqdn simple-cm.cern.ch
  - &simple_lc01_ip_address 188.184.88.69
  - &simple_lc01_fqdn simple-lc01.cern.ch
  - &simple_lc02_ip_address 188.184.83.48
  - &simple_lc02_fqdn simple-lc02.cern.ch
  - &simple_lc03_ip_address 188.185.76.205
  - &simple_lc03_fqdn simple-lc03.cern.ch
  - &simple_lc04_ip_address 188.184.86.181
  - &simple_lc04_fqdn simple-lc04.cern.ch
```

SIMPLE Framework: Example

Details about your site's infrastructure

```
20  site_infrastructure:
21    - fqdn: *simple_cm_fqdn
22      ip_address: *simple_cm_ip_address
23    - fqdn: *simple_lc01_fqdn
24      ip_address: *simple_lc01_ip_address
25    - fqdn: *simple_lc02_fqdn
26      ip_address: *simple_lc02_ip_address
27    - fqdn: *simple_lc03_fqdn
28      ip_address: *simple_lc03_ip_address
29    - fqdn: *simple_lc04_ip_address
30      ip_address: *simple_lc04_fqdn
```

Use variables

```
63  supported_virtual_organizations:
64    - *default_vo_alice
65    - *default_vo_atlas
66    - *default_vo_ops
```

Pick from several default variables

SIMPLE Framework: Example

Describe the grid services should be deployed at the site

