Pegasus Workflows on OLCF - Summit

George Papadimitriou
georgpap@isi.edu

USC Viterbi
School of Engineering
Information Sciences Institute

http://pegasus.isi.edu

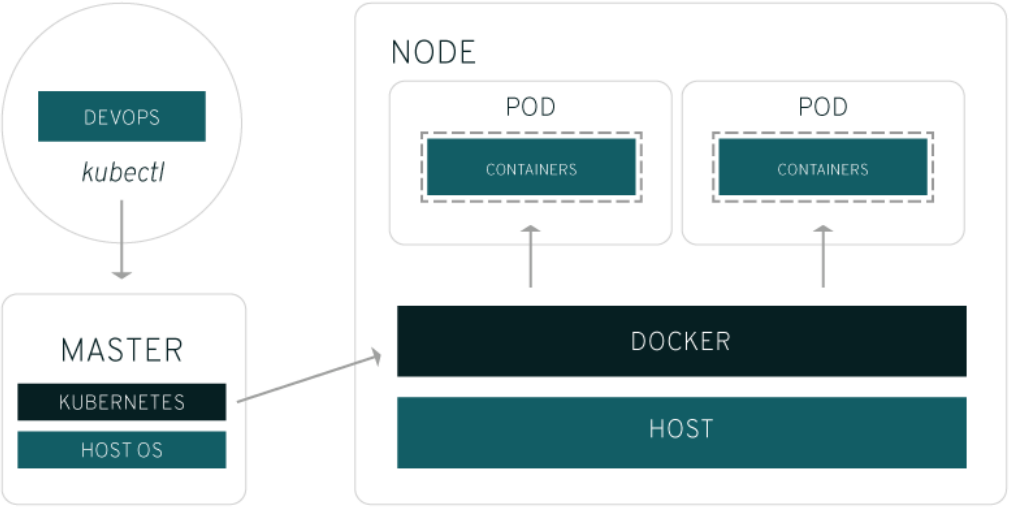
Outline

- Kubernetes/OpenShift
  - What is Kubernetes (Specs, Pods, Services)
  - Why use Kubernetes in HPC
  - Openshift at OLCF
  - Pegasus Deployment on Openshift at OLCF
- How to Deploy
  - Prerequisites
  - Instructions
- Demo
  - Pegasus Workflow on Summit

Kubernetes

Kubernetes: Brief Overview

- **Kubernetes** is an open-source platform for running and coordinating containerized application across a cluster of machines.
- It can be useful for:
  - Orchestrating containers across multiple hosts
  - Control and automate deployments
  - Scale containerized applications on the fly
  - And more…



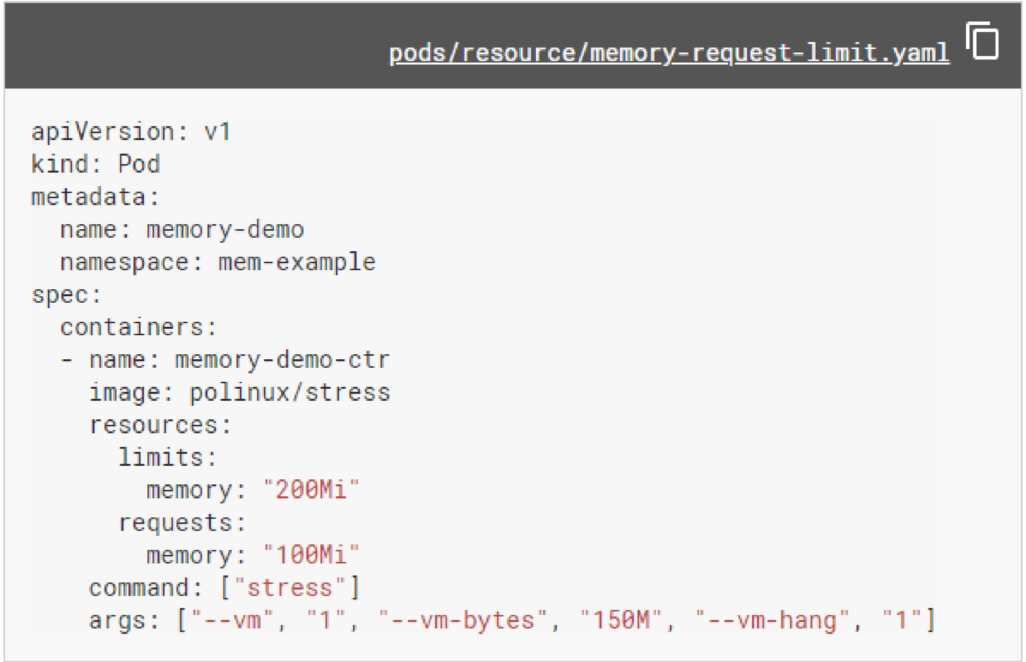
Reference:
https://www.redhat.com/en/topics/containers/what-is-kubernetes

- **Key objects** in the Kubernetes architecture are:
  - Master: Controls Kubernetes nodes – assign tasks
  - Node: Perform the assigned tasks
  - Pod: A group of one or more containers deployed on a single node
  - Replication Controller: Controls how many copies of a pod should be running
  - Service: Allow pods to be reached from the outside world
  - Kubelet: Runs on the nodes and starts the defined containers

Kubernetes: Configuring Objects

- Within Kubernetes, specification files describe the applications, services and objects being deployed

- Specification files can be written in YAML and JSON formats and can be used to
  - Deploy Pods
  - Create and mount volumes
  - Expose services etc.



pods/resource/memory-request-limit.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo
  namespace: mem-example
spec:
  containers:
  - name: memory-demo-ctr
    image: polinux/stress
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

