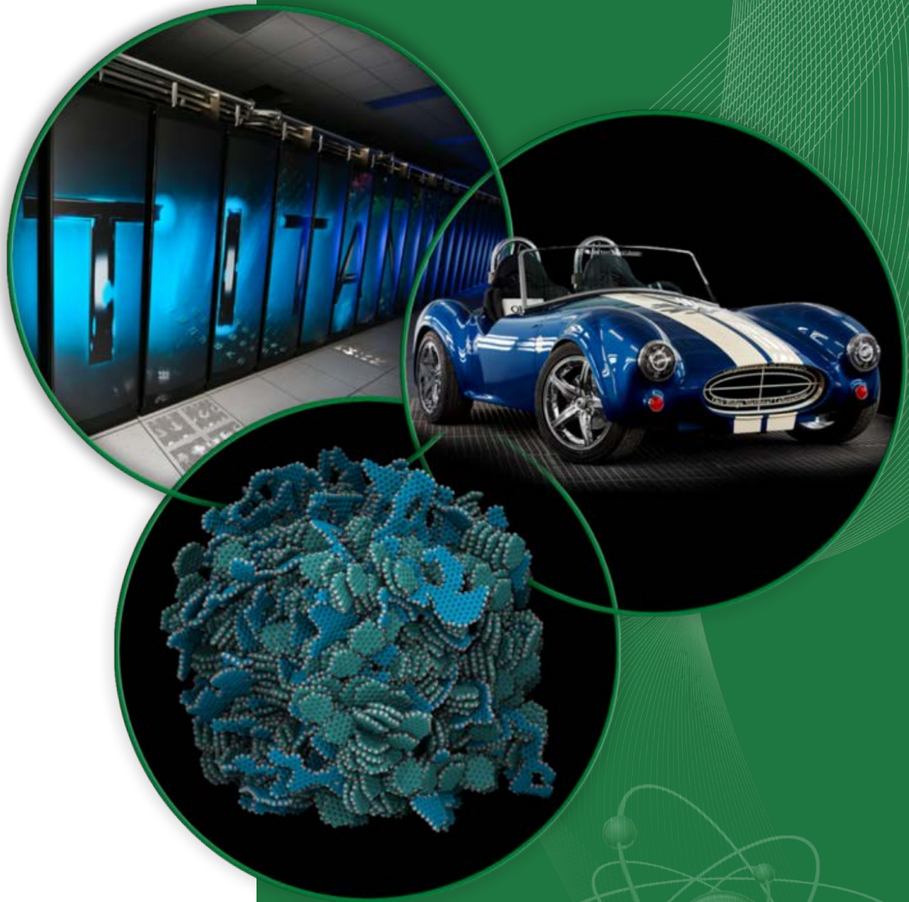
OLCF Container Orchestration for HPC Middleware

Jason Kincl, HPC Systems Administrator



Multiple Container Strategies at OLCF

- **Container orchestration**: Automate deploying and operating service containers with Kubernetes/OpenShift
  - Focused on framework for providing resources (cpu, memory, network, …) for running services
  - Uses own scheduler and is a separate resource from HPC
- **HPC container runtimes** with Singularity
  - Focused on using containers to run applications in a batch job
  - Uses scheduler from batch job submission system
  - Allows users to provide a portable environment to run jobs on HPC resources

What is HPC Middleware?

