Forthcoming SSCC Technical Seminars

Seminar No	Date	Presenter /s	Title
35	12/10/22	Richard Brearton	Simple high-performance computing with Python and NumPy
36	**26/10/22**	**-Space Available-**	**-Space Available-**
37	09/11/22	Chris Burrows	DLS Vacuum Coating Research & Development Facility
38	23/11/22	Alejandra Gonzalez Beltran & Steve Collins	Diamond Data Store
39	**07/12/22**	**-Space Available-**	**-Space Available-**
40	**11/01/23**	**-Space Available-**	**-Space Available-**

Volunteers to present future seminars are required.
Please contact Shoaib Khaja or Mark Heron with a proposed topic.

diamond

Kubernetes for EPICS IOCs

SSCC Technical Seminar

giles knap

28/09/2022

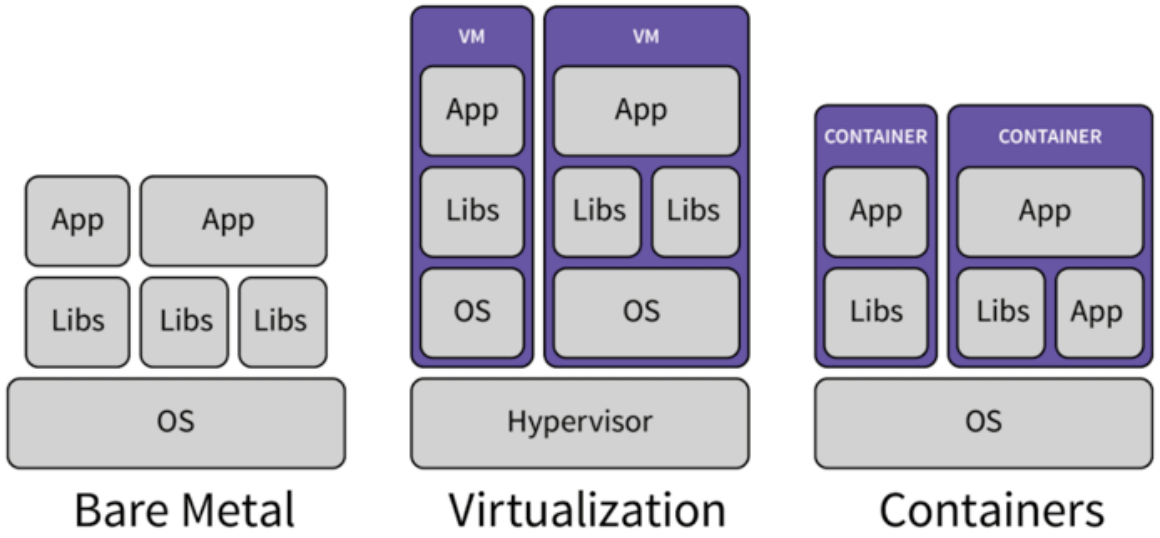
https://github.com/epics-containers/ec-talk

diamond

Kubernetes for EPICS IOCs

- Applying modern industry standards to manage EPICS IOCs
  - Containers
    - Package IOC software and execute it in a lightweight virtual environment
  - Kubernetes
    - Centrally orchestrate all IOCs at a facility
  - Helm Charts
    - Deploy IOCs into Kubernetes with version management
  - Repositories
    - Source, container and helm repositories manage all the above assets
  - Continuous Integration / Delivery
    - Source repositories automatically build assets when changes are pushed
  - Developer Environment – dev containers
    - Developers use the same container as Kubernetes, guaranteeing a matching environment.

Introduction to Containers



Containers, like VMs, isolate an application and its dependencies into a self-contained unit that can run anywhere.