Reference:
https://kubernetes.io/docs/tasks/configure-pod-container/

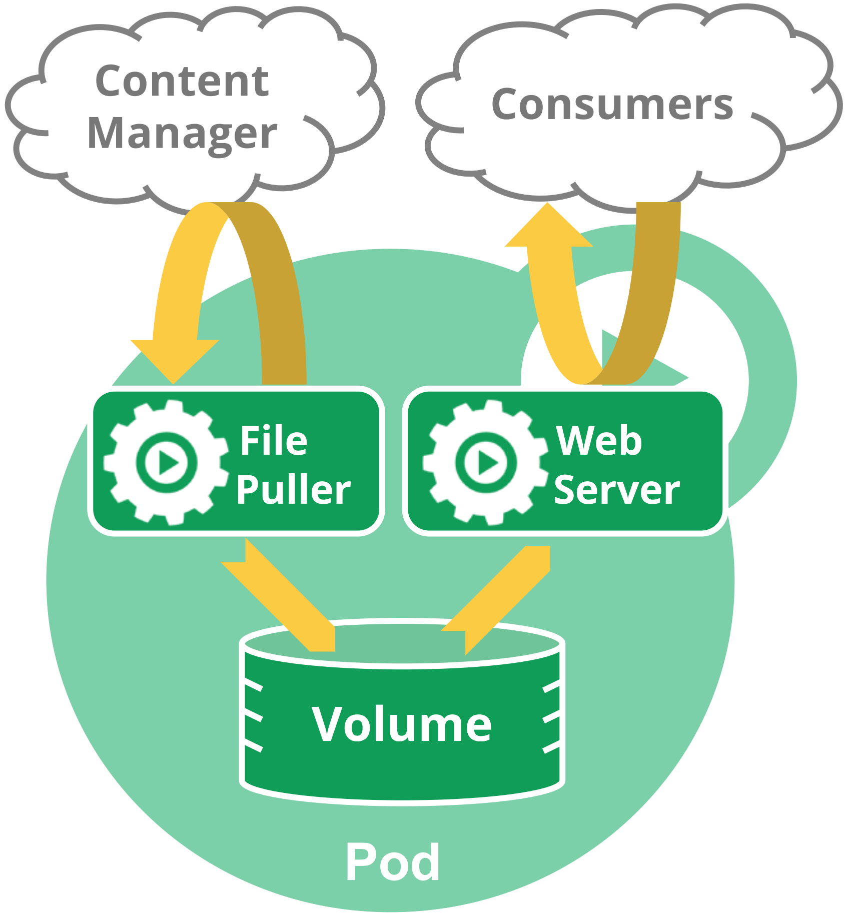
Kubernetes: Pods

- A **Pod** is the basic execution unit of a Kubernetes application
- Pods represent processes running on the cluster
- One can have one or multiple containers running within a Pod.

- **Networking:** Each Pod is assigned a unique IP address within the cluster

- **Storage:** A Pod can specify a set of shared storage Volumes. Volumes persist data and allow Pods to maintain state between restarts.

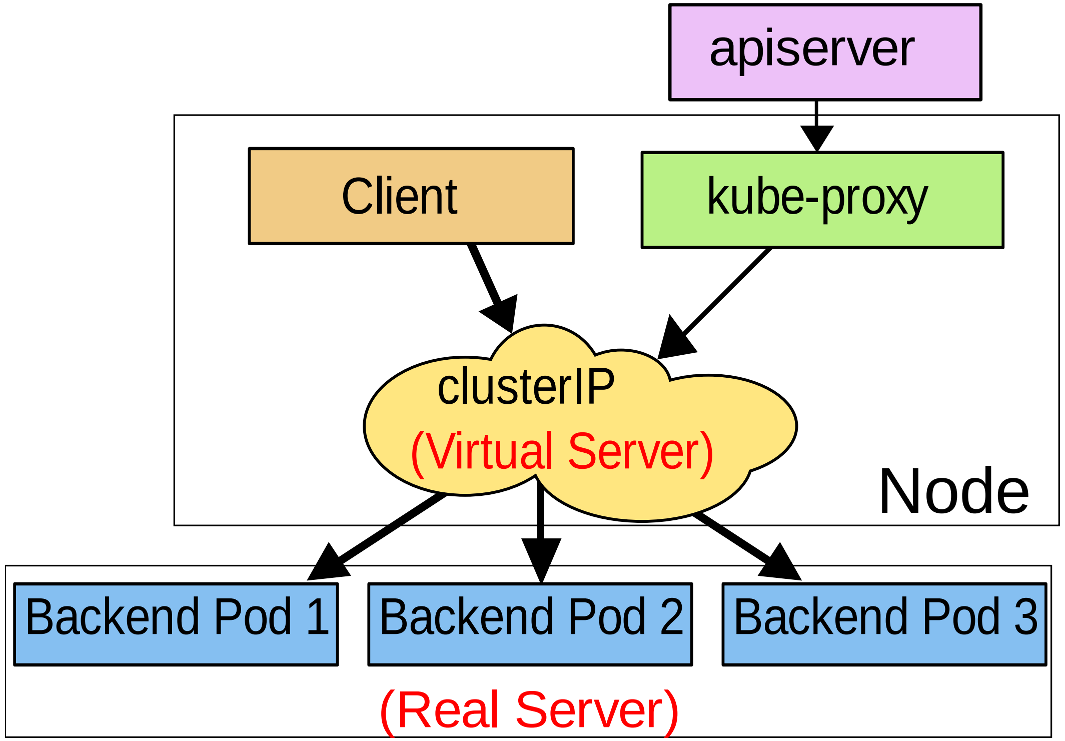
- **Lifecycle:** A Pod starts running on its assigned cluster-node until the container(s) exit or it is removed for some other reason (e.g. user deletes it).



