DODAS
to support small research groups

Tracolli Mirco on behalf of DODAS Team
INFN Perugia

Workshop CCR: Rimini, 11 - 15 june 2018

Outline

- What is DODAS
  - Origin
  - Design
- How DODAS is useful for the communities
- Workflow customization example
- Support and further information about DODAS

DODAS

Dynamic On Demand Analysis Service: it's a Platform as a Service tool built combining several solutions and products developed by INDIGO-DataCloud. Currently, it’s a Thematic Service in the context of EOSC-hub H2020 project.

In detail, DODAS is a service for generating over cloud resources an on-demand container based solution to:

- Instantiate a standalone HTCondor batch system
- Instantiate cluster Big Data processing

Architecture

Data analysts

INDIGO - DataCloud

Model

Service

Container | Container | Container

Framework

Resource Manager

Actual implementation

Service

docker

Marathon, Chronos, Spark etc.

Apache MESOS

Ansible

DODAS User

Infrastructure
(Openstack etc.)

Auth

State of the art

Two different and working type of clusters:

- HTCondor standalone batch system
- Big Data: Spark + HDFS linked storage

as a Service

For the first solution we have two different customization workflow:

- CMS use case
- AMS use case:
  Talk on Friday 15: **“The AMS and DAMPE computing models and their integration into DODAS”**, *Matteo Duranti (PG)*

Requirements from communities perspectives

- Custom workflow:
  - Softwares, libraries, frameworks
  - Custom container images
  - Custom roles

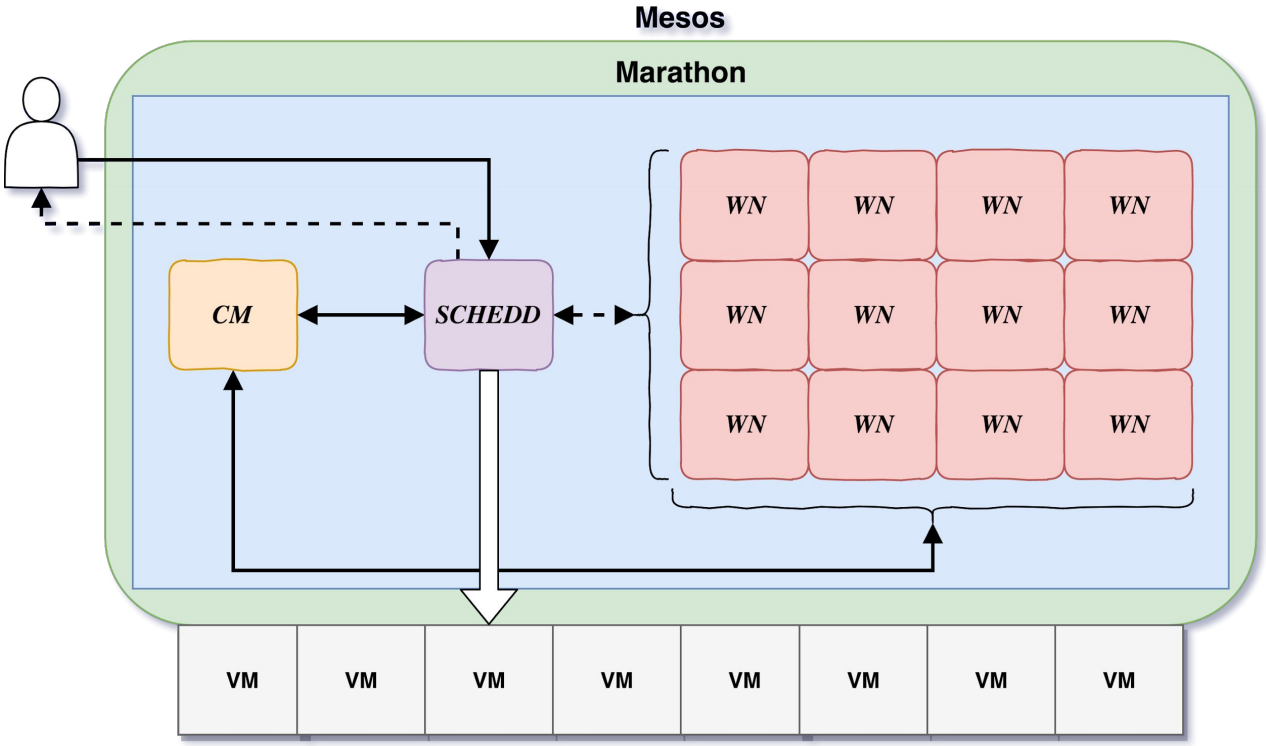
  Personalize through:
  - TOSCA template
  - Ansible scripts
  - Docker images