```
32  lightweight_components:
33    - name: simple-htcondor-ce
34      type: compute_element
35      repository_url: "https://github.com/maany/simple
36      repository_revision: "master"
37      execution_id: 0
38      deploy:
39        - node: *simple_lc01_fqdn
40          container_count: 1
41      config:
42        - reserve_swap: 0
43      supplemental_config:
44        - {condor_knob}:{value}
45
```

Simple's repository for HTCondorCE

SIMPLE Framework: Example

- Execute the framework

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

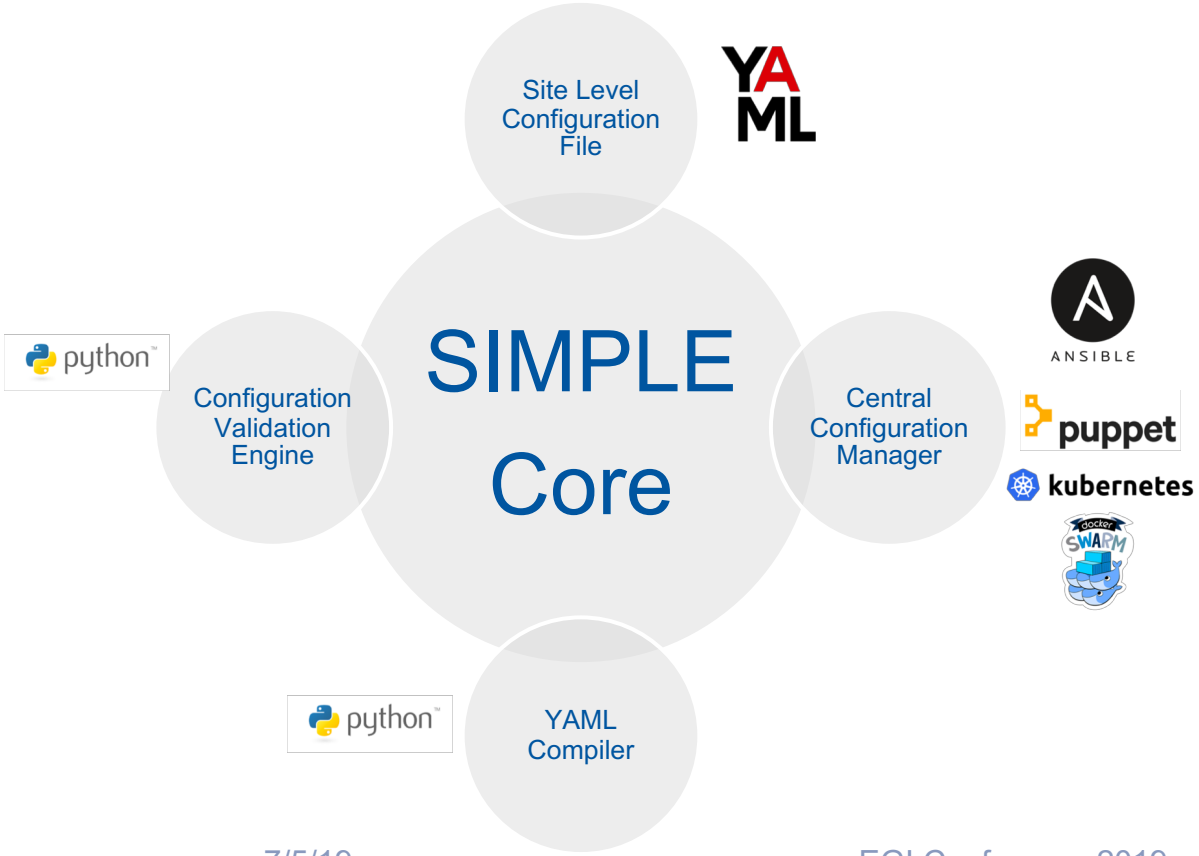
SIMPLE Framework: Example

- Summing up:
  - Install puppet and simple grid puppet module on all nodes.
  - Write a **site-level-config-file.yaml.**
  - Execute the framework.

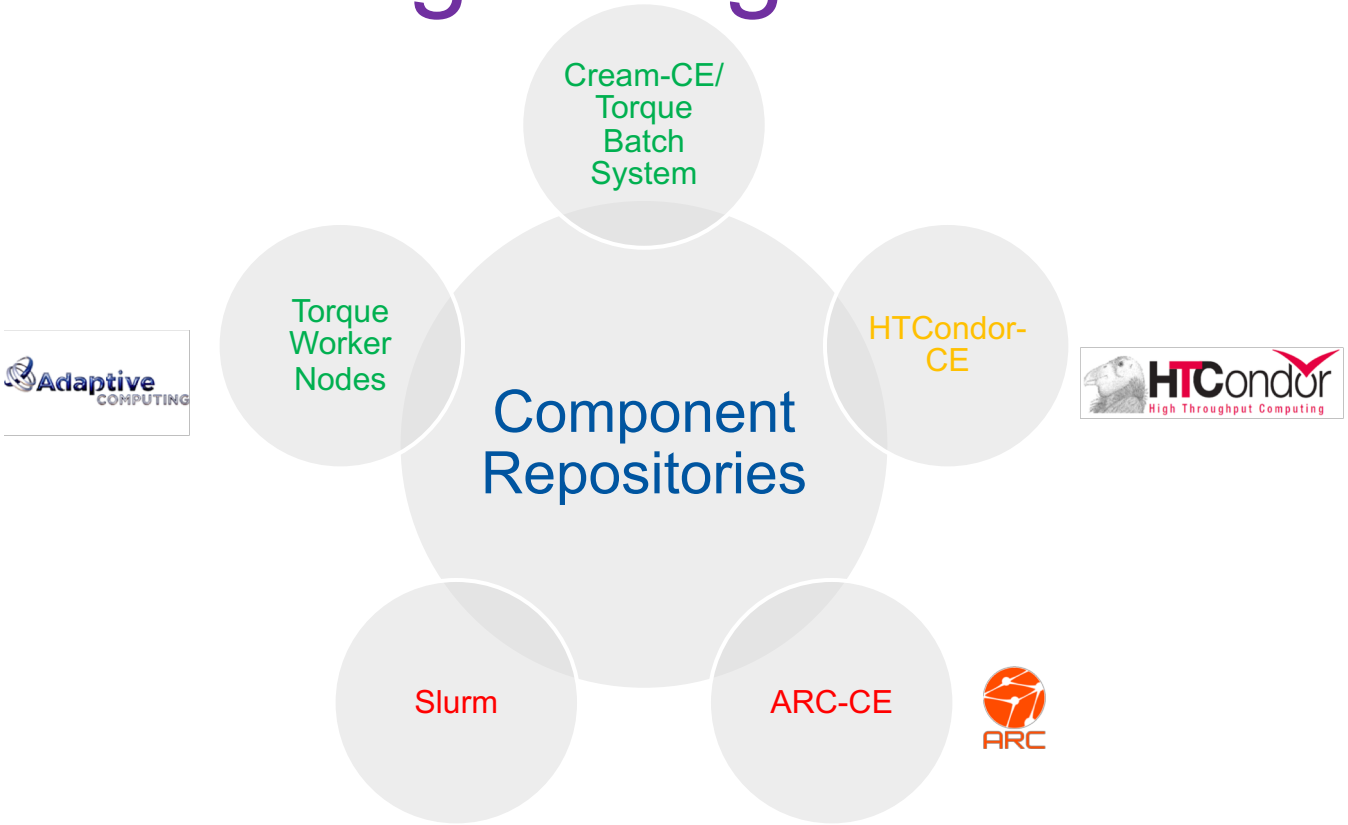