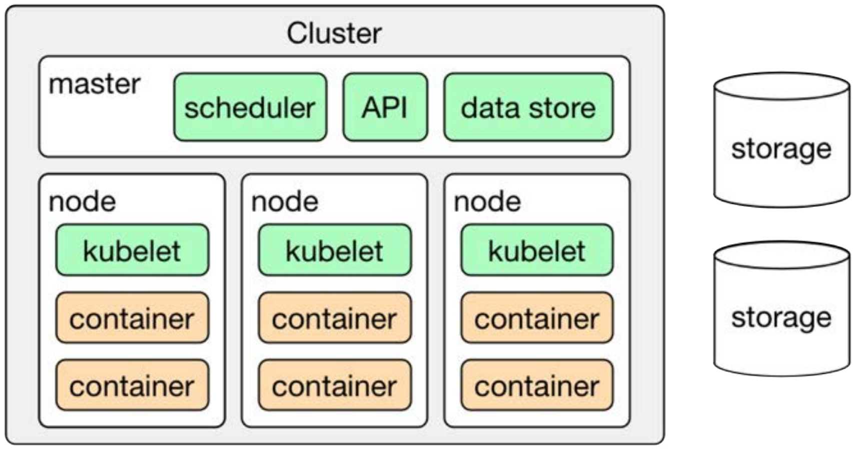
- The collection of applications and services that helps a project achieve it’s scientific goals
- Some examples:
  - Workflows
    - Data movement and job submission
    - Continuous integration and testing
    - Automation
  - Collaboration web portals for viewing data stored in OLCF
    - Jupyter notebooks
  - Streaming data

Basic HPC Workflow Requirements

- Need ways for users to manage their workflow system
  - Diverse ecosystem of workflow systems makes it difficult for NCCS Operations to support every one
- Why not just use SSH keys?
  - Our moderate security controls require remote actions to be authenticated with RSA two-factor credentials
  - Instead we are working on providing ability for running workflow services locally
- Upon surveying existing workflow systems we came up with the following requirements:
  - Run a persistent service locally as a “daemon” that stays up
  - Talk to batch submission system for current queue information and job submission
  - Interact with files on GPFS/Lustre/NFS

OpenShift

- Distribution of Kubernetes developed by Red Hat
- Kubernetes manages containerized applications across nodes and provides mechanisms for deployment, maintenance, and application-scaling.
- Analogous to but separate from a batch scheduler on a compute cluster



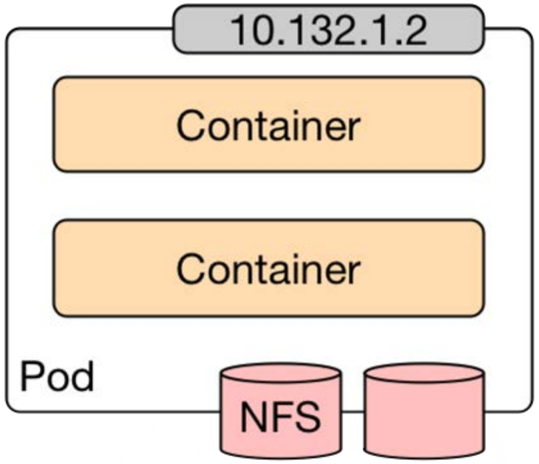
Kubernetes Architecture

- Configuration: YAML or JSON data that describes the application being deployed
- Configuration can define:
  - Containers to run
  - HTTP routes and network ports to expose outside of the cluster
  - Mounting data volumes

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.10
```

Kubernetes – Pods

- Atomic unit of Kubernetes
- Made up of one or more containers deployed together on one host
- Pod lifecycle is defined, pod is assigned to run on a node and runs until the container(s) exit or it is removed for some other reason
- Volumes can be attached that do not share pod lifecycle for persistent data
- Each pod gets its own IP address that is accessible in the cluster



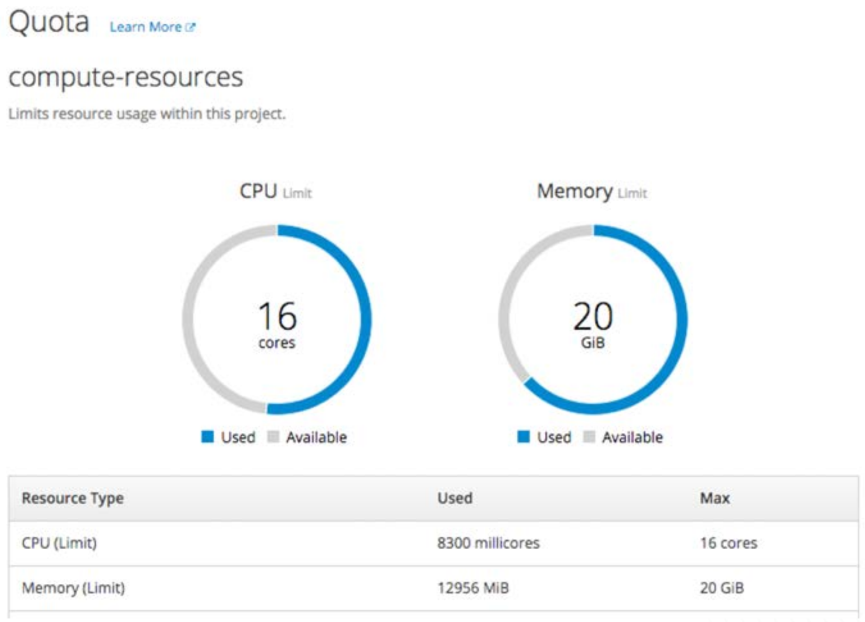
Kubernetes – Replication Controllers

- Pod will not recreate itself if killed or deleted for some reason such as cluster maintenance or quota limit exceeded
- A ReplicationController ensures desired number of pods is running in the cluster
- For example: “I want to have three pods running nginx:1.10 image”

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.10
```