- AuthN/Z:
  - JWT standard support
  - Guarantee integration with legacy systems through plugins

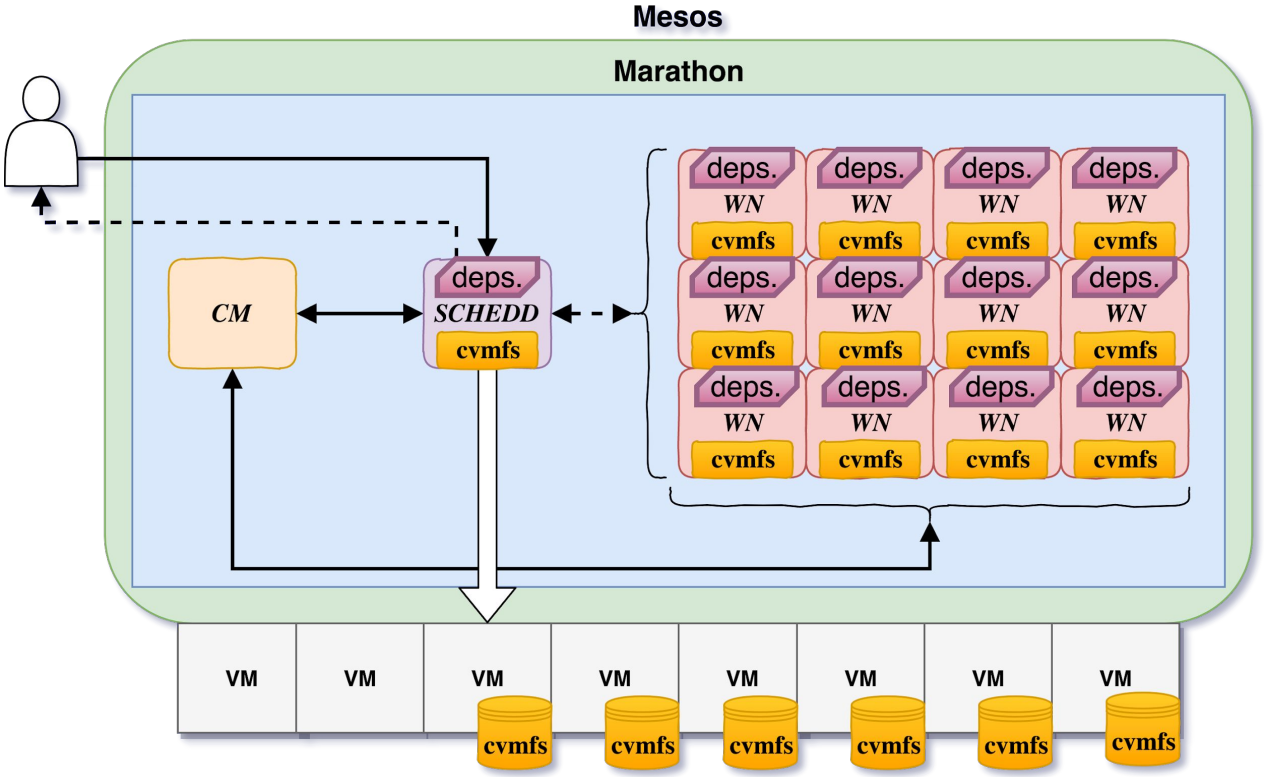
  - Indigo IAM

    - Talk **“Oltre X.509: autenticazione e autorizzazione con OpenID-Connect e OAuth utilizzando l’INDIGO-DataCloud Identity and Access Management (IAM) Service”,** *Andrea Ceccanti (CNAF)*, past Tuesday 12 June

DODAS HTCondor Service



AMS customization example



AMS customization - TOSCA

```
htcondor_services:
  type: tosca.nodes.indigo.HTCondorServices
  properties:
    master_ips: {
      get_attribute: [ mesos-master-server, private_address ] }
    htcondor_config_schedd_ip: {
      get_attribute: [ mesos-slaveschedd-server, private_address, 0 ] }
    config_mode: "master"
  requirements:
    - host: mesos_master
```