SIMPLE Framework

- The SIMPLE HT-Condor repositories should be ready for use in production in next few weeks. (Accounting/ BDII/Default configurations)
- Join the mailing list to get notified:
  - E-Groups : http://cern.ch/go/Hz7S
  - Google Group: http://cern.ch/go/l9wZ

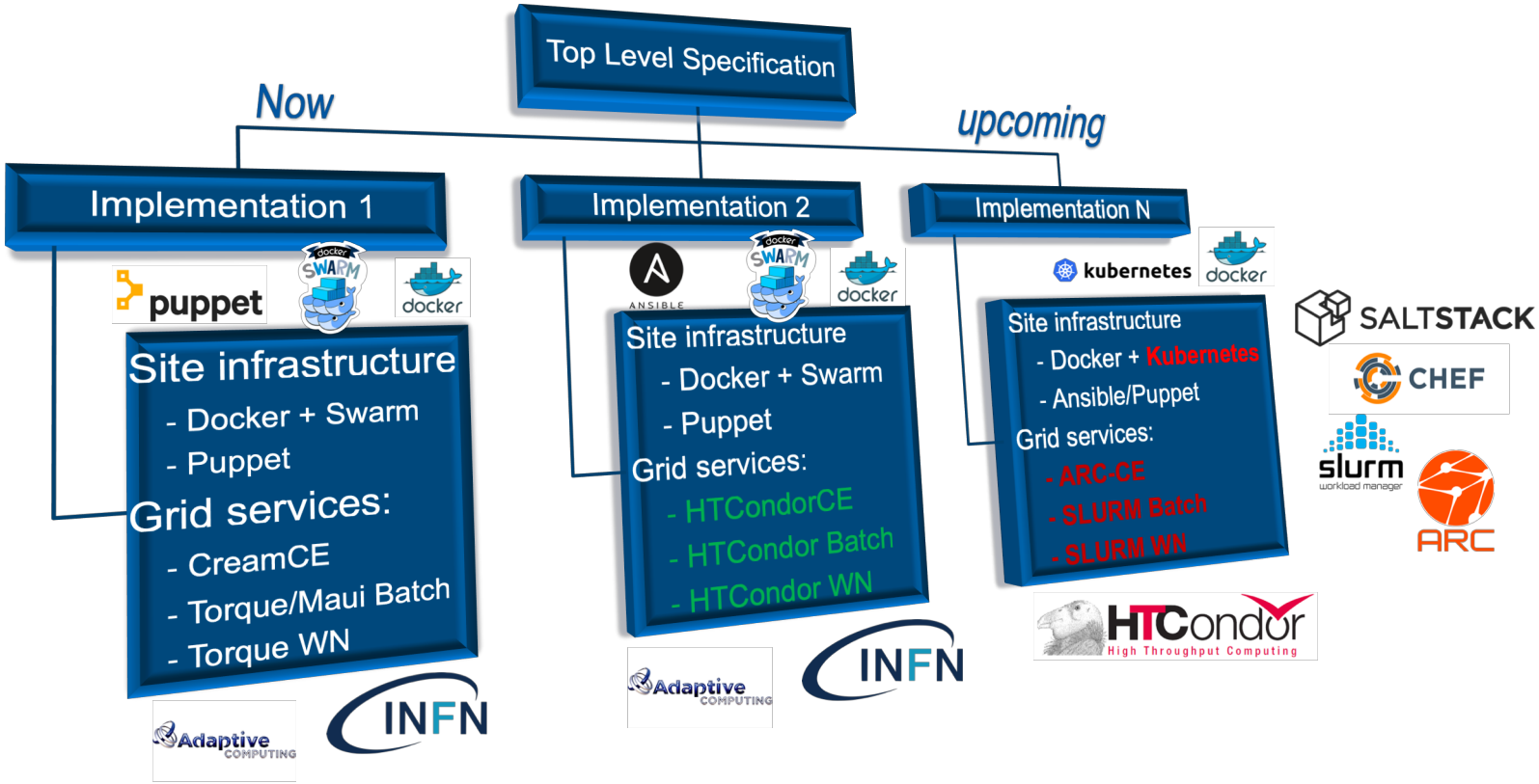
SIMPLE – Core Components



SIMPLE – Lightweight Elements



SIMPLE – Project Structure



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: CreamCE, TorqueWN

Community Driven!

- Open Source community!
- Looking for:
  - **ARC/Slurm experts** to help support these grid services through SIMPLE.
  - **Site admins** who wish to try out/ beta test/ HTCondorCE/ HTCondor Batch system.

The Community

Project Homepage

http://cern.ch/go/9lHd

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/l9wZ

