Using Continuous Integration with EasyBuild in DEEP-SEA

Manoel Römmer, Jülich Supercomputing Centre

2023-12-01 – The DEEP-SEA Seminar Series

The Research leading to these results has received funding from the European Commission's FP7, H2020, and EuroHPC Programmes, under Grant Agreements nř 287530, 610476, 754304, and 955606

Continuous Integration in DEEP-SEA

Objective: *Ensure a proper automated integration process by using software engineering techniques.*

Continuous Integration in DEEP-SEA

Objective: *Ensure a proper automated integration process by using software engineering techniques.*

Currently done by employing general CI infrastructure by various Work Packages:

- Various automated checks and tests
- Automatically running benchmarks
- Automatically building and deploying software on DEEP

Continuous Integration in DEEP-SEA

Objective: *Ensure a proper automated integration process by using software engineering techniques.*

Currently done by employing general CI infrastructure by various Work Packages:

- Various automated checks and tests
- Automatically running benchmarks
- Automatically building and deploying software on DEEP

as part of an automated workflow, directly from GitLab.

The EasyBuild CI

Automatically build and deploy software on DEEP

The EasyBuild CI

Automatically build and deploy software on DEEP

- Build and deploy *bleeding edge* software stage from GitLab on the DEEP system

The EasyBuild CI

Automatically build and deploy software on DEEP

- Build and deploy *bleeding edge* software stage from GitLab on the DEEP system
- (Optionally) rebuild *reverse dependencies*, to keep every component in the Stage coherent

The EasyBuild CI

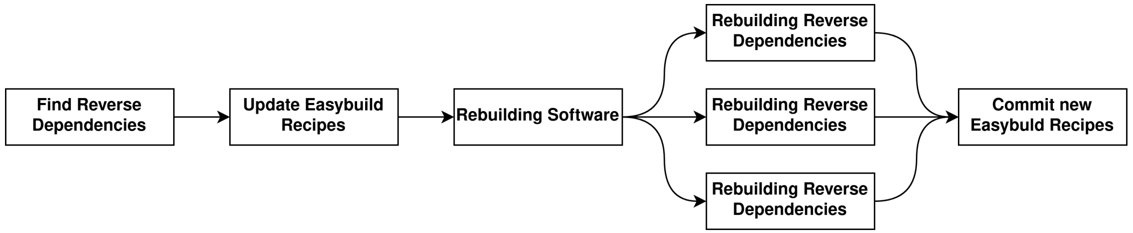
Automatically build and deploy software on DEEP

- Build and deploy *bleeding edge* software stage from GitLab on the DEEP system
- (Optionally) rebuild *reverse dependencies*, to keep every component in the Stage coherent

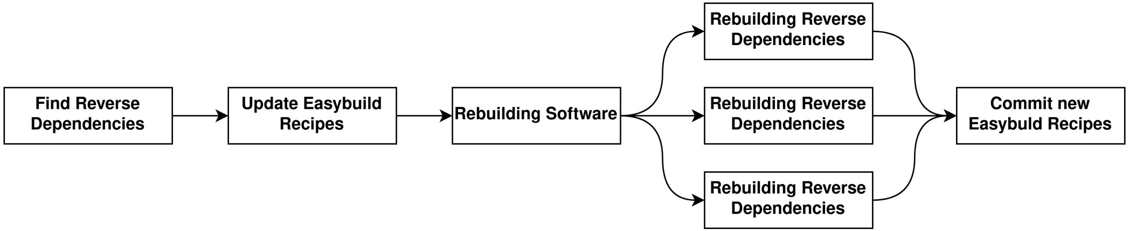
The software installations on DEEP are mostly handled by

EASYBUILD.io
building software with ease

The EasyBuild CI (cont'd)


Find Reverse Dependencies
Update Easybuild Recipes
Rebuilding Software
Rebuilding Reverse Dependencies
Rebuilding Reverse Dependencies
Rebuilding Reverse Dependencies
Commit new Easybuld Recipes

The EasyBuild CI (cont’d)


Find Reverse Dependencies
Update Easybuild Recipes
Rebuilding Software
Rebuilding Reverse Dependencies
Rebuilding Reverse Dependencies
Rebuilding Reverse Dependencies
Commit new Easybuld Recipes

This pipeline lives in the `deep-sea/wp3/software/easybuild-repository-deep-sea` repository, and you can trigger it as a *downstream pipeline* from your own repository.

