ASIPP

Management of module repositories for Integrated Modeling and Analysis Workflow System

Full life-cycle provenance and cross-site migration of physics modules

Xiaojuan LIU ,Zhi YU, Nong XIANG

Institute of Plasma Physics ,Chinese Academy of Sciences

Outline

- Why do we need a management system for the provenance and migration of physics modules?
- Infrastructure of module repository management system .
- Full life-cycle provenance of physics module.
- Cross-site migration of module repository.
- Summary

ASIPP

The provenance of the code-building process should not be ignored.

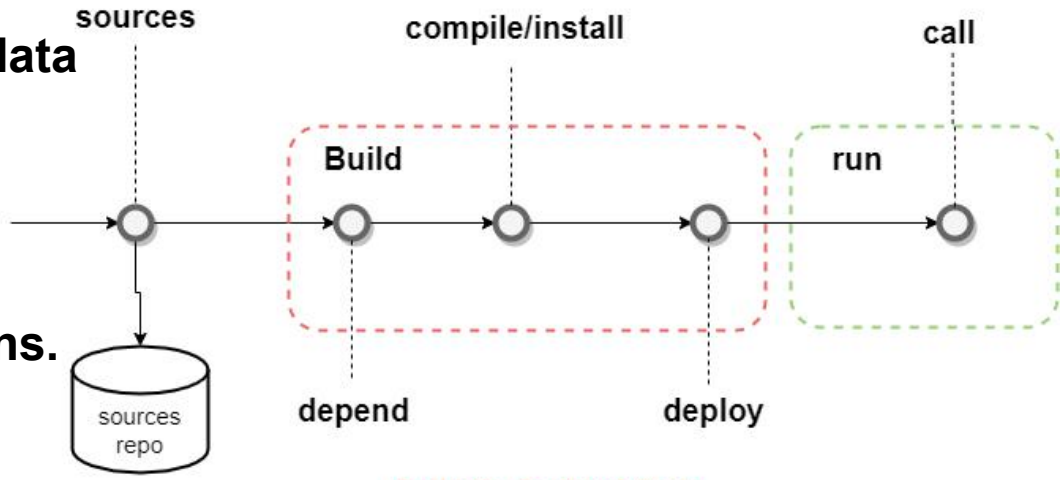
- **In a complex, large-scale scientific project, "provenance" is important for reproducible results.**

- ***Detailed requirements for the ITER Integrated Modelling Infrastructure***
  - tracking and recording the simulatons (data, components ...)
  - data description and **provenance** shall be recorded, together with information about its validity.

- ***Final Report on Open Science Use Cases for Fusion Information[1]***
  - **provenance metadata**, including data processing code parameters and version (svn version or git hash).

- ***2021 Workflows Community Summit[2]***
  - Theme #1: **FAIR (Findable, Accessible, Interoperable, and Reusable)** computational workflows.

- The workflow description **only records the provenance of the data flow**,emphasizing the application steps, parameter settings, etc.
- **Provenance of code should not be ignored.**
- **Incomplete provenance will lead to irreproducible calculations.**



ASIPP

1.https://www.fair4fusion.eu/
2.da Silva, Rafael Ferreira, et al. "Workflows Community Summit: Bringing the Scientific Workflows Community Together." arXiv preprint arXiv:2103.09181 (2021).

How to manage a large and complex module repository for integrated modeling?

Framework	transport-based		workflow-based			
Name	ASTRA	CRONOS	OMFIT		IMAS	
Physical Modules	NCLASS, GLF23M, SPIDER, ESC, NUBEAM, TORBEAM, TORIC, FRTC,.....	NCLASS, GLF23, SPOT, SELPHIHE, /C3PO/LUKE, PION, NEMO/SPOT, REMA, KINEZERO, MISHKA, CASTOR, HPI2, HELENA,.....	ONETWO, GLF23, NUBEAM, GENRAY, NEO, GYRO, TGLF, TEQ, DCON, NCLASS, TSC-EQ, CQL3D, TORIC,.....	EFIT, GATO, GKS, PEST3, SOLPS, SURFMN, TORBEAM, BOUT++, M3Dc1, NIMROD, AORSA,.....	IMAS Installer, FC2K, IDStools, IMASPy, PyAL, UDA, XMLlib, Viz, Kepler, HCD, ASCOT,.....	CASPER, CHESASE, FoPla, GENRAY, GRAYSCALE, HCD2CORE_PROFILES, HCD2CORE_SOURCES, NBISIM, NEMO, PION, RISK, SMITER, SPOT,.....

