PSI Center for Scientific Computing,
Theory and Data

AWI Department Meeting

Overview of Group 7902

Markus Janousch
PSI, 15 October 2024

Content

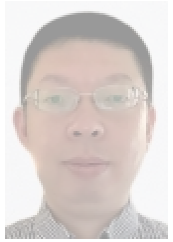
- Overview (Markus)
- TOMCAT new processing pipeline (Alain)
- Jira future (Alain)
- BEC deployment (Ivan)
- Indexer (Hans-Christian is sick)

DataProcessing and Development Group (7902)

Alain Studer

Hans-Christian Stadler Kleeb

Ivan Usov



Jun Zhu

Markus Janousch

N.N. (SDATE)

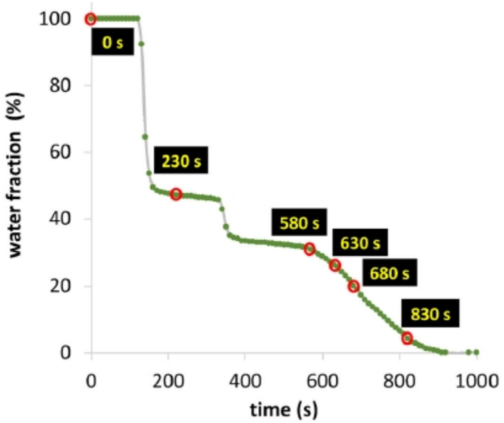
SDATE – Smart Data Acquisition for Tomoscopy Experiments*

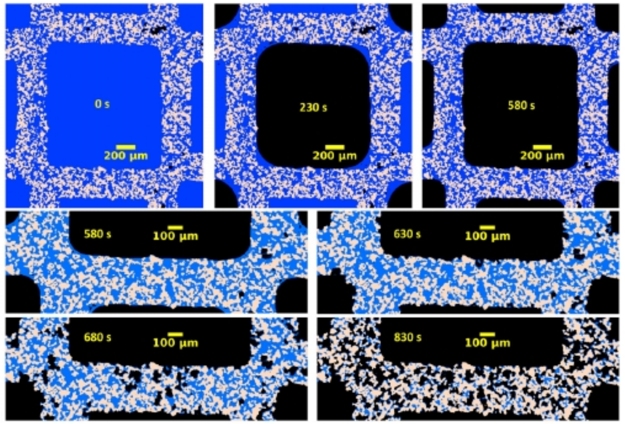
Call from SDSC in 2023 - Data Science Projects for Large Infrastructures

Time resolved tomography (Tomoscopy) at TOMCAT will produce 10 – 100 TB of data each day after the SLS 2.0 upgrade.

Compression of images is very inefficient (only factor 2 - 4).

Aim for data **reduction**.

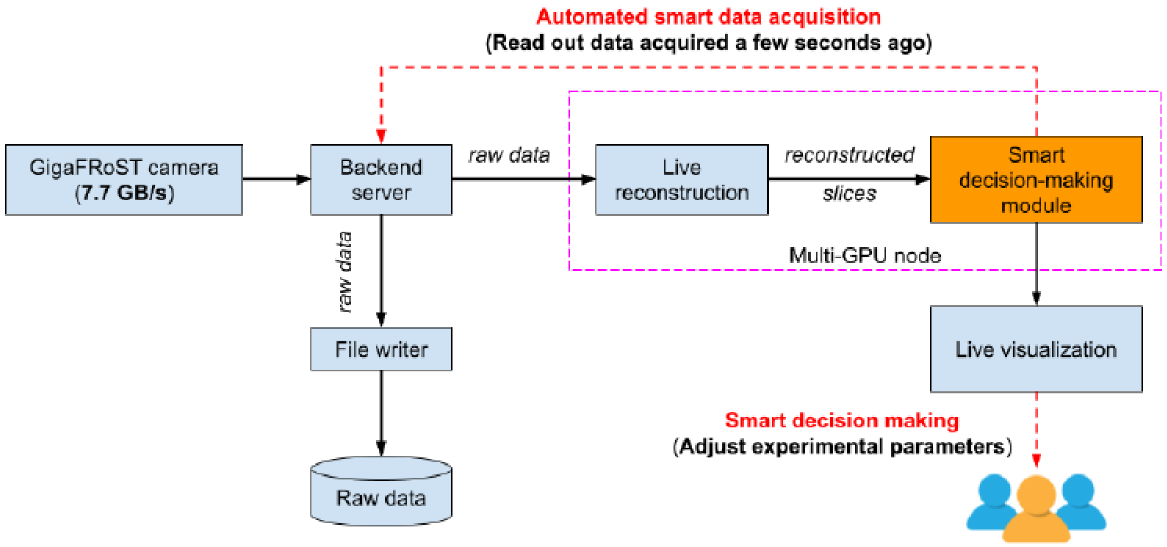




Novák, V., Blažek, M., Schlepütz, C. M., Kočí, P. & Stampanoni, M. Drying of water from porous structures investigated by time-resolved X-ray tomography. Drying Technology 0, 1–19 (2022).

* Jun Zhu, Markus Janousch, Christian Schlepütz, Goran Lovric, Leonardo Hax Damiani (AWI, CPS, CAS)

- Achieve reduction via AI. Train ML-algorithms with existing data for “changepoint detection”.
- Reduce data taking rate when in steady state
- Increase around a “changepoint”
- Change of acquisition rate either through a closed feedback loop or as a hint to the experimenter.


