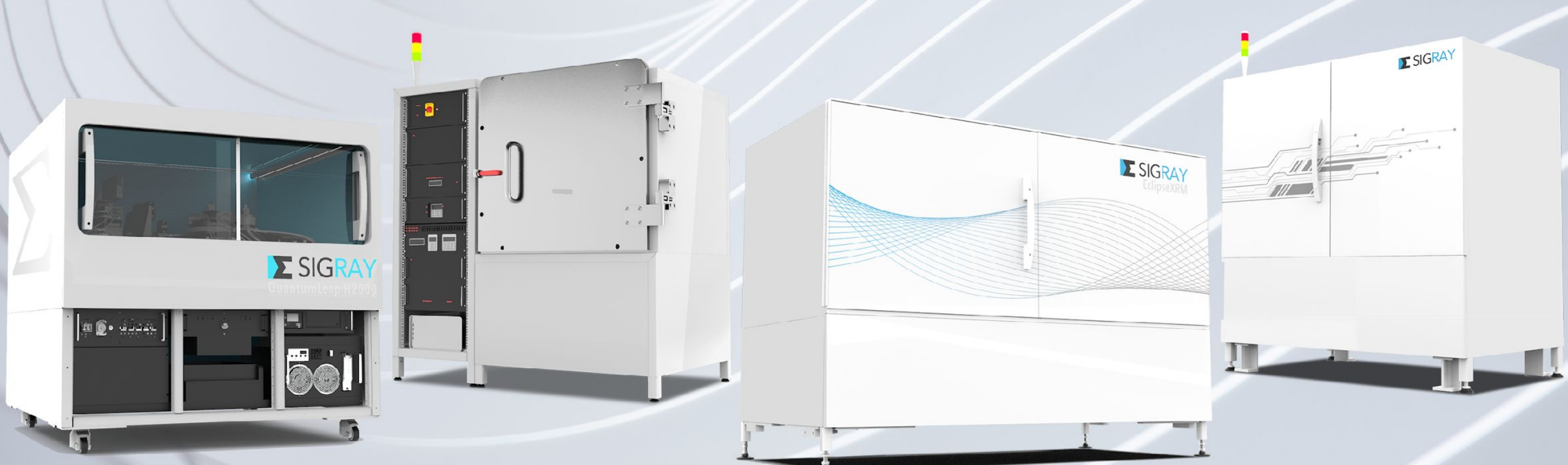
EPICS Deployment at Sigray using Docker

Presenter: Hong Truong
Co-authors: Benjamin Stripe, Ernesto Paiser, Ibrahim Saleh, Richard Farnsworth



Intro to Sigray

Mission: Bring next-generation x-ray analytical capabilities from the synchrotron to the laboratory

Founded in 2013

- Dr. Wenbing Yun (OSA Fellow and serial entrepreneur that founded Xradia, now Carl Zeiss X-ray Microscopy) and Sylvia Lewis

Our Technology:

- Strong IP: 64 patents, 30+ pending, many trade secrets
- Disruptive x-ray components (source & optics)
- 5 world leading product families

Rapidly Growing:

- 34k sq. ft. facility in Concord, CA (San Francisco Bay Area) and 82 employees
- Global installation base of leading universities and companies (semiconductor & pharma)

Refer to Benajmin Stripe's presentation for more information.

Contents

- Motivation
- Technologies
- How it works
- Usage
- Deploying updates
- Takeaways

Motivation

The pre-existing setup

- A machine has multiple devices, each controlled with an EPICS IOC
- IOC settings may be different between machines
    - Differences can include what db, template, or iocsh files are called, environment settings, IP addresses, etc.
    - Difficult to maneuver IOCs to allow us to do this natively
- We already had dockers for each IOC
    - Each docker had its own folder
    - Not in version control
- Difficulties
    - Hard to know what dockers are existed
    - Hard to track the configuration differences between machines
        - This leads to making it hard to identify and propagate fixes that were made on one machine and not others
    - Hard to reproduce a machine

Motivation

We wanted to ...

- Track the configuration of machines after shipment to be able to reproduce the machine
  - Set up new machines
  - Reinstall dockers if a computer fails
  - Debug issues with shipped machines
- Make it easy to share fixes and feature updates across all machines
  - Can involve changing synApps module versions, manual fixes to IOC source code via docker, system startup service scripts, etc.
- Standardize shared configuration to make debugging and setup easier
  - Ports, IP addresses, paths
- Allow offline updates (and builds) due to customer restrictions
- Make it easy to create new IOCs

Motivation

Who would be the user?

- A mix of users
  - Non-Linux users unfamiliar with a terminal or bash
  - Varying ranges of familiarity with Python and docker
- Controls engineers and software engineers
  - Add new IOCs and docker services
- Systems engineers
  - Customize files for the system (e.g. hdf5 xml files for areaDetector and db files)
- Field service engineers
  - Perform updates
  - Reload autosave files

Technologies

- Docker
  - Allow conflicting dependencies
  - Freeze dependencies
  - Easy to deploy
- Docker Compose
  - Allow inheritance of docker service definitions
  - Can define multiple dockers and their build and run settings in one file
- Portainer
  - Docker management
- Procserv
  - Wrapper for running IOCs as background processes with telnet access
- Bash
  - Helper scripts
  - Easier to call most commands directly
- Python
  - Helper scripts and GUI
  - Will replace most of the bash scripts because it's more readable and easier to debug
- Poetry
  - Python dependency manager
- Tests
  - Bats framework for bash tests
  - pytest for everything else

