Interactive Scientific Computing on the Anvil Composable Platform

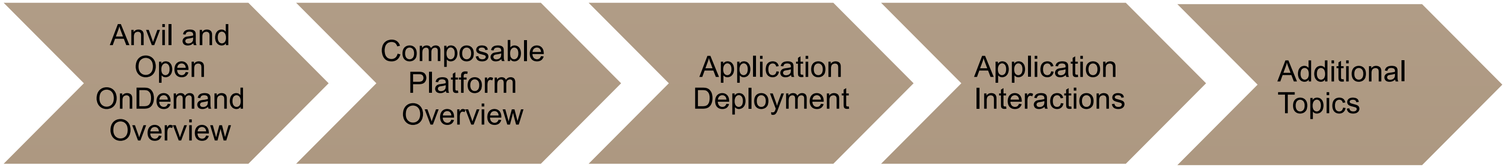
PEARC ‘21 Tutorial

Tutorial Overview

Goals

- Introduce participants to the Anvil cluster and its components
- Provide an overview of Anvil’s interactive computing capabilities
  - Open OnDemand
  - Composable Subsystem
- Highlight flexibility and ease of use through tutorial exercises

Content



Anvil and Open OnDemand Overview

Composable Platform Overview

Application Deployment

Application Interactions

Additional Topics

Presenters

Erik Gough, Eric Adams, Brian Werts, Sam Weekly, Steve Kelley
(Research Computing, Purdue University)

Robert Settlage (Open OnDemand, Virginia Tech)

Logistics

- Audience
  - Introductory, no previous experience required
- Requirements
  - A modern web browser
  - Firefox or Chrome is recommended
- Materials
  - Code of Conduct is in Pathable under Files
  - Tutorial Exercises: https://tinyurl.com/pearc21-anvil-tutorial
  - Each participant will receive a training account username and password via Zoom message during the Anvil overview
  - Accounts are removed after the tutorial

Anvil

Category I: A National Composable Advanced Computational Resource for the Future of Science and Engineering

NSF Innovative HPC Solicitation

NSF Solicitation 19-587

Advanced Computing Systems & Services: Adapting to the Rapid Evolution of Science and Engineering Research

- The intent of this solicitation is to request proposals from organizations willing to serve as service providers (SPs) within the NSF Innovative High-Performance Computing (HPC) program to provide advanced cyberinfrastructure (CI) capabilities and/or services in production operations to support the full range of computational- and data-intensive research across all of science and engineering (S&E)

Anvil – NSF Award Information

- NSF award #2005632
- Total budget is ~$10M for system acquisition
- $2.5M for operation & support each year
- 5 years of operations
- Currently scheduled to be in production on

October 1, 2021

- Will be allocated via NSF XSEDE (and its follow-on program)

Challenges for the CI Ecosystem

Meet ever-increasing demands for computational resources

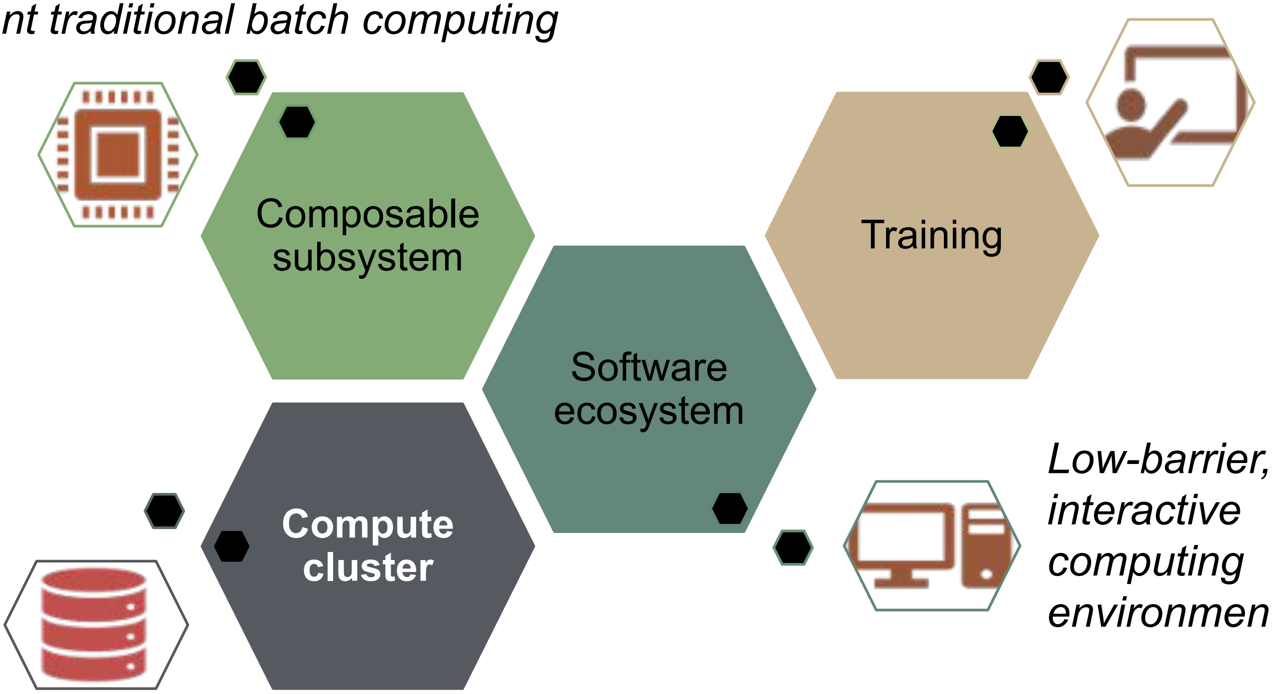
Support evolving and expanding applications