References:
https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/
https://kubernetes.io/docs/concepts/workloads/pods/pod/
https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/
https://kubernetes.io/docs/concepts/storage/volumes/

Kubernetes: Services

- A **Service** provides an abstract way to expose an application running on a set of Pods as network service to the rest of the world
- Since Pods are ephemeral, services allow users to access the backend applications via a common way
- Service types are:
  - ClusterIP: Exposes the service on a cluster-internal IP
  - NodePort: Exposes the service on each Node's IP at a static port
  - LoadBalancer: Exposes the service externally and loadbalances it
  - ExternalName: Maps the service to a name, returns a CNAME record



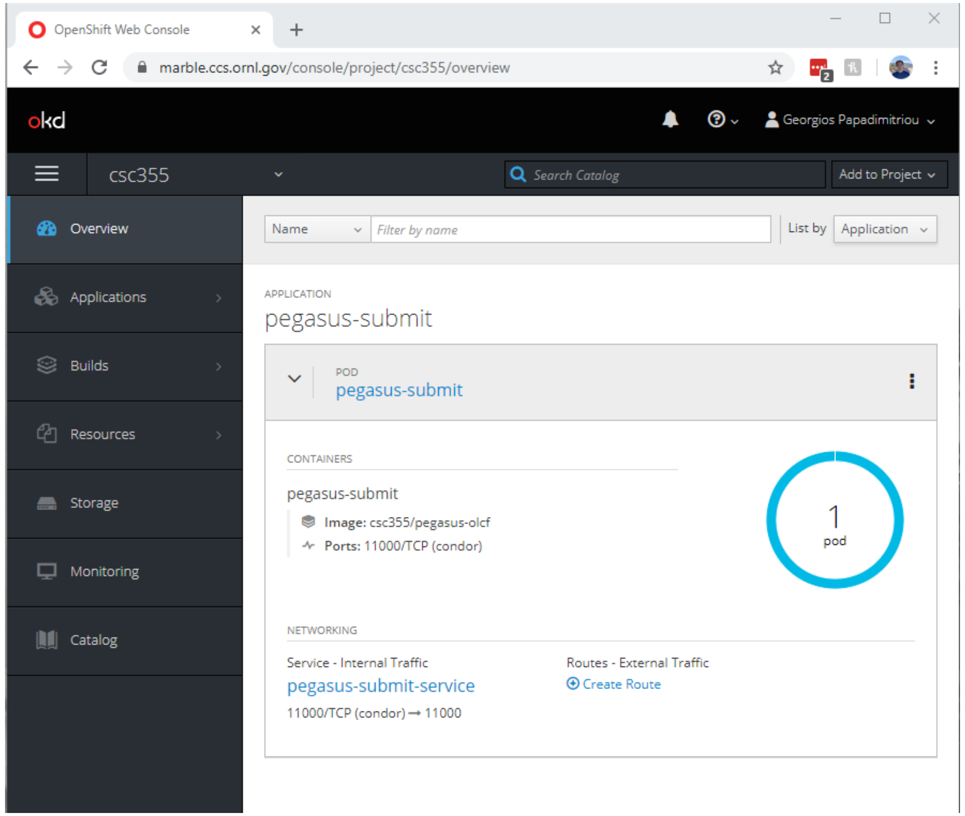
Reference:
https://kubernetes.io/docs/concepts/services-networking/service/

Kubernetes: Why it can be useful in HPC

- Running services on login nodes can be cumbersome (build from scratch, compile all dependences etc.) and sometimes prohibited by the system administrators.
- Maintaining an application/service up to day is easier
- **Assist workflow execution**
  - Create submission environments
  - Handle data movement and job submissions
  - Automation and Reproducibility
- **Create collaborative web portals**
  - Jupyter Notebooks
  - Workflow Design (e.g. Wings)
- **Streaming Data**
  - Consuming
  - Publishing

Kubernetes (OpenShift) at OLCF

- OLCF has deployed OpenShift, a distribution of Kubernetes developed by RedHat
- OpenShift provides a command line and a web interface to manage your Kubernetes objects (pods, deployments, services, storage etc.)
- OLCF's deployment has automation mechanisms that allow users to submit jobs to the batch system and access the shared file systems (NFS, GPFS)
- All containers run as an automation user that is tied to a project



OpenShift Web Console

marble.ccs.ornl.gov/console/project/csc355/overview

APPLICATION
pegasus-submit

POD
pegasus-submit

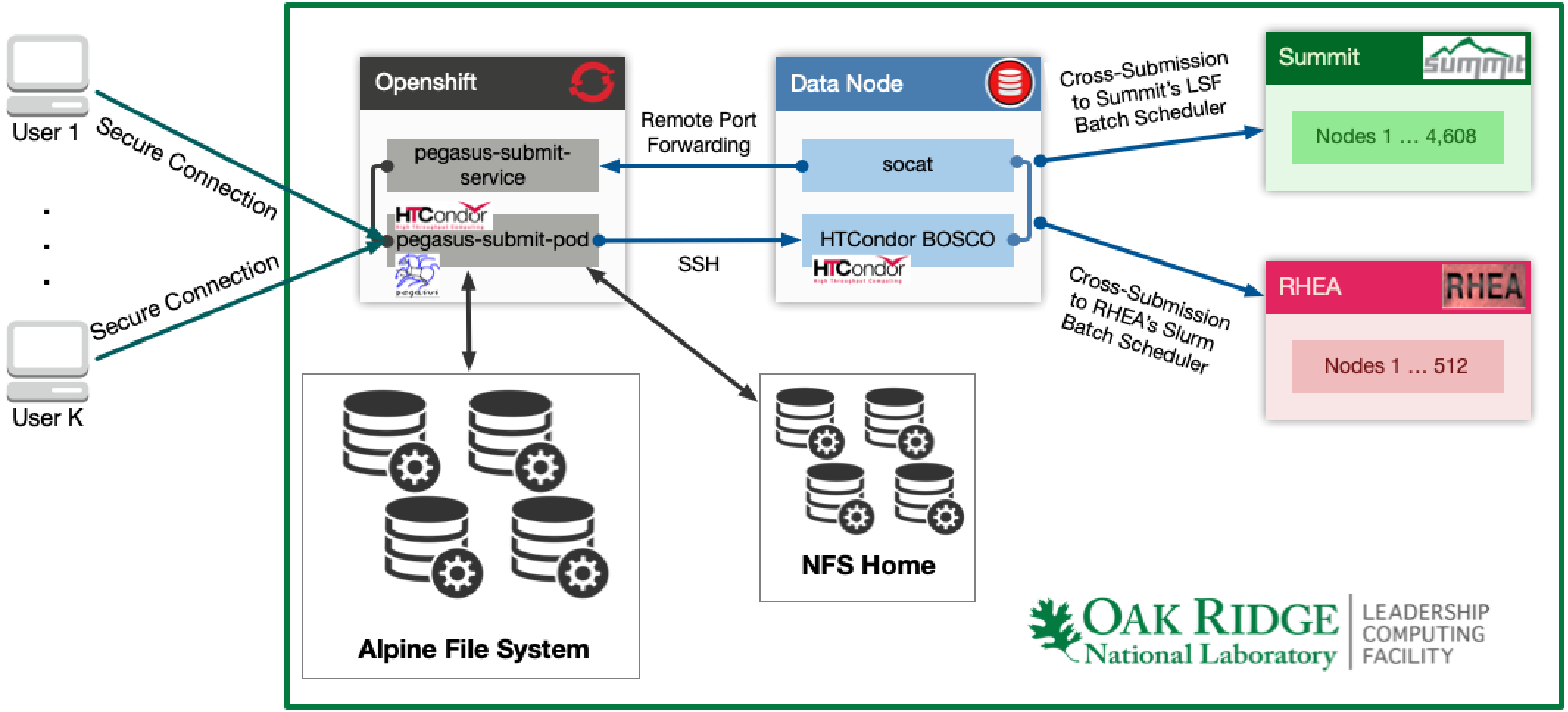
CONTAINERS
pegasus-submit
Image: csc355/pegasus-olcf
Ports: 11000/TCP (condor)