Automated smart data acquisition
(Read out data acquired a few seconds ago)
GigaFRoST camera (7.7 GB/s)
Backend server
raw data
Live reconstruction
reconstructed slices
Smart decision-making module
Multi-GPU node
raw data
File writer
Raw data
Live visualization
Smart decision making
(Adjust experimental parameters)

Training

Basic Python training now available at PSI. One course from MIT course (free) and “ Learn Python the hard way” ($ 29.-).

Started a Knowledge Base on Confluence with a direct link to training pages.

Collect the documentation and training material that is available in AWI.

PSI Center for Scientific Computing,
Theory and Data

TOMCAT – Sirius Collaboration
and
Jira Cloud Migration

Alain Studer
15 October 2024

Part 1

- TOMCAT-SIRIUS Collaboration

Collaboration with LNLS

- LNLS: Brazilian Synchrotron (SIRIUS)
- Collaboration between TOMCAT and CNPEM
- Sirius provides a GPU implementation of the Tomography Reconstruction Pipeline (called RAFT)
- The TOMCAT reconstruction pipeline is CPU based

Status and Goal

- There is a test installation of RAFT on RA
- RAFT has adopted existing TOMCAT CLI
- TOMCAT can use CPU or GPU based reconstruction interchangeably
- First we need to integrate, benchmark, test etc..

Advantage of GPU Implementation

- Standard usage of Reconstruction Pipeline:

  All reconstruction parameters are know in advance

- For some use cases however, parameters must be iteratively adapted
- This is not ideal with actual implementation

Iterative Reconstruction Algorithm

- 1) User sets (initial) guess for specific parameter
- 2) Load raw data from disk to memory (~100GB)
- 3) Reconstruct, save reconstructions to disk (~100GB)
- 4) Load reconstructions from disk to memory for

  3D rendering (~100GB)
- 5) Based on visual inspection, users decide if results are ok
- 6) If not, goto 1)

Drawback of Current Algorithm

- Many data transfers from/to memory (from/to disk)
- The GPU based reconstruction can keep raw data in memory for subsequent reconstructions.
- Since visualization is best done on a GPU node, also this step should be more convenient.

Questions?

Part 2

- Jira Cloud Migration:Status & Outlook

Jira Cloud Migration

- Meeting last week (status update)
- Discussion soon changed from ‘Migration’ to ‘Jira’ (Is Jira the appropriate tool?)
- Main problem: License costs and group management (in combination)

Crucial Point

- CAS wants to continue working as is now
- All PSI/CPS should be Jira group members

  (Everybody should be able to assign tickets to CAS)
- This means that all PSI/CPS members must have a license (licenses are named)
- CPS is reluctant to pay for licenses.

Problem on CAS Side

- For cost reasons, the CAS/CPS Jira group must only comprise a (small) subset of CPS members
- CAS as admins must do the group management

  (Define, create, maintain the group members/permission)
- The CAS representative indicated that CAS is not willing to take this additional burden.

Problem on CPS Side

- If only a few people on PSI side have licenses

  what happens if somebody without a license wants to create a ticket?

- Either buy a additional license (cost + admin)….
- …or chose a representative within CPS which creates the ticket on behalf of the person having the problem

Proposed Jira Alternatives

- MS 365
- Service Now
- Git

Questions?