System administrator's troubles:

- **Complex** library and toolchain dependencies;
- **Multiple** development languages;
- **Multiple** versions of the toolchain coexist;
- Needs to be deployed on **multiple** HPC sites and kept in sync.

Requirements for module repository management system:

- Support **full life-cycle provenance of each module**;
- Support for provenance management of the **entire module repository.**
- Support for centralized management of **cross-site deployments.**

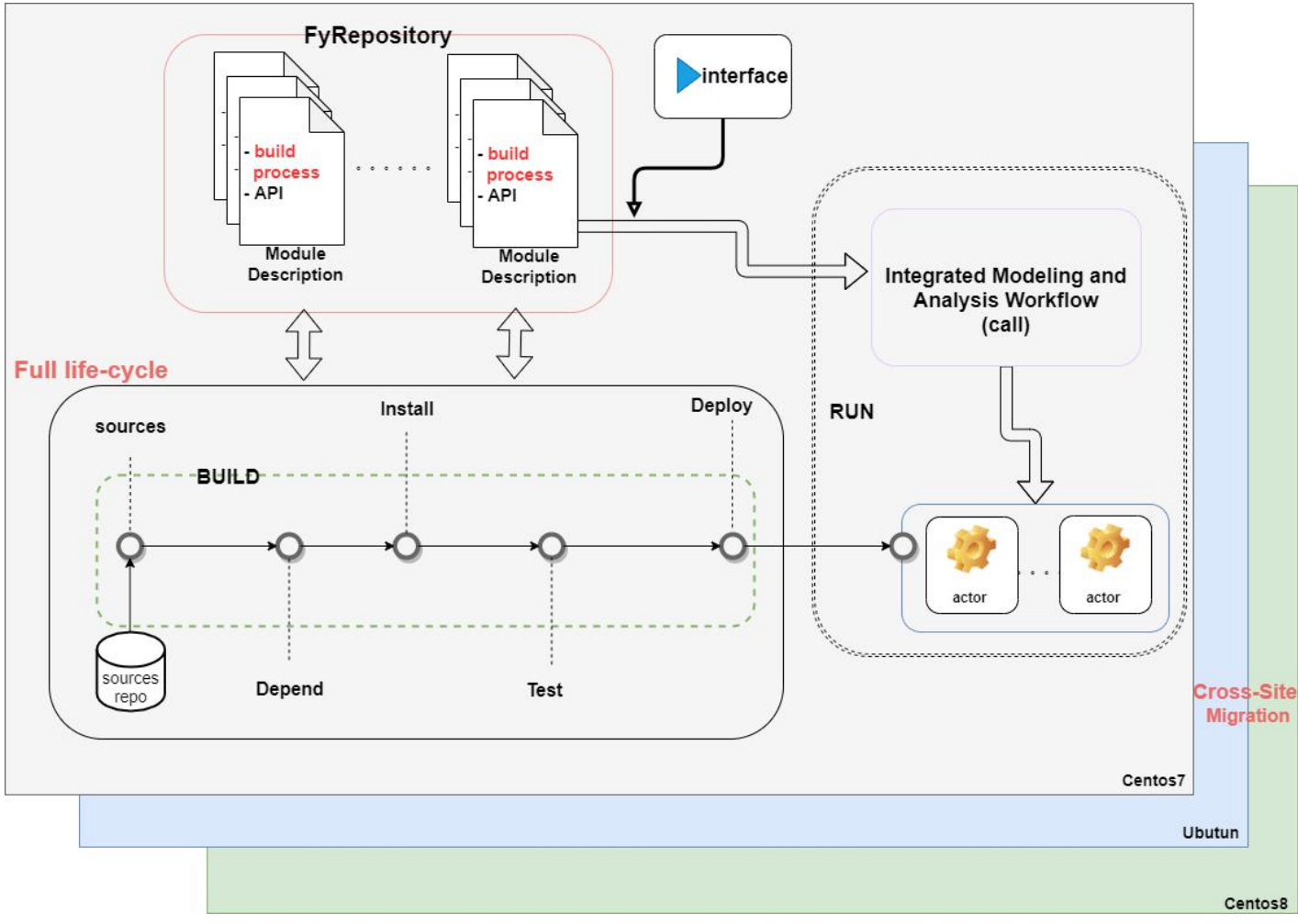
ASIPP

Infrastructure of the module repository management system

- **Standardize each stage** of the life-cycle of module:
  - Explicit executable command is presented to the user
  - Discover the hidden messages:
    - sources ,dependence, install, test, deploy

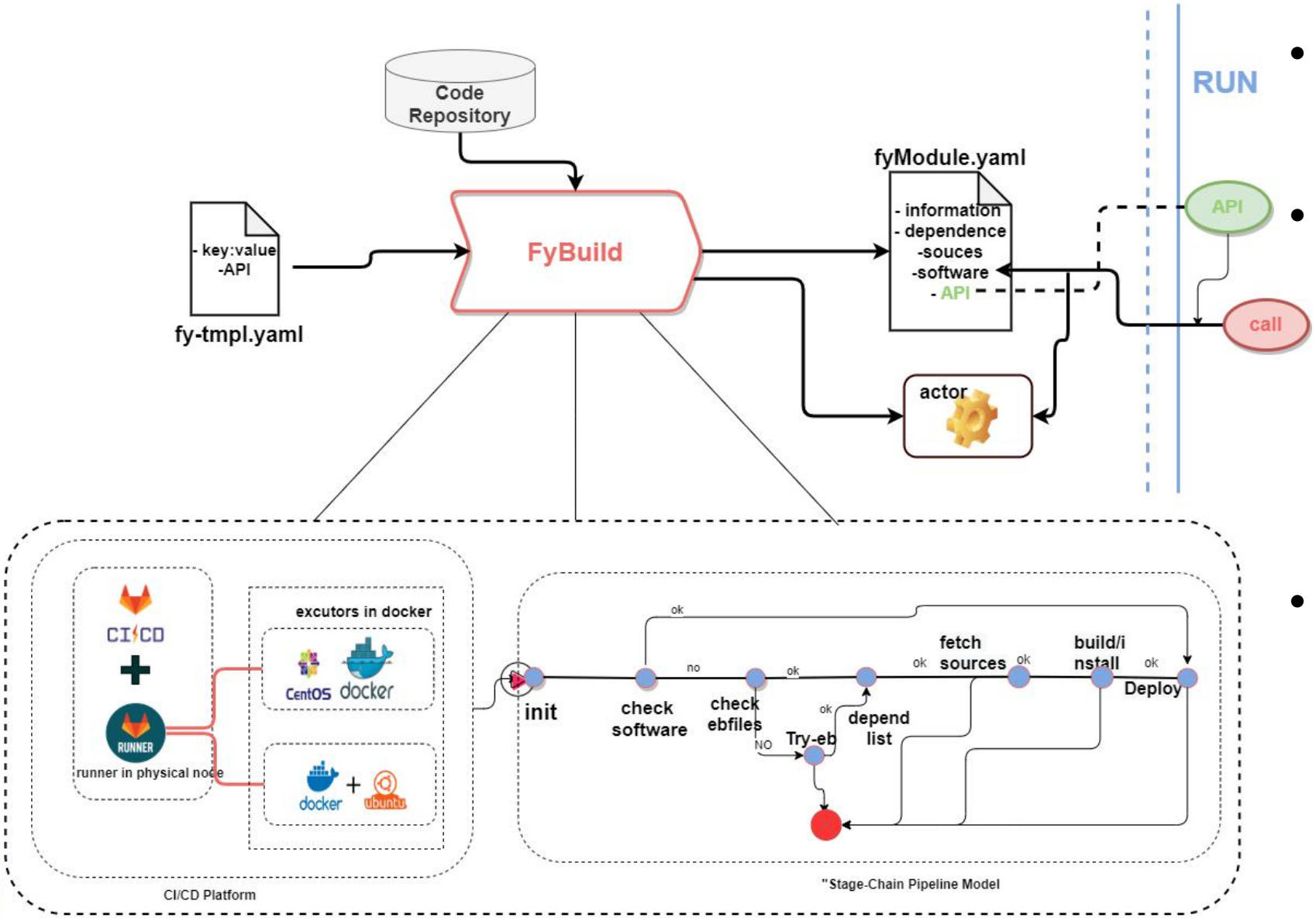
- **Dynamic tracking** of the life-cycle of the module:
  - Record the results of each stage to the **module description** file
  - One-to-one correspond to the physical module