Prerequisites

EasyBuild Recipe

```
name = 'Scalasca'
version = '2.6.1'

toolchain = {'name': 'gpsmpi', 'version': '2022a'}

source_urls = [
    'https://apps.fz-juelich.de/scalasca/releases/scalasca/%(version_major_minor)s/dist'
]
sources = [SOURCELOWER_TAR_GZ]
checksums = ['a0dbc3de82a6c0fe598de9e340513cff2882c199410a632d3a7f073ba921c7e7']

dependencies = [
    ('CubeGUI', '4.8'),
    ('CubeLib', '4.8'),
    ('OTF2', '3.0.2'),
    ('Score-P', '8.0'),
]
```

Existing EasyBuild recipes used as a *template* to generate new recipes for CI run

JSC GitLab Repository

The CI workflow lives in the JSC GitLab EasyBuild repository.

JSC GitLab Repository

The CI workflow lives in the JSC GitLab EasyBuild repository.

- gitlab.jsc.fz-juelich.de/deep-sea/wp3/software/easybuild-repository-deep-sea

JSC GitLab Repository

The CI workflow lives in the JSC GitLab EasyBuild repository.

- gitlab.jsc.fz-juelich.de/deep-sea/wp3/software/easybuild-repository-deep-sea
- Please have your EasyBuild recipes merged there in the 2023 branch

JSC GitLab Repository

The CI workflow lives in the JSC GitLab EasyBuild repository.

- gitlab.jsc.fz-juelich.de/deep-sea/wp3/software/easybuild-repository-deep-sea
- Please have your EasyBuild recipes merged there in the 2023 branch

How use it then? You can *trigger* this pipeline from your own GitLab repository!

JSC GitLab Repository

The CI workflow lives in the JSC GitLab EasyBuild repository.

- gitlab.jsc.fz-juelich.de/deep-sea/wp3/software/easybuild-repository-deep-sea
- Please have your EasyBuild recipes merged there in the 2023 branch

How use it then? You can *trigger* this pipeline from your own GitLab repository!

- Ideally: Your code repository in the JSC GitLab
- Alternatively: Create a repository just for your CI workflow

A Quick Word From Project Management

Public EasyBuild repository as Software Release:
gitlab.jsc.fz-juelich.de/deep-sea/easybuild-repository-public-release

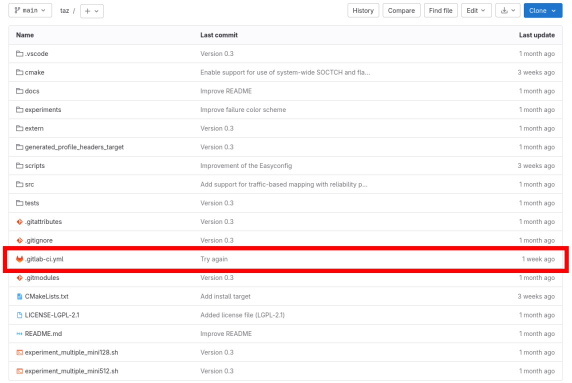
Internal EasyBuild repository:
gitlab.jsc.fz-juelich.de/deep-sea/wp3/software/easybuild-repository-deep-sea/

↑ Please use this one to trigger CI ↑

CI Configuration from YOUR y Repository

CI Configuration from YOUR y Repository

You need to configure a pipeline with the `.gitlab-ci.yml`

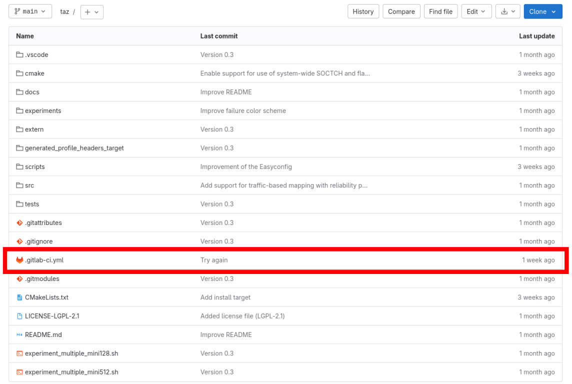


Name	Last commit	Last update
.vscode	Version 0.3	1 month ago
cmake	Enable support for use of system-wide SOCTCH and fla...	3 weeks ago
docs	Improve README	1 month ago
experiments	Improve failure color scheme	1 month ago
extern	Version 0.3	1 month ago
generated_profile_headers_target	Version 0.3	1 month ago
scripts	Improvement of the Easyconfig	3 weeks ago
src	Add support for traffic-based mapping with reliability p...	1 month ago
tests	Version 0.3	1 month ago
.gitattributes	Version 0.3	1 month ago
.gitignore	Version 0.3	1 month ago
.gitlab-ci.yml	Try again	1 week ago
.gitmodules	Version 0.3	1 month ago
CMakeLists.txt	Add install target	3 weeks ago
LICENSE-LGPL-2.1	Added license file (LGPL-2.1)	1 month ago
README.md	Improve README	1 month ago
experiment_multiple_mini128.sh	Version 0.3	1 month ago
experiment_multiple_mini512.sh	Version 0.3	1 month ago

