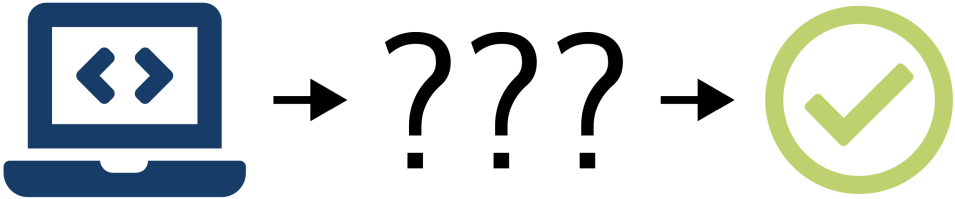


CONTINUOUS INTEGRATION FOR HPC

JLESC, March 23, 2023 | Jakob Fritz, Ivo Kabadshow, Robert Speck | Jülich Supercomputing Center, Forschungszentrum Jülich

JÜLICH Forschungszentrum

CI for HPC

JÜLICH Forschungszentrum

CI for HPC

What is it?

Focus on Continuous Integration (CI)
Continuous Benchmarking (CB) is also interesting and relevant, but CI should be up-and-running first!

CI for HPC

What is it?

Focus on Continuous Integration (CI)
Continuous Benchmarking (CB) is also interesting and relevant, but CI should be up-and-running first!

Automatically test your code every time, that it is updated in the repository.

CI for HPC

What is it?

Focus on Continuous Integration (CI)
Continuous Benchmarking (CB) is also interesting and relevant, but CI should be up-and-running first!

Automatically test your code every time, that it is updated in the repository.

Within CI for HPC there are multiple aspects

- CI for software that is developed to run on HPC
- running CI on HPC-Infrastructure

CI for HPC

For whom and why is this relevant?

- Users, that want to run their code on one or multiple nodes

CI for HPC

For whom and why is this relevant?

- Users, that want to run their code on one or multiple nodes
- Software Engineers, that develop software that shall run on HPC
  - Testing the software helps to find errors early
  - reduces callbacks from users
  - makes collaboration on code easier, see above

CI for HPC

For whom and why is this relevant?

- Users, that want to run their code on one or multiple nodes
- Software Engineers, that develop software that shall run on HPC
  - Testing the software helps to find errors early
  - reduces callbacks from users
  - makes collaboration on code easier, see above
- Software Engineers, whose code shall work on different architectures

CI for HPC

For whom and why is this relevant?

- Users, that want to run their code on one or multiple nodes
- Software Engineers, that develop software that shall run on HPC
  - Testing the software helps to find errors early
  - reduces callbacks from users
  - makes collaboration on code easier, see above
- Software Engineers, whose code shall work on different architectures
- Software Engineers, who want regular performance checks of their code ($\rightarrow$ CB)

Table of Contents

JÜLICH Forschungszentrum

Integrate Gitlab-CI with Github-Repositories

Integrate Gitlab-CI with Github-Repositories

Why is this relevant?

It helps to combine

- The larger community and visibility of Github
- The possibilities of self-hosted Gitlab-Instances
- The self-hosted CI-Runners of Gitlab

Integrate Gitlab-CI with Github-Repositories

How is this done?

Preparation:

- Create new, empty repo on a Gitlab-Instance
- Create a token in Gitlab
- Add token as secret in Github

Integrate Gitlab-CI with Github-Repositories

How is this done?

Preparation:

- Create new, empty repo on a Gitlab-Instance
- Create a token in Gitlab
- Add token as secret in Github

Usage:

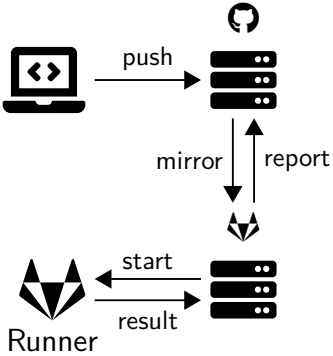
- Add `.gitlab-ci.yml`-file to your Github-repo. With code to be executed after mirroring to Gitlab.
- Add following to Github-Actions (in `.github/workflows`)