Base HTCondor service

Custom AMS service

```
ams_services:
  type: tosca.nodes.indigo.AmsCondorMasterConfig
  properties:
    master_ips: {
      get_attribute: [ mesos-master-server, private_address ] }
    htcondor_config_schedd_ip: {
      get_attribute: [ mesos-slaveschedd-server, private_address, 0 ] }
    number_of_wn_instances: { get_input: number_of_wn_instances }
    number_of_slaves: { get_input: number_of_slaves }
    cpu_x_wn: { get_input: cpu_x_wn }
    ram_x_wn: { get_input: ram_x_wn }
    docker_cpu_x_wn: { get_input: docker_cpu_x_wn }
    docker_ram_x_wn: { get_input: docker_ram_x_wn }
    ams_use_local_squid: { get_input: ams_use_local_squid }
    ams_default_squid: { get_input: ams_default_squid }
    ams_default_squid_port: { get_input: ams_default_squid_port }
    ams_repo_server_url: { get_input: ams_repo_server_url }
    ams_repo_public_key_path: { get_input: ams_repo_public_key_path }
    ams_repo_http_proxy: { get_input: ams_repo_http_proxy }
    ams_repo_repository_name: { get_input: ams_repo_repository_name }
    ams_repo_public_key_url: { get_input: ams_repo_public_key_url }
    ams_repo_public_key: { get_input: ams_repo_public_key }
  requirements:
    - host: mesos_master
    - host: htcondor_services
```

AMS customization - Ansible

```
#·[ ... ]

-·name:·install·cvmfs·repo
··apt:·deb=https://ecsft.cern.ch/dist/cvmfs/cvmfs-release/cvmfs-release-latest_all.deb
··tags:·ams_config

-·name:·Install·cvmfs·packages
··apt:·name={{item}}·state=present·update_cache=yes
··with_items:
·····-·cvmfs
·····-·cvmfs-config-default
··tags:·ams_config

#·[ ... ]

-·name:·set·AMS·repo
··blockinfile:
····dest:·/etc/cvmfs/config.d/ams.cern.ch.local
····create:·yes
····content:·|
·····#CVMFS_SERVER_URL="http://cvmfs-stratum-one.cern.ch/opt/@org@"
·····CVMFS_SERVER_URL="http://cvmfs-stratum-one.cern.ch/cvmfs/ams.cern.ch"
·····CVMFS_HTTP_PROXY=DIRECT
··tags:·ams_config_cvmfs_repo

#·[ ... ]
```