CIDER

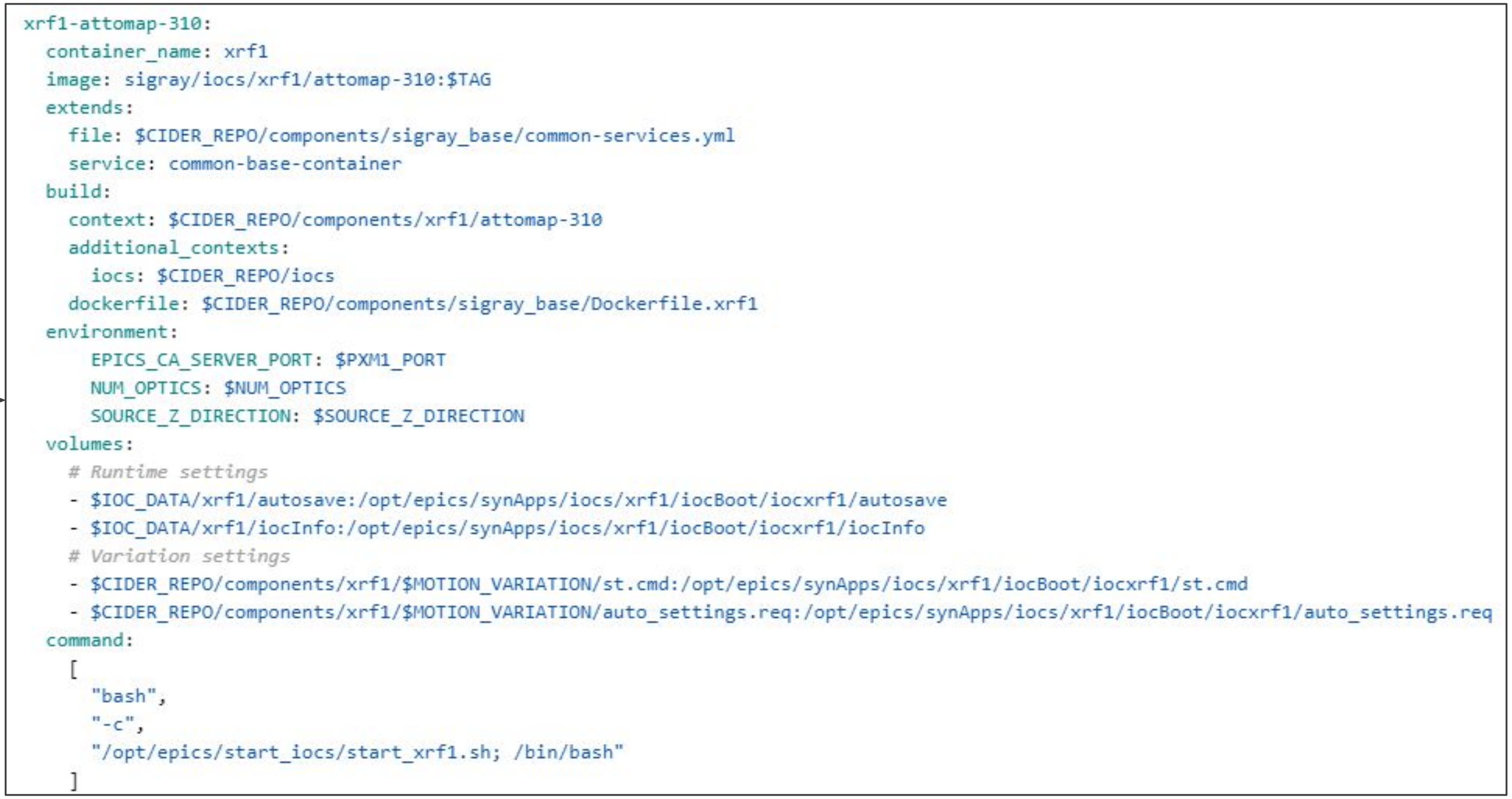
Overview

- Configuration, Installation, and Deployment of EPICS Repositories
- Basic idea
  - One repository to store all docker-related files and machine configuration files
  - Have one base docker image for online dependencies
    - Mostly installing synApps modules with assemble_synApps.sh
  - Each IOC docker image builds on top of the base docker image and includes **no online dependencies at build time**
  - Each machine mounts its own set of files at runtime, overwriting existing IOC files
  - Use environment variables and inheritance to avoid code duplication
- Allows us to:
  - View existing docker services for IOCs (and other utilities) and add them for specific machines
  - Make and track modifications to the IOCs per machine
  - Standardize values in an environment file and propagate them to all dockers

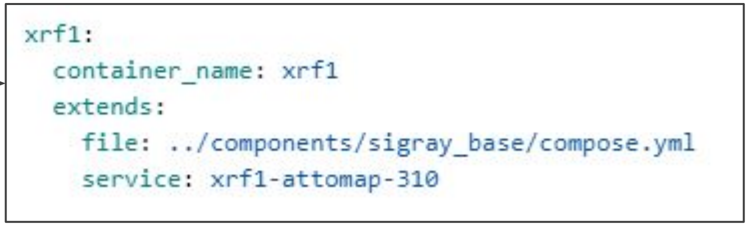
CIDER

File Structure Example

- components/
  - component_a/
    - common-services.yml
    - compose.yml
    - Dockerfile
    - common/
    - variation_1/
    - …
- ioc_submodules/
- machine_compose_files/
  - compose.am-1124.yml
- environment_files/
  - common_env_vars
  - am-1124.config



```
xrf1-attomap-310:
  container_name: xrf1
  image: sigray/iocs/xrf1/attomap-310:$TAG
  extends:
    file: $CIDER_REPO/components/sigray_base/common-services.yml
    service: common-base-container
  build:
    context: $CIDER_REPO/components/xrf1/attomap-310
    additional_contexts:
      iocs: $CIDER_REPO/iocs
    dockerfile: $CIDER_REPO/components/sigray_base/Dockerfile.xrf1
  environment:
      EPICS_CA_SERVER_PORT: $PXM1_PORT
      NUM_OPTICS: $NUM_OPTICS
      SOURCE_Z_DIRECTION: $SOURCE_Z_DIRECTION
  volumes:
    # Runtime settings
    - $IOC_DATA/xrf1/autosave:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/autosave
    - $IOC_DATA/xrf1/iocInfo:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/iocInfo
    # Variation settings
    - $CIDER_REPO/components/xrf1/$MOTION_VARIATION/st.cmd:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/st.cmd
    - $CIDER_REPO/components/xrf1/$MOTION_VARIATION/auto_settings.req:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/auto_settings.req
  command:
    [
      "bash",
      "-c",
      "/opt/epics/start_iocs/start_xrf1.sh; /bin/bash"
    ]
```

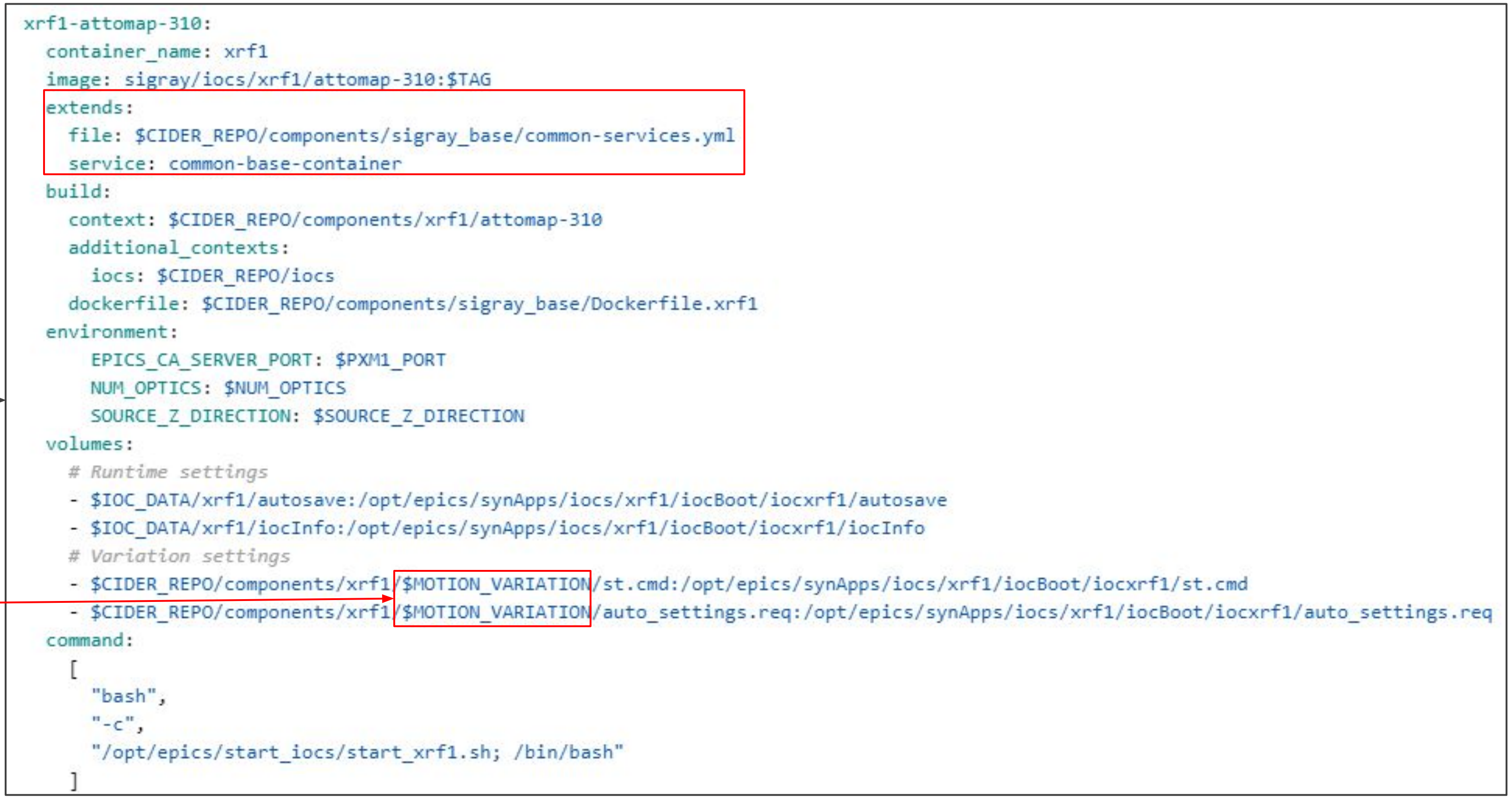


```
xrf1:
  container_name: xrf1
  extends:
    file: ../components/sigray_base/compose.yml
    service: xrf1-attomap-310
```

