Dynamic On-Demand Analysis Service (DODAS) & Kubernetes @CMS

Daniele Spiga - INFN - (spiga@infn.it)
On behalf of DODAS Team

Pre-GDB - Kubernetes many faces
CERN, Tuesday 10 December 2019

Outline

- Introduction to DODAS
  - Main concepts
- DODAS and Kubernetes
  - Stateless sites
    - Managing Compute and Data
- A note on AuthN/Z
- Summary and Future work

What is DODAS

Dynamic On Demand Analysis Service: DODAS

A INFN solution designed with the goal to enable users **to create and provision infrastructure deployments, automatically and repeatedly**, on “any cloud provider” **with almost zero effort**.

- Implement the **infrastructure as code paradigm**: driven by a templating engine to specify high-level requirements. Declarative approach **allows to describe “What” instead of “How”**
  - Let the underlying system to abstract providers and automatically instantiate and setup the computing system(s)
- Allows to instantiate **on-demand container-based clusters** (Mesos/**Kubernetes**) to execute software applications:
  - E.g. HTCondor batch system, Spark cluster, Data Caches…
  - **But also composition of services e.g. to manage stateless (WLCG-compliant) sites**

… and where it comes from

DODAS was initially prototyped within the **INDIGO-DataCloud project (2017)**

- Having in mind a primary use case: to develop an **effective solution for dynamic resource provisioning@CMS** (targeting Opportunistic computing)

Since then it has been **evolved**:

- In terms of supported **use cases** (from HTCondor to BigData platforms)
- In terms of adopted **technologies** (Mesos/Marathon, Kubernetes)
- In terms of supported **communities** (see later)

Currently the project is also supported by **EOSC-hub H2020 EU project as a Thematic Service.**

DODAS main concepts

Designed to:

- Support **user tailored** computing **environments**
- **Automate** configuration and deployment of custom services and/or dependencies
- Support declarative approach to define input parameters, customize workflows, **treating a collection of resources together as a single unit**

Highly flexible and modular solution enabling multiple usage patterns:

- Leverage clusters, possibly customising the stack, building highly reliable, highly scalable, applications
  - without worrying about creating and configuring the underlying infrastructure
    - **TOSCA + Ansible + Application setup (e.g Helm)**
- Generate Clusters on demand (**K8s on demand**), possibly customizing the underlying infrastructure
  - **TOSCA + Ansible**
  - And leave users **to focus just on applications**
    - **(e.g. Helm)**
- (Abstract the IaaS provisioning VMs/DB etc… not in scope with today's pre-GDB discussion)

DODAS and Kubernetes

TOSCA

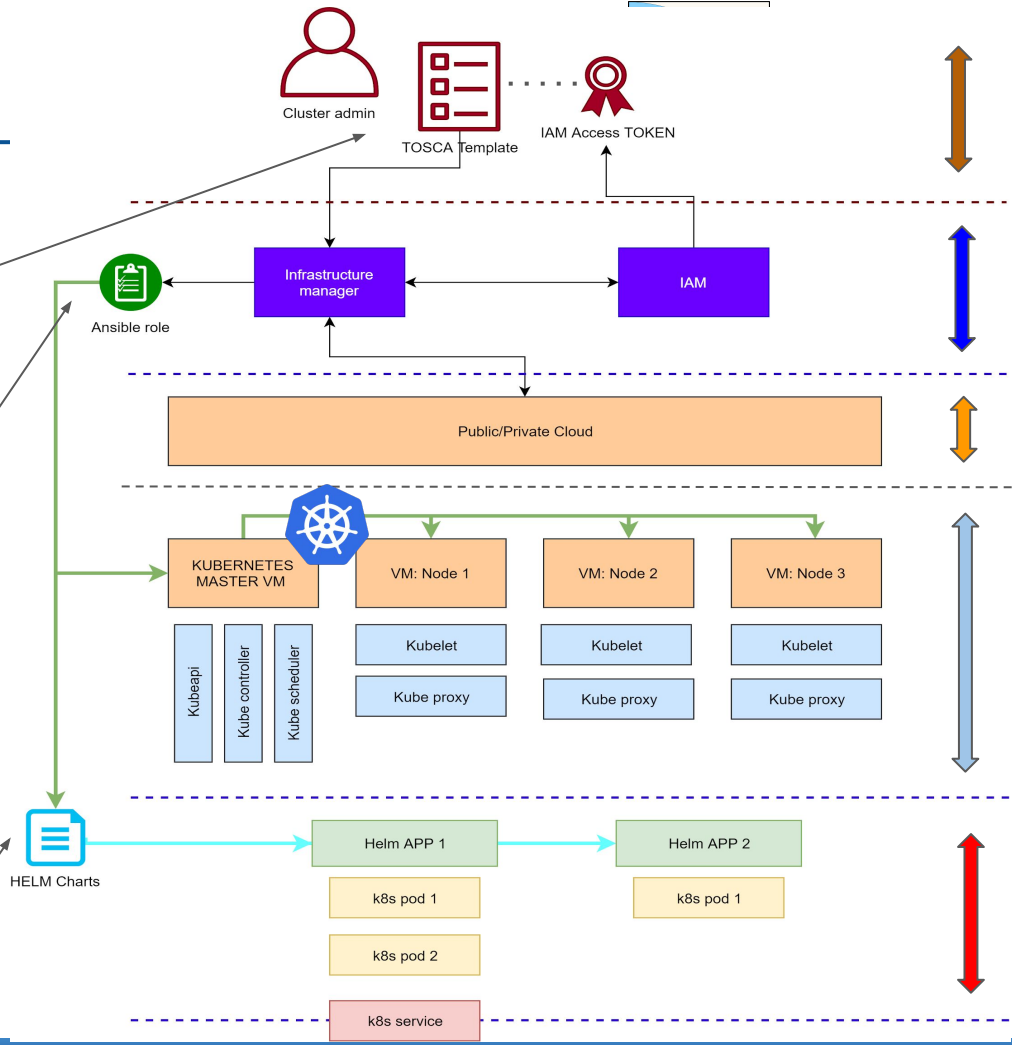
- Define the infrastructure (the HW)
- Define services (k8s) & Applications to setup (through Ansible)
- Declare (“any”) input parameters