Develop the next-generation workforce

Anvil Deliverables

Training the next-generation workforce

Complement traditional batch computing



Composable subsystem

Training

Software ecosystem

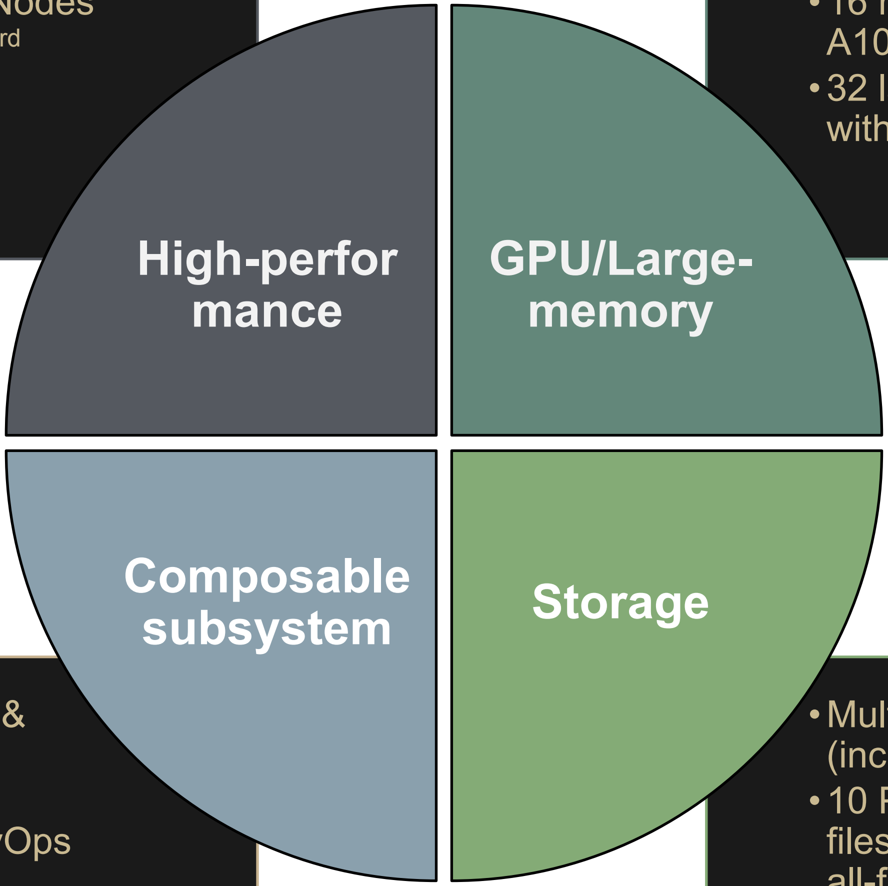
Compute cluster

Low-barrier, interactive computing environments

Deliver 1.1 billion CPU-core hours per year, dedicated GPU and large memory nodes for data-intensive computations

ANVIL COMPUTE CLUSTER

System Resources



High-performance

- 1000 Compute Nodes
- 128 core AMD 3rd Gen EPYC processors
- 5.3 PF peak performance

GPU/Large-memory

- 16 nodes with 4 NVIDIA A100 GPUs each
- 32 large memory nodes with 1 TB of RAM

Composable subsystem

- 8 large memory & storage nodes
- Kubernetes – Rancher for DevOps

Storage

- Multi-tier storage (including object storage)
- 10 PB of parallel filesystem, and 3 PB of all-flash storage
- Globus data transfer

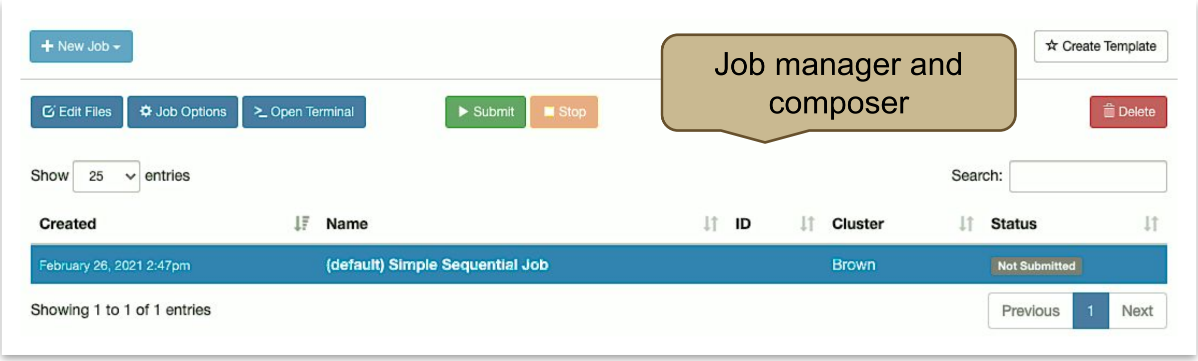
Part 1: Interactive Computing with Open OnDemand

Interactive Computing

Low Barrier & Familiar Access to Compute

OPEN OnDemand

Cendio ThinLinc



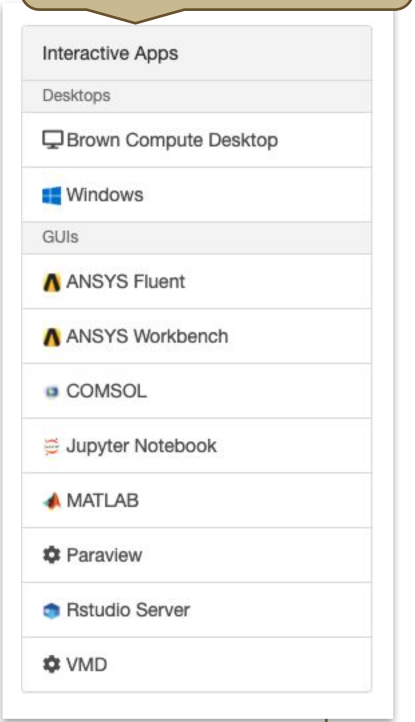
Show 25 entries Search:

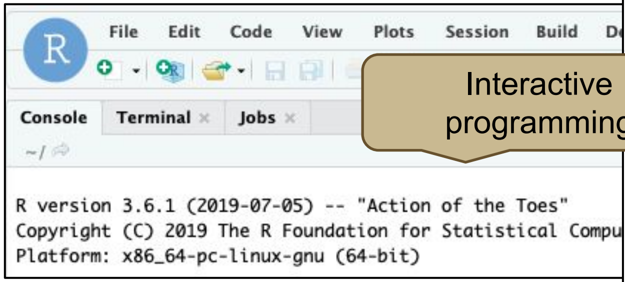
Created	Name	ID	Cluster	Status
February 26, 2021 2:47pm	(default) Simple Sequential Job		Brown	Not Submitted

Showing 1 to 1 of 1 entries

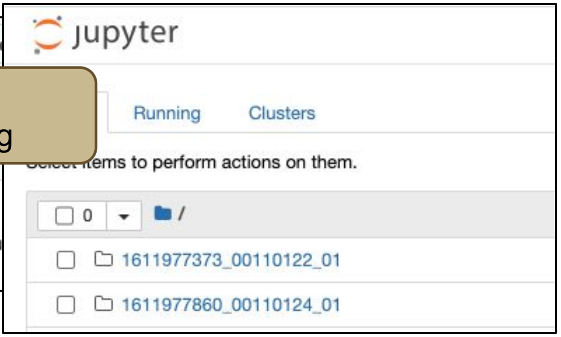
Job manager and composer

Interactive scientific applications





```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Compu
Platform: x86_64-pc-linux-gnu (64-bit)
```



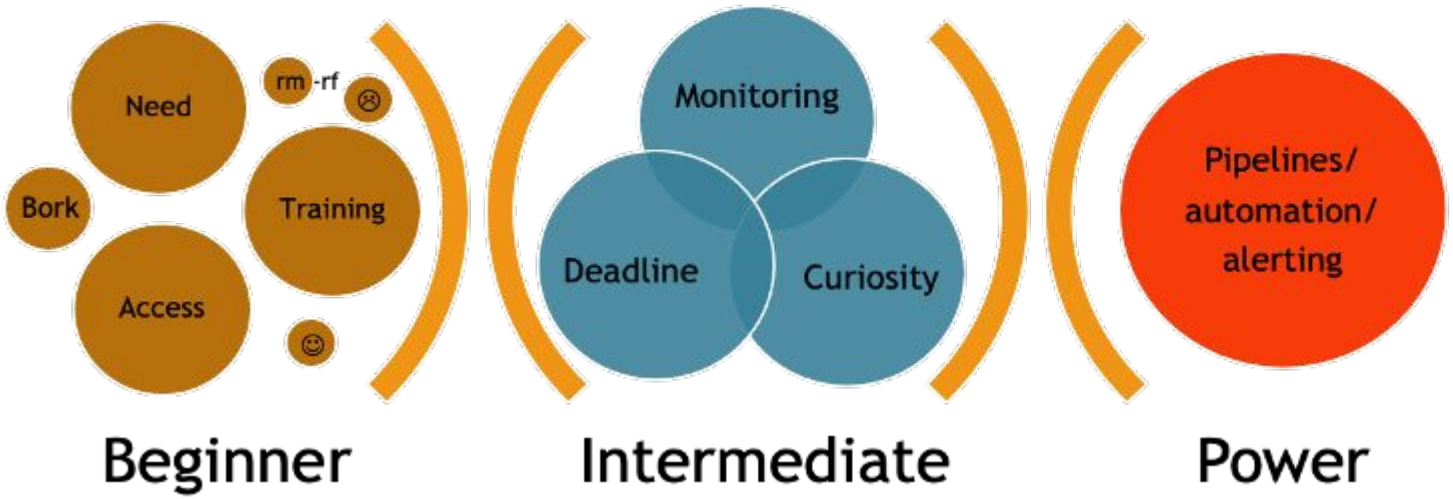
Select items to perform actions on them.

□ 1611977373_00110122_01

□ 1611977860_00110124_01

Interactive programming

Facilitate progression of users



University at Buffalo
Center for Computational Research

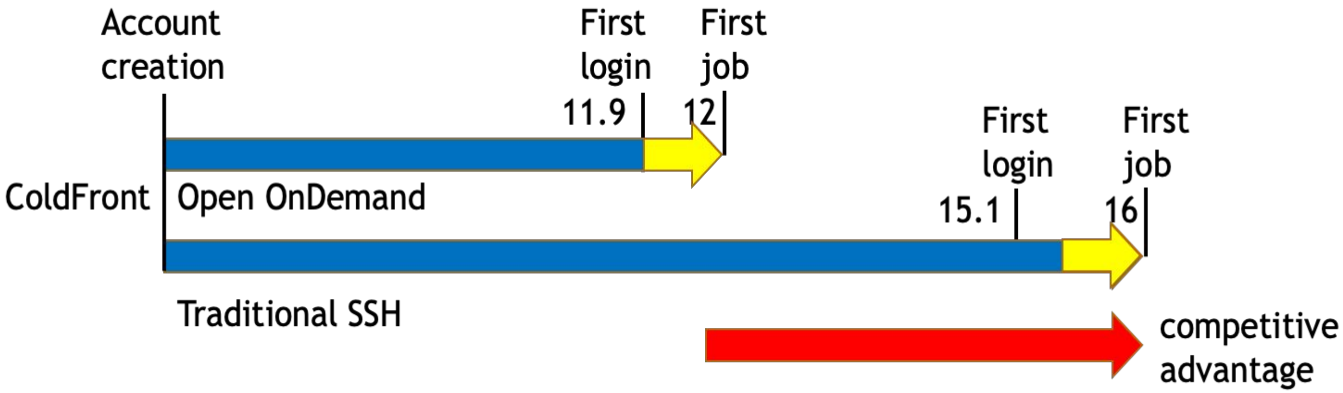
Ohio Supercomputer Center

VIRGINIA TECH

Facilitate progression of users

76

Less hours
To first job



University at Buffalo
Center for Computational Research

Ohio Supercomputer Center

VIRGINIA TECH

New features coming soon ...