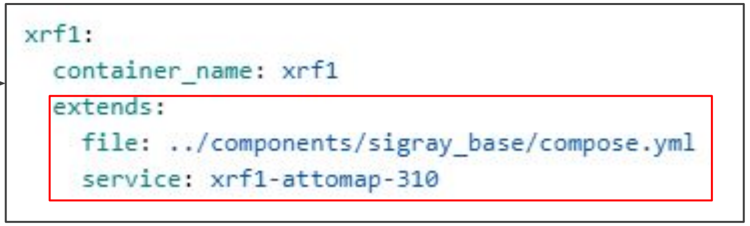
CIDER

File Structure Example

- components/
  - component_a/
    - common-services.yml
    - compose.yml
    - Dockerfile
    - common/
    - variation_1/
    - …
- ioc_submodules/
- machine_compose_files/
  - compose.am-1124.yml
- environment_files/
  - common_env_vars
  - am-1124.config



```
xrf1-attomap-310:
  container_name: xrf1
  image: sigray/iocs/xrf1/attomap-310:$TAG
  extends:
    file: $CIDER_REPO/components/sigray_base/common-services.yml
    service: common-base-container
  build:
    context: $CIDER_REPO/components/xrf1/attomap-310
    additional_contexts:
      iocs: $CIDER_REPO/iocs
    dockerfile: $CIDER_REPO/components/sigray_base/Dockerfile.xrf1
  environment:
      EPICS_CA_SERVER_PORT: $PXM1_PORT
      NUM_OPTICS: $NUM_OPTICS
      SOURCE_Z_DIRECTION: $SOURCE_Z_DIRECTION
  volumes:
    # Runtime settings
    - $IOC_DATA/xrf1/autosave:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/autosave
    - $IOC_DATA/xrf1/iocInfo:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/iocInfo
    # Variation settings
    - $CIDER_REPO/components/xrf1/$MOTION_VARIATION/st.cmd:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/st.cmd
    - $CIDER_REPO/components/xrf1/$MOTION_VARIATION/auto_settings.req:/opt/epics/synApps/iocs/xrf1/iocBoot/iocxrf1/auto_settings.req
  command:
    [
      "bash",
      "-c",
      "/opt/epics/start_iocs/start_xrf1.sh; /bin/bash"
    ]
```



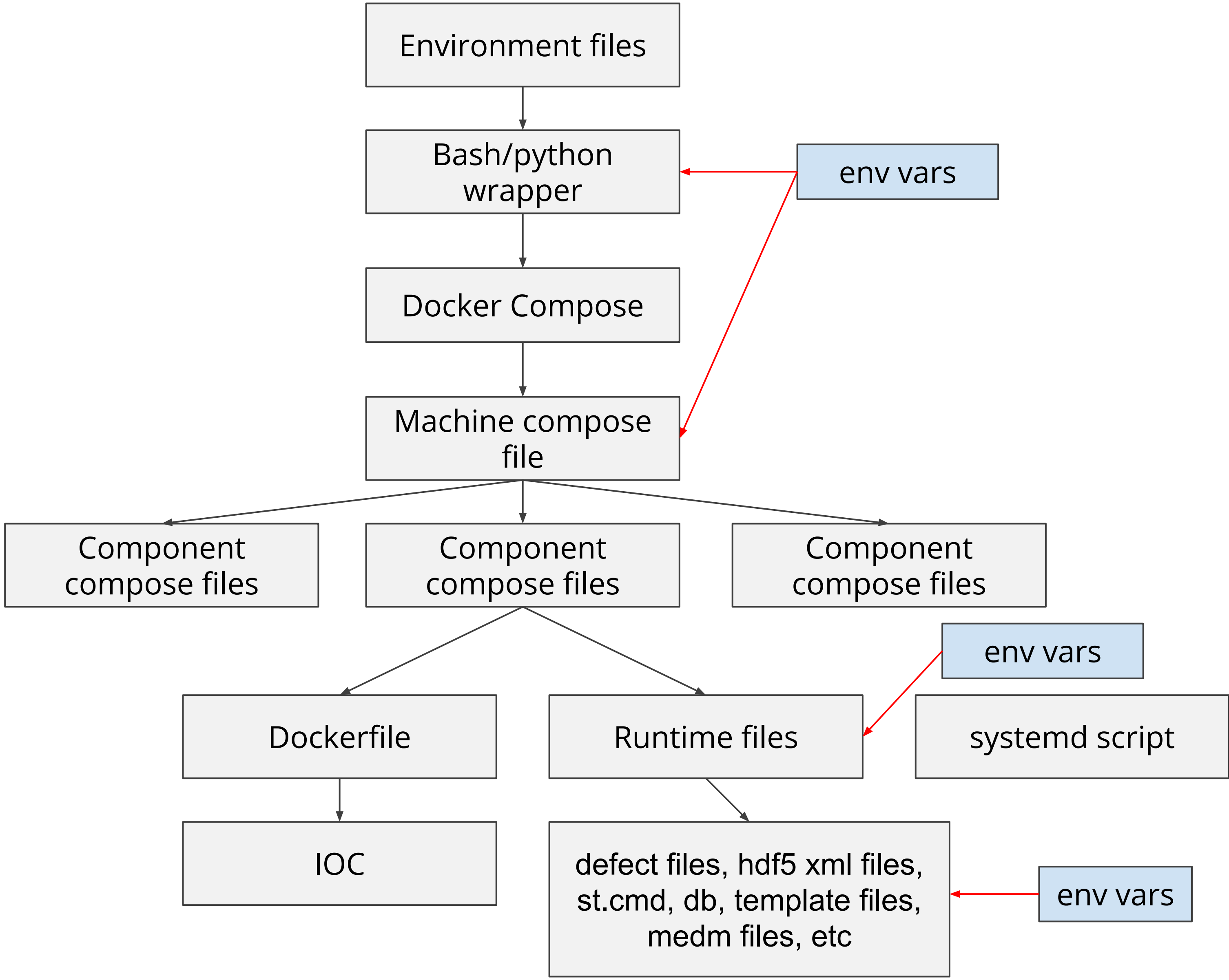
```
xrf1:
  container_name: xrf1
  extends:
    file: ../components/sigray_base/compose.yml
    service: xrf1-attomap-310
```

CIDER

Machine Callgraph

The machine is configured using:

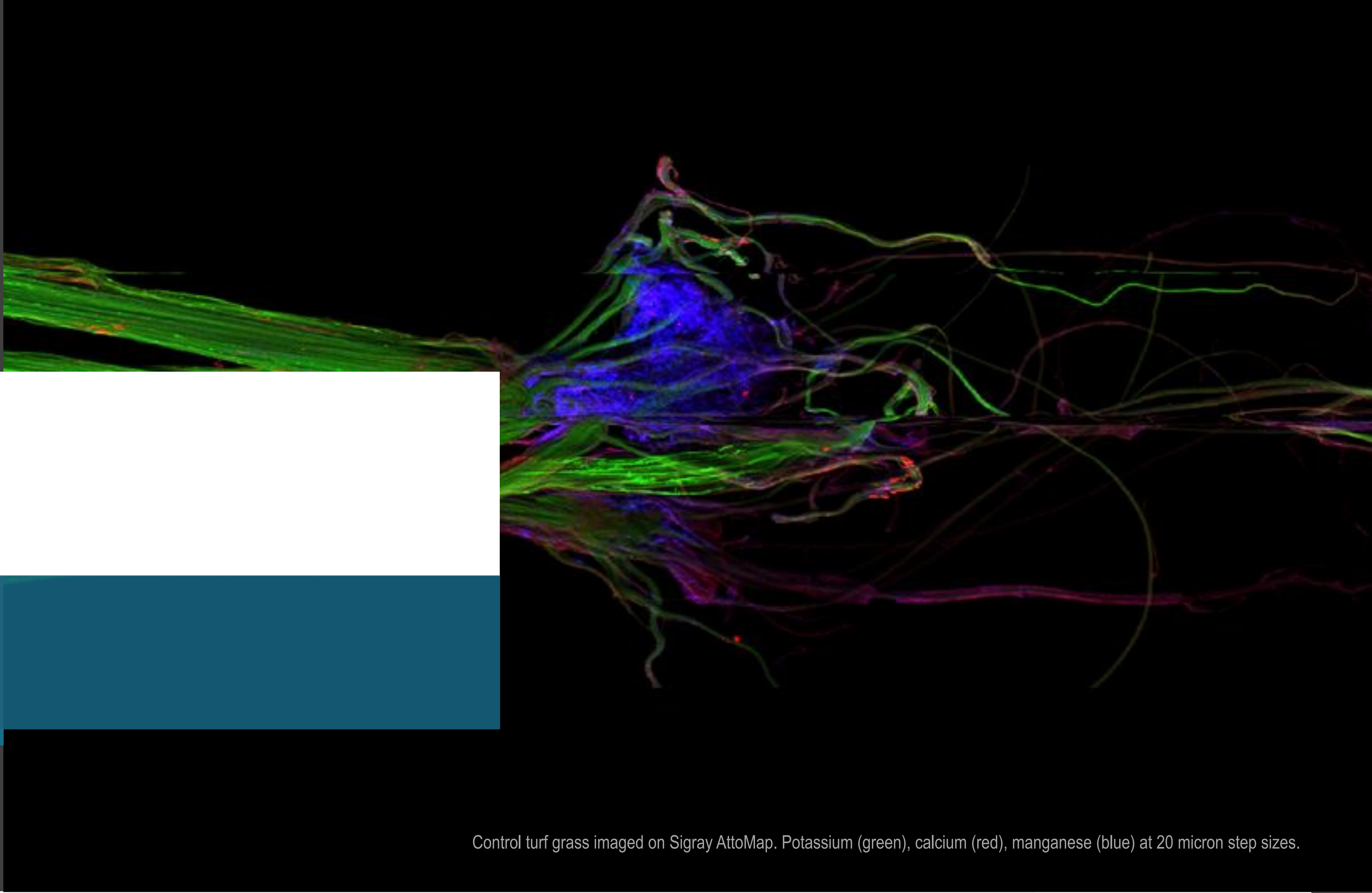
- Environment files dictate:
    - Machine compose file
    - What paths are mounted
    - IOC settings (e.g. sequencer, st.cmd, .db. etc)
- A machine compose file to dictate what IOC dockers it uses
    - Simply extends component definitions to reduce code duplication
- Docker bind mounts to overwrite specific IOC files



SIGRAY

Usage

How we use CIDER



Control turf grass imaged on Sigray AttoMap. Potassium (green), calcium (red), manganese (blue) at 20 micron step sizes.