CI Configuration from YOUR y Repository

You need to configure a pipeline with the `.gitlab-ci.yml`

- If you already have a JSC GitLab repo with CI workflow, integrate it there
- If not, consider putting your code in JSC GitLab
- If not, create a repository with only your CI trigger



Name	Last commit	Last update
.vscode	Version 0.3	1 month ago
cmake	Enable support for use of system-wide SOCTCH and fla...	3 weeks ago
docs	Improve README	1 month ago
experiments	Improve failure color scheme	1 month ago
extern	Version 0.3	1 month ago
generated_profile_headers_target	Version 0.3	1 month ago
scripts	Improvement of the Easyconfig	3 weeks ago
src	Add support for traffic-based mapping with reliability p...	1 month ago
tests	Version 0.3	1 month ago
.gitattributes	Version 0.3	1 month ago
.gitignore	Version 0.3	1 month ago
.gitlab-ci.yml	Try again	1 week ago
.gitmodules	Version 0.3	1 month ago
CMakeLists.txt	Add install target	3 weeks ago
LICENSE-LGPL-2.1	Added license file (LGPL-2.1)	1 month ago
README.md	Improve README	1 month ago
experiment_multiple_mini128.sh	Version 0.3	1 month ago
experiment_multiple_mini512.sh	Version 0.3	1 month ago

How To Integrate

Upstream Workflow Summary

1. Have *one* EasyBuild recipe for your software merged in `easybuild-repository-deep-sea` Repository

Upstream Workflow Summary

1. Have *one* EasyBuild recipe for your software merged in `easybuild-repository-deep-sea` Repository

2. In your own pipeline:
   - Bundle your sources (e.g. `.tar.gz`) and move it somewhere well known `/p/project/deepsea/ci-stage-sources`

Upstream Workflow Summary

1. Have *one* EasyBuild recipe for your software merged in `easybuild-repository-deep-sea` Repository

2. In your own pipeline:
   - Bundle your sources (e.g. `.tar.gz`) and move it somewhere well known `/p/project/deepsea/ci-stage-sources`

3. Trigger the downstream CI pipeline:
   - For most, setting the `$SRC` and `$SRC_URLS` variable to your source file name should be enough
   - Some special care must be taken for Python EasyBuild recipes (and bundles in general). If you have one of these, talk to me!

Triggering a Rebuild

```yaml
stages: [prep, staging]
default:
  tags: [deep-sea, jacamar, shell]

create_tarball:
  stage: prep
  script:
  - mkdir -p -m 777 /p/project/deepsea/ci-stage-sources/s/Score-P/
  - tar -cvz --exclude=.git* -f /p/project/deepsea/ci-stage-sources/s/Score-P/${CI_PIPELINE_ID}.tar.gz .
  - chmod g+rw /p/project/deepsea/ci-stage-sources/s/Score-P/${CI_PIPELINE_ID}.tar.gz

trigger-downstream:
  stage: staging
  variables:
    EB_FILE_CURRENT: "Score-P-7.1-gpsmpi-2021b.eb"
    SRC: "${CI_PIPELINE_ID}.tar.gz"
    SRC_URLS: "file:///p/project/deepsea/ci-stage-sources/s/Score-P/"
    BUILD_INVERSE_DEPENDENCIES: "TRUE"
  trigger:
    project: deep-sea/wp3/software/easybuild-repository-deep-sea
    strategy: depend
    branch: ci-2023-dev
```

Triggering a Rebuild

```
stages: [prep, staging]
default:
  tags: [deep-sea, jacamar, shell]
```

Triggering a Rebuild

```
stages: [prep, staging]
default:
  tags: [deep-sea, jacamar, shell]

create_tarball:
  stage: prep
  script:
  - mkdir -p -m 777 /p/project/deepsea/ci-stage-sources/s/Score-P/
  - tar -cvz --exclude=.git* -f /p/project/deepsea/ci-stage-sources/s/Score-P/${CI_PIPELINE_ID}.tar.gz .
  - chmod g+rw /p/project/deepsea/ci-stage-sources/s/Score-P/${CI_PIPELINE_ID}.tar.gz
```