- interface redesign
- more XDMoD integrations
- Kubernetes connector
- OpenStack (or other cloud) connector
- pinned favorite apps for one-click access
- pipeline support

University at Buffalo
Center for Computational Research

Ohio Supercomputer Center

VIRGINIA TECH

Open OnDemand Demo

Quick OOD Walkthrough

In some random order …

- files app
- interactive apps
- active jobs
- job composer

Start demo ...

Exercise 1: OOD Login and Desktop Session

1.1 Logging into Open OnDemand

1. Open https://gateway.pearc.rcac.purdue.edu in a web browser
2. Log in with your provided credentials

1.2 Start an Open OnDemand Desktop

1. Click Interactive Apps -> Bell Compute Desktop
2. Enter “pearc21” for the **Queue**
3. Enter 4 for **Number of hours**
4. Enter 4 for **Processor Cores requested**
5. Click **Launch**
6. Wait a bit while your job is scheduled
7. Once the status changes to Running, click the **Launch Bell Compute Desktop** button
   a. Increase the Image Quality and Compression if desired
8. Your desktop session will open in a new tab

You now have a Linux desktop running on a Bell cluster compute node in your browser, pretty cool!

1.3 Open the Tutorial exercises in your OOD Desktop Session

Having the tutorial document open in a browser your OOD Desktop session will make it easier to copy and paste tutorial materials.

1. Open Firefox in your Open OnDemand Desktop session (If you click the Browser icon in the tray, you may have to select Mozilla FIrefox as your preferred Browser)
2. Open the URL https://tinyurl.com/pearc21-anvil-tutorial

Part 2: Composable Platform Overview

Need for Composable Platforms

Researchers are increasingly looking for more diverse computing resources

- Cloud style flexibility
- Containers
- Persistent services
- Data analysis tools
- Web based science gateways and applications

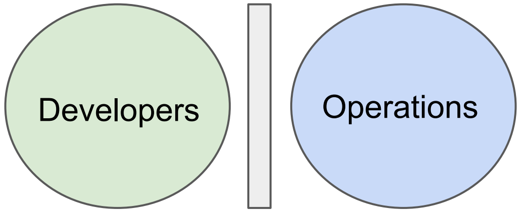
Researchers can use DevOps methodologies to enable portable and reproducible science, decreasing time to analysis

Composable Subsystem

A platform for “SciOps”

- Composable infrastructure provides highly customizable, on-demand provisioning of pools of computing resources (CPU/GPU, mem, disk, network)

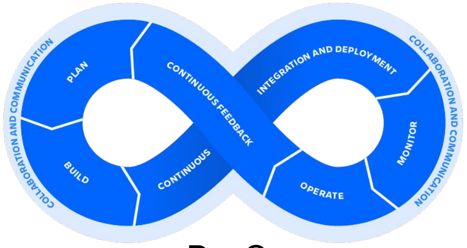
- DevOps Principles
  - Infrastructure as code
  - Version Control
  - Automation
  - Portability
  - Reproducibility