PSI Center for Scientific Computing, Theory and Data

Beamline and Experiment Control (BEC) Deployment

Ivan Usov

15 October 2024

In collaboration with

- Borys Sharapov
- Klaus Wakonig
- Leonardo Sala
- Simon Ebner

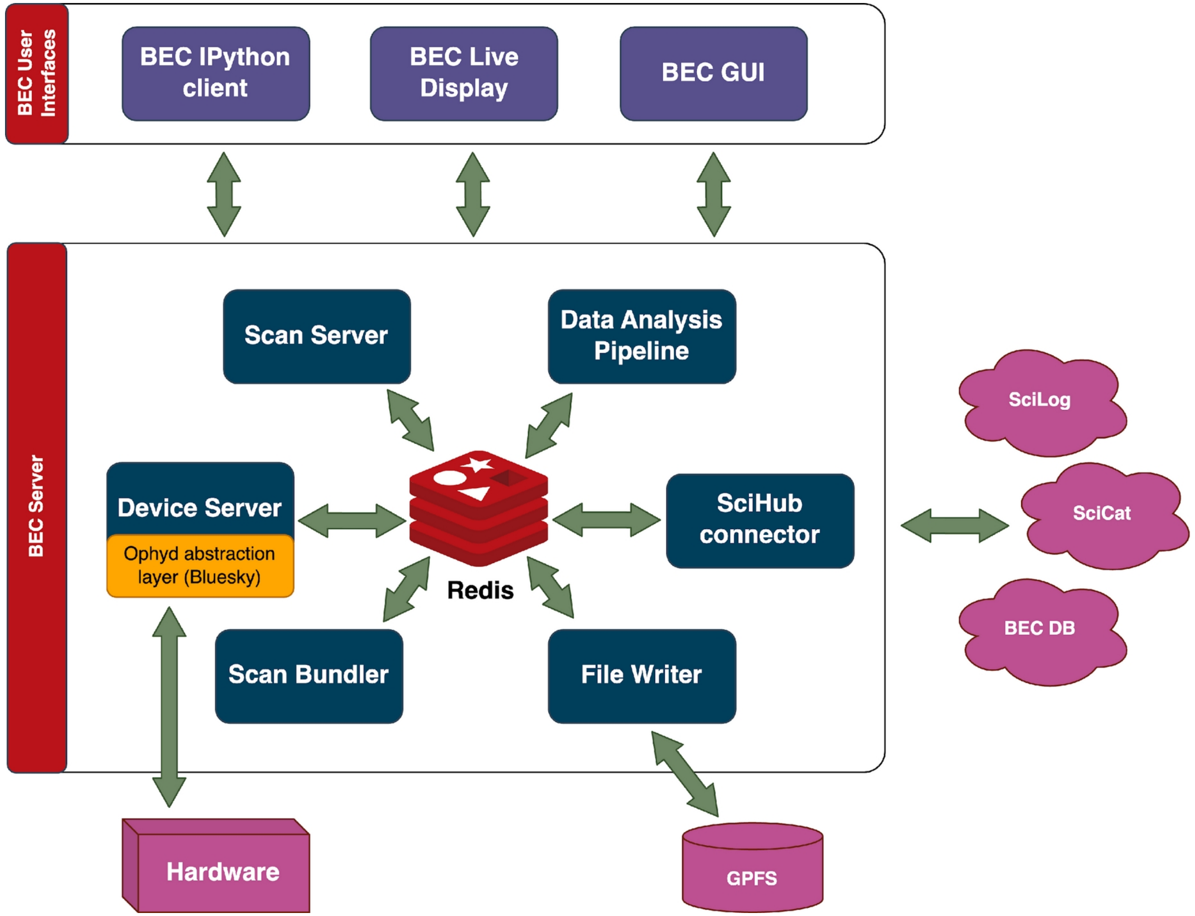
BEC

The Beamline and Experiment Control (BEC) is a new python-based control system for experiments that targets the Swiss Light Source upgrade (SLS 2.0)

- https://gitlab.psi.ch/bec
- https://bec.readthedocs.io/en/latest/

Components (per beamline):

1. redis
2. bec-server (device server, scan server, scan bunder, etc.)
3. bec clients (ipython, bec-widgets)



Motivation

We wanted the beamline staff to start using BEC with a minimal setup effort

- Run BEC independently on the current physical consoles/workstations at beamlines
- Avoid repetitive installation steps on each beamline

Ability to scale to dozens of BEC installations in the future, while having an overview of the components version migration and of operation (monitoring)

Approaches:

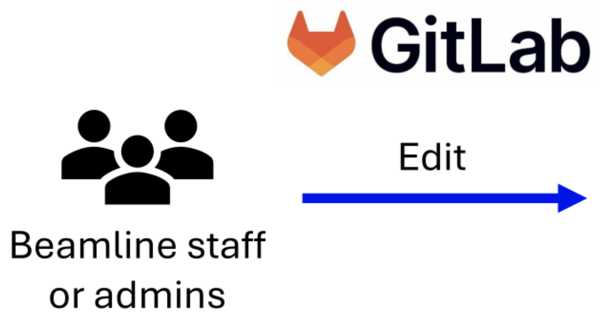
- Hardware virtualization (VMware) with an initial puppet setup
- BEC as a service, not as a software

BEC as a service setup (1)

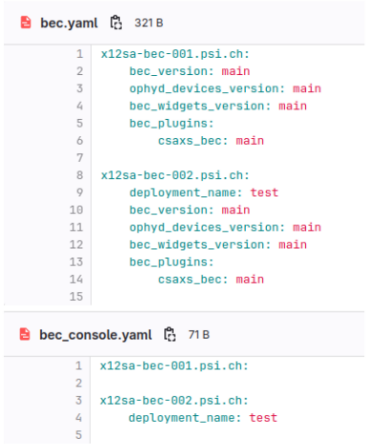
Beamline managers have a write access to their respective git repository with BEC deployment configuration files

BEC deployment configuration

- A simple user interface with declarative yaml files
- The configuration is defined on a per-host basis and specifies versions of BEC components and beamline plugins to be installed in each deployment



Configuration files with parameters for BEC deployment



bec.yaml 321 B

```
x12sa-bec-001.psi.ch:
    bec_version: main
    ophyd_devices_version: main
    bec_widgets_version: main
    bec_plugins:
        csaxs_bec: main

x12sa-bec-002.psi.ch:
    deployment_name: test
    bec_version: main
    ophyd_devices_version: main
    bec_widgets_version: main
    bec_plugins:
        csaxs_bec: main

```

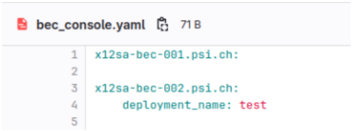
bec_console.yaml 71 B

```
x12sa-bec-001.psi.ch:

x12sa-bec-002.psi.ch:
    deployment_name: test

```

BEC as a service setup (2)