NETWORKING
Service - Internal Traffic
pegasus-submit-service
11000/TCP (condor) → 11000

Routes - External Traffic
Create Route

Reference:
https://www.olcf.ornl.gov/wp-content/uploads/2017/11/2018UM-Day3-Kincl.pdf

Kubernetes (OpenShift) at OLCF: Pegasus Deployment



Kubernetes at OLCF: Pegasus Deployment - Advantages

- Pegasus workflow **environments** at OLCF have been **simplified**.

- Using the Kubernetes cluster at OLCF, we can deploy Pegasus submit nodes as services, within a few seconds.

- The deployment uses HTCondor’s BOSCO SSH style submissions on the DTNs and achieves submissions to the SLURM and LSF batch schedulers.

- This approach allows a single workflow to be configured to use **all** of OLCF’s resources. E.g. Execute transfers on the DTNs, run simulations and heavy processing on Summit and then do lightweight post processing steps on RHEA.

How to Deploy

We will follow the tutorial: https://pegasus.isi.edu/tutorial/summit/tutorial_setup.php

How to Deploy: Prerequisites

- Pegasus Kubernetes Templates for OLCF:
  - https://github.com/pegasus-isi/pegasus-olcf-kubernetes
- Openshift’s Origin Client:
  - https://github.com/openshift/origin/releases
- A working RSA Token to access OLCF’s systems
- An automation user for OLCF’s systems
- Allocation on OLCF’s Openshift Cluster (https://marble.ccs.ornl.gov)

How to Deploy: Useful Origin Client Commands

- oc login: acquires an access token, authenticate against a cluster
- oc status: returns/prints the status of your deployments
- oc describe: shows details of a specific resource
- oc create: creates a Kubernetes resource from specification
- oc start-build: initiates the creation of a container image
- oc logs: returns/prints the Kubernetes log for a resource
- oc exec: executes a command in a container
- oc delete: deletes a resource

How to Deploy: Pegasus - Kubernetes Templates

- **bootstrap.sh** Generates customized Dockerfile and Kubernetes pod and service specifications for your deployment.

- **Specs/pegasus-submit-build.yml** Contains Kubernetes build specification for the pegasus-olcf image.

- **Specs/pegasus-submit-service.yml** Contains Kubernetes service specification that can be used to spawn a Nodeport service that exposes the HTCondor Gridmanager Service running in your submit pod, to outside world.

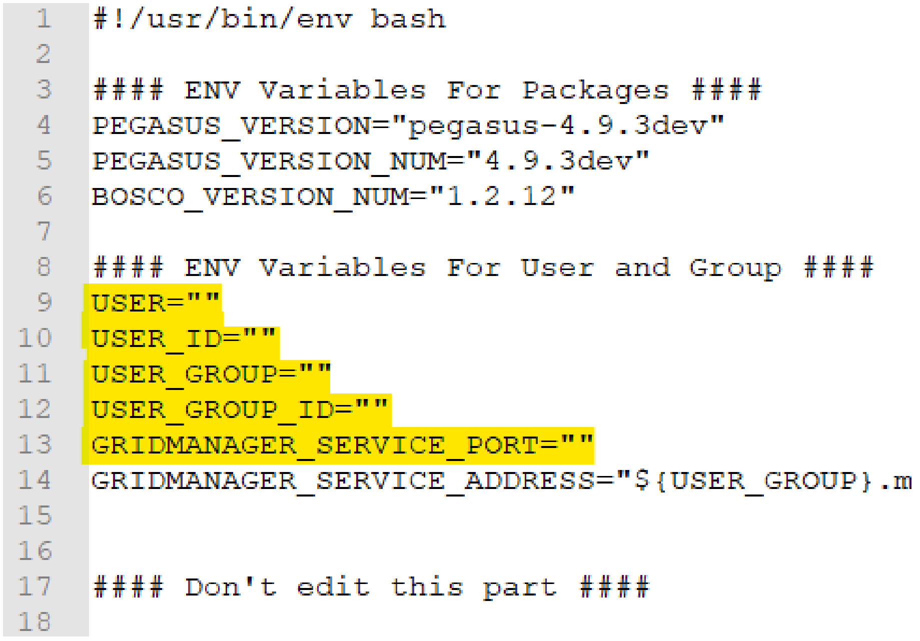
- **Specs/pegasus-submit-pod.yml** Contains Kubernetes pod specification that can be used to spawn a pegasus/condor pod that has access to Summits's GPFS filesystem and its batch scheduler.

How to Deploy: Customize Templates

In **bootstrap.sh** update the section "ENV Variables For User and Group" with your automation user's name, id, group name, group id and the Gridmanager Service Port, which must be in **the range 30000-32767**.