Usage

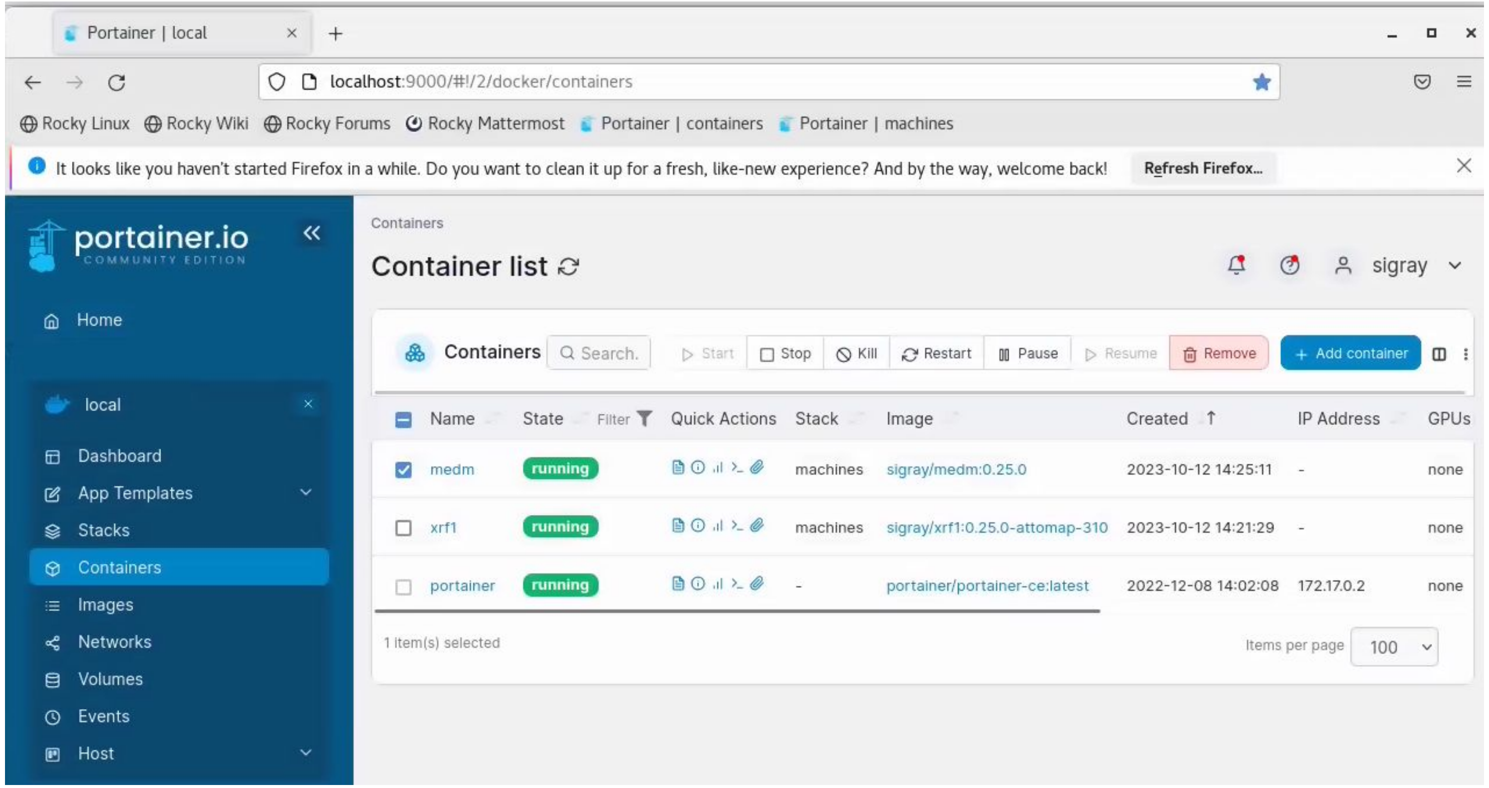
Machine setup

- Set up new machines
  - Set up udev rules (for serial-specific devices) and network settings
  - Create configuration files
  - Build and run docker services
  - Save settings
    - Autosave files
    - OS version
    - udev rules
    - Network configuration
- Total setup time: 30-60 minutes (including build time)
  - Assuming there aren’t any new devices or issues
  - Not including Jenkins and PRs

Usage

Tools - Portainer

- Use Portainer to restart the dockers/IOCs with updates



It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

Container list

Containers

	Name	State	Quick Actions	Stack	Image	Created	IP Address	GPUs
☑	medm	running		machines	sigray/medm:0.25.0	2023-10-12 14:25:11	-	none
□	xrf1	running		machines	sigray/xrf1:0.25.0-attomap-310	2023-10-12 14:21:29	-	none
□	portainer	running		-	portainer/portainer-ce:latest	2022-12-08 14:02:08	172.17.0.2	none

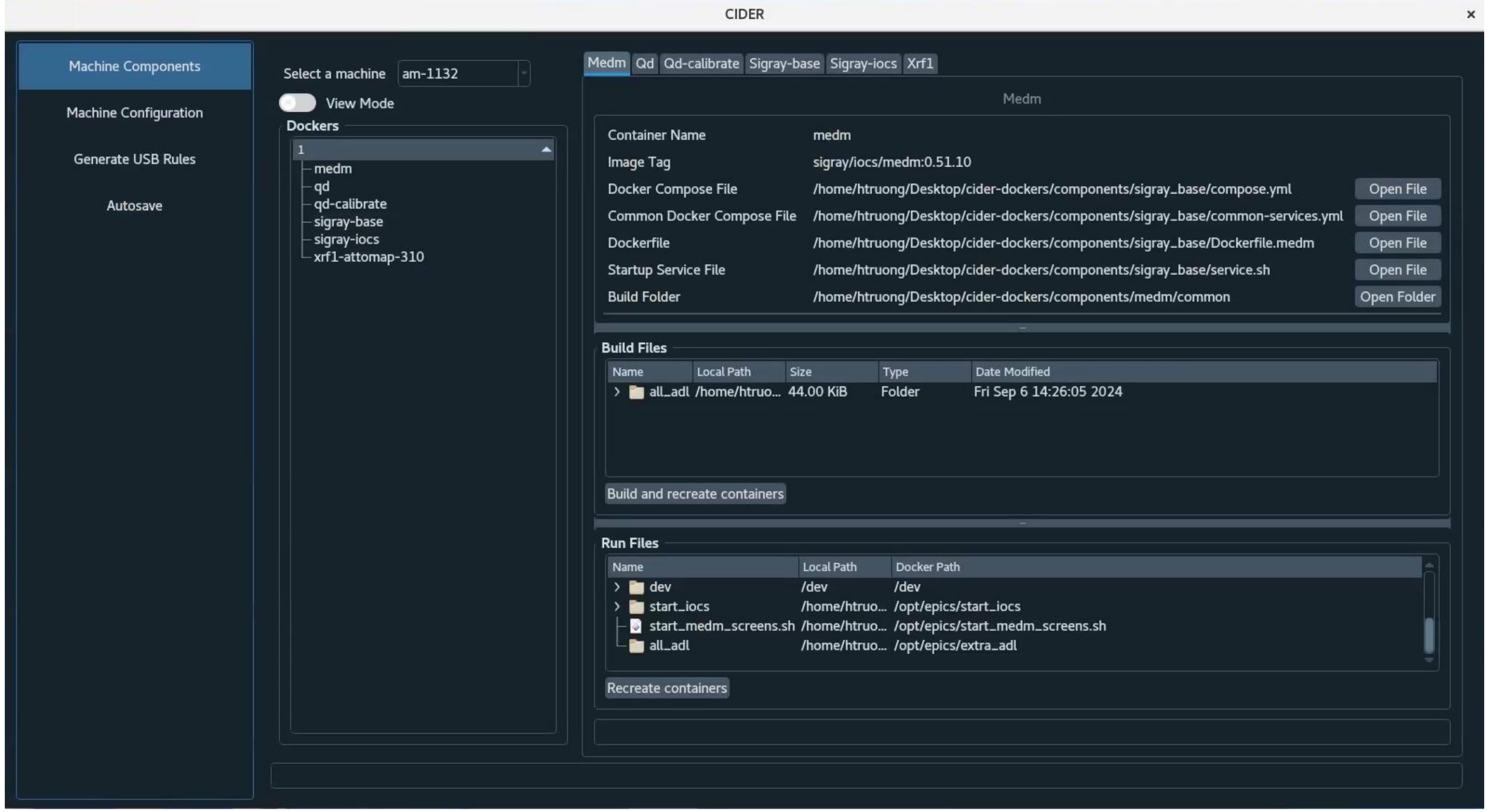
1 item(s) selected

Items per page 100

Usage

Tools - GUI

- Easy access to files for system engineers
- Rebuilding dockers and recreating containers
  - Portainer does not quite have this feature



Select a machine am-1132

Dockers

1
- medm
- qd
- qd-calibrate
- sigray-base
- sigray-iocs
- xrf1-attomap-310

Medm

Container Name	medm
Image Tag	sigray/iocs/medm:0.51.10
Docker Compose File	/home/htruong/Desktop/cider-dockers/components/sigray_base/compose.yml
Common Docker Compose File	/home/htruong/Desktop/cider-dockers/components/sigray_base/common-services.yml
Dockerfile	/home/htruong/Desktop/cider-dockers/components/sigray_base/Dockerfile.medm
Startup Service File	/home/htruong/Desktop/cider-dockers/components/sigray_base/service.sh
Build Folder	/home/htruong/Desktop/cider-dockers/components/medm/common

Build Files

Name	Local Path	Size	Type	Date Modified
all_adl	/home/htruo...	44.00 KiB	Folder	Fri Sep 6 14:26:05 2024