Developers

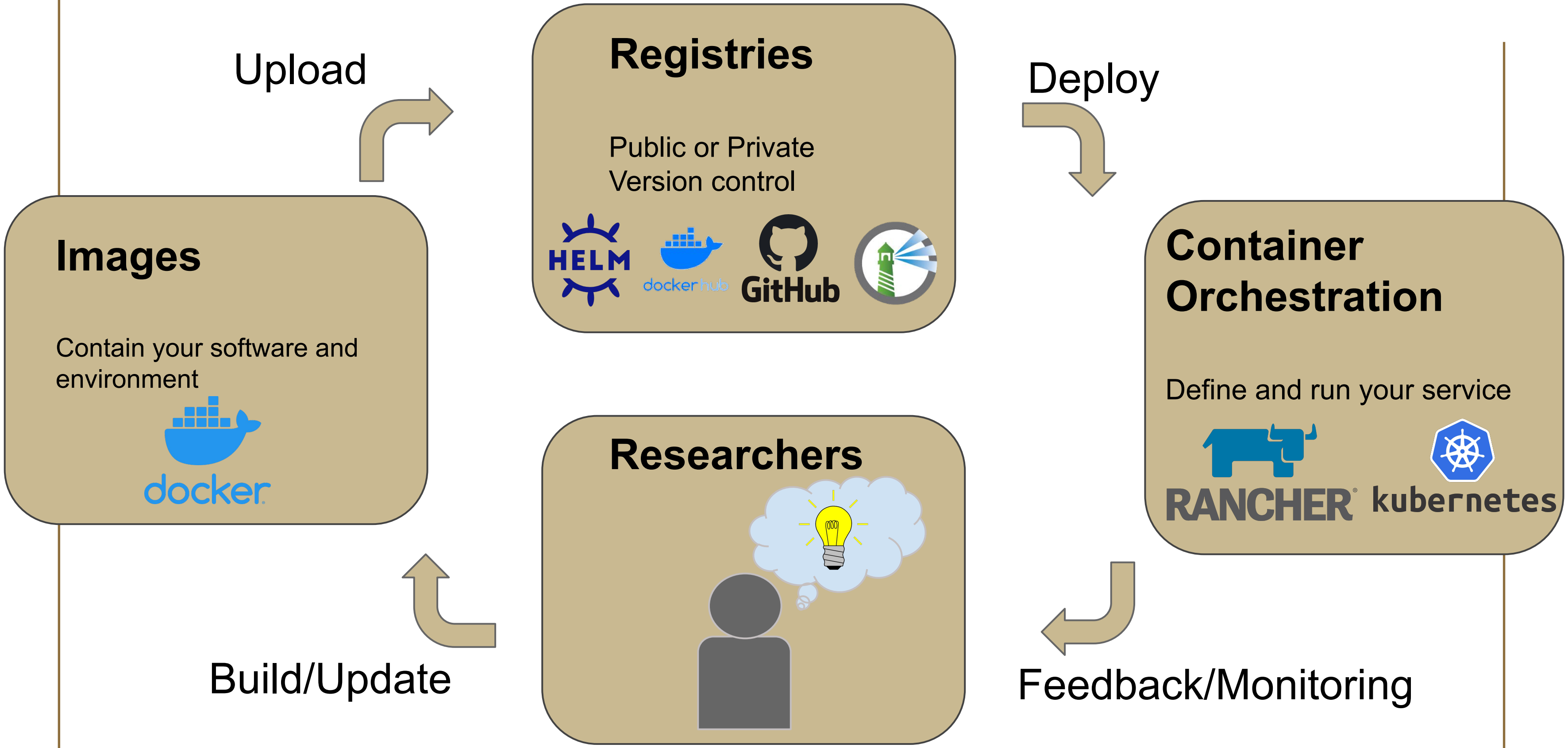
Operations

Traditional


COLLABORATION AND COMMUNICATION
PLAN
BUILD
CONTINUOUS
CONTINUOUS FEEDBACK
INTEGRATION AND DEPLOYMENT
MONITOR
OPERATE
COLLABORATION AND COMMUNICATION

DevOps
Source

Composable Infrastructure



Upload

Registries

Public or Private
Version control

Deploy

Container Orchestration

Define and run your service

Feedback/Monitoring

Researchers

Build/Update

Images

Contain your software and environment

"SciOps" with the Composable Subsystem

Docker

an open-source software project for automating the deployment of applications inside software containers

Containerization technology

- Brought containerization to the masses
- Provides lightweight OS-level virtualization
- Wide adoption in the containerization ecosystem

Wide scale adoption

- Lots of documentation
- Large and active community
- Many surrounding tools and utilities available for use
- Used in many large scale deployments

Increases application scalability

- Cross platform
- Ease of horizontal scaling
- Write locally, ship anywhere
- Isolated environments

Increases developer velocity

- Consistent environments
- Bundled dependencies
- Increased deployment speed
- Solves the “it works in my machine” problem
- Makes sharing code easy

Kubernetes

an open-source container-orchestration system for automating deployment, scaling, and management.

Orchestrates Containers

- Auto-scaling
- Failover
- Monitoring
- Alerting
- On premise, in cloud or hybrid

Networking Features

- Enforced security policies
- Highly scalable
- Flexible via plugins
- SDN based solutions

Provides robust storage interfaces

- Ceph block and filesystem storage
- NFS
- SMB/CIFS
- Many others

Velocity focused

- Templated infrastructure
- Version controlled code
- CI/CD integrations

Rancher

an open-source management platform for Kubernetes container systems

Deployment Flexibility

- Can manage multiple cluster types and locations
- Built in tools to help with administration and addons
- Can bring in pre-existing K8s clusters
- Runs on prem, in cloud, or hybrid as a self hosted or paid service

Rancher Interfaces (UI & CLI)

- Easy to use
- Helps with user onboarding to K8s native environments
- Single management for many clusters

Kubernetes management

- Provisioning clusters
- Catalog management
- Managing projects
- Pipelines

Authorization and RBAC

- Centralized user authentication
- Access control and security policies
- Identity integration
- Projects and members
- Users and roles

Container Registries

a repository used to store container images and related artifacts for container-based environments

Centralized image management

- Control of images
- Cost savings
- Speed
- Accountability

Security and access

- Vulnerability audits and alerting
- Identity integration
- RBAC or ACLs
- Image patching
- Image trust signing

CI/CD integration

- Automated deployment pipelines
- Public or private access

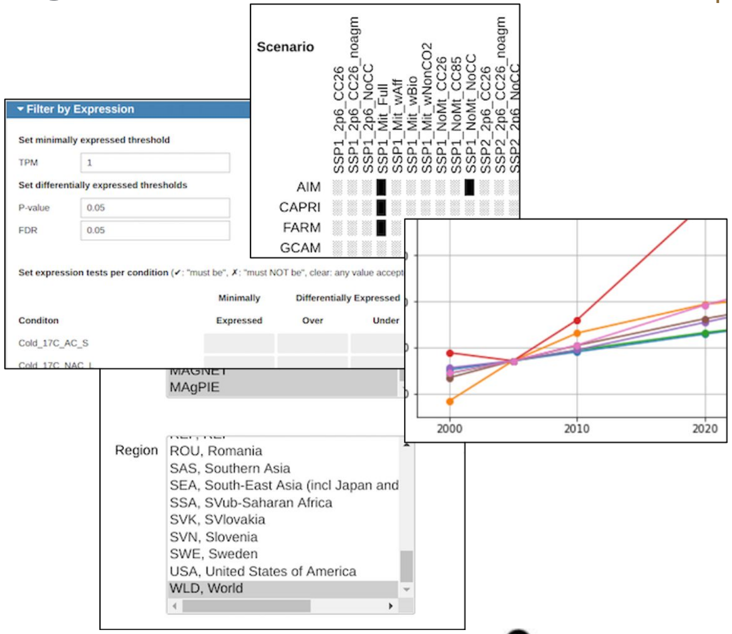
Registry examples

- Docker Hub
- GitHub
- Harbor
- Quay

Composable Subsystem In Practice

Applications, Services, Gateways

- Host science gateways
- Access other compute components on-demand (GPU, large memory)
- Host long-running, resilient, scalable data processing pipelines
- Host public interactive applications



