epics-containers

A deep dive!

Oxfordshire EPICS Meeting
Nov 3rd, 2023

giles knap
Beamline Controls – Diamond Light Source

diamond

What?

Applying modern industry practices for software delivery to EPICS IOCs

Containers: Package IOC software and execute it in a lightweight virtual environment

Kubernetes: Centrally orchestrates all IOCs at a facility

Helm Charts: Deploy IOCs into Kubernetes with version management

Repositories: Source and container repositories manage all the above assets

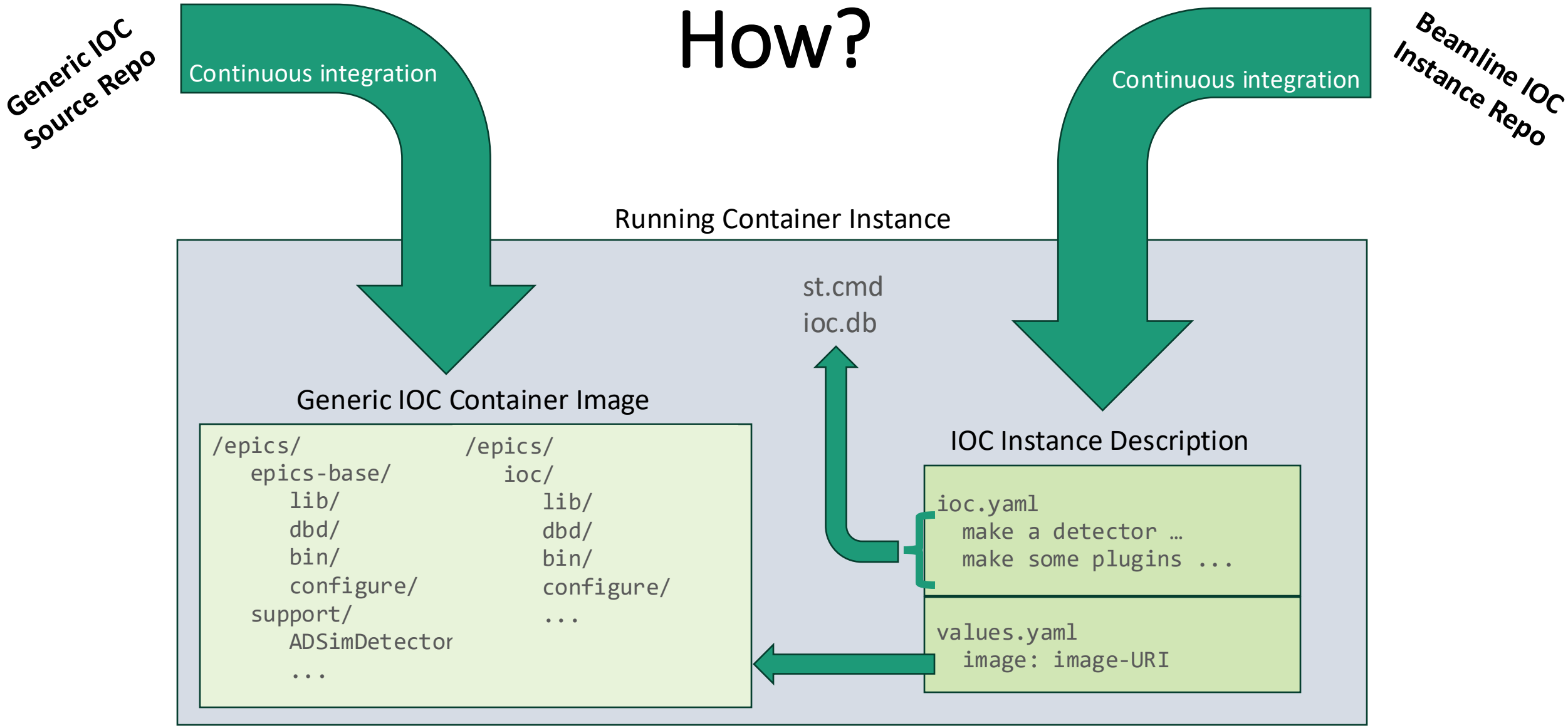
Continuous Integration: Source repositories automatically build assets from source when updated

Why?