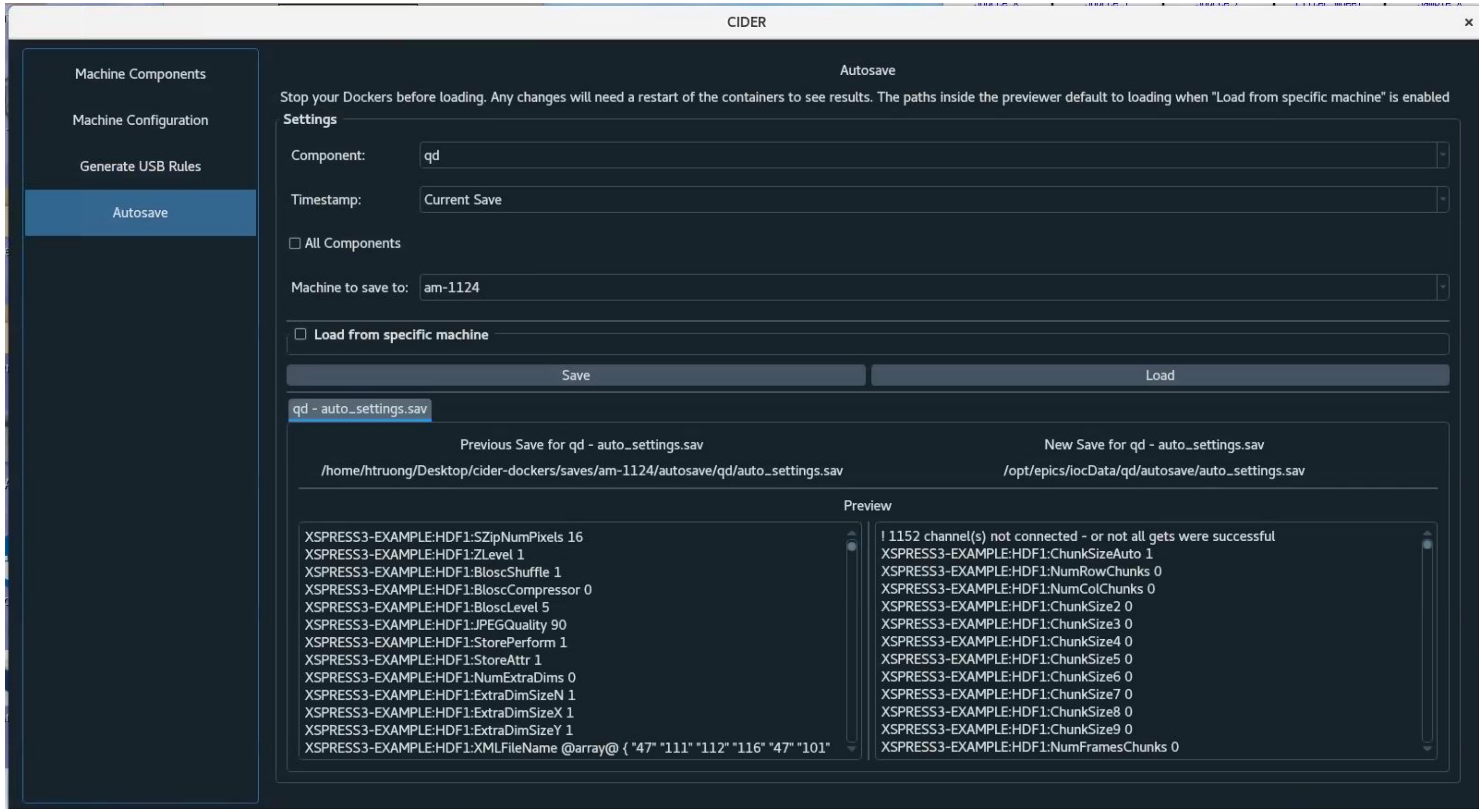
Run Files

Name	Local Path	Docker Path
dev	/dev	/dev
start_iocs	/home/htruo...	/opt/epics/start_iocs
start_medm_screens.sh	/home/htruo...	/opt/epics/start_medm_screens.sh
all_adl	/home/htruo...	/opt/epics/extra_adl

Usage

Tools - GUI

- Load and save autosave files
    - Doing this manually can be quite tedious if there is more than one autosave file
    - This is confusing for non-EPICS users to do manually



Autosave

Stop your Dockers before loading. Any changes will need a restart of the containers to see results. The paths inside the previewer default to loading when "Load from specific machine" is enabled

Settings

Component: qd

Timestamp: Current Save

□ All Components

Machine to save to: am-1124

□ **Load from specific machine**

Previous Save for qd - auto_settings.sav
/home/htruong/Desktop/cider-dockers/saves/am-1124/autosave/qd/auto_settings.sav

New Save for qd - auto_settings.sav
/opt/epics/iocData/qd/autosave/auto_settings.sav

Preview

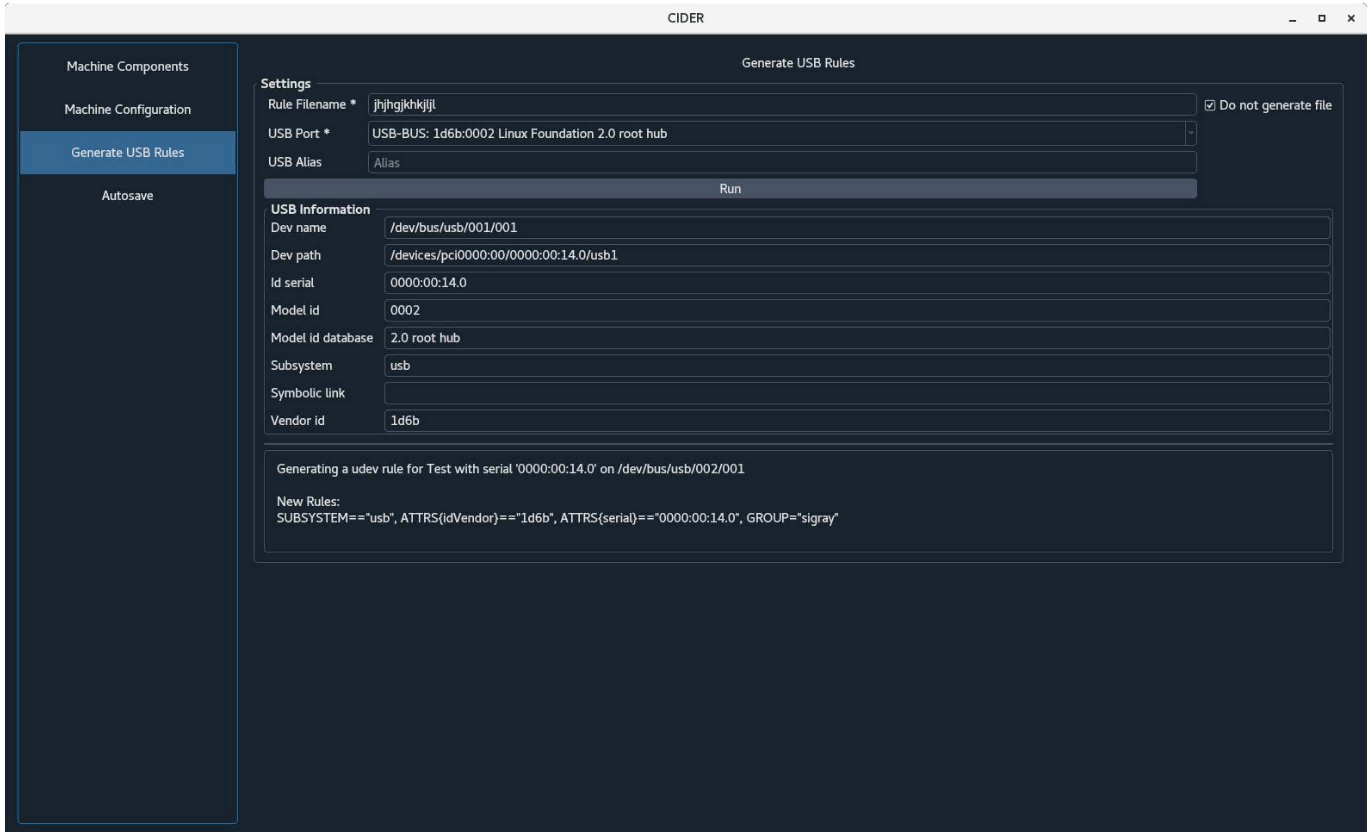
```
XSPRESS3-EXAMPLE:HDF1:SZipNumPixels 16
XSPRESS3-EXAMPLE:HDF1:ZLevel 1
XSPRESS3-EXAMPLE:HDF1:BloscShuffle 1
XSPRESS3-EXAMPLE:HDF1:BloscCompressor 0
XSPRESS3-EXAMPLE:HDF1:BloscLevel 5
XSPRESS3-EXAMPLE:HDF1:JPEGQuality 90
XSPRESS3-EXAMPLE:HDF1:StorePerform 1
XSPRESS3-EXAMPLE:HDF1:StoreAttr 1
XSPRESS3-EXAMPLE:HDF1:NumExtraDims 0
XSPRESS3-EXAMPLE:HDF1:ExtraDimSizeN 1
XSPRESS3-EXAMPLE:HDF1:ExtraDimSizeX 1
XSPRESS3-EXAMPLE:HDF1:ExtraDimSizeY 1
XSPRESS3-EXAMPLE:HDF1:XMLFileName @array@ { "47" "111" "112" "116" "47" "101"
```

```
! 1152 channel(s) not connected - or not all gets were successful
XSPRESS3-EXAMPLE:HDF1:ChunkSizeAuto 1
XSPRESS3-EXAMPLE:HDF1:NumRowChunks 0
XSPRESS3-EXAMPLE:HDF1:NumColChunks 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize2 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize3 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize4 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize5 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize6 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize7 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize8 0
XSPRESS3-EXAMPLE:HDF1:ChunkSize9 0
XSPRESS3-EXAMPLE:HDF1:NumFramesChunks 0
```