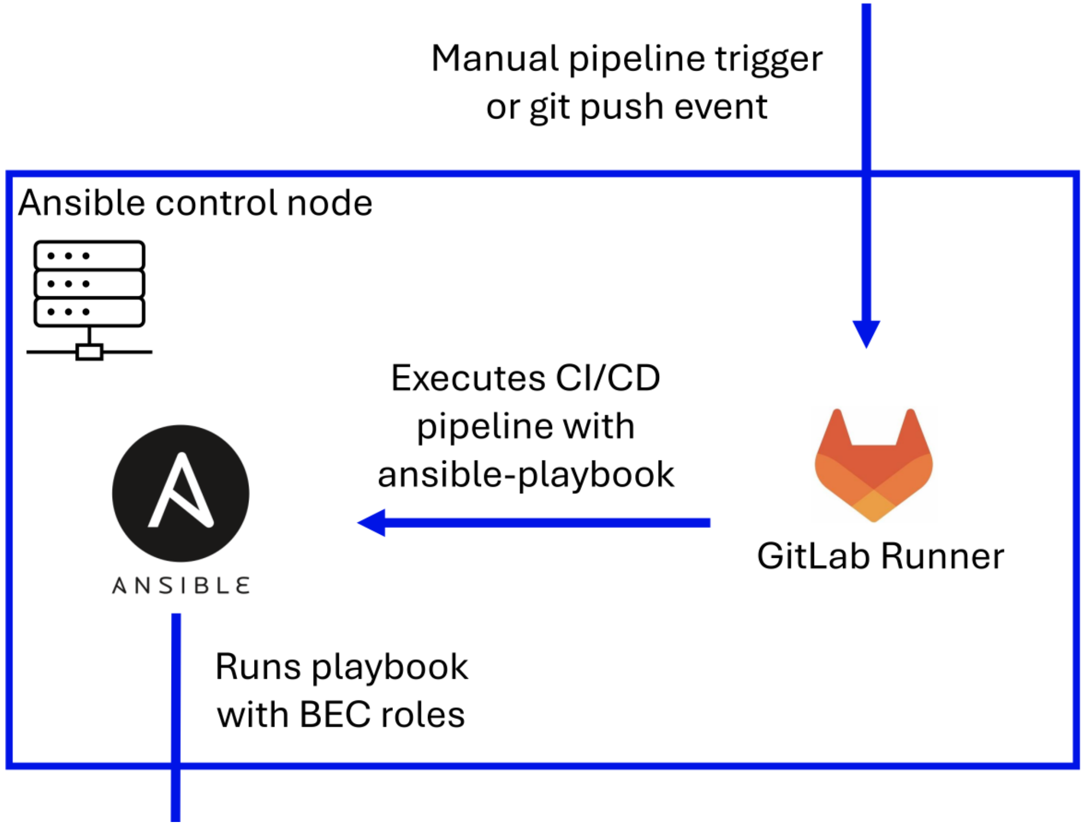
GitLab Runner and CI/CD pipelines

- GitLab Runner is installed and configured on an ansible control node
- Can be triggered by a git push event or from GitLab web interface
- GitLab Runner executes a CI/CD pipeline that, in-turn, runs an Ansible playbook



bec_console.yaml 71 B

```
x12sa-bec-001.psi.ch:

x12sa-bec-002.psi.ch:
    deployment_name: test

```



Manual pipeline trigger
or git push event

Ansible control node

Executes CI/CD
pipeline with
ansible-playbook

ANSIBLE

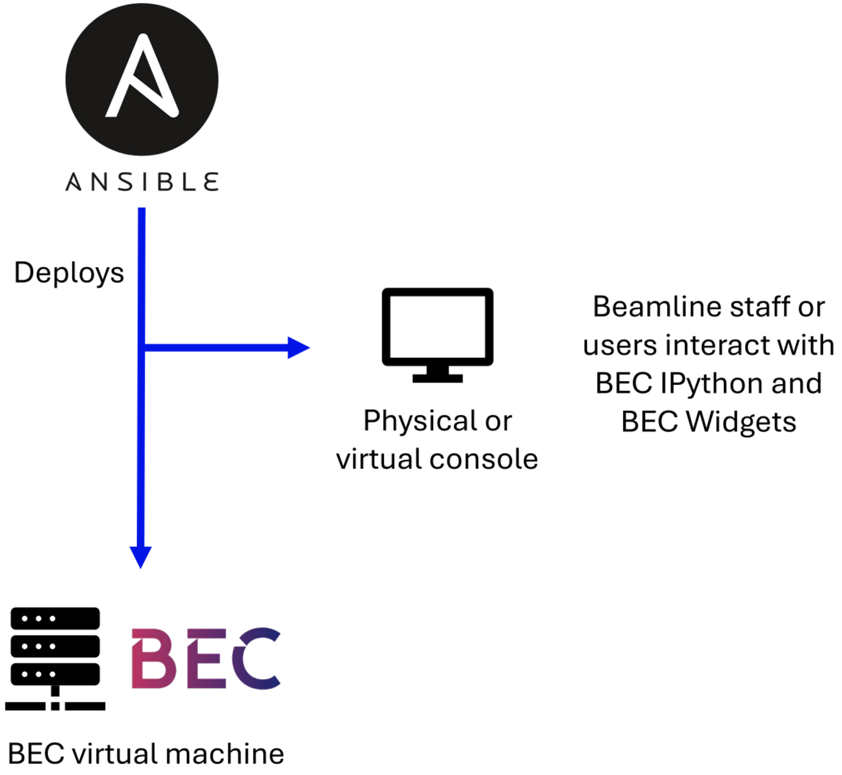
GitLab Runner

Runs playbook
with BEC roles

BEC as a service setup (3)

Execution of a playbook with imported **psi.bec** and **psi.bec_console** ansible roles

(A straightforward scaling to other service deployments by including additional ansible roles in the playbook of a specific beamline)