Lightweight

- starts with just one process
- shares the kernel with the host OS

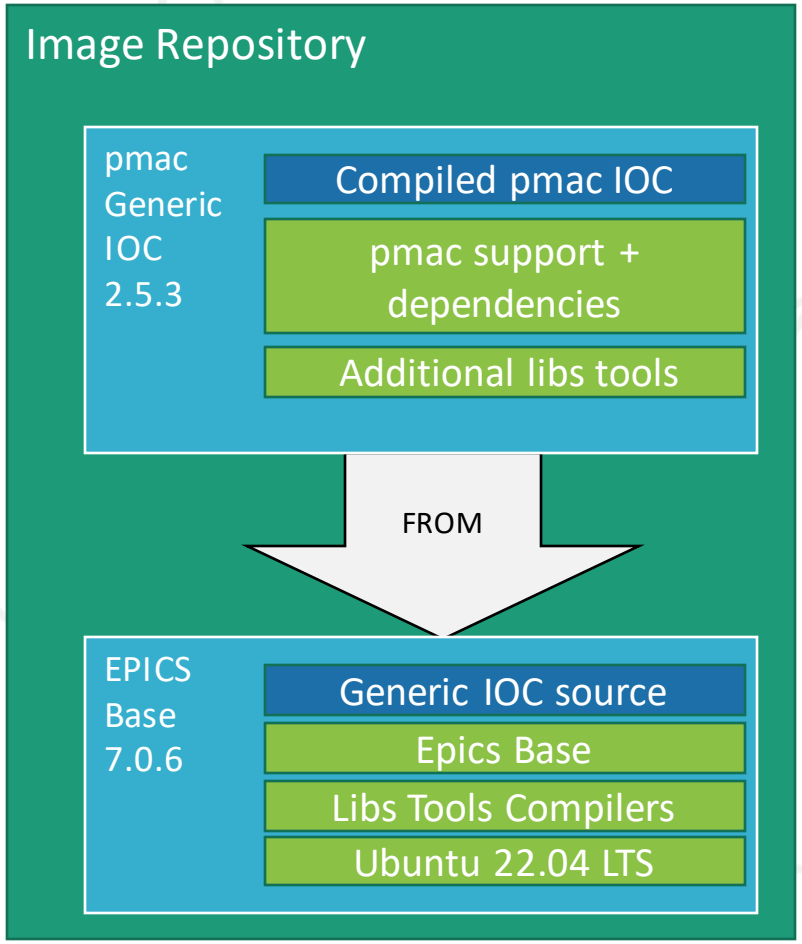
A container's virtual filesystem is initialized with an image.

Image registries like Docker Hub hold many useful predefined images.

New custom images can be derived from these by adding extra layers of software.

diamond

IOCs in containers



- IOC images represent a **generic** IOC
- The same image is used for every IOC that controls a given class of device
- An IOC **instance** is a container based on a **generic** IOC image with an added startup script that makes it unique
- This example shows the filesystem layers in the **generic** IOC for the standard DLS Motion Controller