Example: setup and configuration of a CVMFS repository

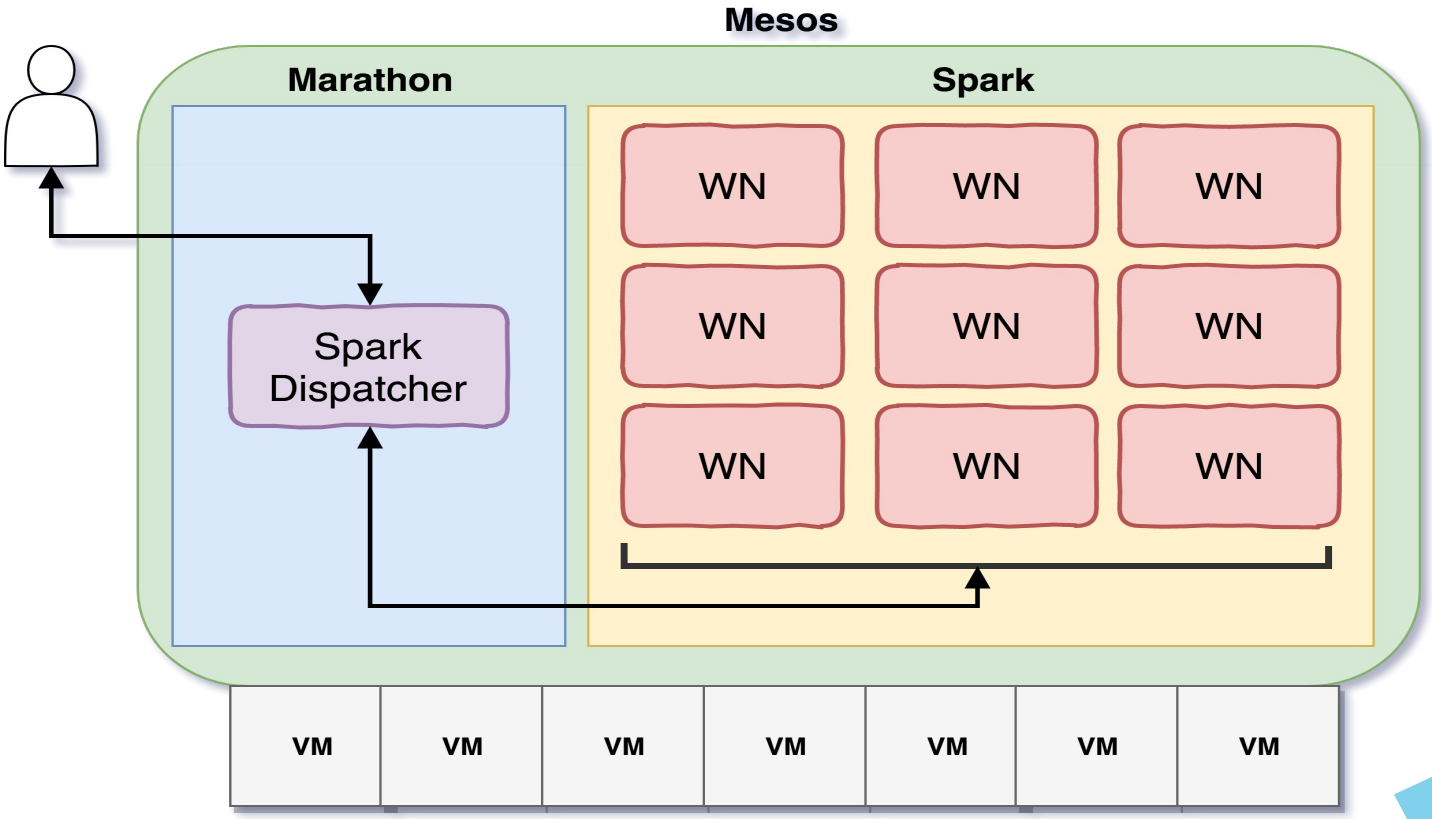
AMS customization Docker image

```
FROM cloudpg/dodas-htcondor

RUN yum --setopt=tsflags=nodocs -y install binutils \
        boost-devel \
        cmake \
        curl \
        cvs \
        emacs \
        freetype \
        freetype-devel \
        fuse \
        gcc \
        gcc-c++ \
        gcc-gfortran \
        git \
        glibc-devel \
        glibc-headers \
        gsl-devel \
        initscripts \
```

Extend base docker image with necessary packages, frameworks, libraries etc.

Another example
Big Data Cluster - Schema



Summary

- DODAS is a service to allow the exploitation of any cloud with almost zero effort
  - Support different type of infrastructures
  - Customization at any level of the stack
- DODAS service is a ready for community exploitation
- As Thematic Service (EOSC-hub) is available for exploitation
  - Relies on resources Cloud@CNAF and ReCaS@Bari

People interested can ask for support here: dodas-support@lists.infn.it

Link to the user guide: https://dodas.gitbook.io/dynamic-on-demand-analysis-service/

References

- Base HTCondor image: https://hub.docker.com/r/cloudpg/dodas-htcondor/
- Personalized Docker image: https://hub.docker.com/r/cloudpg/dodas-ams/
- Configurable Ansible role: https://github.com/indigo-dc/ansible-role-htcondor_config/tree/condor_base
- Personalized Ansible role: https://github.com/indigo-dc/ansible-role-ams_config/tree/condor_base
- Spark on Mesos: https://spark.apache.org/docs/latest/running-on-mesos.html