Ansible based installation using:

- Kubeadm (initialization)
- Flannel (default but others available)
- nginx ingress (optional)
- k8s dashboard (optional)

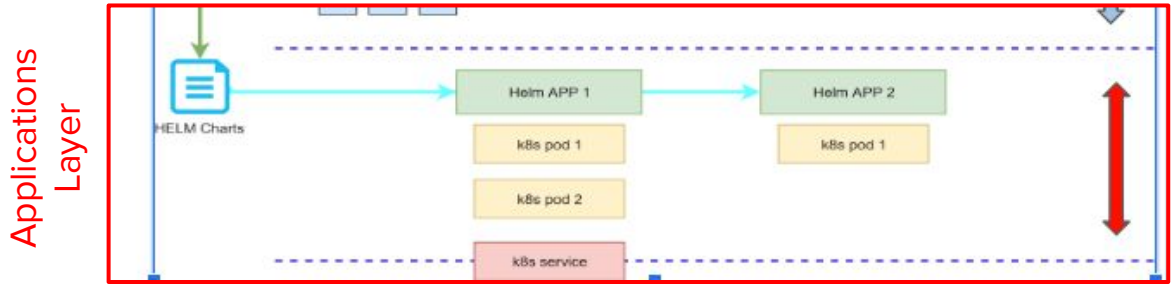
Helm (Applications layer)

- Configure at runtime. Dynamically load and compile values (from TOSCA through ansible)
- Install applications



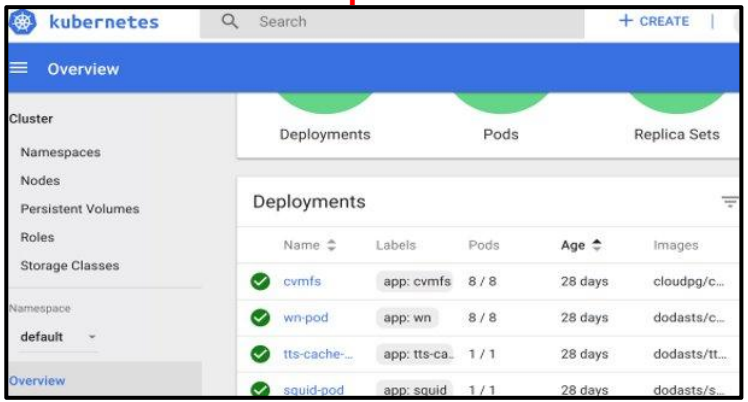
Implementing **stateless sites** with DODAS