- IOCs are decoupled from the OS
  - No modifying support modules to suit facility infrastructure
  - Allows us to use upstream support modules with no need for local forks
  - Develop and test anywhere
  - Protection from many security vulnerabilities
- Simple server setup: any Linux + container runtime only.
  - Very easy server OS upgrades
- Remove maintenance of internal management tools
- Kubernetes provides (not just for IOCs):
  - Shared software deployment and management
  - Shared Logging and monitoring
  - Shared resource management: Disk / CPU / Memory
  - Just Google it when things go wrong

diamond

How?


Generic IOC Source Repo
Continuous integration
Continuous integration
Beamline IOC Instance Repo
Running Container Instance
st.cmd
ioc.db
Generic IOC Container Image
/epics/
epics-base/
lib/
dbd/
bin/
configure/
support/
ADSimDetector
...
/epics/
ioc/
lib/
dbd/
bin/
configure/
...
IOC Instance Description
ioc.yaml
make a detector …
make some plugins ...
values.yaml
image: image-URI

Supporting Tools

- **ibek –** IOC builder for EPICS and Kubernetes
  - Runs inside the container at build time
    - Helpers for building Support and Generic IOC in the container environment
  - Runs inside the container at runtime startup
    - Makes IOC instance startup script and database from a YAML description
    - Extracts IOC instance engineering screens from the container

- **ec –** the epics containers CLI for developers
  - Runs outside the container
    - Helpers for building and deploy IOC Instances
    - Helpers for local debugging and testing of Generic IOCs
    - Also used by Continuous Integration – **simplifies Bash Scripts**

- **PVI -** Process Variables Interface
  - Provides structure for the PV interface to a device
  - Auto generates engineering screens for the device (bob/edm/adl)
  - Supplies DB metadata for use with Bluesky

diamond

How to make an IOC Instance?

1. Create a beamline repo

- using template on GitHub
- add a subfolder to `iocs`
- create new IOC Instance config

values.yaml

bl45p > iocs > bl45p-ea-ioc-01 > values.yaml > ...

```yaml
image: ghcr.io/epics-containers/ioc-adaravis-linux-runtime:2023.10.1
```

A helm chart values override file for the beamline helm chart. The only required field is the URL of the Generic IOC to use.

ioc.yaml

bl45p > iocs > bl45p-ea-ioc-01 > config > ioc.yaml > ...

```yaml
# yaml-language-server: $schema=https://github.com/epics-containers/ioc-ad

ioc_name: bl45p-ea-ioc-01
description: Sample and overview cameras IOC for BL45P

# Just a comment to force rebuild

entities:
  - type: epics.EpicsCaMaxArrayBytes
    max_bytes: 6000000

  - type: ADAravis.aravisCamera
    CLASS: AVT_Mako_G234B
    PORT: DET.DET
    P: BL45P-EA-MAP-01
    R: ":DET:"
    ID: 172.23.59.11
```

*An **ibek** IOC YAML file listing the support entities that we want to instantiate for this IOC instance.*

2. Deploy the IOC instance:

- Tag and push the beamline repo
- **ec** ioc deploy bl45p-ea-ioc-01 2023.11.1

diamond

How to make a Generic IOC?

1. Create a Generic IOC repo

- using template on GitHub
- edit Dockerfile and Readme

2. Deploy Generic IOC to your container registry:

- Tag and push the generic IOC repo
- CI does the rest

To make a new generic IOC add the additional support modules required to the template Dockerfile

Dockerfile

ioc-adaravis > Dockerfile

```
1  ##### build stage ##############################################################
2
3  ARG TARGET_ARCHITECTURE
4  ARG BASE=7.0.7ec2
5  ARG REGISTRY=ghcr.io/epics-containers
6
7  FROM  ${REGISTRY}/epics-base-${TARGET_ARCHITECTURE}-developer:${BASE} AS developer
8
9  # the devcontainer mounts the project root to /epics/ioc-adaravis
10 WORKDIR /epics/ioc-adaravis/ibek-support
11
12 # copy the global ibek files
13 COPY ibek-support/_global/ _global
14
15 COPY ibek-support/iocStats/ iocStats
16 RUN iocStats/install.sh 3.1.16
17
18 ... other support modules in dependency order ...
19
20 COPY ibek-support/ADAravis/ ADAravis/
21 RUN ADAravis/install.sh R2-3
22
23 # create IOC source tree, generate Makefile and compile IOC Instance
24 RUN ibek ioc build
25
26 ##### runtime preparation stage ################################################
27
28 ... below is runtime stage - essentially boilerplate ...
29
```

How to Add a new Support Module

1. Add a folder in the ibek-support repo

- Shared ibek-support on GitHub
- Or private ibek-support per facility

2. Add a new install.sh file:

- Use the new support in your Generic IOC Dockerfile

Install.sh example for ADSimDetector. Most install.sh would look identical to this but You can add custom steps as Needed.