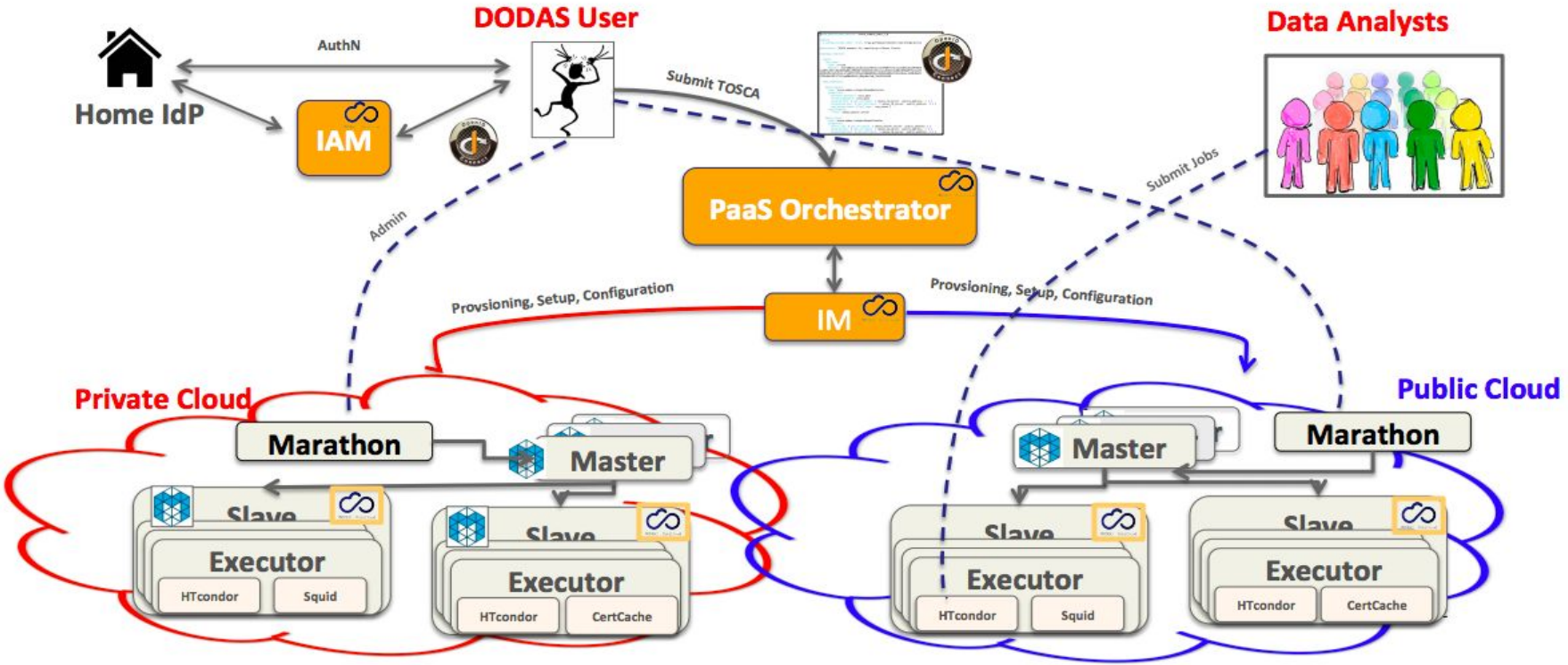
Backup

AMS customization Ansible

```
{
  "id": "condorschedd",
  "cpus": 2.4,
  "mem": 3200.95,
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "{{htcondor_config_condor_image}}",
      "forcePullImage": true,
      "privileged": true,
      "network": "HOST"
    },
    "volumes": [
      {
        "containerPath": "/cvmfs",
        "hostPath": "/cvmfs",
        "mode": "RW"
      },
      {
        "containerPath": "/home/uwdir",
        "hostPath": "/tmp/uwdir",
        "mode": "RW"
      }
    ]
  }
```

```
{
  "id": "condorwn",
  "cpus": {{ams_docker_cpu_x_wn}},
  "mem": {{ams_docker_ram_x_wn}},
  "instances": {{number_of_wn_instances}},
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "{{htcondor_config_condor_image}}",
      "forcePullImage": true,
      "privileged": true,
      "network": "BRIDGE"
    },
    "volumes": [
      {
        "containerPath": "/cvmfs",
        "hostPath": "/cvmfs",
        "mode": "RW"
      }
    ]
  },
  [ ... ]
```

A insider look at DODAS



Docker htcondor image

```
FROM cloudpg/centos-7-grid-tini-sshd

WORKDIR /etc/yum.repos.d

RUN useradd -ms /bin/bash condor \
    && wget http://research.cs.wisc.edu/htcondor/yum/repo.d/htcondor-development-rhel7.repo \
    && wget http://research.cs.wisc.edu/htcondor/yum/repo.d/htcondor-stable-rhel7.repo \
    && wget http://research.cs.wisc.edu/htcondor/yum/RPM-GPG-KEY-HTCondor \
    && rpm --import RPM-GPG-KEY-HTCondor \
    && yum --setopt=tsflags=nodocs -y update \
    && yum --setopt=tsflags=nodocs -y install \
        condor-all \
        gcc \
        gcc-c++ \
        make \
        openssh-clients \
        openssh-server \
        python-devel \
        python-pip \
    && yum clean all \
    && pip install --upgrade pip setuptools \
    && pip install j2cli paramiko psutil kazoo requests \
    && systemctl disable condor

# Root home
WORKDIR /root
```