diamond

Example Container Definition

Dockerfile and ioc.yaml

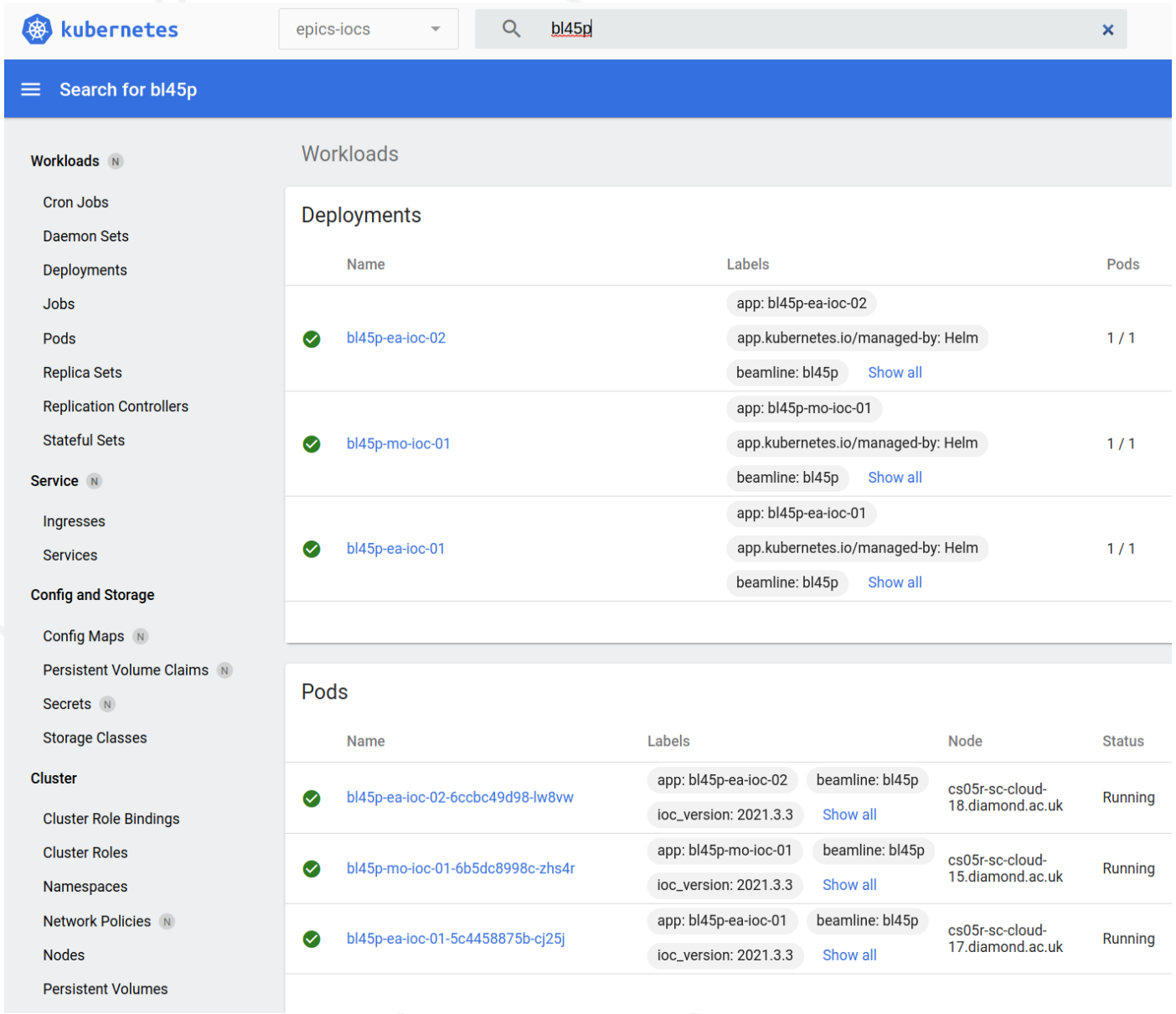
From the ioc-pmac project

Takeaways from ioc-pmac Dockerfile

- Repeatability
  - The Dockerfile precisely defines the environment in which the IOC will run
- Version control
  - The file is text only so git can track the history of changes to the environment
- Layered
  - The container's filesystem adds a new hashed layer for each command in the Dockerfile.
    - Faster builds because layers are cached
    - Saved disk and memory – layers are shared by all container instances
- Staged Build
  - Separate targets provide a lightweight runtime image and fully loaded build+developer image
- Generic IOC template
  - Make your own generic IOC by changing the YAML file and the system installs only