Usage

Tools - GUI

- Other system setup tasks, like generating udev rules for USB devices
    - Very easy to mess up with a typo and very hard to debug



Settings

Rule Filename * jhjhgjkhkjljl ☑ Do not generate file

USB Port * USB-BUS: 1d6b:0002 Linux Foundation 2.0 root hub

USB Alias Alias

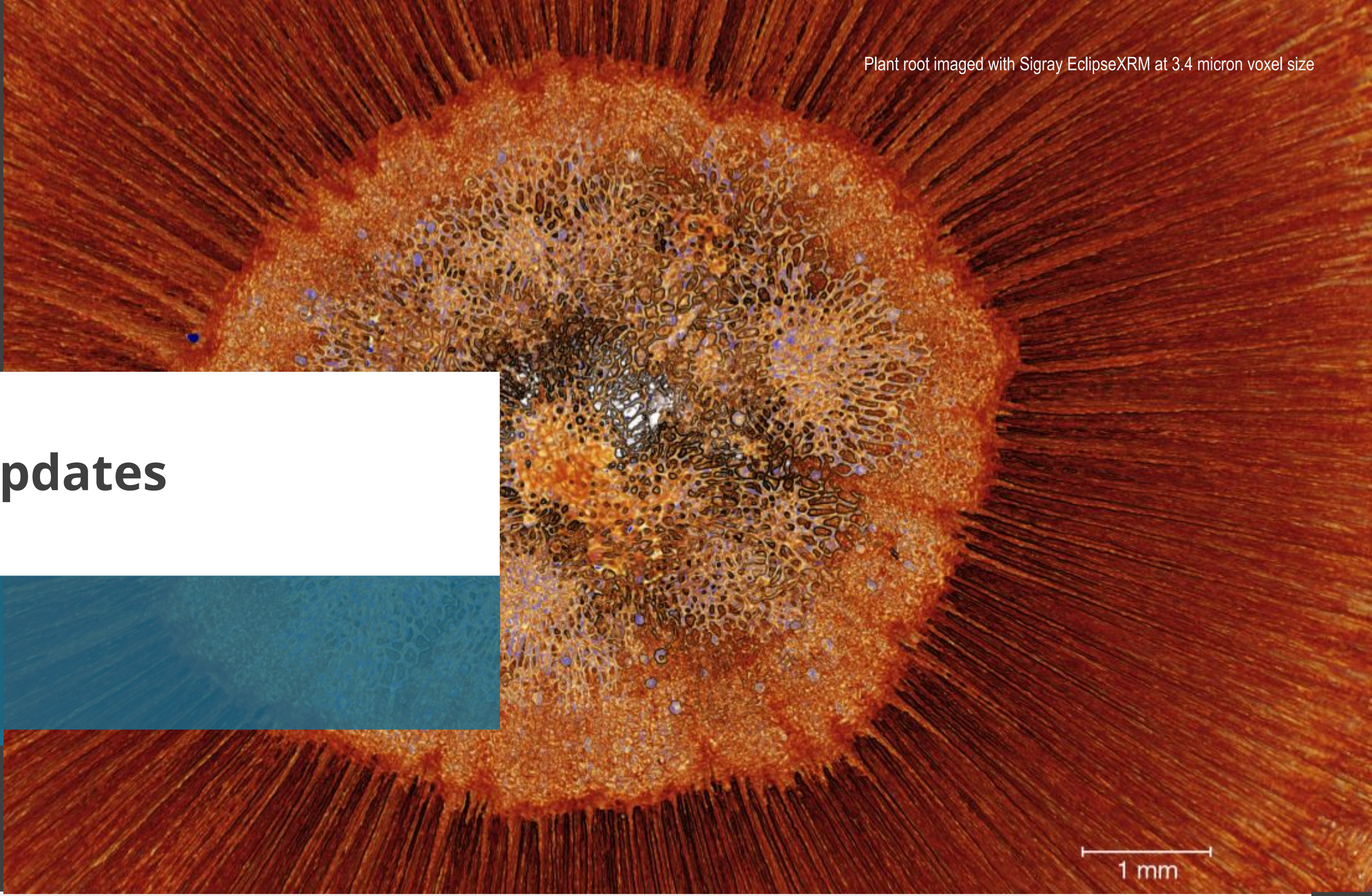
USB Information

Dev name	/dev/bus/usb/001/001
Dev path	/devices/pci0000:00/0000:00:14.0/usb1
Id serial	0000:00:14.0
Model id	0002
Model id database	2.0 root hub
Subsystem	usb
Symbolic link	
Vendor id	1d6b

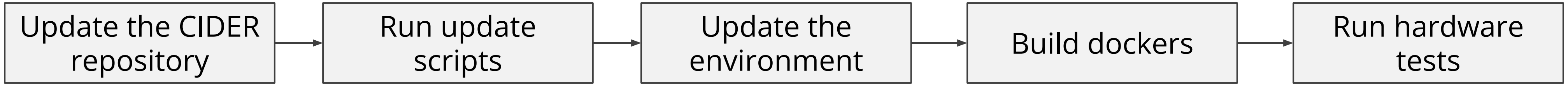
```
Generating a udev rule for Test with serial '0000:00:14.0' on /dev/bus/usb/002/001

New Rules:
SUBSYSTEM=="usb", ATTRS{idVendor}=="1d6b", ATTRS{serial}=="0000:00:14.0", GROUP="sigray"
```