Deploys

Physical or virtual console

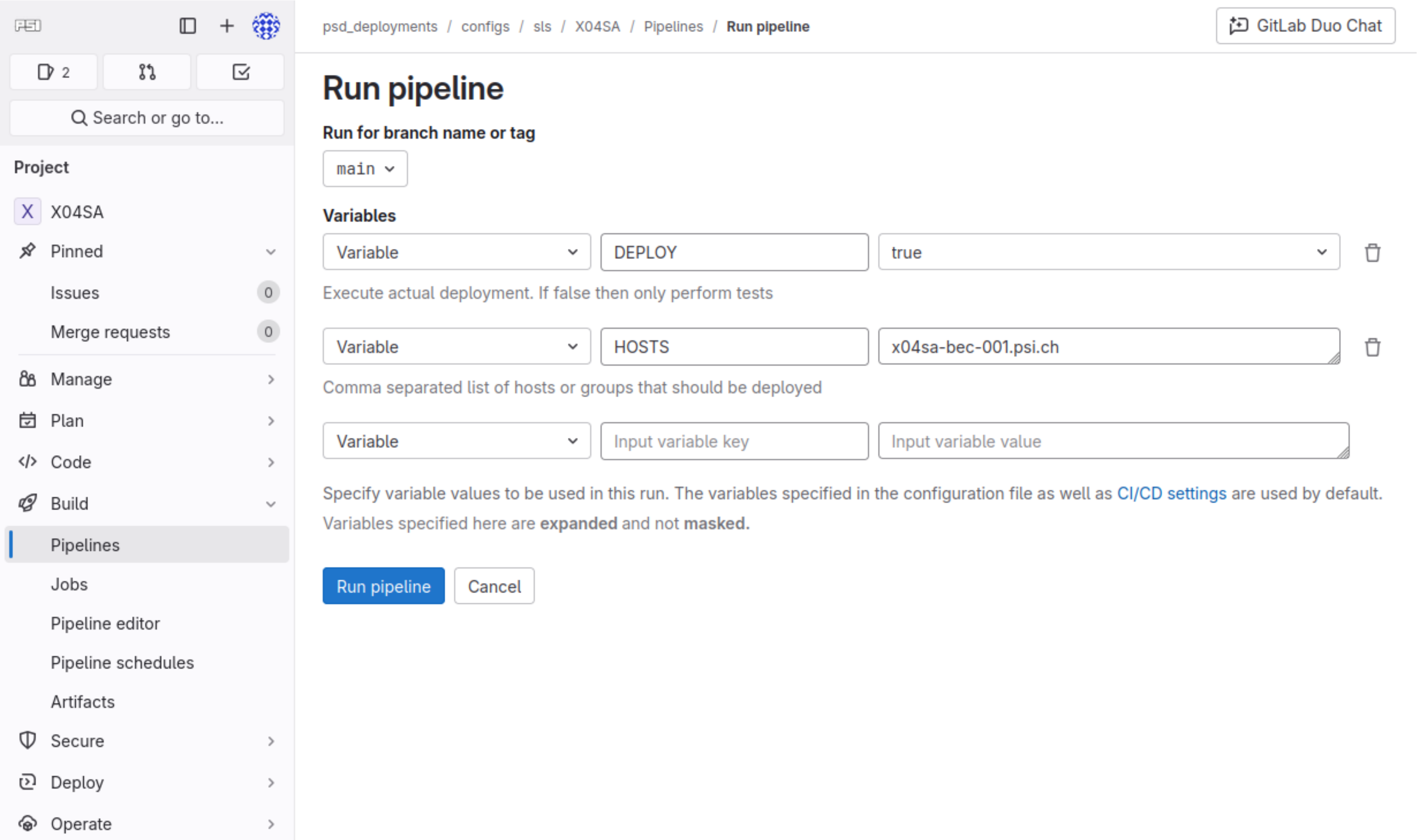
Beamline staff or users interact with BEC IPython and BEC Widgets

BEC virtual machine

BEC virtual machine provides:

- Source code of BEC components installed in editable/developer mode
- Beamline-specific plugins
- Corresponding python virtual environment
- Integration with remote services, like ElasticSearch and BEC DB (in progress)