Base dependencies and exposed ports

```
# condor_collector
EXPOSE 9618
# condor_negotiator
EXPOSE 9614
# condor_ckpt_server
EXPOSE 5651-5654
# condor_ports
EXPOSE 1024-2048
```

```
# Condor env default values
ENV CONDOR_DAEMON_LIST="COLLECTOR, MASTER, NEGOTIATOR, SCHEDD, STARTD"
ENV CONDOR_HOST="\$(FULL_HOSTNAME)"
ENV CCB_ADDRESS_STRING=""
ENV NETWORK_INTERFACE_STRING=""
ENV CONDOR_SCHEDD_SSH_PORT=31042
ENV TUNNEL_FROM="UNDEFINED"
ENV TUNNEL_TO="UNDEFINED"
ENV SEC_DAEMON_AUTHENTICATION_METHODS=CLAIMTOBE
ENV SEC_CLIENT_AUTHENTICATION_METHODS=CLAIMTOBE
ENV SEC_NEGOTIATOR_AUTHENTICATION_METHODS=CLAIMTOBE
ENV SEC_ADVERTISE_STARTD_AUTHENTICATION_METHODS=CLAIMTOBE
ENV NUM_SLOTS=1
ENV NUM_SLOTS_TYPE_1=1
ENV SLOT_TYPE_1="cpus=1, mem=4096"
ENV FLOCK_FROM=""
ENV FLOCK_TO=""
ENV FLOCK_TO_COL_NEG=""
ENV HOST_ALLOW_FLOCK=""
```

Personalize HTCondor environment

```
RUN mkdir -p /opt/dodas/htc_config \
    && mkdir -p /opt/dodas/fs_remote_dir \
    && mkdir -p /opt/dodas/health_checks \
    && mkdir -p /etc/skel/.ssh
COPY condor.sh /opt/dodas/
COPY ./health_checks/check_condor_processes.py /opt/dodas/health_checks/
COPY ./health_checks/check_cvmfs_folders.py /opt/dodas/health_checks/
COPY ./health_checks/check_ssh_server.py /opt/dodas/health_checks/
COPY ./health_checks/check_condor_master_ip.sh /opt/dodas/health_checks/
COPY ./health_checks/check_condor_schedd_tunnel.sh /opt/dodas/health_checks/
COPY cache.py /opt/dodas/
COPY ./config/condor_config.template /opt/dodas/htc_config/
```

Init scripts and health checks

```
RUN ln -s /opt/dodas/condor.sh /usr/local/sbin/dodas_condor \
    && ln -s /opt/dodas/health_checks/check_condor_processes.py /usr/local/sbin/dodas_check_condor_processes \
    && ln -s /opt/dodas/health_checks/check_cvmfs_folders.py /usr/local/sbin/dodas_check_cvmfs_folders \
    && ln -s /opt/dodas/health_checks/check_ssh_server.py /usr/local/sbin/dodas_check_ssh_server \
    && ln -s /opt/dodas/health_checks/check_condor_master_ip.sh /usr/local/sbin/dodas_check_condor_master_ip \
    && ln -s /opt/dodas/health_checks/check_condor_schedd_tunnel.sh /usr/local/sbin/dodas_check_condor_schedd_tunnel \
    && ln -s /opt/dodas/cache.py /usr/local/sbin/dodas_cache

# CentOS uname characteristics
RUN mv /bin/uname /bin/uname_old
COPY ./bin/uname /bin/

ENTRYPOINT ["/usr/bin/tini", "--", "/usr/local/sbin/dodas_condor"]
```

Different resource manager

A possible alternative is Kubernetes because:

- Fit the container model of DODAS
- Can manage groups of apps (PODS)
- It is a solution cloud oriented

Cons:

- Different management of app lifecycle
- Lack of framework layer (to manage resources or different software that can't be containerized)

Why Ansible?

- Template based
- Extended by plugin or modules
- Based on Python (more multi platform oriented)
- Don’t need a service but only access to hosts