Triggering a Rebuild

```
stages: [prep, staging]
default:
  tags: [deep-sea, jacamar, shell]

create_tarball:
  stage: prep
  script:
  - mkdir -p -m 777 /p/project/deepsea/ci-stage-sources/s/Score-P/
  - tar -cvz --exclude=.git* -f /p/project/deepsea/ci-stage-sources/s/Score-P/${CI_PIPELINE_ID}.tar.gz .
  - chmod g+rw /p/project/deepsea/ci-stage-sources/s/Score-P/${CI_PIPELINE_ID}.tar.gz

trigger-downstream:
  stage: staging
  variables:
    EB_FILE_CURRENT: "Score-P-8.0-gpsmpi-2022a.eb"
    SRC: "${CI_PIPELINE_ID}.tar.gz"
    SRC_URLS: "file:///p/project/deepsea/ci-stage-sources/s/Score-P/"
    BUILD_INVERSE_DEPENDENCIES: "TRUE"
  trigger:
    project: deep-sea/wp3/software/easybuild-repository-deep-sea
    strategy: depend
    branch: ci-2023-dev
```

Effect on EasyBuilds

Required:

- `EB_FILE_CURRENT`: A single EasyBuild recipe used as template

Effect on EasyBuilds

Required:

- `EB_FILE_CURRENT`: A single EasyBuild recipe used as template

Optional (kinda):

- `SRC`: Replaces the `sources` field in recipe
- `SRC_URLS`: Replaces the `source_urls` field in recipe
- `BUILD_INVERSE_DEPENDENCIES`: Enables reverse dependency rebuilt (value `"TRUE"` or `"FALSE"`)

Effect on EasyBuilds

Required:

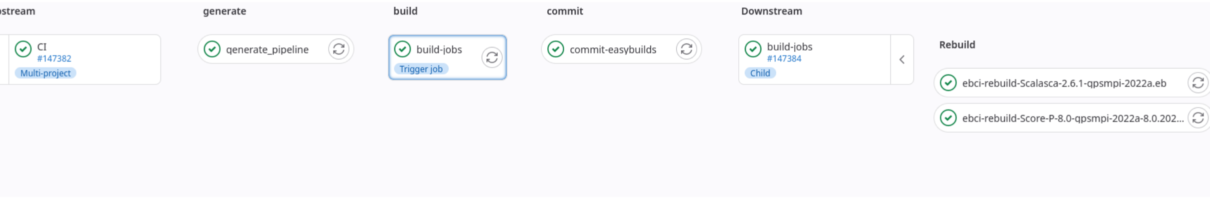
- `EB_FILE_CURRENT`: A single EasyBuild recipe used as template

Optional (kinda):

- `SRC`: Replaces the `sources` field in recipe
- `SRC_URLS`: Replaces the `source_urls` field in recipe
- `BUILD_INVERSE_DEPENDENCIES`: Enables reverse dependency rebuilt (value `"TRUE"` or `"FALSE"`)

Also available:

- `REBUILD_ONLY`: Limits the reverse dependency rebuild by package names

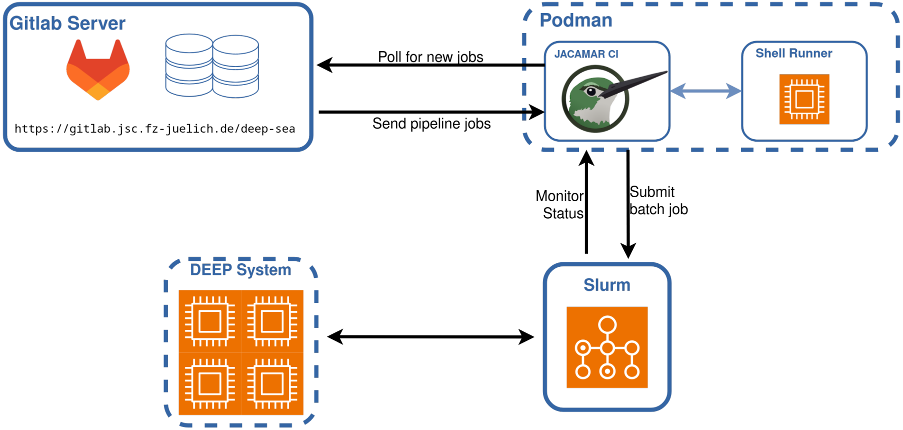

stream
generate
build
commit
Downstream
CI
#147382
Multi-project
generate_pipeline
build-jobs
Trigger job
commit-easybuilds
build-jobs
#147384
Child
Rebuild
ebci-rebuild-Scalasca-2.6.1-gpsmpi-2022a.eb
ebci-rebuild-Score-P-8.0-gpsmpi-2022a-8.0.202...

```
$ module use /p/project/deepsea/ci-stage-2023-dev/easybuild/modules/all
$ module avail Score-P

---------- /p/project/deepsea/ci-stage-2023-dev/easybuild/modules/all ---------
MPI/GCC/11.3.0/psmpi/5/Score-P/8.0.20231123.105500 (u)
MPI/GCC/11.3.0/psmpi/5/Score-P/8.0.20231123.122040 (u, D)

Where:
g:  built for GPU
u:  Built by user

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching
any of the "keys".

$ module load MPI/GCC/11.3.0/psmpi/5/Score-P/8.0.20231123.122040
```

Thank You For Your Attention

Further Content

Continuous Integration Infrastructure

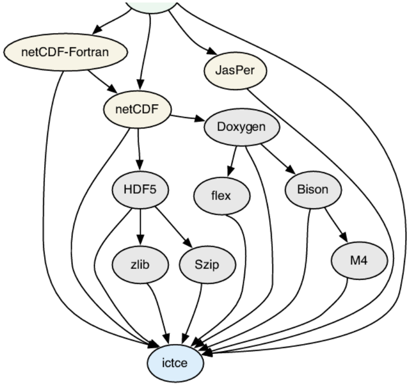


Reverse Dependency Rebuilding

- `netCDF` depends on `HDF5`
- When we push a new version of `HDF5` we should also rebuild and test `netCDF`

Reverse Dependency Rebuilding

- `netCDF` depends on `HDF5`
- When we push a new version of `HDF5` we should also rebuild and test `netCDF`



Reverse Dependency Rebuilding

EasyBuild CI needs to find reverse dependencies, update their dependency information and rebuild them

Reverse Dependency Resolution

- Not directly supported by EasyBuild
  - We dump EasyBuilds full dependency graph, use it to locate reverse dependencies
  - We actually need a present Easybuild (base-) recipe for that

Reverse Dependency Resolution

- Not directly supported by EasyBuild
  - We dump EasyBuilds full dependency graph, use it to locate reverse dependencies
  - We actually need a present Easybuild (base-) recipe for that
- Configurable by upstream user (that’s You):
  - Do you need reverse dependency rebuild: Yes/No?
  - All your reverse dependencies or only a select few?
  - More flexible configuration options in the works!

Rewriting EasyBuild Recipes

```
name = 'Scalasca'
version = '2.6.1'

toolchain = {
    'name': 'gpsmpi',
    'version': '2022a'
}

dependencies = [
    ('CubeGUI', '4.8'),
    ('CubeLib', '4.8'),
    ('OTF2', '3.0.2'),
    ('Score-P', '8.0'),
]
```

$\rightarrow$

```
name = 'Scalasca'
version = '2.6.1.20230726.064424'

toolchain = {
    'name': 'gpsmpi',
    'version': '2022a'
}

dependencies = [
    ('CubeGUI', '4.8'),
    ('CubeLib', '4.8'),
    ('OTF2', '3.0.2'),
    ('Score-P', '8.0.20230726.064424'),
]
```

We now have a robust and flexible solution to update EasyBuild Recipes!

Downstream EasyBuild Pipeline Generation

In the previous design: Build job orchestration supposed to be done by JUBE

Downstream EasyBuild Pipeline Generation

In the previous design: Build job orchestration supposed to be done by JUBE

Now: Moved to custom solution that generates downstream pipeline

Downstream EasyBuild Pipeline Generation

In the previous design: Build job orchestration supposed to be done by JUBE

Now: Moved to custom solution that generates downstream pipeline

- Jacamar can directly submit (build) jobs to SLURM
- Jobs can be directly monitored from Gitlab CI/CD control panel
- Shell runners are available for other tasks

Downstream EasyBuild Pipeline Generation

In the previous design: Build job orchestration supposed to be done by JUBE

Now: Moved to custom solution that generates downstream pipeline

- Jacamar can directly submit (build) jobs to SLURM
- Jobs can be directly monitored from Gitlab CI/CD control panel
- Shell runners are available for other tasks

Of course you can still, use JUBE to run your benchmarks on top of this stack!

How to Integrate?

Trigger?

```
workflow:
  rules:
    - if: $CI_COMMIT_MESSAGE =~ /-draft$/
      when: never
    - if $CI_PIPELIEN_SOURCE == "push"
```

When to trigger a rebuild?

- Depends on development workflow
- On push, merge request, release, manually,. . .