- **Mapping the site environment to a collection of module description files**
  - **FyRepository** is a collection of description files.
  - Represents an ecosystem of module repository
  - Support for **cross-site** redeployment.



ASIPP

Full life-cycle provenance of the single module



- **Module template file :**
  - Some basic information
  - Fixed API description
- **FyBuild Tookits:**
  - "Stage-Chain" Pipeline Model:
    - Define the stages of the module life cycle
  - CI/CD Platform:
    - Trigger build
    - Assign the appropriate execution environment
  - Function codes
- **Products:**
  - FyModule.yaml:
    - Populate information in key-value format for each stage of the life-cycle
  - Generate an executable actor

FyMoudule.yaml —— an example of a module description file

fybuild-dev > ! genray-mpi-201213-gompi-2020a.yaml

```yaml
$id: genray-mpi/201213-gompi-2020a
$schema: SpModule#SpModuleLocal
annotation:
  contributors: liuxj
  date: Tue Oct  5 04:27:27 2021
  email: lxj@ipp.ac.cn
description: this is a fy_module file ···
homepage: http://funyun.com/demo.html
information:
  name: genray-mpi
  version: '201213'
install:
  $class: EasyBuild
  toolchain:
    tag: gompi-2020a
  process:
    depend:
      depend_cmd: eb -Dr genray-mpi-201213-gompi-2020a.eb --robot-paths=/gpfs/fuyun/software/EasyBuild
        /4.3.2/easybuild/easyconfigs:/scratch/liuxj/FYDEV-Workspace/FyBuild/EbfilesRespository
      packageslist:···
    fetch:
      sources:
      - name: genray-mpi-201213.tar.gz
        path: /gpfs/fuyun/sources/g/genray-mpi/genray-mpi-201213.tar.gz
        cmd:
        checksum:
        finalpath: /gpfs/fuyun/build/genraympi/201213/gompi-2020a
    build:
      build_cmd: eb genray-mpi-201213-gompi-2020a.eb -l --force --robot --minimal-toolchains
        --experimental --use-existing-modules --info --robot-paths=/gpfs/fuyun/software/EasyBuild
        /4.3.2/easybuild/easyconfigs:/scratch/liuxj/FYDEV-Workspace/FyBuild/EbfilesRespository
      ebfile: /gpfs/fuyun/software/genray-mpi/201213-gompi-2020a/easybuild/genray-mpi-201213-gompi-2020a.eb
```

```yaml
license: GPL
postscript: module purge
prescript:
  - /gpfs/fuyun//modules/all
  - module purge
  - module load genray-mpi/201213-gompi-2020a
API:
  in_ports:···
  out_ports:···
```