Provides remotely deployed CPU clusters and data Caches. Managed through K8s

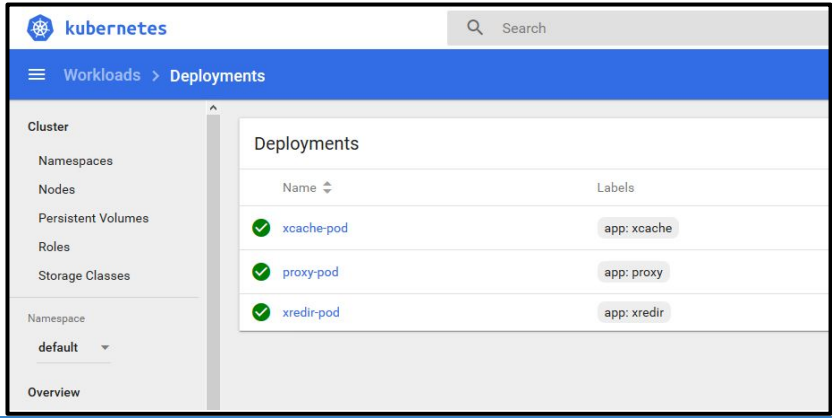


- **No Dedicated** experiment **support**
- **No site specific setup** required

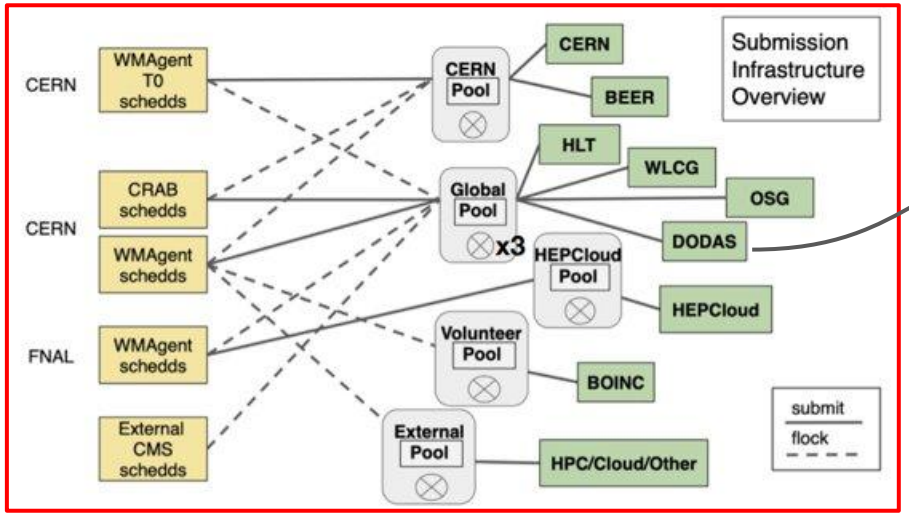
Compute

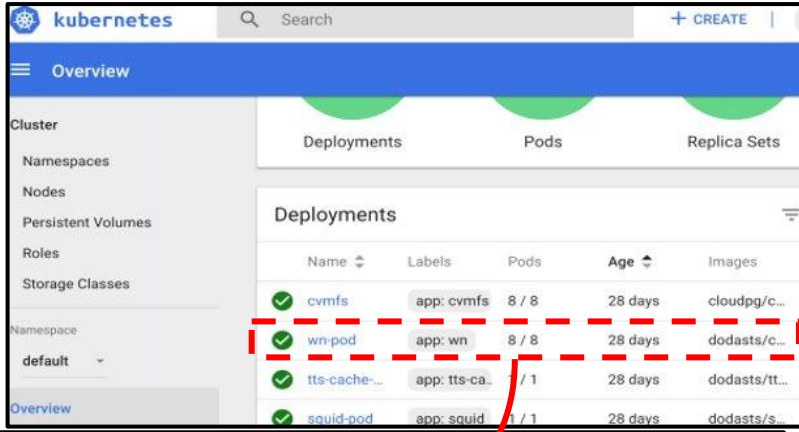


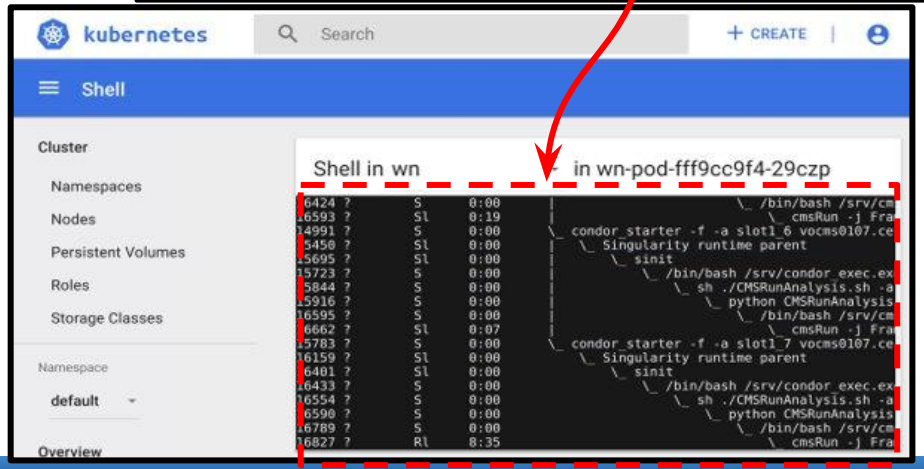
Data



Example: The CMS Integration







A Key component is the AuthN/Z:

DODAS is based on JWT (INDIGO-IAM). To integrate the CMS GlobalPool:

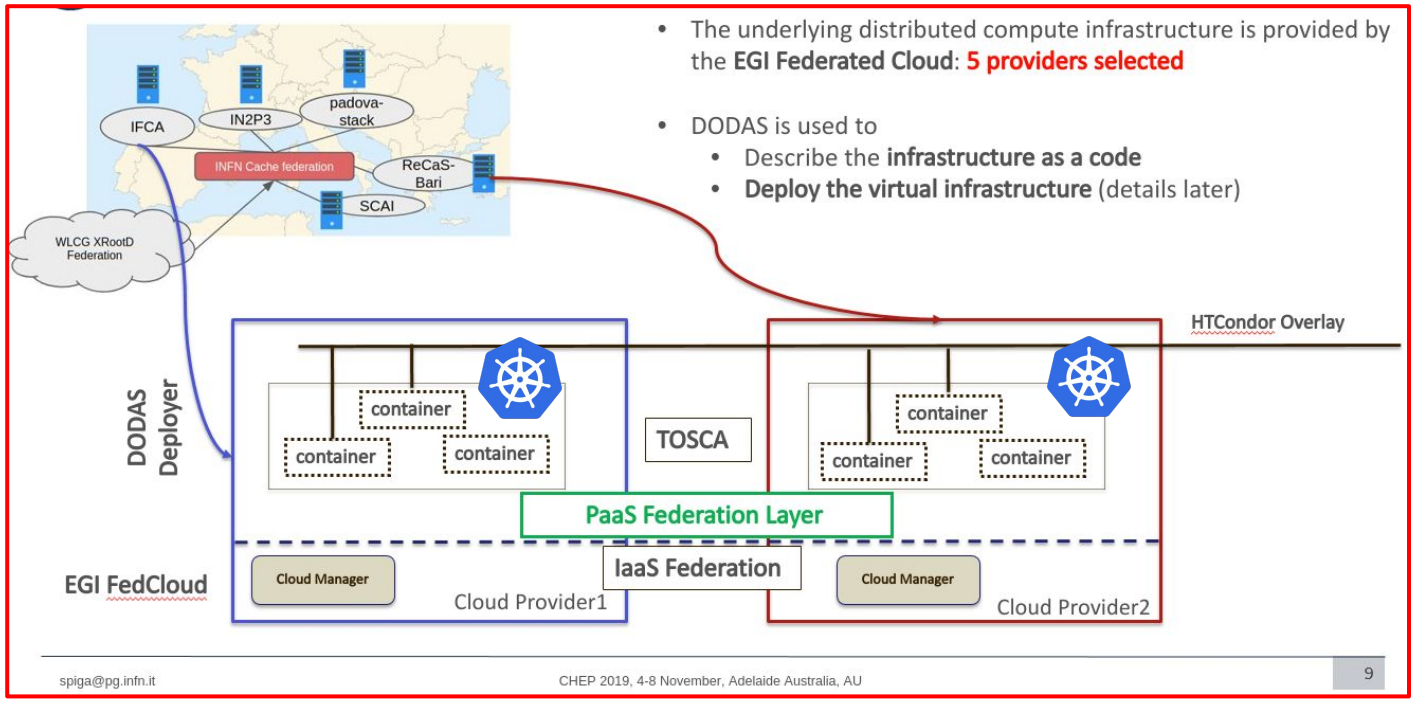
- start with JWT Token as incoming auth credential
- Implements security via IAM token exchange
- On-demand X.509 certificates generated from IAM tokens (and cached) to enable access to CMS

--> Global Pool authorization is based on DN mapping

A recent setup

Many tests and deployments in the past two years (see results @ CHEP2018). Recently we used DODAS to manage **5 stateless sites**... CMS just sees it as a single virtual site, thanks to the **HTCondor overlay**