diamond

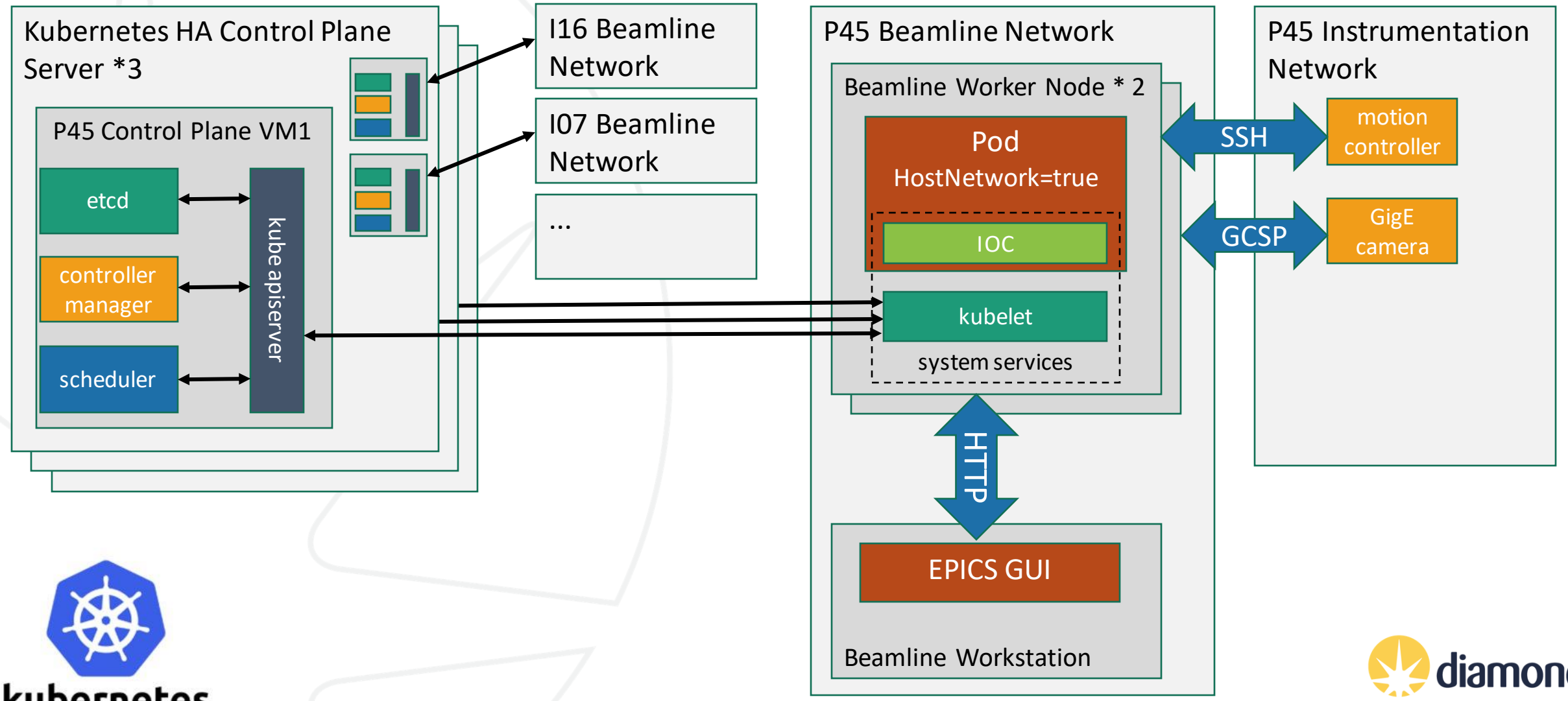
Introduction to Kubernetes https://kubernetes.io



- Kubernetes efficiently manages containers across clusters of host machines.
- It builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community
- Today it is by far the dominant orchestration technology for containers
- Many household names have adopted it:
  - CERN
  - Spotify
  - Twitter
  - Many more .. https://kubernetes.io/case-studies/
- This screenshot shows the standard K8s dashboard managing P45 IOCs

diamond

Proposed DLS Kubernetes Cluster Topology



kubernetes

diamond

Example Kubernetes Manifest

An IOC Instance

BL45P motion IOC 90

Takeaways from the Kubernetes Manifest

- Declarative
  - Describes the application requirements – Kubernetes allocates the best resources to meet those requirements
- Idempotent
  - Re-applying the same manifest has no affect. Modifying it and applying adds the changes only
- Lots of YAML to write
  - Therefore, we use helm to create IOC manifests

diamond

Introduction to Helm

- Helm is the most popular package manager for K8S applications
- The packages are called Helm Charts
- Charts contain templated YAML files to define a set of resources to apply to a Kubernetes cluster
- Helm has functions to deploy Charts to a cluster and manage multiple versions of the chart within the cluster
- It also supports registries for storing version history of charts, this is allogenous to container registries

Additional Tools

- Everything mentioned so far represents standard, widely available tools that are free to open-source projects
- To make IOC development even easier, the following optional tools are recommended:
  - **vscode** – a code editor that provides tight integration with dev containers.
  - **ibek** – IOC builder for EPICS on Kubernetes – generates IOCs from YAML descriptions. A DLS tool.
  - **ec** – a command line assistant for EPICS containers commands. A DLS tool.