timedate ,owner

basic information

Toolkit

stages:
- checkdepent
- fetch sources
- build/install

Parse API

call actor

Intergration Workflow

- as an intermediate bridge between the base environment and module call

FyModule.yaml: as the generic interface of physics codes

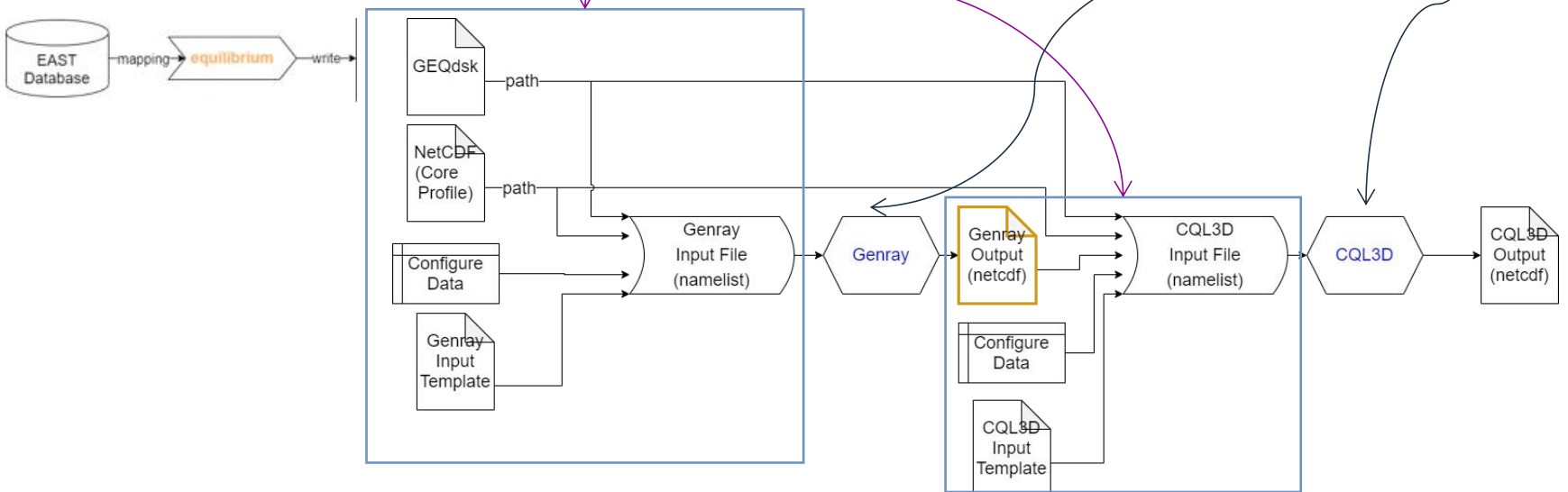
```python
# open connection to EAST experiment database
doc = Collection(f"EAST+mdsplus://{DATA_DIR}/db/mdsplus/~t/?tree_name=efit_east",
                 mapping_data_path=DATA_DIR/"mapping") \
    .open(shot=55555, run=0, time_slice=100, mode="r") # open document

# load genray (Fortran version)
genray = spdm.module.physics.genray[{"version": "10.13_200117", "tag": "-gompi-2020a"}](
    profile_in=FILE_DIR/"genray/genray_profs_in.nc",
    equilibrium=FILE_DIR/"genray/g0.04800",
    config={"$class": "file/namelist",  "template": FILE_DIR/"genray/genray.in"}
)


# load cql3d (Fortran version)
cql3d = spdm.module.physics.cql3d(
    equilibrium=genray.outputs.equilibrium,
    genray_result=genray.outputs.result, # using the output of genery as input
    config={"$class": "file/namelist",
            "default": {
                "setup": {
                    "nstop": 1,
                    "nplot": 1,
                    "nplt3d": 0
                }
            },
            "template": FILE_DIR/"cql3d/cqlinput"})
```

```
/gpfs/fuyun/software/
├── cql3d
│   ├── 9bcdb60-gompi-2019b
│   │   ├── bin
│   │   │   └── xcql3d.gfortran64
├── genray
│   ├── 10.13_200117-gompi-2019b
│   │   ├── bin
│   │   │   └── xgenray.gfortran64
│   ├── 10.13_200117-gompi-2020a
│   │   ├── bin
│   │   │   └── xgenray.gfortran64
```