Cluster Resources

- Resource allocation is different from the traditional core hours or node hours on Titan/Rhea
- Quota system based on CPU and memory limits
- User defines what CPU and memory are required for each container, if container exceeds limits it is killed



Quota Learn More

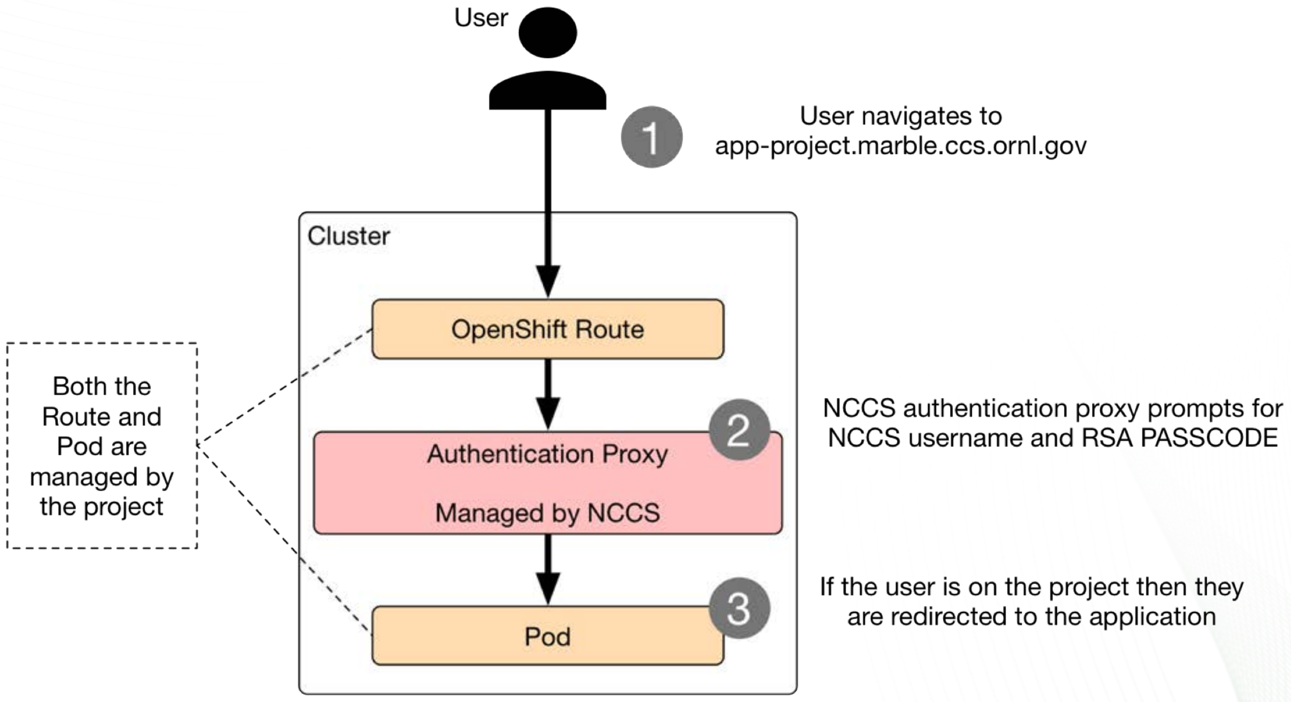
compute-resources

Limits resource usage within this project.