Replace the highlighted text:

- **USER:** with the username of your automation user (eg. csc001_auser)
- **USER_ID:** with the user id of your automation user (eg. 20001)
- **USER_GROUP:** with the project name your automation user belongs to (eg. csc001)
- **USER_GROUP_ID:** with the project group id your automation user belongs to (eg. 10001)
- **GRIDMANAGER_SERVICE_PORT:** with the Kubernetes Nodeport port number the Gridmanager Service should use (eg. 32752)

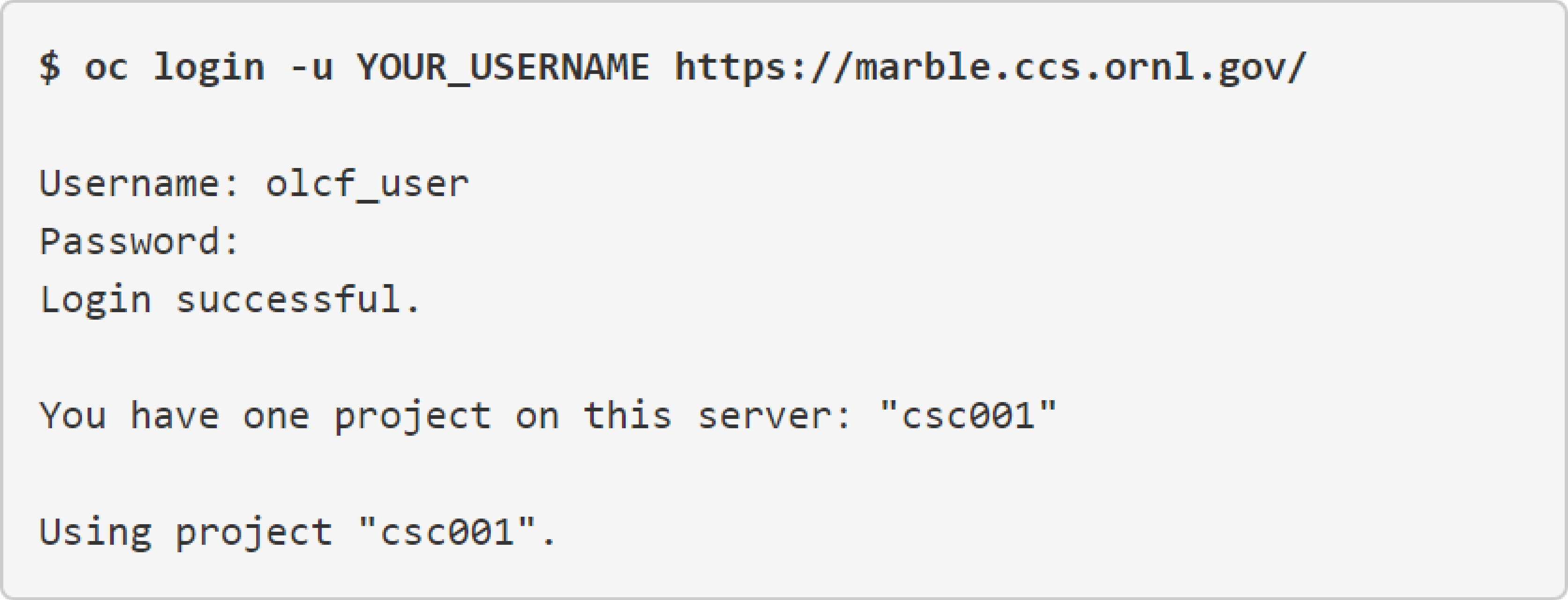


```
#!/usr/bin/env bash

#### ENV Variables For Packages ####
PEGASUS_VERSION="pegasus-4.9.3dev"
PEGASUS_VERSION_NUM="4.9.3dev"
BOSCO_VERSION_NUM="1.2.12"

#### ENV Variables For User and Group ####
USER=""
USER_ID=""
USER_GROUP=""
USER_GROUP_ID=""
GRIDMANAGER_SERVICE_PORT=""
GRIDMANAGER_SERVICE_ADDRESS="${USER_GROUP}.m


#### Don't edit this part ####

```

Execute Script:

```
$ bash bootstrap.sh
```

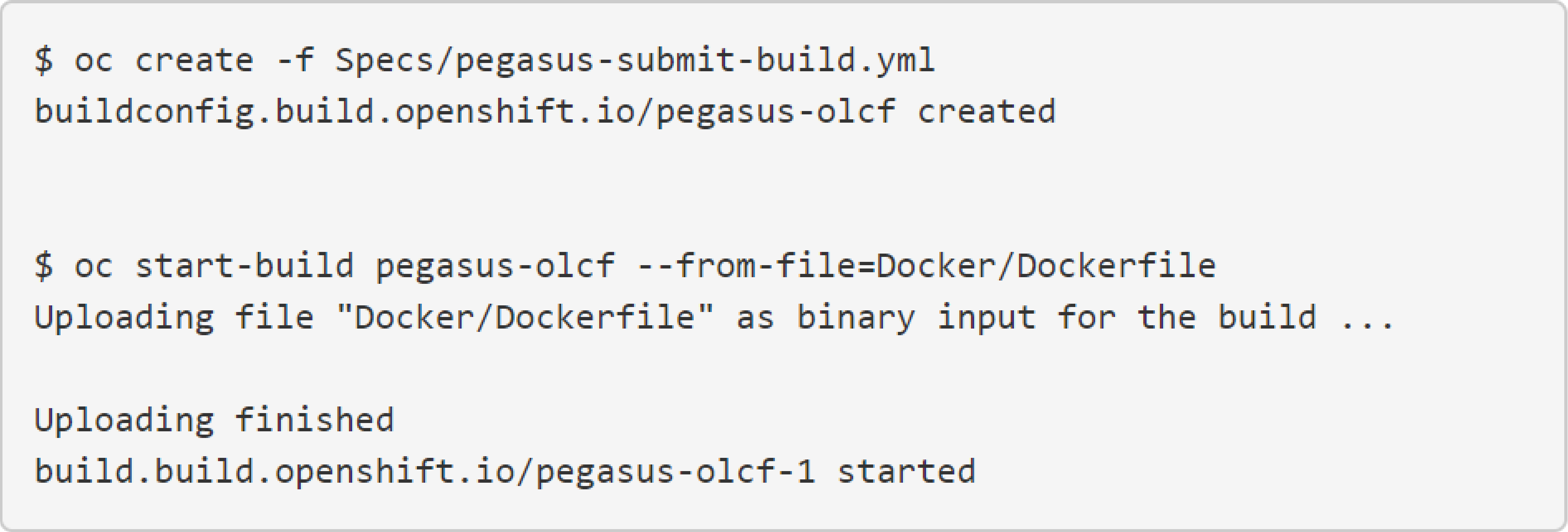
How to Deploy: Acquire an Access Token (Step 1)



```
$ oc login -u YOUR_USERNAME https://marble.ccs.ornl.gov/

Username: olcf_user
Password:
Login successful.

You have one project on this server: "csc001"

Using project "csc001".
```