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

Additional Slides

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 **setup 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
- 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

- **S**olution for **I**nstallation, **M**anagement and **P**rovisioning of **L**ightweight **E**lements
- Support diversity in WLCG sites with **minimal oversight and operation efforts**
- Keep **functionality the same**, but easier for site admins to setup 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:

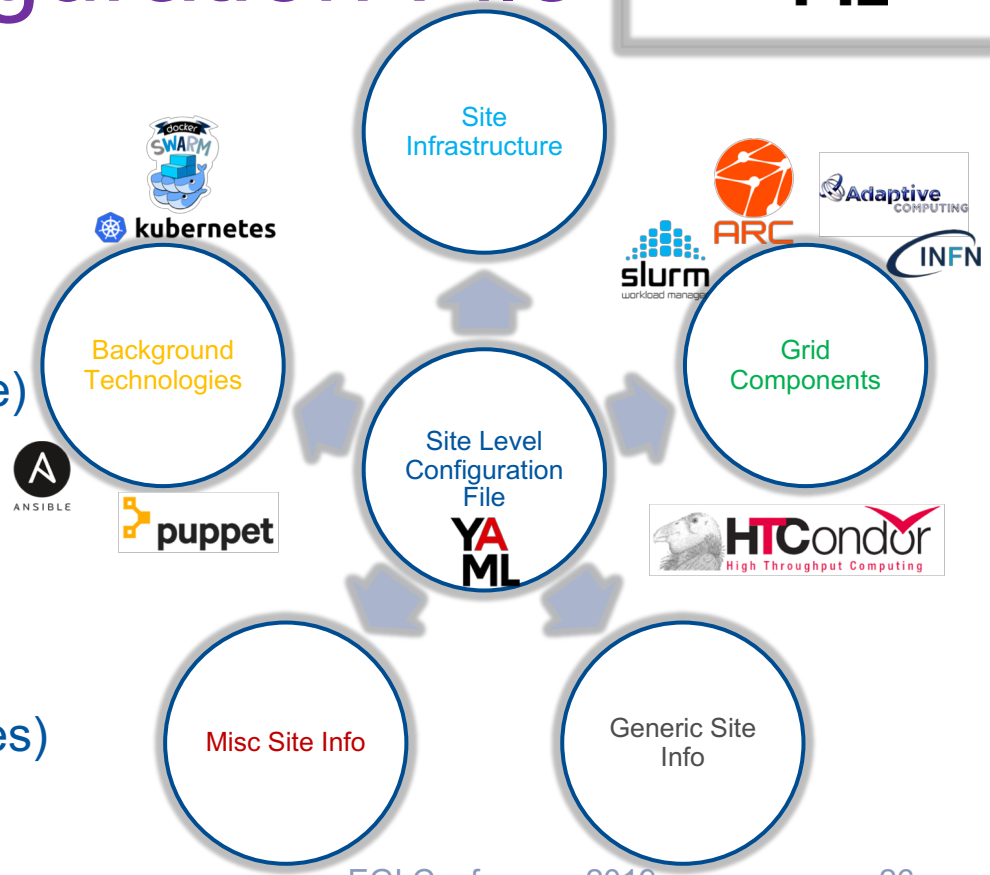
Site-Infrastructure (Hostnames, IP addresses, OS/Kernel, Disk/Memory)

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)



Section: site_infrastructure

Site-Infrastructure

Nodes on which grid services will be deployed