Resource Type	Used	Max
CPU (Limit)	8300 millicores	16 cores
Memory (Limit)	12956 MiB	20 GiB

Exposing services

- OpenShift gives users the ability to expose services outside of the cluster
  - For HTTP-based services, NCCS will handle initial authentication to ensure service is accessed only by members of that project

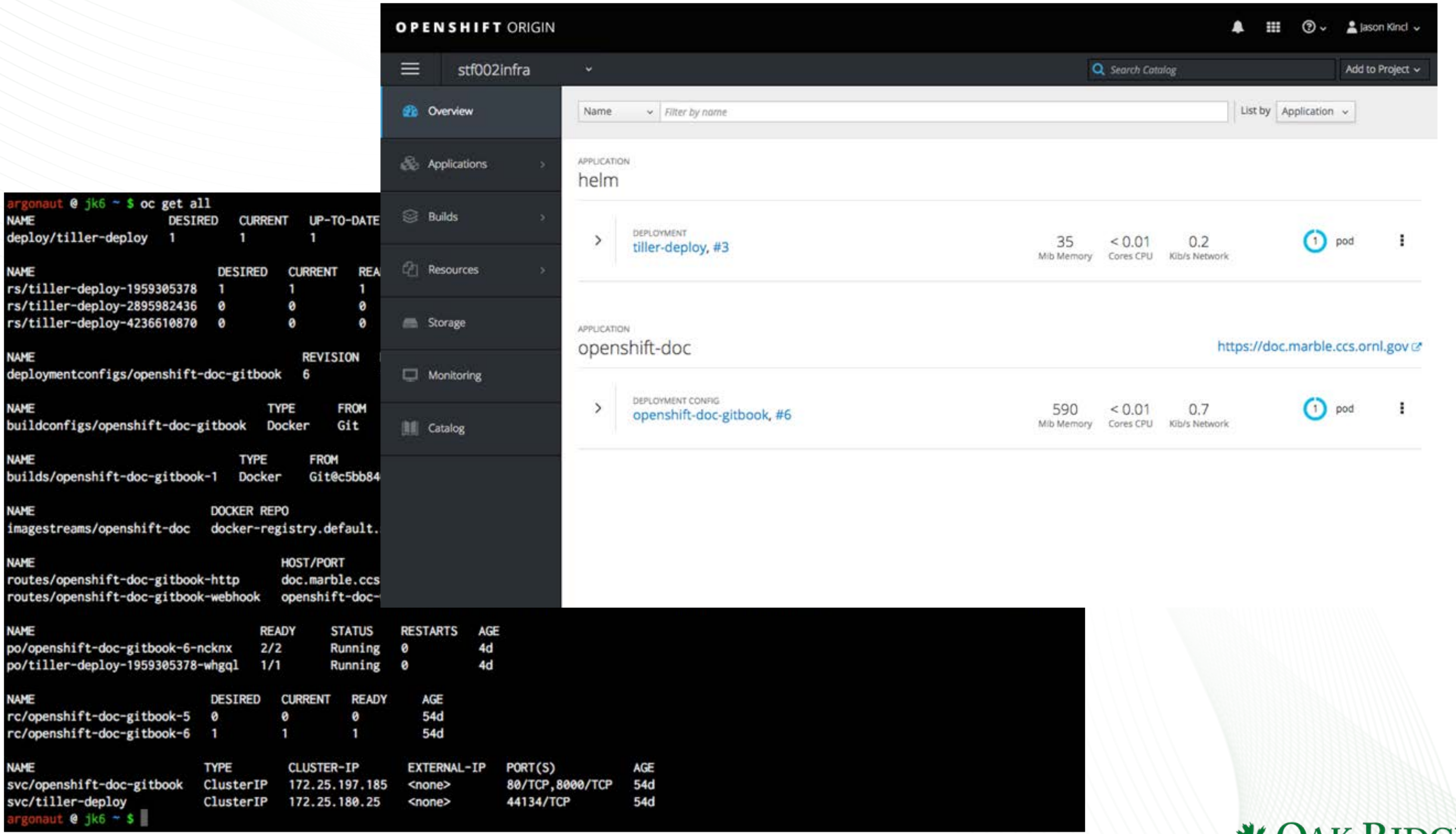

User
1
User navigates to app-project.marble.ccs.ornl.gov
Cluster
OpenShift Route
2
Authentication Proxy
Managed by NCCS
NCCS authentication proxy prompts for NCCS username and RSA PASSCODE
3
Pod
If the user is on the project then they are redirected to the application
Both the Route and Pod are managed by the project

Accessing NCCS resources

- All containers run as an automation user that is tied to a project and has access to the project's allocation and files like a regular user
- Batch job submission from container
  - Users can base their container image off our NCCS golden image which comes with the tools to schedule batch jobs or get queue status
- Accessing shared filesystems (GPFS/Lustre/NFS)
  - Shared filesystems can be mounted in the container by Kubernetes allowing access just like a login or compute node

Accessing OpenShift

- **OpenShift provides a command-line client as well as a web user interface**



```
argonaut @ jk6 ~ $ oc get all
NAME                      DESIRED   CURRENT   UP-TO-DATE
deploy/tiller-deploy      1         1         1

NAME                              DESIRED   CURRENT   REA
rs/tiller-deploy-1959305378       1         1         1
rs/tiller-deploy-2895982436       0         0         0
rs/tiller-deploy-4236610870       0         0         0

NAME                                        REVISION
deploymentconfigs/openshift-doc-gitbook     6

NAME                                  TYPE      FROM
buildconfigs/openshift-doc-gitbook    Docker    Git

NAME                              TYPE      FROM
builds/openshift-doc-gitbook-1    Docker    Git@c5bb84

NAME                          DOCKER REPO