diamond

ibek – IOC builder for EPICS on Kubernetes

- Uses YAML files to describe:
  - Support module definitions
    - startup script commands and parameters
    - database templates and parameters
    - libs and dbds for the IOC Makefile
  - Generic IOC build
    - List of support modules to link into generic IOC binary
  - IOC Instances
    - List of the above support module entities to instantiate
    - With arguments
- YAML Schema
  - Helps developers create IOC YAML

- Generates:
  - Generic IOC container build
    - Clone and build dependencies
    - IOC Makefile
  - Beamline repo build
    - Helm chart to install the IOC into the cluster
  - Generic IOC container startup
    - IOC startup script
    - IOC database

- Optional tool
  - **ibek** is optional for IOC instances
  - Traditional startup script and substitution files will also work

diamond

Example IOC

Beamline Definition

BL45P source repo

https://github.com/epics-containers/bl45p

- Beamline repository
- **ibek** YAML
- Helm chart

Takeaways from Beamline Definition

- Beamline repository holds a description of each IOC instance
- IOC instances are described using ibek YAML
- Building the beamline makes a helm chart for each IOC

Repositories and Continuous Integration

- Kubernetes for EPICS IOCs uses these types of repository:
  - Generic IOC images source repo
    - CI publishes a generic IOC image to the image repo
  - Beamline definition source repo
    - CI publishes a helm chart for each IOC instance to the beamline Helm Repo
  - Image repository
    - Holds container images for generic IOCs – updated by CI of Generic IOCs
  - Helm Charts repository
    - Holds versioned helm charts for IOC instance – updated by CI of Beamline definitions
- Helm deploys to Kubernetes directly from the repository
- Kubernetes pulls generic IOC images directly from the image repository
- No intermediate shared filesystem is required.

Example Repositories

CI and Repositories for BL45P

https://github.com/epics-containers

Developer Environment - dev containers

- Locally run IOCs will:
  - Execute in the same environment as Kubernetes
  - Build in the same environment as CI
- Dev container can run anywhere:
  - provides a complete developer environment, with all work saved to the workstation filesystem
- These CLI tools will be essential to the developer
  - **kubectl** – send commands to the Kubernetes cluster
  - **helm** – package manager for Kubernetes applications
  - **podman** – container management for the local workstation
  - **ec** – the Epics Containers assistant CLI - *developed by DLS*
- Documentation will provide prescriptive workflows for creating / debugging IOCs and support modules

diamond

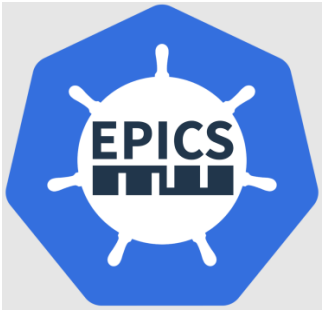
ec - Epics Containers assistant CLI

- Written in Python
- Provides a very thin layer of assistance to save typing and help with adoption.
- Has command line completion and CLI help
- Only uses **kubectl**, **helm** and **podman**.
- Prints the system commands as a learning aid e.g.

```
$ ec deploy bl45p-mo-ioc-01 0.0.1-b0
+ helm upgrade --install bl45p-mo-ioc-01 oci://ghcr.io/epics-containers/bl45p-mo-ioc-01 --version 0.0.1-b0
```

diamond

Current Status



- The test beamline BL45P has its IOCs running on Beamline worker nodes managed by DLS' Pollux cluster.
- We are planning to apply this approach for MX Bridge and Diamond II
- All source and assets for the BL45P POC work are published at
  - https://github.com/epics-containers
- The organization includes documentation, so anyone to try it out
  - https://epics-containers.github.io/
- The docs include a tutorial that walks through setting up a mini-Kubernetes cluster and deploying an ADSimDetector IOC.
  - https://epics-containers.github.io/main/tutorials/setup_k8s.html
- Please Join the organization and contribute your own ideas!