Run deployment pipeline



psd_deployments / configs / sls / X04SA / Pipelines / **Run pipeline**

Run pipeline

Run for branch name or tag

main

Variables

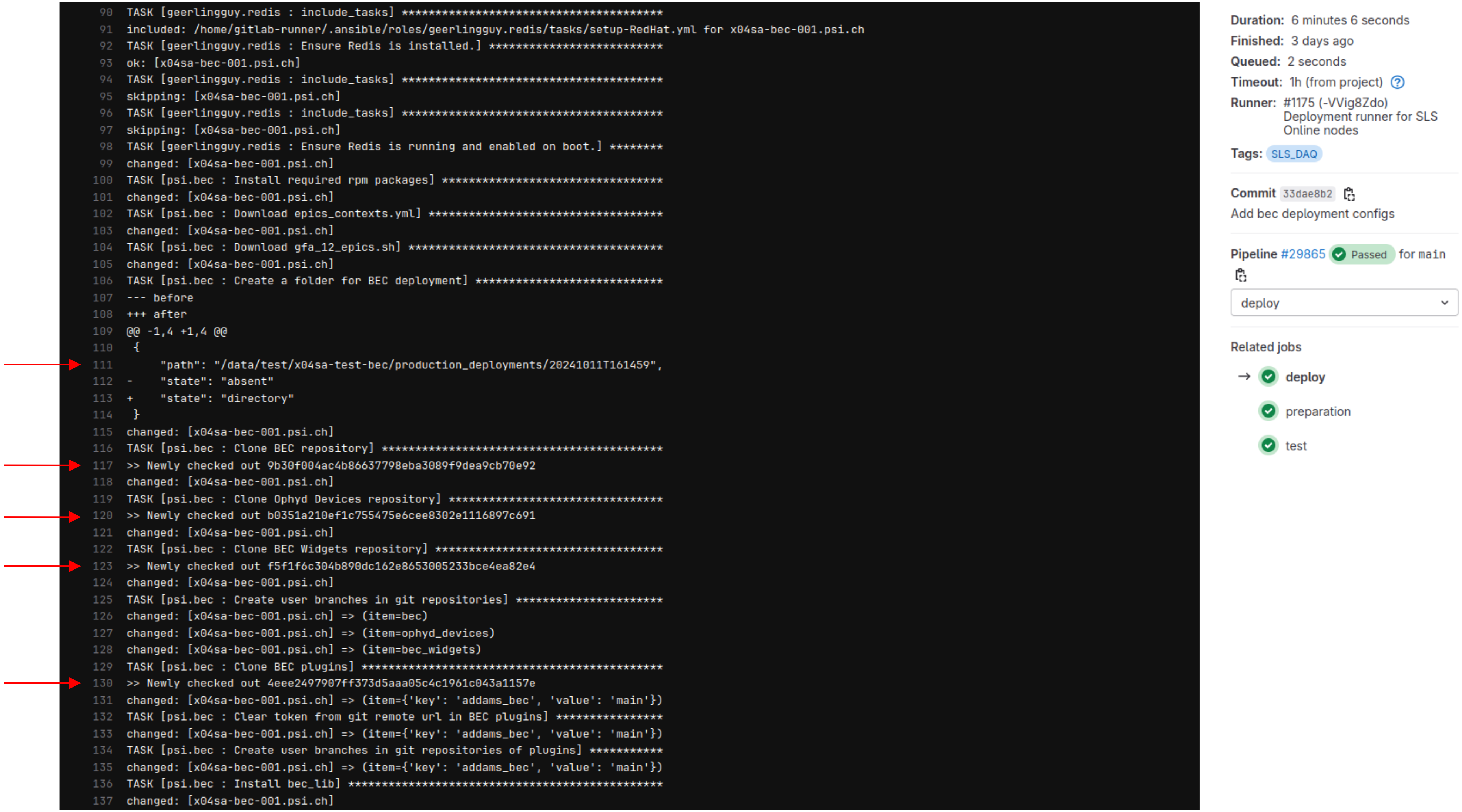
Variable	DEPLOY	true
Variable	HOSTS	x04sa-bec-001.psi.ch
Variable	Input variable key	Input variable value

Execute actual deployment. If false then only perform tests

Comma separated list of hosts or groups that should be deployed

Specify variable values to be used in this run. The variables specified in the configuration file as well as CI/CD settings are used by default. Variables specified here are **expanded** and not **masked.**

Self-documenting pipeline



```
TASK [geerlingguy.redis : include_tasks] ***************************************
included: /home/gitlab-runner/.ansible/roles/geerlingguy.redis/tasks/setup-RedHat.yml for x04sa-bec-001.psi.ch
TASK [geerlingguy.redis : Ensure Redis is installed.] **************************
ok: [x04sa-bec-001.psi.ch]
TASK [geerlingguy.redis : include_tasks] ***************************************
skipping: [x04sa-bec-001.psi.ch]
TASK [geerlingguy.redis : include_tasks] ***************************************
skipping: [x04sa-bec-001.psi.ch]
TASK [geerlingguy.redis : Ensure Redis is running and enabled on boot.] ********
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Install required rpm packages] *********************************
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Download epics_contexts.yml] ***********************************
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Download gfa_12_epics.sh] **************************************
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Create a folder for BEC deployment] ****************************
--- before
+++ after
@@ -1,4 +1,4 @@
 {
     "path": "/data/test/x04sa-test-bec/production_deployments/20241011T161459",
-    "state": "absent"
+    "state": "directory"
 }
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Clone BEC repository] ******************************************
>> Newly checked out 9b30f004ac4b86637798eba3089f9dea9cb70e92
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Clone Ophyd Devices repository] ********************************
>> Newly checked out b0351a210ef1c755475e6cee8302e1116897c691
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Clone BEC Widgets repository] **********************************
>> Newly checked out f5f1f6c304b890dc162e8653005233bce4ea82e4
changed: [x04sa-bec-001.psi.ch]
TASK [psi.bec : Create user branches in git repositories] **********************
changed: [x04sa-bec-001.psi.ch] => (item=bec)
changed: [x04sa-bec-001.psi.ch] => (item=ophyd_devices)
changed: [x04sa-bec-001.psi.ch] => (item=bec_widgets)
TASK [psi.bec : Clone BEC plugins] *********************************************
>> Newly checked out 4eee2497907ff373d5aaa05c4c1961c043a1157e
changed: [x04sa-bec-001.psi.ch] => (item={'key': 'addams_bec', 'value': 'main'})
TASK [psi.bec : Clear token from git remote url in BEC plugins] ****************
changed: [x04sa-bec-001.psi.ch] => (item={'key': 'addams_bec', 'value': 'main'})
TASK [psi.bec : Create user branches in git repositories of plugins] ***********
changed: [x04sa-bec-001.psi.ch] => (item={'key': 'addams_bec', 'value': 'main'})
TASK [psi.bec : Install bec_lib] ***********************************************
changed: [x04sa-bec-001.psi.ch]
```