imagestreams/openshift-doc    docker-registry.default.

NAME                                   HOST/PORT
routes/openshift-doc-gitbook-http      doc.marble.ccs
routes/openshift-doc-gitbook-webhook   openshift-doc-

NAME                               READY     STATUS    RESTARTS   AGE
po/openshift-doc-gitbook-6-ncknx   2/2       Running   0          4d
po/tiller-deploy-1959305378-whgql  1/1       Running   0          4d

NAME                          DESIRED   CURRENT   READY     AGE
rc/openshift-doc-gitbook-5    0         0         0         54d
rc/openshift-doc-gitbook-6    1         1         1         54d

NAME                        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)            AGE
svc/openshift-doc-gitbook   ClusterIP   172.25.197.185  <none>        80/TCP,8080/TCP    54d
svc/tiller-deploy           ClusterIP   172.25.180.25   <none>        44134/TCP          54d
argonaut @ jk6 ~ $
```

APPLICATION
helm

DEPLOYMENT
tiller-deploy, #3 — 35 MiB Memory, < 0.01 Cores CPU, 0.2 KiB/s Network, 1 pod

APPLICATION
openshift-doc — https://doc.marble.ccs.ornl.gov

DEPLOYMENT CONFIG
openshift-doc-gitbook, #6 — 590 MiB Memory, < 0.01 Cores CPU, 0.7 KiB/s Network, 1 pod

OAK RIDGE National Laboratory

Current Clusters

- NCCS Moderate cluster – Marble
  - Available now
  - Same zone as Summit/Titan/Rhea/Atlas
- NCCS Open cluster – Onyx
  - Coming soon
  - Supports ORNL XCAMS user/password authentication instead of two-factor tokens (but no access to moderate resources like Atlas filesystem)
  - Access to Wolf filesystem

Some Pilot Projects

- Continuous Integration with Jenkins and testing on Titan
- Project dashboard for viewing job output
- Running Fireworks workflow DB backend
- BigPanDA workflow services
- Database-based data portal for accessing project data

Interested? Get in touch with us!

- Currently in a pilot phase, if you have a use case that may benefit please reach out to us

Email: help@olcf.ornl.gov

Web: https://www.olcf.ornl.gov/for-users/user-assistance/

OAK RIDGE National Laboratory