Filter by Expression

Set minimally expressed threshold

TPM 1

Set differentially expressed thresholds

P-value 0.05

FDR 0.05

Set expression tests per condition (✓: "must be", ✗: "must NOT be", clear: any value accept

Conditon	Minimally Expressed	Differentially Expressed Over	Under
Cold_17C_AC_S			
Cold_17C_NAC_L			

Scenario: SSP1_2p6_CC26, SSP1_2p6_CC26_noagm, SSP1_2p6_NoCC, SSP1_Mit_Full, SSP1_Mit_wAff, SSP1_Mit_wBio, SSP1_Mit_wNonCO2, SSP1_NoMt_CC26, SSP1_NoMt_CC85, SSP1_NoMt_NoCC, SSP2_2p6_CC26, SSP2_2p6_CC26_noagm, SSP2_2p6_NoCC

AIM, CAPRI, FARM, GCAM

MAGNET
MAgPIE

Region: ROU, Romania; SAS, Southern Asia; SEA, South-East Asia (incl Japan and; SSA, SVub-Saharan Africa; SVK, SVlovakia; SVN, Slovenia; SWE, Sweden; USA, United States of America; WLD, World

RabbitMQ

hubzero

APACHE Spark

APACHE kafka
A distributed streaming platform

GO

React

Terminology

- A **container** is the abstraction that holds an application and all of its dependencies needed for deployment
- These are stored and transported as an **image.**

- A **workload** is an application running in K8s.
- A **pod** is a group of one or more containers with shared resources and runtime specification
- A **namespace** is a virtual cluster abstraction backed by the same physical cluster.
- **Ingress** is an service that manages external access to the services in a cluster, it is used primarily for HTTP/HTTPS traffic, but can handle other traffic as well.
- An alternative service is the **load balancer** which provides a gateway for external connections to access your cluster.

- A **project** is a group of one or more namespaces that can be operated on as a like group.

Composable Hands On Exercise

Exercise 2: Rancher User Interface Access

2.1 Navigate to and login the Rancher UI

1. Open Firefox in your Open OnDemand Desktop session (If you click the Browser icon in the tray, you may have to select Mozilla FIrefox as your preferred Browser)
2. Open the URL https://beta.geddes.rcac.purdue.edu
3. To log into Rancher, click **Use a Local User**
4. Fill in your training account information and click the **Log In as Local User** button

Note: The Rancher URL is only accessible from your OOD Desktop session

2.2 Explore the Rancher UI

A guided tour of the Rancher web interface from tutorial presenter

2.3 Create a new namespace within your Rancher project

Refer to https://tinyurl.com/pearc21-anvil-tutorial for the steps to create a namespace

Part 3: Application Deployment

Application Deployment

Workloads

- A **workload** is an application running in Kubernetes
- Workloads run inside one or more **pods**
- A pod is a group of one or more **containers**
- Containers are created from Docker **images**
- Workload resources
  - Deployment - Deploy one or more pods
  - DaemonSet - Deploy a pod on all Kubernetes nodes
  - StatefulSet - Deploy one or more pods, tracking state
  - Job - Run a Pod and exit (batch)
  - Cron Job - Tasks that run on a schedule
- Rancher defaults to using scalable Deployments, which works well for most users' needs

Demo: Deploying a workload via the Rancher UI

Application Deployment

Persistent Storage

- Containers are ephemeral
- If your application needs to retain data, then you need some type of persistent storage
- Pods can be configured to mount persistent storage at a location in the filesystem
- Terminology
  - Storage Class* - a type of storage
  - Persistent Volume (PV) - a piece of a Storage Class
  - Persistent Volume Claim (PVC) - a request for storage from a user

* Anvil users will have access to Ceph Block and Filesystem classes, along with object storage