How to Deploy: Build the Container Image (Step 2)

Create a new build and build the image:



1

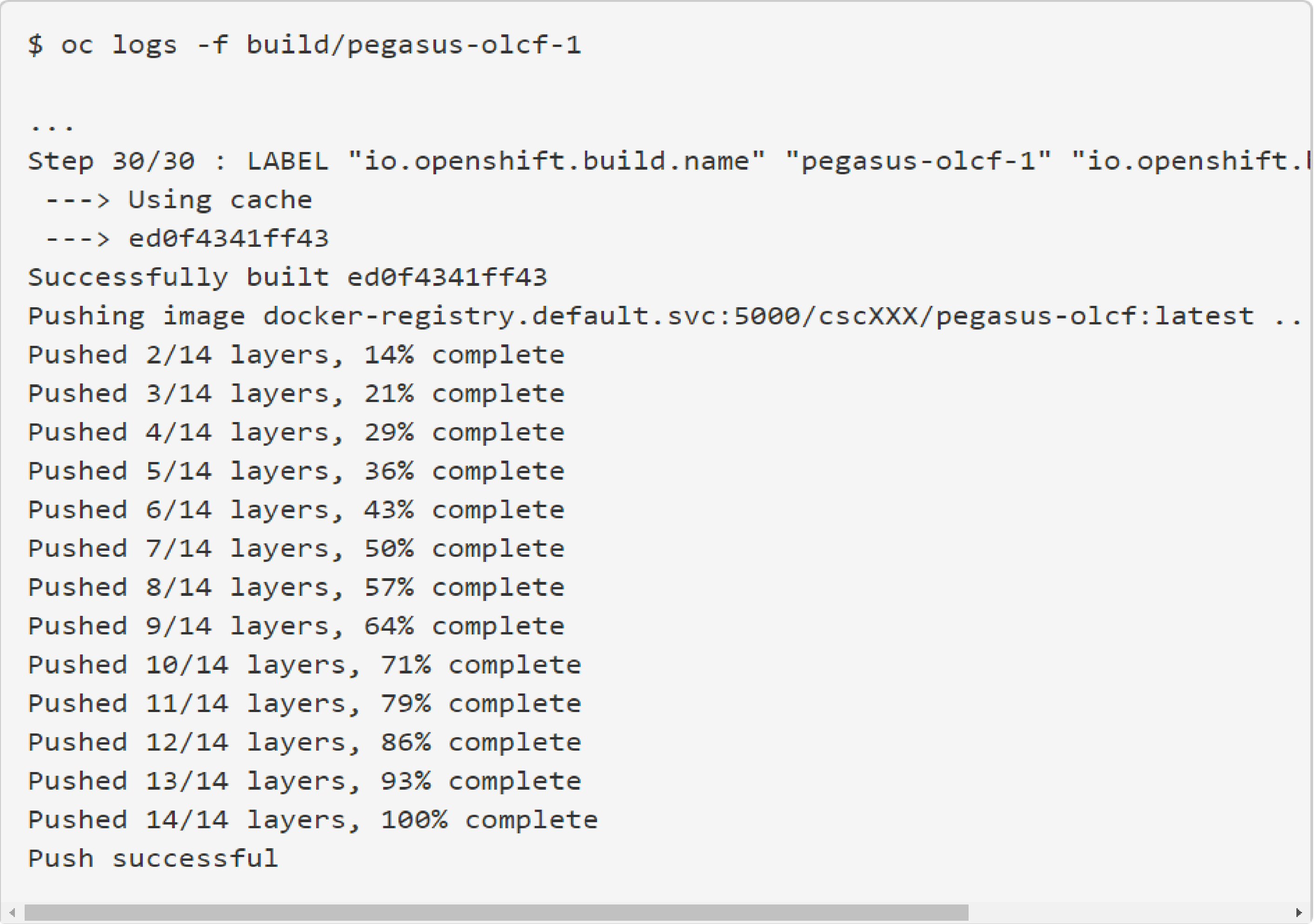
```
$ oc create -f Specs/pegasus-submit-build.yml
buildconfig.build.openshift.io/pegasus-olcf created
```

2

```
$ oc start-build pegasus-olcf --from-file=Docker/Dockerfile
Uploading file "Docker/Dockerfile" as binary input for the build ...

Uploading finished
build.build.openshift.io/pegasus-olcf-1 started
```

How to Deploy: Build the Container Image (Step 2)