Duration: 6 minutes 6 seconds
Finished: 3 days ago
Queued: 2 seconds
Timeout: 1h (from project)
Runner: #1175 (-VVig8Zdo)
Deployment runner for SLS
Online nodes
Tags: SLS_DAQ

Commit 33dae8b2
Add bec deployment configs

Pipeline #29865 Passed for main

deploy

Related jobs

→ ✓ **deploy**
✓ preparation
✓ test

deployment path with timestamp

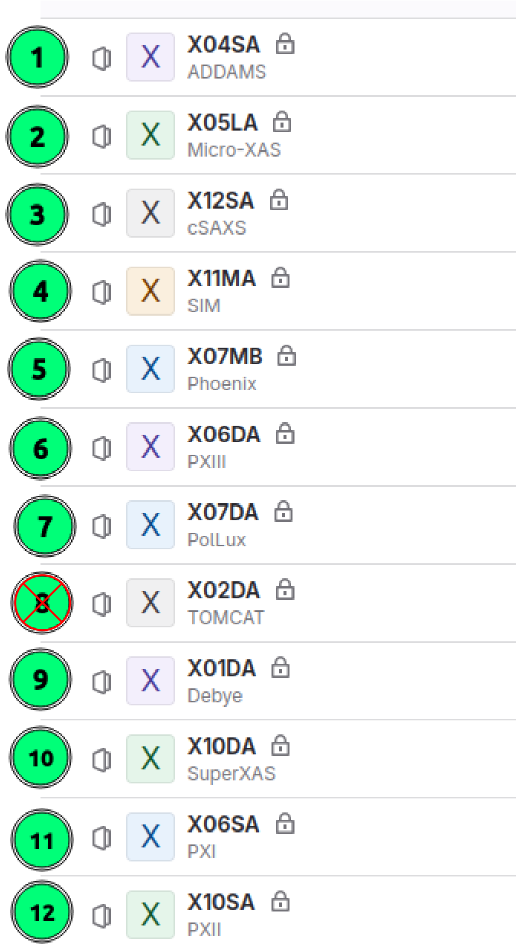
git refs of bec components

Status at SLS beamlines

- 11 beamlines with BEC deployed
  - X02DA - only std_daq tests

- 13 deployments
  - X05LA and X12SA - 2 BEC VMs each

- PSI-wide accessible Wiki for beamline managers: https://git.psi.ch/groups/psd_deployments/-/wikis/home



1 X X04SA ADDAMS
2 X X05LA Micro-XAS
3 X X12SA cSAXS
4 X X11MA SIM
5 X X07MB Phoenix
6 X X06DA PXIII
7 X X07DA PolLux
8 X X02DA TOMCAT
9 X X01DA Debye
10 X X10DA SuperXAS
11 X X06SA PXI
12 X X10SA PXII

Development VMs and non-SLS deployments

Development machines: **awi-bec-dev-[01:06].psi.ch** are used for development and testing of bec-related ansible roles

We can provide test BEC VMs for interested users also outside of SLS:

- **detector-group-bec-01.psi.ch**

It is deployed from one of the dev machines (as an ansible control node) with minor changes to the deployment structure

- https://gitlab.psi.ch/bec/ansible_bec

Monitoring

Icinga2

- Out-of-the-box via puppet with the adjusted criticality of checks (e.g. Disk Usage, Memory) and alerting severity (send email alerts 24x7)

ElasticSearch

- Setup and configuration via ansible (also use puppet?)
- Central collection of bec-server and client logs (for all beamlines?)
- Individual metrics of redis and bec-server processes