Persistent Storage

Persistent Storage Example

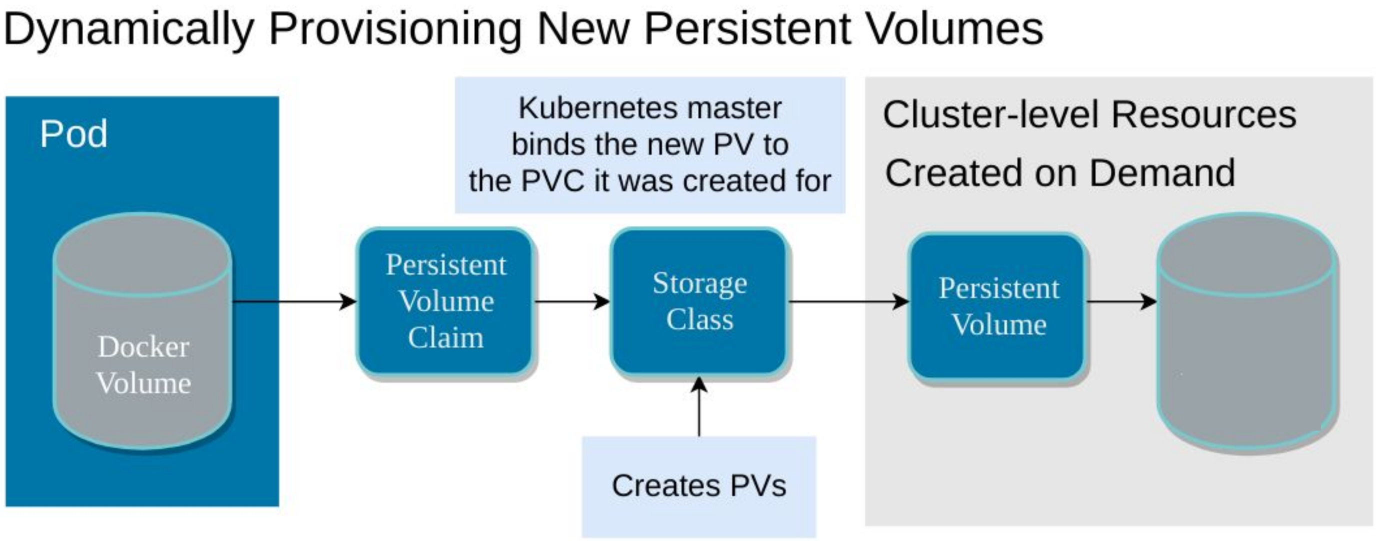

Dynamically Provisioning New Persistent Volumes
Pod
Docker Volume
Persistent Volume Claim
Kubernetes master binds the new PV to the PVC it was created for
Storage Class
Creates PVs
Cluster-level Resources Created on Demand
Persistent Volume

Image source

Persistent storage complexity is masked by the Rancher UI. Users just need to click to create a PVC and provide a directory to mount it in a Pod

Composable Hands On Exercise

Exercise 3: Application Deployment

In this exercise, we will use Rancher to deploy a Postgres database that uses persistent storage from Longhorn. We will use kubectl and the Rancher UI to populate the database with Geographical Information System (GIS) data of the Michigan highway system.

Tutorial Materials: https://tinyurl.com/pearc21-anvil-tutorial

Part 4: Interacting with Applications

Networking

Networking Basics

- DNS maps a hostname to an IP address
  - myapp.pearc.rcac.purdue.edu -> 192.168.138.90
- Applications listen for connections on a socket, the combination of an IP address and port
  - <ip address>:<port>
  - 192.168.138.90:443
  - myapp.pearc.geddes.rcac.purdue.edu:443
- Communication between pods in Kubernetes occurs on a private cluster network
  - Handled via the Container Network Interface (CNI)
  - Network isolation applied at the Project level
- Kubernetes has its own DNS service
  - Provides internal addresses: myapp.pearc.geddes.local
  - And external: myapp.pearc.geddes.rcac.purdue.edu

Kubernetes Services

Services

- A Kubernetes **Service** is a way to expose an application on a network
- You choose the network!
  - Kubernetes internal network
  - Purdue private network
  - Purdue public network
- Service Types
  - **ClusterIP** - Open a port on an internal network
  - **NodePort** - Open a port on every Kubernetes node
  - **HostPort** - Open a port on a Kubernetes node (not recommended)
  - **LoadBalancer** - Open a port on a dynamically assigned IP address
  - And others...
- Configurable under the “Port Mapping” section when deploying a workload

Load Balancing

Layer-4 Load Balancer

- Allocates an IP address on your desired network, which listens on your desired port and distributes traffic to pods in a workload
- mydb.pearc.geddess.rcac.purdue.edu:3306
- The DNS record is created automatically
  - <servicename>.<namespace>.geddes.rcac.purdue.edu
  - IPs are not static! You might get a different IP if you redeploy a LoadBalancer service, so just use the DNS name

Ingress

- A “Layer 7” load balancer that distributes traffic to web-based services based on hostname and path
- Uses the Nginx Ingress Controller
- HTTP (80) and HTTPS (443) traffic
- myapp.pearc.geddes.rcac.purdue.edu/myapi -> workload1
- myapp.pearc.geddes.rcac.purdue.edu/myotherapi -> workload2

Note: Make sure your application is secure before you configure one of these

Composable Hands On Exercise

Exercise 4: Accessing Applications Externally

In this exercise, we will use a LoadBalancer service to automatically assign an IP address on a private network at Purdue and open the postgres port (5432). A DNS name will automatically be configured for your service. We will then interact with the database via a Jupyter notebook launched via Open OnDemand.

Tutorial materials: https://tinyurl.com/pearc21-anvil-tutorial

Part 5: Additional Topics

Scalability

Horizontal Pod Autoscaler (HPA)

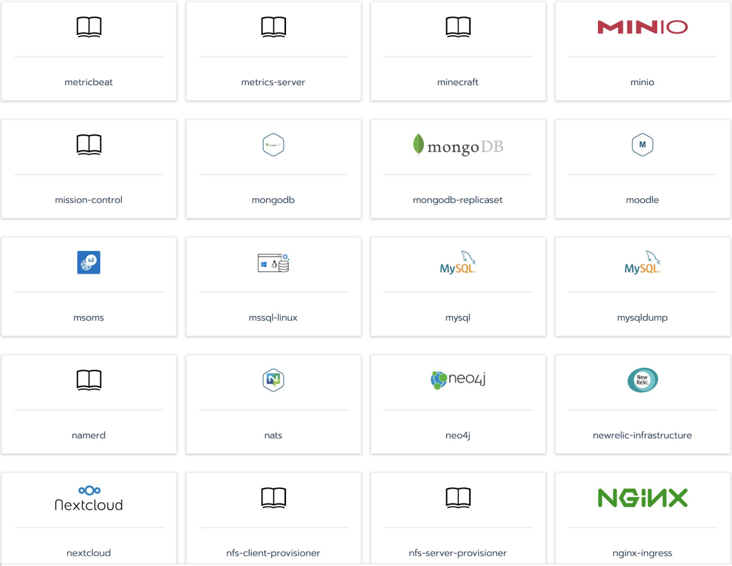
- Provides a mechanism to automatically scale the number of Pods in a workload
- Configurable resource metrics
  - CPU utilization
  - Memory usage
- “Add more replicas of my Pod if average CPU utilization is greater than 80%”
- Configurable minimum and maximum replica count
- Replicas will automatically scale down when resources go below the threshold

Application Deployment

Helm Charts

- Helm charts allow users to flexibly define and deploy applications in Kubernetes
- Accessed under the **Apps** tab in the Rancher UI
- A global set of charts is available to all users and users can add their own charts, making them available to all Project members

“Push button” applications



Composable Hands On Exercise

Exercise 5: Advanced Topics and CYO Adventure

In the final exercise, users will deploy a science gateway, make it accessible on the Internet and use the Horizontal Pod Autoscaler to scale the application. Participants can also investigate the Apps tab to deploy “push button” applications.