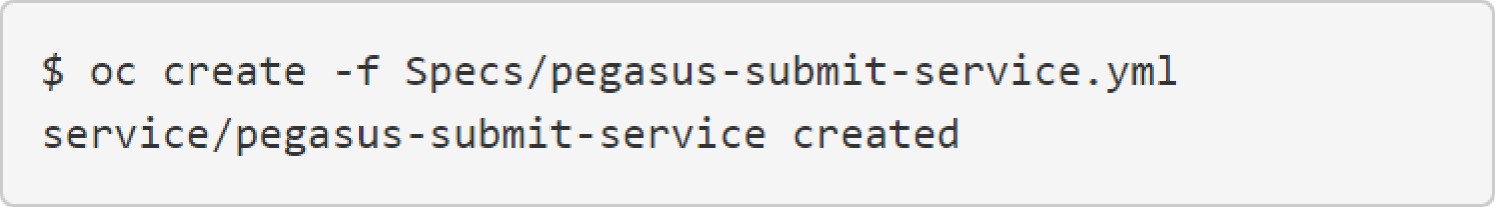
Trace the progress of the build:



```
$ oc logs -f build/pegasus-olcf-1

...
Step 30/30 : LABEL "io.openshift.build.name" "pegasus-olcf-1" "io.openshift.
 ---> Using cache
 ---> ed0f4341ff43
Successfully built ed0f4341ff43
Pushing image docker-registry.default.svc:5000/cscXXX/pegasus-olcf:latest ..
Pushed 2/14 layers, 14% complete
Pushed 3/14 layers, 21% complete
Pushed 4/14 layers, 29% complete
Pushed 5/14 layers, 36% complete
Pushed 6/14 layers, 43% complete
Pushed 7/14 layers, 50% complete
Pushed 8/14 layers, 57% complete
Pushed 9/14 layers, 64% complete
Pushed 10/14 layers, 71% complete
Pushed 11/14 layers, 79% complete
Pushed 12/14 layers, 86% complete
Pushed 13/14 layers, 93% complete
Pushed 14/14 layers, 100% complete
Push successful
```

How to Deploy: Start the Kubernetes Service (Step 3)

Start a Kubernetes Service that will expose your pod’s services:

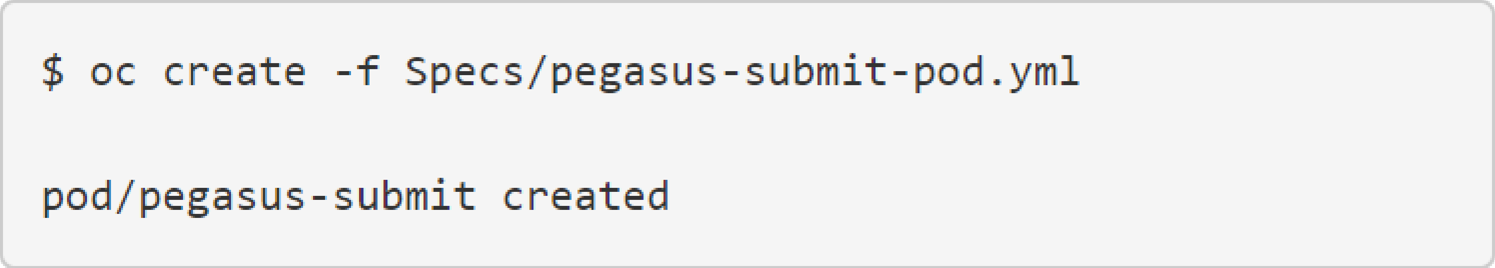


```
$ oc create -f Specs/pegasus-submit-service.yml
service/pegasus-submit-service created
```

Note: In case this step fails, go back to the bootstrap.sh change the service port number and execute it again.
Proceed from this step, there is no need to rebuild the container.

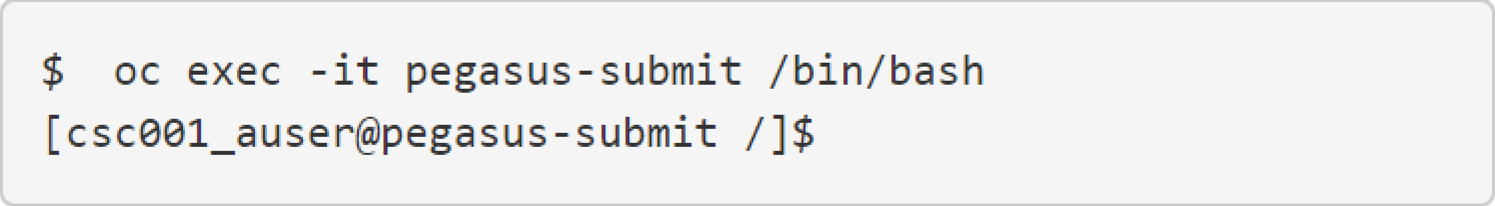
How to Deploy: Start the Pegasus Pod (Step 4)

Start a Kubernetes Pod with Pegasus and HTCondor:



```
$ oc create -f Specs/pegasus-submit-pod.yml

pod/pegasus-submit created
```

Logon to the Pod:

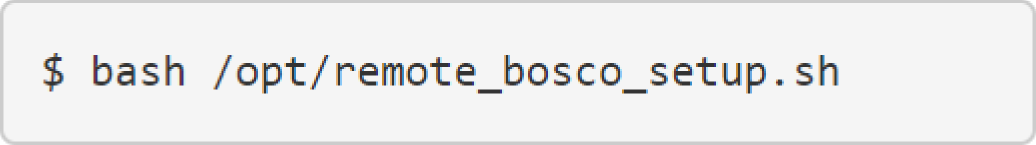


```
$  oc exec -it pegasus-submit /bin/bash
[csc001_auser@pegasus-submit /]$
```

How to Deploy: Configuring for Batch Submissions (Step 5)

If this is the first time you bringing up the Pegasus container in Kubernetes we need to configure it for batch submissions.