```
36  site_infrastructure:
37    - fqdn: ce01.mysite.domain
38      ip_address: 192.168.10.2
39    - fqdn: wn01.mysite.domain
40      ip_address: 192.168.10.70
41    ...
```

Section: lightweight_components

lightweight_components

URL for grid service container repository

Node on which grid service should be deployed

Config params for grid service container

```
75  - name: WN-Pbs
76    type: worker_node
77    repository_url: "https://github.com/WLCG-Lightweight-Sites/wlcg_lightweight_site_wn_pbs"
78    repository_revision: "master"
79    execution_id: 1
80    lifecycle_hooks:
81      pre_config:
82        - /etc/simple_grid/lifecycle/wn_pre_config.sh
83      pre_init:
84        - /etc/simple_grid/lifecycle/wn_pre_inst1.sh
85      post_init:
86        - /etc/simple_grid/lifecycle/wn_post_inst1.sh
87    deploy:
88      - node: wn01.mysite.domain
89        container_count: 2
90    preferred_tech_stack:
91      level_2_configuration: yaim
92    config:
93        ce_host: *pbs_runtime_var_ce_host
94        batch_server: pbs
95    supplemental_config:
96      some_additional_parameter: some_value
```

Section: lightweight_components

lightweight_components (advanced features)

```
75  - name: WN-Pbs
76    type: worker_node
77    repository_url: "https://github.com/WLCG-Lightweight-Sites/wlcg_lightweight_site_wn_pbs"
78    repository_revision: "master"
79    execution_id: 1
80    lifecycle_hooks:
81      pre_config:
82        - /etc/simple_grid/lifecycle/wn_pre_config.sh
83      pre_init:
84        - /etc/simple_grid/lifecycle/wn_pre_inst1.sh
85      post_init:
86        - /etc/simple_grid/lifecycle/wn_post_inst1.sh
87    deploy:
88      - node: wn01.mysite.domain
89        container_count: 2
90    preferred_tech_stack:
91      level_2_configuration: yaim
92    config:
93        ce_host: *pbs_runtime_var_ce_host
94        batch_server: pbs
95    supplemental_config:
96      some_additional_parameter: some_value
```

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

```
1  ### Variable declaration:
2  vars:
3    - &lightweight_component01_ip_address 192.168.0.4
4    - &lightweight_component01_fqdn lightweight_component01.cern.ch
5    - &lightweight_component02_ip_address 192.168.0.5
6    - &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.

```
 98  supported_virtual_organizations:
 99    - *default_vo_alice
100    - *default_vo_dteam
101    - *default_vo_ops
```

Advanced features

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

```
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

```
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:

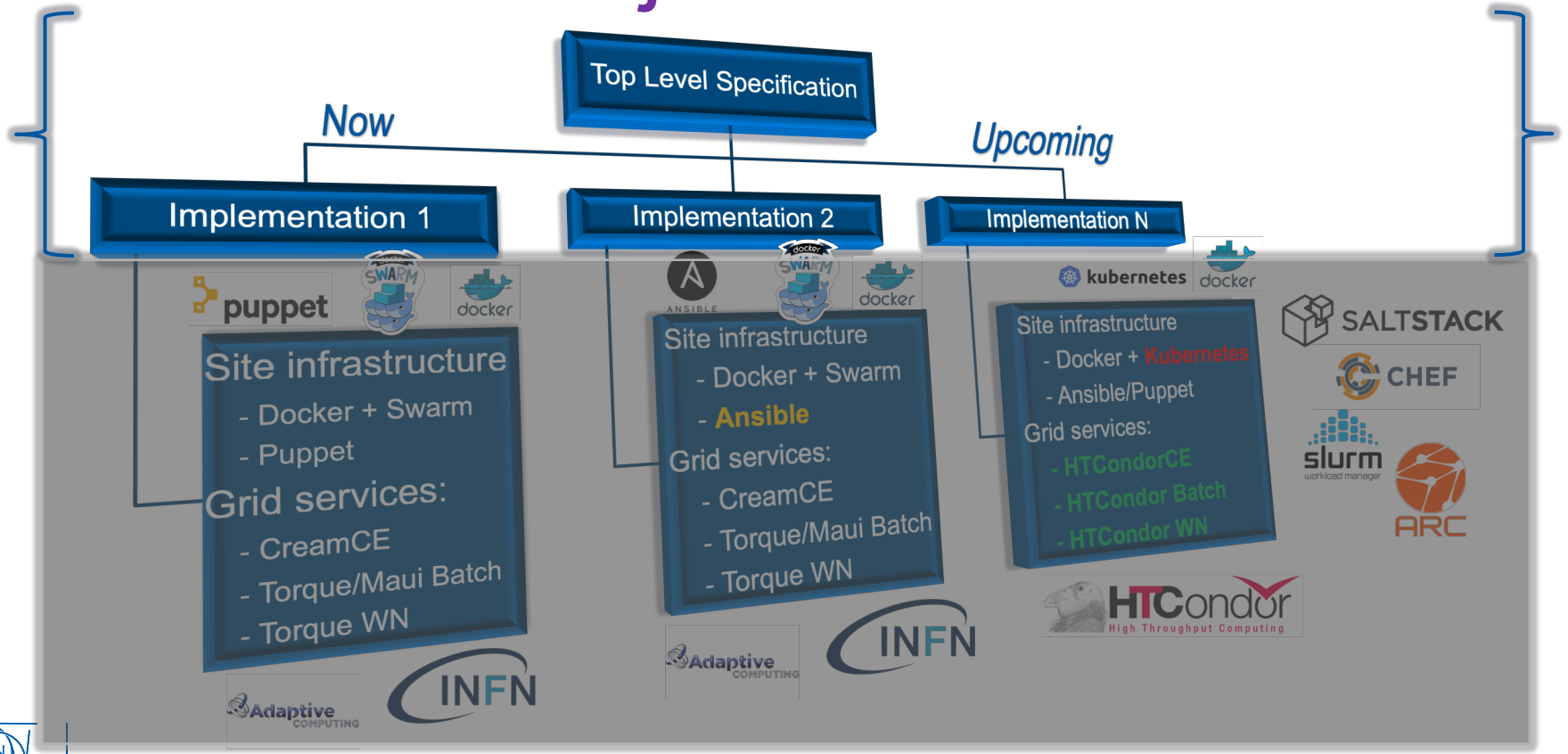
- Install Puppet on all the nodes.
- Install SIMPLE Grid Puppet module on all the nodes.
- Write the site-level-configuration file
- Execute the SIMPLE Grid Framework.

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.
  - **VO's** to be supported by their sites.
  - **Pool Account**s 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 – Project Structure



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 – 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.

Try it out via SIMPLE Dev-Kit

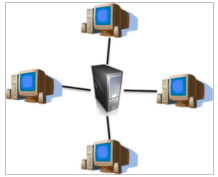
- Simulate a grid site on your machine.
- Used to Develop/Test/Debug the framework
- Works locally until the pre-deployment stage of execution pipeline.
- https://github.com/maany/simple_grid_puppet_dev_kit/tree/master

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!!