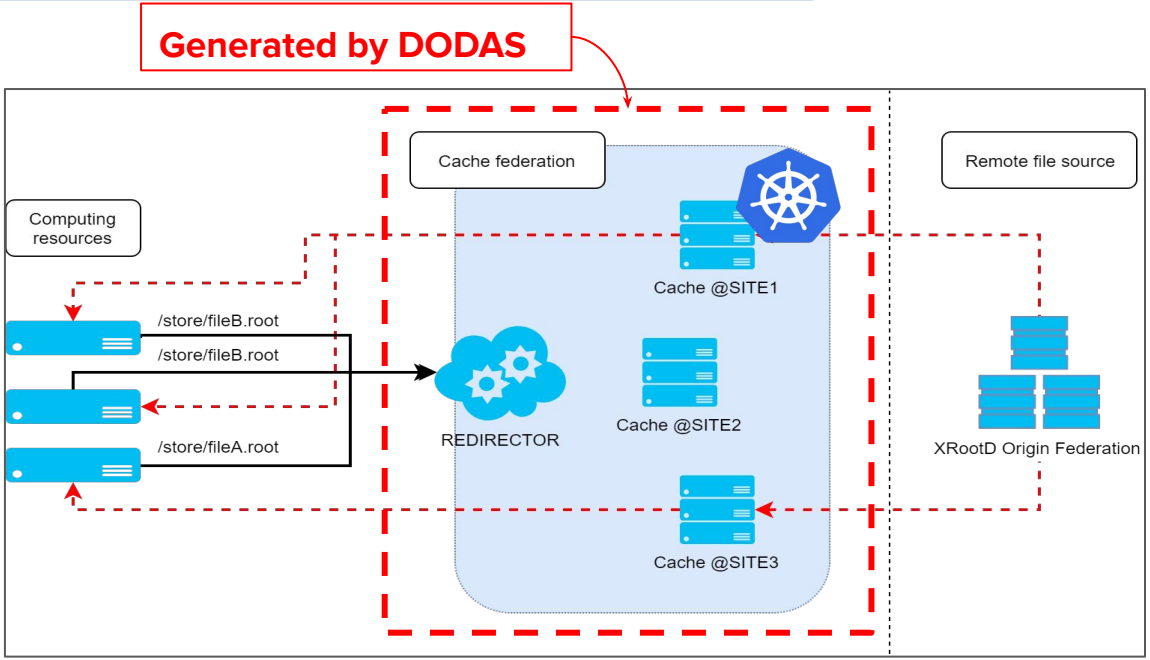
From CHEP 2019



DODAS and Data Caches

We rely on **XRootD** technology and we support configuration of a variety of services.

- **Data Server**:
- **Redirector**:
- **XCache**:



Cache development in collaboration with **SoCaL US project**

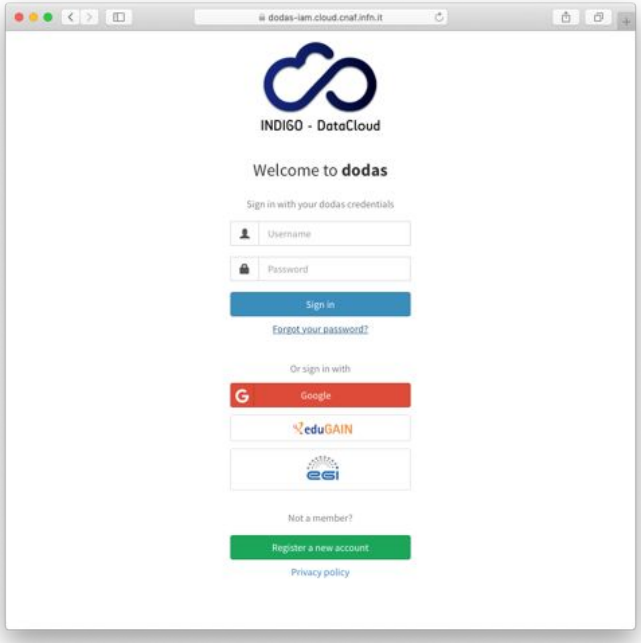
```
$> helm repo add cache https://cloud-pg.github.io/CachingOnDemand/
$> helm repo update
$> helm install cache/cachingondemand
```

Aside note : AuthN/Z

DODAS relies on the INDIGO-Identity Access Management to manage Authentication and Authorization