$ install.sh

ioc-adaravis > ibek-support > ADSimDetector > $ install.sh

```
#!/bin/bash
###############################################################################
###### install script for ADSimDetector Module ################################
###############################################################################

# ARGUMENTS:
#  $1 VERSION to install (must match repo tag)
VERSION=${1}
NAME=ADSimDetector

# log output and abort on failure
set -xe

# get the source and fix up the configure/RELEASE files
ibek support git-clone ${NAME} ${VERSION} --org http://github.com/areaDetector/
ibek support register ${NAME}

# declare the libs and DBDs that are required in ioc/iocApp/src/Makefile
ibek support add-libs simDetector
ibek support add-dbds simDetectorSupport.dbd

# compile the support module
ibek support compile ${NAME}
# prepare *.bob, *.pvi, *.ibek.support.yaml for access outside the container.
ibek support generate-links
```

diamond

How to Add a new Support Module

1. Add a folder in the ibek-support repo

- Shared ibek-support on GitHub
- Or private ibek-support per facility

2. Add a new install.sh file:

- Use the new support in your Generic IOC Dockerfile

3. Add an ibek support YAML description:

- This allows us to describe our IOC instances as ibek IOC YAML.

Support YAML example for ADSimDetector. Declares inst arguments, startup script line and database template substitutions.

```yaml
module: ADSimDetector

defs:
  - name: simDetector
    description: |-
      Creates a simulation detector

    args:
      - type: str
        name: P
        description: Device Prefix
      - type: str
        name: R
        description: Device Suffix
      - type: id
        name: PORT
        description: Port name for the detector
# ... other arguments omitted for brevity ...

    databases:
      - file: $(ADSIMDETECTOR)/db/simDetector.template
        args:
          P:
          R:
          PORT:
          TIMEOUT:
          ADDR:

    pre_init:
      - type: text
        value: |
          # simDetectorConfig(portName, maxSizeX, maxSizeY, dataType, maxBuffers, maxMemory)
          simDetectorConfig("{{PORT}}", {{WIDTH}}, {{HEIGHT}}, {{DATATYPE}}, {{BUFFERS}}, {{MEMORY}})
```

