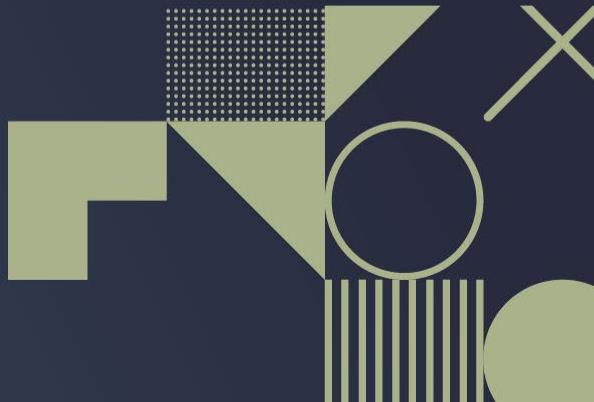
Utilizing Kubernetes and Open OnDemand to Support Virtual Classroom Labs

Alan Chalker, Ph.D.
Trey Dockendorf
Ohio Supercomputer Center



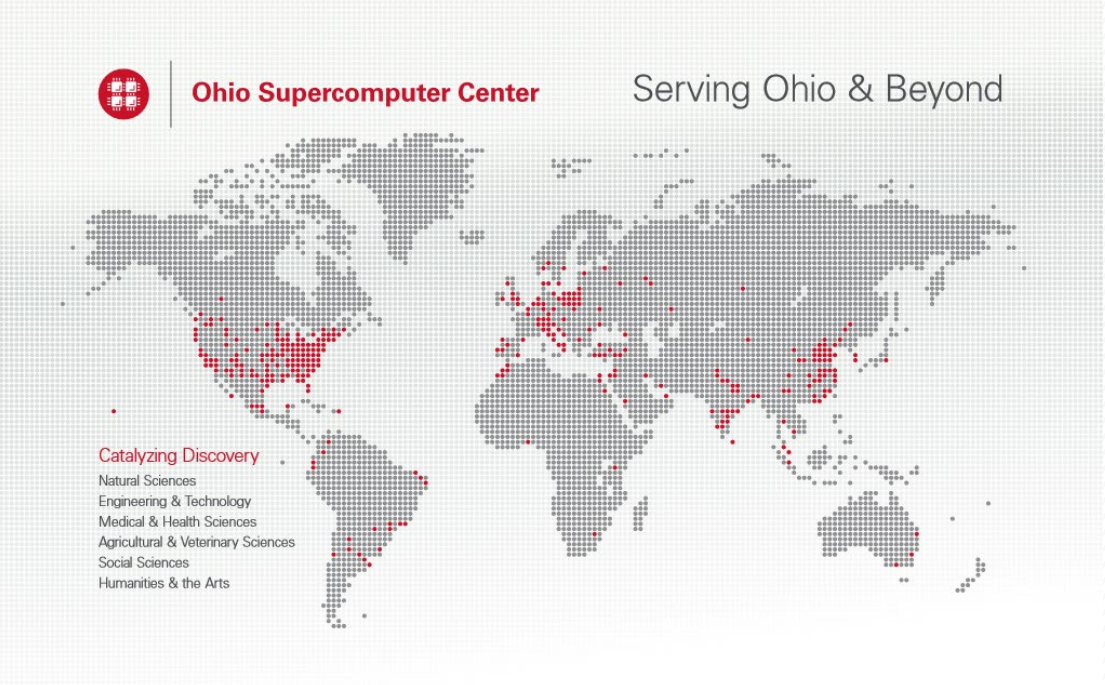
Supercomputing for anyone, anywhere!

Ohio academic institutions

- Faculty
- Postdocs
- Research scientists
- Grad students
- Instructors

Commercial, nonprofit & more

- Business and industry
- Nonprofit organizations
- Government agencies
- Hospitals and health care
- Organizations outside of Ohio



Virtual Computer Lab Support

In FY21, OSC supported 5,565 enrollees, 249 courses, 77 departments, 20 universities with:

- Custom OnDemand dashboard at class.osc.edu
- Installing specific software packages as requested
- Streamlining student account creation
- Scheduler reservations as needed
- Introduction to supercomputing for classroom

75/249 Ohio Higher-Ed Courses
Up 232%

Kubernetes Workload

- Lightweight interactive workloads
- Minimal CPU requirements compared to traditional HPC workloads
- Exclusively Jupyter and Rstudio (so far)
- 16 different classes this autumn semester are using Kubernetes under the hood.
- OSC staff make the decision to leverage Kubernetes based on the expected usage pattern of the class – all hidden from students.