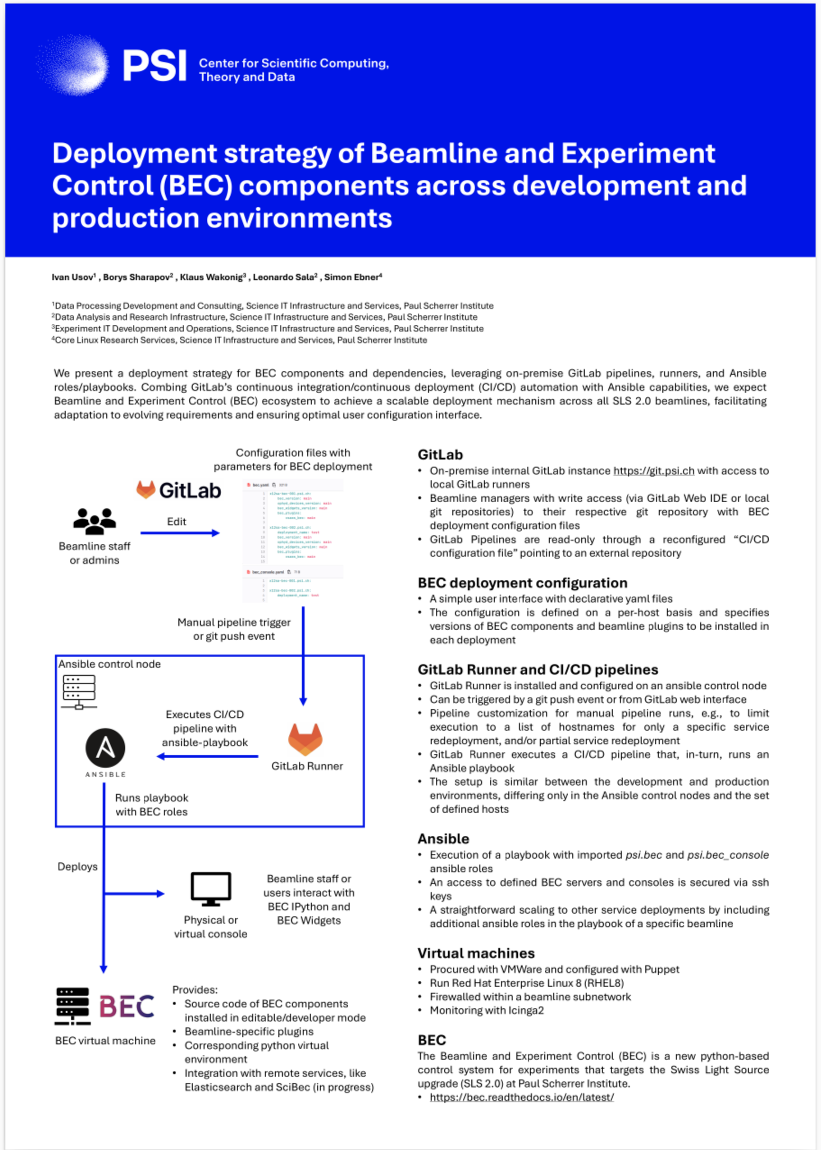
Poster at NOBUGS 2024

We are not the only one to use the “gitlab ci/cd+ansible” setup.

Feedback from discussions during poster sessions:

- A great way of self-documenting scalable deployment
- How beamlines find out when to update a version of a bec-component
  - Compatibility matrix between components?
  - Bundled distributions of compatible components?
- Beamline staff vs admins to actually run redeployments
  - ~10% beamline staff and ~90% admins?

Link to the poster:
https://indico.esrf.fr/event/114/contributions/841/



PSI Center for Scientific Computing, Theory and Data

Deployment strategy of Beamline and Experiment Control (BEC) components across development and production environments

Ivan Usov[1] , Borys Sharapov[2] , Klaus Wakonig[3] , Leonardo Sala[2] , Simon Ebner[4]

[1]Data Processing Development and Consulting, Science IT Infrastructure and Services, Paul Scherrer Institute
[2]Data Analysis and Research Infrastructure, Science IT Infrastructure and Services, Paul Scherrer Institute
[3]Experiment IT Development and Operations, Science IT Infrastructure and Services, Paul Scherrer Institute
[4]Core Linux Research Services, Science IT Infrastructure and Services, Paul Scherrer Institute

We present a deployment strategy for BEC components and dependencies, leveraging on-premise GitLab pipelines, runners, and Ansible roles/playbooks. Combing GitLab’s continuous integration/continuous deployment (CI/CD) automation with Ansible capabilities, we expect Beamline and Experiment Control (BEC) ecosystem to achieve a scalable deployment mechanism across all SLS 2.0 beamlines, facilitating adaptation to evolving requirements and ensuring optimal user configuration interface.

Configuration files with parameters for BEC deployment

GitLab

Beamline staff or admins

Edit

Manual pipeline trigger or git push event

Ansible control node

Executes CI/CD pipeline with ansible-playbook

ANSIBLE

GitLab Runner

Runs playbook with BEC roles

Deploys

Physical or virtual console

Beamline staff or users interact with BEC IPython and BEC Widgets

BEC

BEC virtual machine

Provides:
- Source code of BEC components installed in editable/developer mode
- Beamline-specific plugins
- Corresponding python virtual environment
- Integration with remote services, like Elasticsearch and SciBec (in progress)

GitLab

- On-premise internal GitLab instance https://git.psi.ch with access to local GitLab runners
- Beamline managers with write access (via GitLab Web IDE or local git repositories) to their respective git repository with BEC deployment configuration files
- GitLab Pipelines are read-only through a reconfigured “CI/CD configuration file” pointing to an external repository

BEC deployment configuration

- A simple user interface with declarative yaml files
- The configuration is defined on a per-host basis and specifies versions of BEC components and beamline plugins to be installed in each deployment

GitLab Runner and CI/CD pipelines

- GitLab Runner is installed and configured on an ansible control node
- Can be triggered by a git push event or from GitLab web interface
- Pipeline customization for manual pipeline runs, e.g., to limit execution to a list of hostnames for only a specific service redeployment, and/or partial service redeployment
- GitLab Runner executes a CI/CD pipeline that, in-turn, runs an Ansible playbook
- The setup is similar between the development and production environments, differing only in the Ansible control nodes and the set of defined hosts

Ansible

- Execution of a playbook with imported *psi.bec* and *psi.bec_console* ansible roles
- An access to defined BEC servers and consoles is secured via ssh keys
- A straightforward scaling to other service deployments by including additional ansible roles in the playbook of a specific beamline

Virtual machines

- Procured with VMWare and configured with Puppet
- Run Red Hat Enterprise Linux 8 (RHEL8)
- Firewalled within a beamline subnetwork
- Monitoring with Icinga2

BEC

The Beamline and Experiment Control (BEC) is a new python-based control system for experiments that targets the Swiss Light Source upgrade (SLS 2.0) at Paul Scherrer Institute.

- https://bec.readthedocs.io/en/latest/