diamond

Benefits

- Containers are decoupled from the host OS and each other.
  - Great for collaboration:
    - All BL45P IOC dependencies are vanilla EPICS Community support modules
  - RHEL upgrades:
    - don't affect IOCs or EPICS developer tools!
  - Run anywhere:
    - develop, test, demo on a laptop or home machine.
- Standard tools:
  - If something goes wrong: Google it.
- Kubernetes provides economy of scale through centralized:
  - Software deployment and management
  - Logging and Monitoring
  - Resource management: Disk, CPU, Memory
- Remove maintenance of internal IOC management tools
- Remove need for shared filesystem

diamond

Thanks for listening

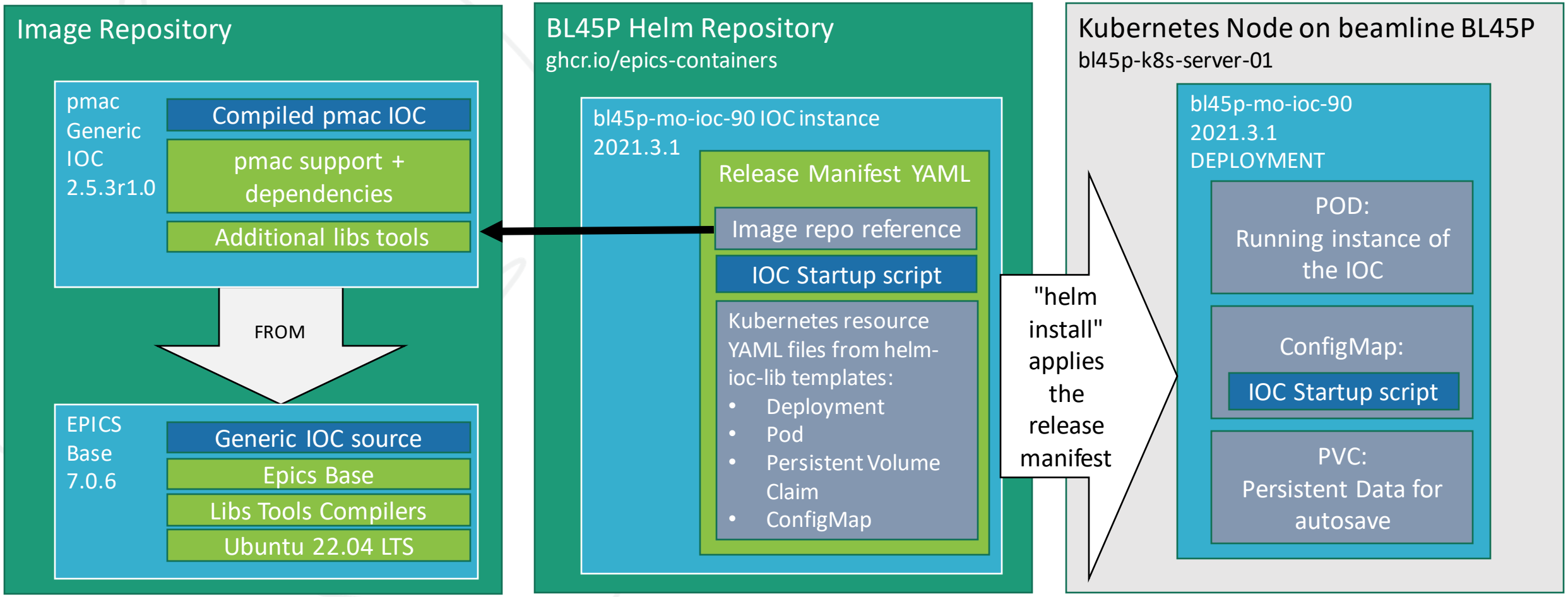
Any questions?

Features Provided by Kubernetes and Helm

- Auto start IOCs when servers come up
- Restart crashed IOCs
- Manually Start and Stop IOCs
- Allocate the server which runs an IOC
- Move IOCs if a server fails
- Throttle IOCs that exceed CPU limit
- Restart IOCs that exceed Memory limit
- Deploy versioned IOCs to the beamline
- Track historical IOC versions
- Rollback to a previous IOC version
- Monitor IOC status and versions
- View the current log
- View historical logs (via graylog)
- Connect to an IOC and interact with its shell

diamond

Summary: Deploying bl45p-mo-ioc-90



- A Helm Chart defines an IOC instance: IMAGE + STARTUP SCRIPT + K8S DEPLOYMENT YAML
- The entire definition of the P45 beamline is held in https://github.com/orgs/epics-containers/packages