Overview of Technologies

Kubernetes

- Open-source container orchestration

Open OnDemand

- Web interface to make HPC access easier
- Provides a way for sites to make things like interactive jobs easy to deploy and use
- Web processes run as logged in HPC user
- Supports multiple resource manages: SLURM, Torque, Kubernetes

Kyverno

- Kubernetes policy engine
- Deploy policies using Kubernetes resources, ie standard Kubernetes YAML resources

Challenges

- Treat Kubernetes "jobs" similar to traditional HPC jobs and allow usage of SLURM or Kubernetes from same app code
- Kubernetes pods can run as root and on systems with shared filesystems like GPFS, this can be very dangerous.
- How to ensure a user running a pod is doing so using their UID/GIDs that ensures operations like filesystem access are taking place as that user
- How to charge users for their usage of Kubernetes similar to job charging in traditional HPC batch environment

Kubernetes Design Patterns

- All user pods run in user specific namespace of pattern user-$USER which is bootstrapped by OnDemand at login
- RBAC for user-$USER namespaces limits user operations to just the things needed to run OnDemand jobs
- Kubernetes authenticates with Keycloak OIDC IDP and the OIDC tokens for OnDemand are allowed to be used for Kubernetes via OAuth2 audience
- Deploy job-pod-reaper tool to cleanup pods after “walltime” is exceeded
  - Uses annotation to set what walltime should be
- Deploy k8-namespace-reaper to cleanup unused namespaces

OnDemand Infrastructure

- When OnDemand PUN starts, a pre-PUN hook is executed as root to bootstrap Kubernetes resources for each user
  - Namespace, network policy, RBAC
- Hook is passed username and OIDC token, and this token is used to set kube config credentials using OnDemand token
  - Possible thanks to OAuth2 audience configured in Keycloak
- The kubeconfig is done per-environment so unique context for production vs test vs dev. Main OnDemand instance can use same context as classroom OnDemand instance
- OSC uses one OnDemand instance for general usage and separate instance for classrooms

OnDemand Kubernetes Support

- Kube config is bootstrapped by OnDemand code, credentials come from pre-PUN hook
- All kubectl commands run as user logged into OnDemand using bootstrapped kube config
- OnDemand forces UID, GID and supplemental groups of user logged into OnDemand
- Labels assigned to pod such as "account" label, other labels allowed
- Annotation added to set walltime so pods can be seen as having finite runtime.
- Pods always have a CPU/memory request and limit set, also supports requesting GPU resources from Kubernetes

OnDemand Interactive Apps

- OSC currently supports Jupyter and RStudio Server, both used for classrooms
- Single Jupyter or RStudio Server OnDemand app that supports both SLURM and Kubernetes. Kubernetes is selected as "Kubernetes" cluster vs selecting "Owens" or "Pitzer" HPC clusters
- Classrooms have form options removed for most classrooms and instead forced to use specific number of cores and forced onto Kubernetes
  - Matched based on group membership and map stored in OnDemand

OnDemand Kubernetes Pods

- Kubernetes pod mounts numourous locations, $HOME of user, GPFS, Lmod apps and init, SLURM munge socket, sockets for SSSD
- Pods use a "cluster" container image that essentially looks like HPC compute node in terms of OS packages
- Pods treated like HPC compute nodes so Lmod apps that can run on HPC batch jobs can run in Kubernetes
- One limitation is other user $HOME locations not accessible in pod given how OSC mounts $HOME locations

Solutions to security using Kyverno

- Deploy policies that enforce user's pods run as that user
  - Ensure pod user UID and GID match the requesting user based on LDAP
  - Ensure pod supplemental groups match those of user based on LDAP
  - Ensure user pods cannot escalate privileges or access host filesystems outside of filesystems needed to run OnDemand jobs

- LDAP user mapping is performed by k8-ldap-configmap tool that generates ConfigMap resources from LDAP data that Kyverno can use in policies

Solutions for accounting using Kyverno

- Deploy policies that ensure accounting is possible
  - Require pods to have account label
  - Ensure the account label is valid when compared to LDAP data

- Deploy policies that ensure controlled usage of Kubernetes
  - Ensure CPU and Memory requests and limits exist
  - Ensure pod lifetime annotation is present and set max lifetime
  - Ensure pods are pulling images from trusted image registries

Example policies - runAsUser

```
validate:
  message: >-
    Invalid user UID specified in fields
    spec.securityContext.runAsUser or spec.containers[*].securityContext.runAsUser or
    spec.initContainers[*].securityContext.runAsUser
  anyPattern:
  - spec:
      securityContext:
        runAsUser: "{{ uidMap.data.\"{{ request.object.metadata.namespace }}\" }}"
      =(initContainers):
        - =(securityContext):
            =(runAsUser): "{{ uidMap.data.\"{{ request.object.metadata.namespace }}\" }}"
      containers:
        - =(securityContext):
            =(runAsUser): "{{ uidMap.data.\"{{ request.object.metadata.namespace }}\" }}"
  - spec:
      =(initContainers):
        - securityContext:
            runAsUser: "{{ uidMap.data.\"{{ request.object.metadata.namespace }}\" }}"
      containers:
        - securityContext:
            runAsUser: "{{ uidMap.data.\"{{ request.object.metadata.namespace }}\" }}"
```