A VO-scoped authentication and authorization service that

- supports multiple authentication mechanisms
- provides users with a persistent, VO-scoped identifier
- exposes identity information, attributes and capabilities to services via JWT tokens and
- standard OAuth & OpenID Connect protocols
- can integrate existing VOMS-aware services
- supports Web and non-Web access, delegation and token renewal



Finally: Not Only CMS

DODAS is also under evaluation (at different level of testing and integration) by **communities other than CMS**

- **AMS Experiment** is already testing/evaluating DODAS to run analysis over opportunistic resources
- **Fermi** analysts are already using (for daily activities) DODAS
- **Virgo** is integrating a pipeline for testing the whole flow

Summary

DODAS is a high modular deployer manager built on the concept of Infrastructure as a code to create and provision infrastructure deployments, **automatically and repeatedly**. Today

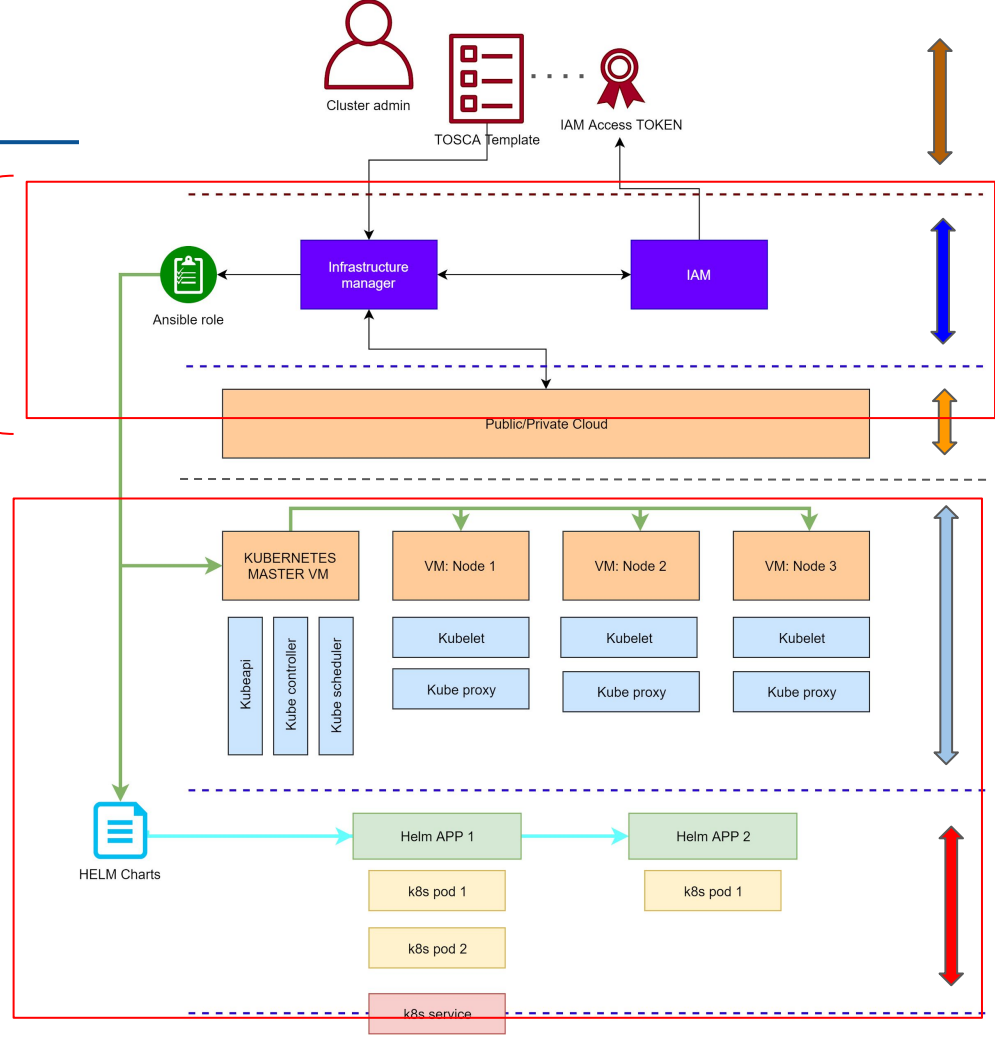
- We primarily discussed how we use DODAS to support **CMS stateless sites on K8s**
  - includes compute and data creation and orchestration and federation (XCache service)
- We shown how DODAS can **provision K8s on demand**
  - Allowing the customisation of the underlying environment (e.g. dependencies) and the definition of the computational stack (DBs, FS, storages/caches)

DODAS also supports an **on-demand analysis facilities** offering (**on top of K8s**):

- HTCondor batch systems on demand, supporting HTCondor federations
  - Floking, routing and HTC/HPC mixing (CHEP2019)
- Spark cluster, TFaaS, etc..