If you have a different use case that you are interested in, feel free to try it out on the platform.

Presenters are available to help with any questions for the remainder of the scheduled tutorial time.

Tutorial Materials: https://tinyurl.com/pearc21-anvil-tutorial

Tutorial Wrap Up

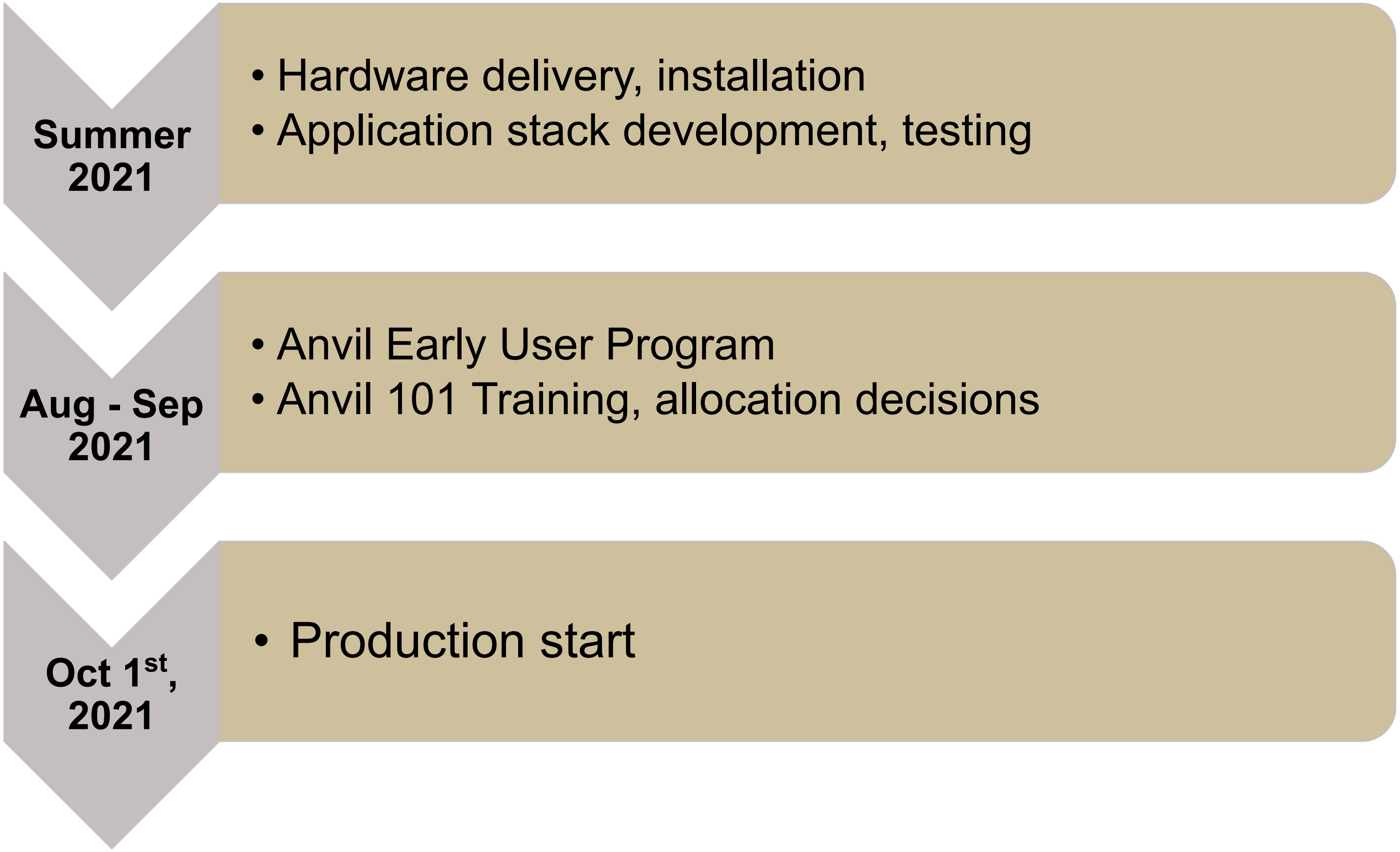
Accounts and Allocations

- Resource Types: Anvil, Anvil GPU, storage, composable

- Request one or more resources through XSEDE submission process for allocations beginning October 1, 2021.
  - (one-time) Startup allocations can be requested at any time
  - Research allocation request is reviewed quarterly

- Variety of queues with varying wall-time and job priority restrictions
  - Dedicated high priority interactive queue (OnDemand)
  - Exclusive/shared CPU, GPU queues
  - Long running jobs: 96 hours (max)
  - Large core jobs: 7680 cores (max)
  - Dedicated time could be reserved for large-scale testing

Deployment Timeline



Summer 2021

- Hardware delivery, installation
- Application stack development, testing

Aug - Sep 2021

- Anvil Early User Program
- Anvil 101 Training, allocation decisions

Oct 1st, 2021

- Production start

Thank You!

Anvil Webpage

https://rcac.purdue.edu/anvil

XSEDE allocation request

https://portal.xsede.org/submit-request

Send questions to:

anvil@purdue.edu

Jun 15th – Jul 15th	Anvil available for XSEDE allocation
Aug 01st – Aug 31st	Early User Program
Oct 1st	Anvil enters production