In the shell you got on the previous step execute:

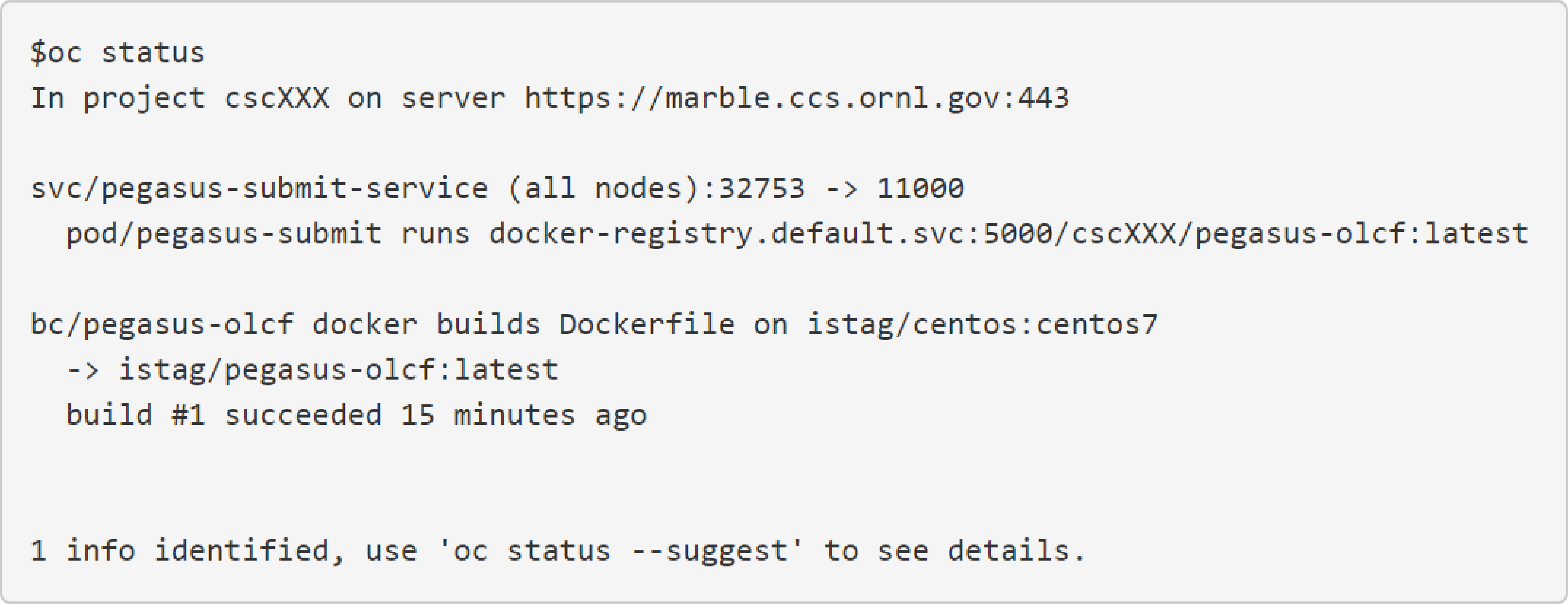


```
$ bash /opt/remote_bosco_setup.sh
```

Note: This script installs some additional files needed to operate on OLCF, and prepares the environment on the DTNs, by installing BOSCO.

How to Deploy: Check the status of the deployment

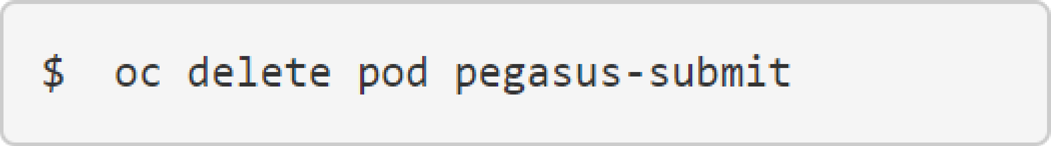
If all goes well you should see something similar to this in your terminal:



```
$oc status
In project cscXXX on server https://marble.ccs.ornl.gov:443

svc/pegasus-submit-service (all nodes):32753 -> 11000
  pod/pegasus-submit runs docker-registry.default.svc:5000/cscXXX/pegasus-olcf:latest

bc/pegasus-olcf docker builds Dockerfile on istag/centos:centos7
  -> istag/pegasus-olcf:latest
  build #1 succeeded 15 minutes ago



1 info identified, use 'oc status --suggest' to see details.
```

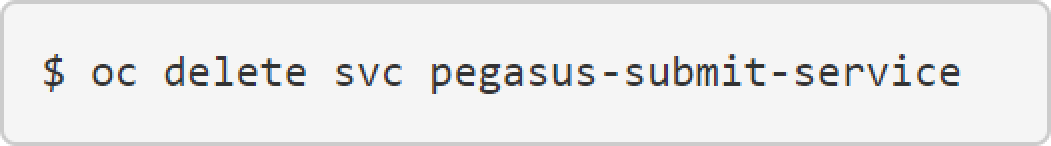
How to Deploy: Deleting the Pod and the Service

Deleting the Pod:



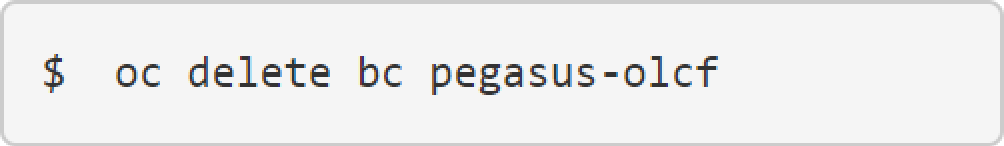
```
$  oc delete pod pegasus-submit
```

Deleting the Service:



```
$ oc delete svc pegasus-submit-service
```

Deleting the container image:



```
$  oc delete bc pegasus-olcf
```

Demo Workflow

We will follow the tutorial: https://pegasus.isi.edu/tutorial/summit/tutorial_submitting_wf.php

Acknowledgements

Special thanks to the OLCF people that helped us make this deployment happen !

Jason Kincl
kincljc@ornl.gov

Valentine Anantharaj
anantharajvg@ornl.gov

Jack Wells
wellsjc@ornl.gov

panorama360

USC Viterbi
School of Engineering
Department of Computer Science

- GitHub: https://github.com/Panorama360
- Website: https://panorama360.github.io

George Papadimitriou
Computer Science PhD Student
University of Southern California

email: georgpap@isi.edu

Pegasus est. 2001

Automate, recover, and debug scientific computations.

Get Started

Pegasus Website

http://pegasus.isi.edu

Users Mailing List

pegasus-users@isi.edu

Support

pegasus-support@isi.edu

Pegasus Online Office Hours

https://pegasus.isi.edu/blog/online-pegasus-office-hours/

Bi-monthly basis on second Friday of the month, where we address user questions and also apprise the community of new developments