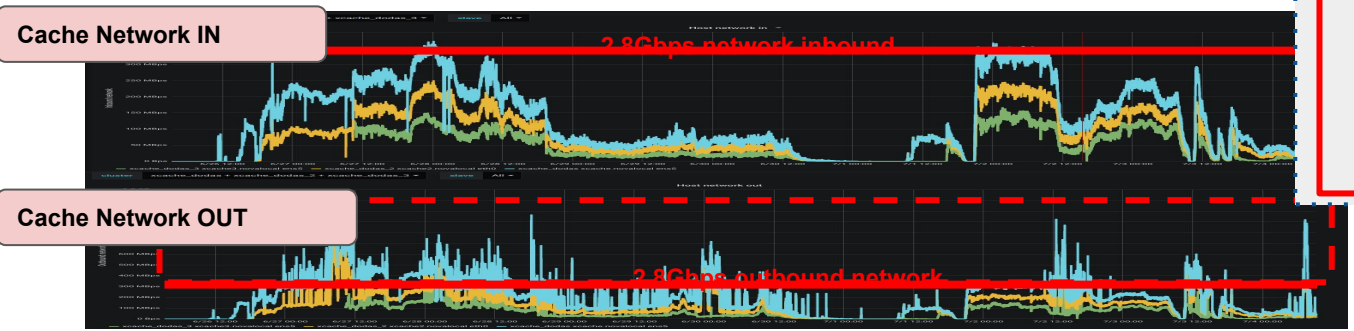
Future work and R&D

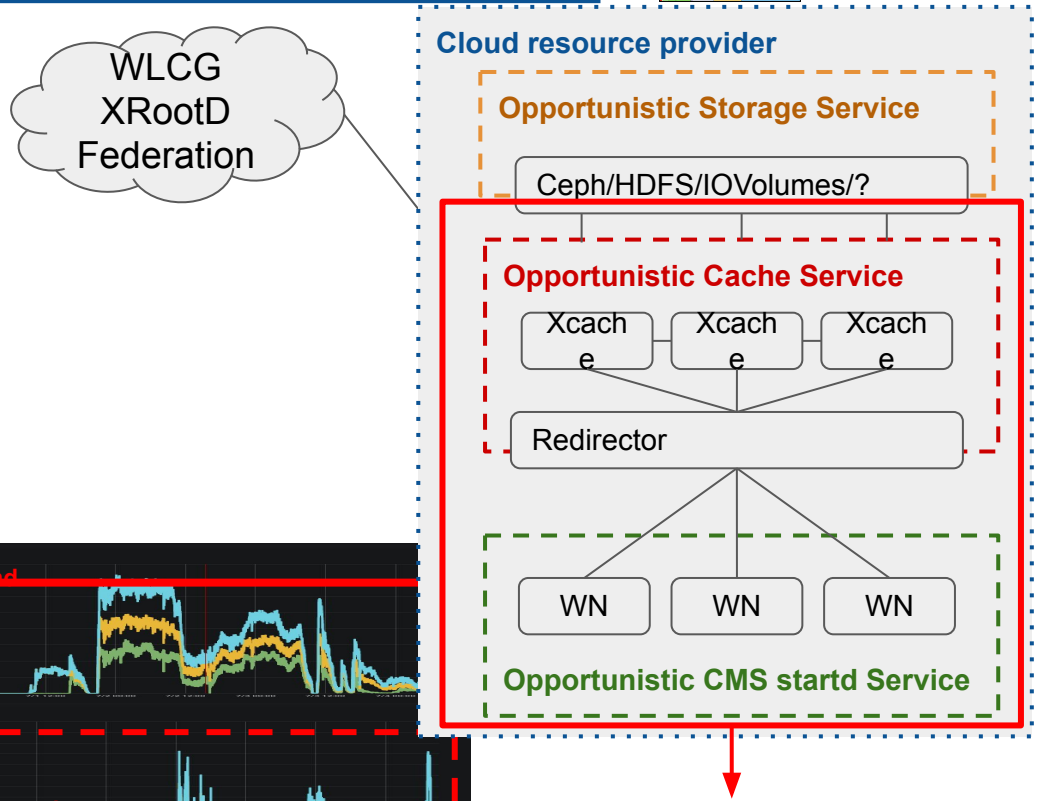
- DODAS specific
  - Improve and evolve the support for bare metal (instead of Cloud API)
  - Improve/evolve User interface (GUI/CLI)
- K8s oriented:
  - Autoscalers with custom metrics
  - Federated k8s
  - Integrated AuthN/Z layer