Deploying Updates


Plant root imaged with Sigray EclipseXRM at 3.4 micron voxel size

Online Updates



Goal: Offline Updates

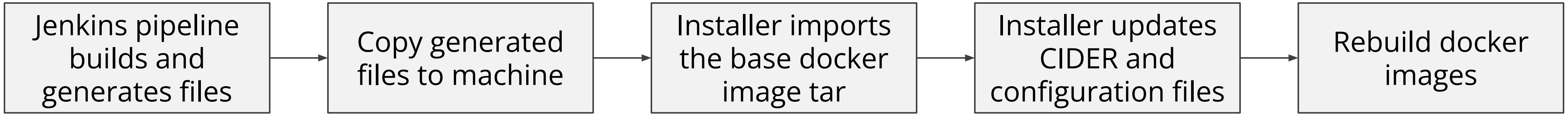
Customer Restrictions

- No internet access allowed
  - Cannot rebuild docker images that access internet (e.g. assemble_synapps)
- File size restrictions
  - Cannot export 7+ docker images that are 2-3 GB each
  - Cannot put all contents into one giant docker image

Goal: Offline Updates

Expected Procedure

- Files
    - Base docker image
        - Contains all shared online dependencies like synApps, Linux packages, etc
    - CIDER
        - Contains environment files, configuration files, and helper scripts
    - GUI folder
        - Generated by pyinstaller, making python dependencies portable
- Procedure



Jenkins pipeline builds and generates files → Copy generated files to machine → Installer imports the base docker image tar → Installer updates CIDER and configuration files → Rebuild docker images

Potassium distribution in flowers acquired on Sigray AttoMap

Takeaways

This includes CIDER, EPICS, and non-EPICS takeaways.

Takeaways

The Bad

- CIDER is difficult to develop with
    - The file structure can be confusing, and developers need to know what files are in the IOC submodule and what files are overwritten by CIDER
    - Having a branch per machine can lead to more toil if changes are frequently made after PRs are merged
    - Perhaps a file structure refactor with local config files and a version controlled configuration repository will improve this
- Having one environment file makes it unclear what variables correspond to what docker service
    - This makes it hard to know what environment variables to define or remove when adding/removing docker services
    - Hopefully the addition of **multiple environment files** in docker compose or just plain old Python to generate the environment files and machine compose files
- Storing configuration files in the same repository is not sustainable
    - Repository size will keep growing as we build more systems
    - Plan to move them into their own repository or make them local

Takeaways

The Bad

- Autosave can be unreliable
  - Turning off power to devices can lead to some PVs being wiped in autosave
  - Some PVs will randomly reset when the IOC or computer restarts
- MEDM is not very dynamic, and it can be hard to make modifications
- EPICS can be confusing to develop with (requires training) and debug
- Some of these issues may totally be due to my insufficient knowledge of EPICS!

Takeaways

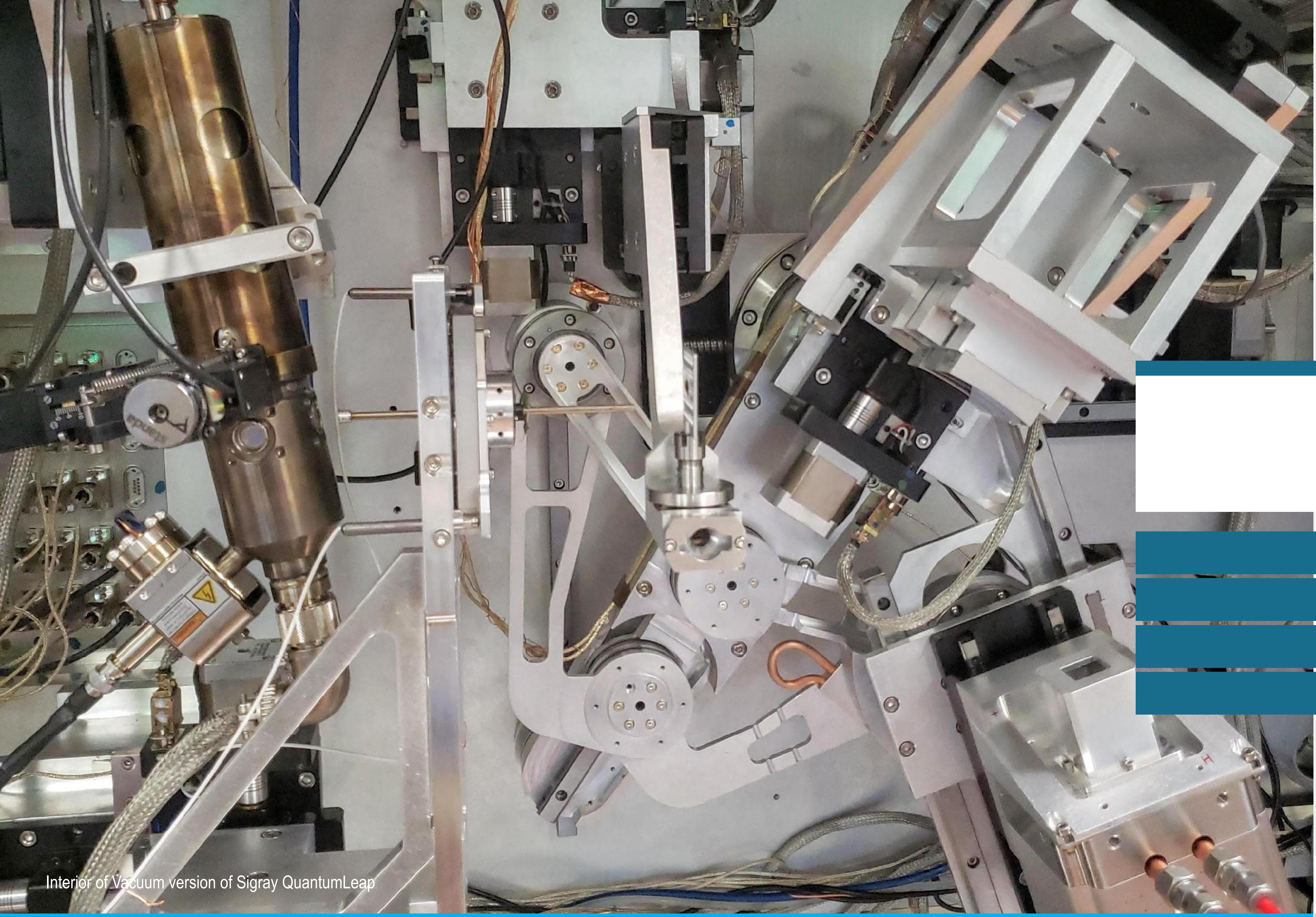
The Good

- Reuse an IOC repository as much as possible to make it easier to propagate changes to all machines
  - Achieved by moving as many build-time settings to run-time as possible to avoid requiring recompilation (and thus a new docker image)
    - Use environment variables in sequencer, st.cmd, .db
  - This also makes testing faster and easier, since we can heavily test the base IOC and sprinkle tests for the runtime settings
- Migration scripts have helped with ensuring machine settings are not lost during updates, even if the configuration files have changed structurally
- Keeping configuration files in version control is great!
  - Track why changes were made, rollback to an older version, diff files, etc
- ADAravis is great for creating new IOCs for GenICam-compatible detectors!
  - Doesn’t always work right off the bat

Takeaways

The Good

- Docker and docker compose has some helpful features that allows us to have a simpler file structure
    - **multiple build contexts**
        - Can organize things in terms of logical categories rather than in terms of docker images
        - Avoids duplication of files that belong in multiple docker images
    - The docker compose file itself can use environment variables
        - Useful for storing all env vars in one file and propagating them
    - (Future) The **include** top-level element
        - Allow us to define dependencies without:
            - Duplicating docker service definitions
            - Defining service definitions in the same docker compose file



Interior of Vacuum version of Sigray QuantumLeap

Thank you!