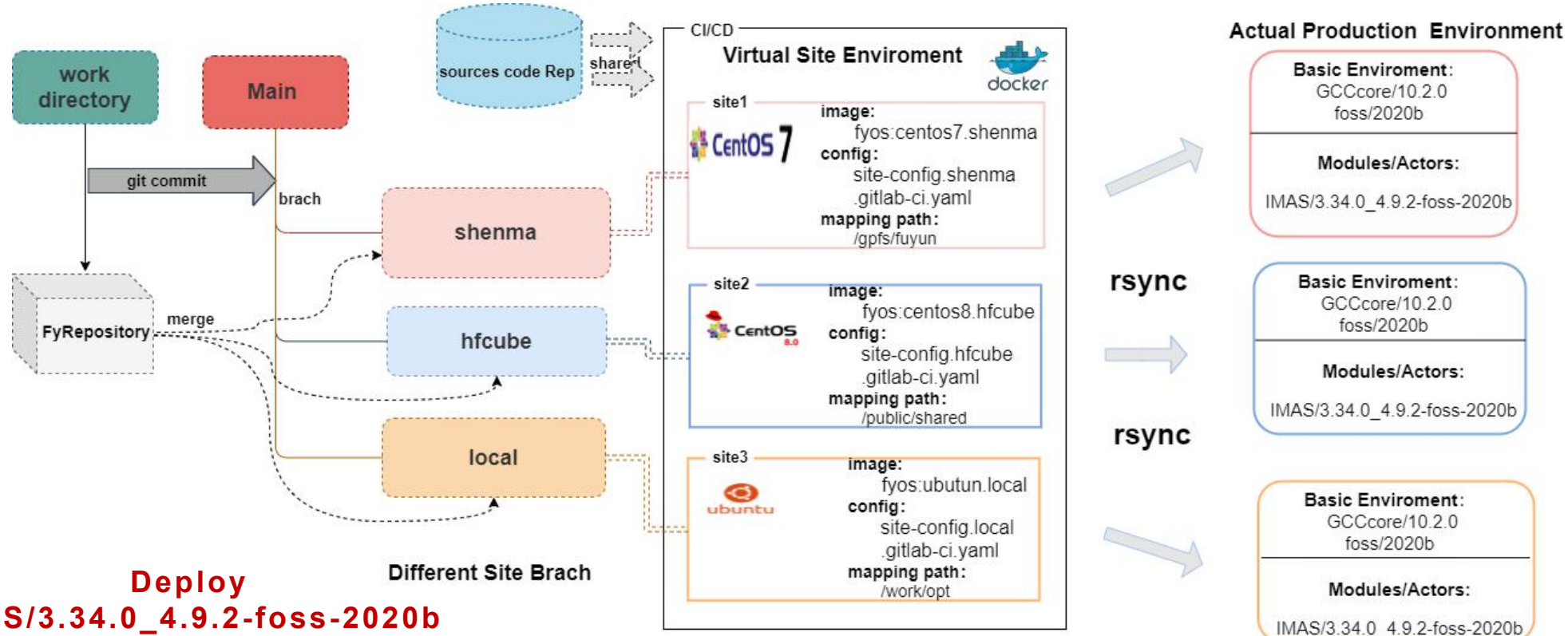
```
/gpfs/fuyun/FyRepository/
├── fymodules
│   ├── cql3d
│   │   └── default-gompi-2020a
│   │       └── fy_module.yaml
│   ├── genray
│   │   ├── 10.13_200117-gompi-2020a
│   │   │   └── genray-10.13_200117-gompi-2020a.yaml
│   │   └── fy_module.yaml
│   └── README.md
```



ASIPP

Cross-site migration of module repository

- **FyRepository** acts as a repository for **git**.
- Different **sites** correspond to different git **branches**.
- The building process takes place in the **docker container**.
- The build results are **synchronized** to the corresponding production environment respectively.



Deploy
IMAS/3.34.0_4.9.2-foss-2020b
to three different site environments

ASIPP

Summary

- A management system for **provenance and migration** of module repository was created.
  - record metadata to fill the gap about the **provenance for full life-cycle of the module**;
  - enable the migration of sites
  - makes it possible to reproduce results
- The core design concept in the implementation of architecture :
  - **Standardize each stage of** the life-cycle of module
  - **Dynamically track** the full life-cycle
  - **Mapping the site environment** to the collection of module description files
- Implementation :
  - FyModule.yaml(description file of the single module) :
    - as a metadata branch ,corresponding to the physical module
    - At the same time, as a bridge between the underlying environment and the module call
  - FyRepository(a collection of module description files):
    - determines the environment ecology of the physical module
    - as a repository source to enables cross-site migration

ASIPP

ASIPP

Thank you for attention