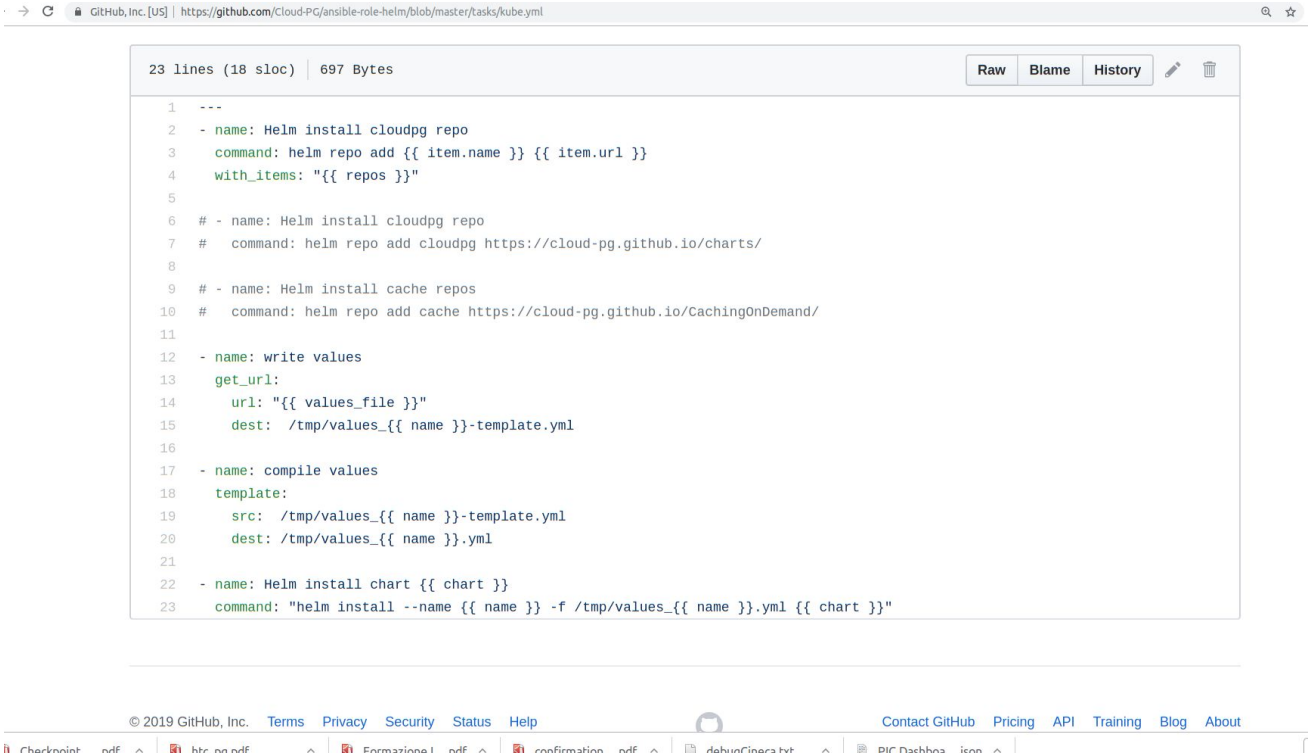
Backup

Putting everything together





Generated by DODAS


GitHub, Inc. [US] | https://github.com/Cloud-PG/ansible-role-helm/blob/master/tasks/kube.yml
23 lines (18 sloc) | 697 Bytes
Raw
Blame
History

- name: Helm install cloudpg repo
command: helm repo add {{ item.name }} {{ item.url }}
with_items: "{{ repos }}"
- name: Helm install cloudpg repo
command: helm repo add cloudpg https://cloud-pg.github.io/charts/
- name: Helm install cache repos
command: helm repo add cache https://cloud-pg.github.io/CachingOnDemand/
- name: write values
get_url:
url: "{{ values_file }}"
dest: /tmp/values_{{ name }}-template.yml
- name: compile values
template:
src: /tmp/values_{{ name }}-template.yml
dest: /tmp/values_{{ name }}.yml
- name: Helm install chart {{ chart }}
command: "helm install --name {{ name }} -f /tmp/values_{{ name }}.yml {{ chart }}"
© 2019 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About