diamond

Need for Developer Containers

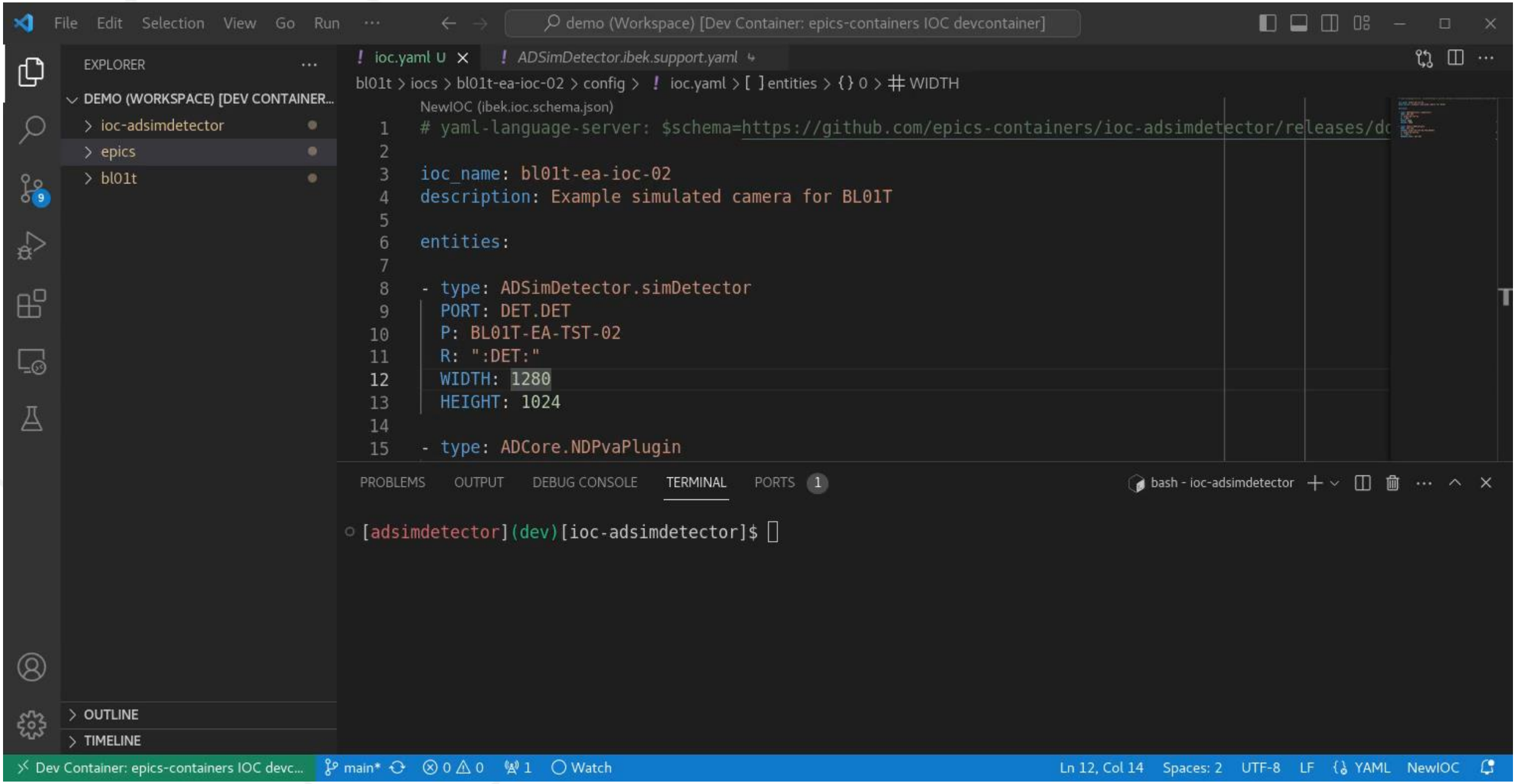
epics-containers development defines 3 levels of changes:

1. Changing IOC instance details only
   - Edit values.yaml or ioc.yaml in your beamline repository
   - Push and tag the changes, re-deploy the instance
2. Changing a Generic IOC
   - Edit Dockerfile in a Generic IOC repository
   - Push changes to publish a new container image
   - Go to 1. to update affected instances
3. Changing Support Modules
   - Edit the support module, verify and push source changes.
   - Go to 2. to update a Generic IOC to include the new support version

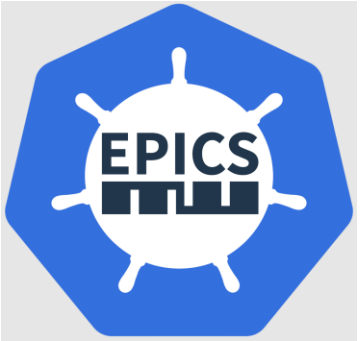
2 and 3 require rebuilding and deploying containers. For this reason, we use Developer Containers for a fast "inner loop".

diamond

Developer Containers



Is Kubernetes Required?

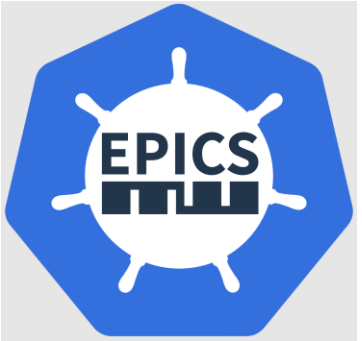


Developers can adopt epics-containers at several levels:

1. Just use precompiled Generic IOCs. Provide your own startup assets and your own approach to managing how containers are launched.
2. As above plus contribute your own Generic IOC container images
   - We recommend using **ibek** and **ibek-support** installers in the container build for consistency and because it is easy.
3. Additionally adopt epics-containers-cli to deploy IOC Instances via:
   - Kubernetes and Helm
   - Local Docker instance per server: *no Kubernetes required*
4. And / Or additionally adopt **ibek** runtime to generate your instance startup assets from IOC YAML
5. As per 4. plus adopt PVI to generate engineering screens for each instance.

diamond

One Take Away



https://github.com/epics-containers

- Includes:
  - Tutorials
  - Documentation
  - Templates
  - Source code
  - A small but growing number of Generic IOC images

diamond

Thanks for listening

Any Questions?

For more information see

https://github.com/epics-containers

diamond