Example policies – account and supplement groups

```
  validate:
    message: "{{ request.object.metadata.namespace }} not authorized to charge against account {{
request.object.metadata.labels.account }}"
    deny:
      conditions:
      - key: "{{ request.object.metadata.labels.account }}"
        operator: NotIn
        value: "{{ userGroupMap.data.\"{{ request.object.metadata.namespace }}\" }}"


  validate:
    message: "{{ request.object.metadata.namespace }} not authorized to use those supplemental groups"
    deny:
      conditions:
      - key: "{{ request.object.spec.securityContext.supplementalGroups[*].to_string(@) }}"
        operator: NotIn
        value: "{{ userGIDMap.data.\"{{ request.object.metadata.namespace }}\" }}"
```

Example LDAP config maps

LDAP user UID map:
user-tdockendorf: "20821"

LDAP user GID map:
user-tdockendorf: "5509"

LDAP user GIDs map:
user-tdockendorf:
'["1021","2399","3241","3285","3309","4391","4496","4547","4548","5087","5301","5325","5353","5356","5358","5509","5527","5607","6393","6557","6558","6951","6952","6957","7175"]'

LDAP user groups map:
user-tdockendorf: '["PZS0708","PZS0703","PAS1936","PDE0001"]'

Supercomputing. Seamlessly.

An intuitive, innovative, and interactive interface to remote computing resources

Open OnDemand helps computational researchers and students efficiently utilize remote computing resources by making them easy to access from any device. It helps computer center staff support a wide range of clients by simplifying the user interface and experience.

<u>Key Benefits & Impact</u>

- Key benefit to you, the end user:
  You can use any web browser to access resources at a computing service provider.
- Key benefit to you, the computer center staff:
  A wide range of clients/needs can utilize your computing resources.
- Overall impact:
  Users are able to use remote computing resources faster and more efficiently.

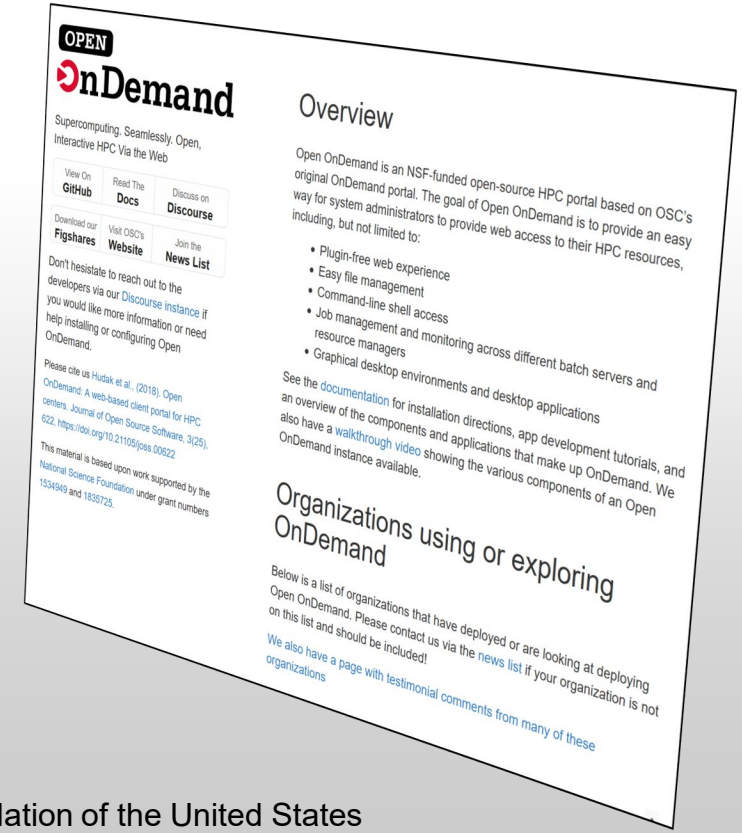
Production Deployments

OPEN OnDemand

Find Out More!

openondemand.org

- Use our Discourse instance for help
- Join our mailing list for updates
- Our webinars are roughly quarterly



This work is supported by the National Science Foundation of the United States under the awards NSF SI2-SSE-1534949 and CSSI-Software-Frameworks-1835725.

Sponsored by