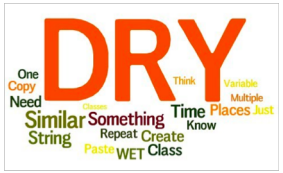
Principles

Abstraction



One node to configure the site

DRY (Don't Repeat Yourself)



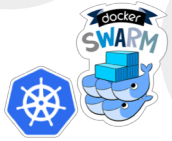
SIMPLE

Extensibility

Community Effort

Modularity

Simple Deployment



kubernetes

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

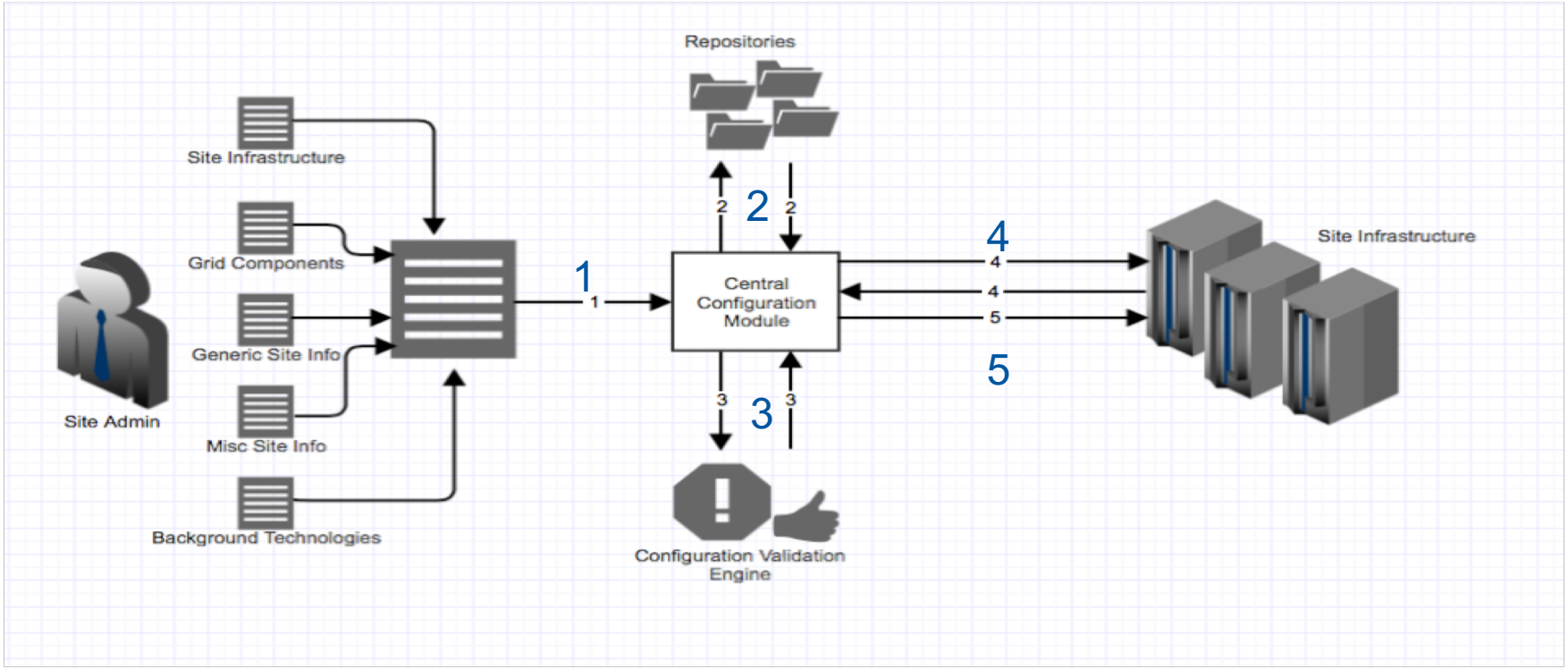
Central Configuration Manager

- 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

Google Summer of Code 2019 Project