.github/workflows/mirror.yml

```yaml
name: Mirror and run GitLab CI
on: [push, pull_request_target]
jobs:
  github2lab:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v1
    - name: Mirror and get status
      uses: jakob-fritz/github2lab_action@main
      env:
        MODE: 'all' # 'mirror', 'get_status', 'all'
        GITLAB_TOKEN: ${{ secrets.GITLAB_TOKEN }}
        GITLAB_HOSTNAME: "codebase.helmholtz.cloud"
        GITLAB_PROJECT_ID: "6627"
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

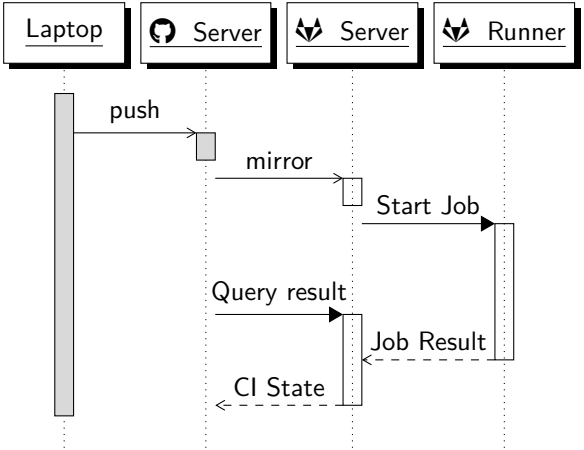
Integrate Gitlab-CI with Github-Repositories

How is this done?



Integrate Gitlab-CI with Github-Repositories

How is this done?



Integrate Gitlab-CI with Github-Repositories

Benefits of this approach

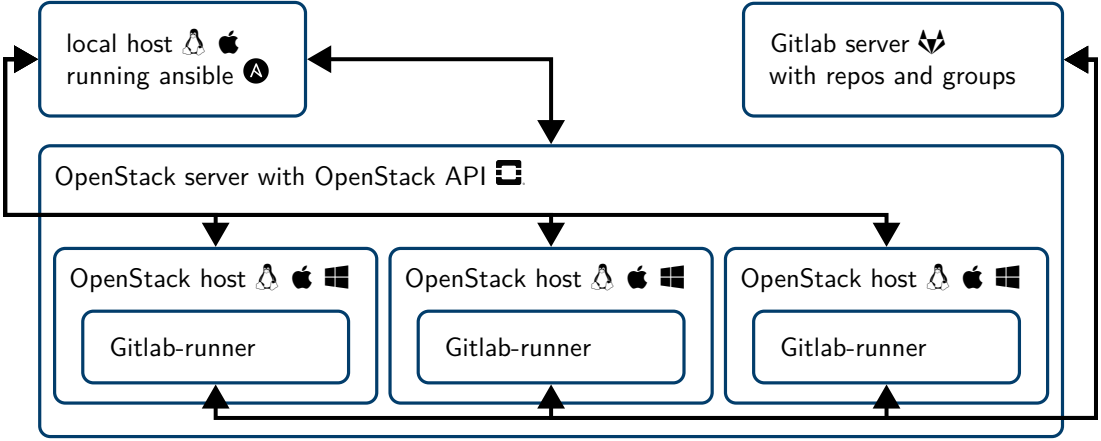
- ✓ Possible to use HPC-Ressources in CI
  even if only a specific Gitlab-Instance is allowed/able to connect to HPC-Ressources
- ✓ Possible to extend to own Runners, that are available in Gitlab

JÜLICH Forschungszentrum

Add SSH-Runner to Gitlab-CI

Add SSH-Runner to Gitlab-CI

Brief recap of previous development



Add SSH-Runner to Gitlab-CI

Brief recap of previous development

1. **Load**
   Load config from files
   ⇒ callable as variable
2. **List**
   Query OpenStack-API for hosts with fitting name
3. **Compare**
   Determine number of hosts per flavor to create or remove.
4. **Create**
   Create a jumphost, in case it does not yet exist.
5. **Deregister**
   Deregister GL-runners that are not needed anymore.
6. **Remove**
   Remove the unused hosts to free resources of OpenStack.
7. **Create**
   Create hosts of the desired flavor & add new hosts to the known hosts file.
8. **Assign**
   Check what runners need to be updated.
   Assign config to runners.
9. **Provision**
   Provision all hosts:
   Install all requirements, run updates and register runners.

Add SSH-Runner to Gitlab-CI

Why SSH-Runner

SSH-Runners can be of use, when

- Docker is not available on an architecture (or not officially supported)
- Docker is not allowed due to policies
- Direct SSH-Access is preferable

Add SSH-Runner to Gitlab-CI

How to achieve it

Config of Runners

- Number
  & individual configuration
  For use with OpenStack

gitlab_openstack.yml (Beginning)

```
instance_names: "AnsibleTestRunner*"
openstack_key_name: "ansible_key"
openstack_key_file: "ansible_key"
name_jumphost: "Jump_host"

default_runner_config:
  state: present
  run_untagged: true
  tags: []
  executor: "docker"
  environment:
    DOCKER_IMAGE: "ubuntu:22.04"
```

Add SSH-Runner to Gitlab-CI

How to achieve it

Config of Runners

- Number
  & individual configuration
  For use with OpenStack

gitlab_openstack.yml (Continued; docker)

```
runners:
  - flavor: "m2.large-disk"
    amount: 1
    config:
      - tags:
          - "docker"
          - "great_software"
          - "split tag"
          - "2 nice tag 1"
          - "HPC"
        run_untagged: true
```

Add SSH-Runner to Gitlab-CI

How to achieve it

Config of Runners

- Number
  & individual configuration
  For use with OpenStack

gitlab_openstack.yml (Continued; SSH)

```
runners:
  - amount: 1
    flavor: "s2.large-disk"
    config:
      - run_untagged: false
        executor: "ssh"
        tags:
          - "ssh-executor"
        environment:
          SSH_HOST: "192.168.0.47"
          SSH_PORT: 22
          SSH_USER: "ubuntu"
          SSH_IDENTITY_FILE: "ssh_runner_key"
          SSH_LOCAL_ID_FILE: "ssh_runner_2"
```

Add SSH-Runner to Gitlab-CI

How to achieve it

Config of Runners

- Number
  & individual configuration
  For manually added hosts

gitlab_specified_hosts.yml (Beginning)

```
default_runner_config:
  state: present
  run_untagged: true
  tags: []
  executor: "docker"
  environment:
    DOCKER_IMAGE: "ubuntu:22.04"
```

Add SSH-Runner to Gitlab-CI

How to achieve it

Config of Runners

- Number
  & individual configuration
  For manually added hosts

gitlab_specified_hosts.yml (Continued; docker)

```
runners:
  - hosts:
      - "Gitlab_Ansible_0"
      - "Gitlab_Runner_1"
    config:
      - tags:
          - "docker"
          - "great_software"
          - "split tag"
          - "2 nice tag 1"
          - "HPC"
        run_untagged: true
        state: present
```

Add SSH-Runner to Gitlab-CI

How to achieve it

Config of Runners

- Number
  & individual configuration
  For manually added hosts

gitlab_specified_hosts.yml (Continued; SSH)

```
runners:
  - hosts:
      - "Gitlab_Ansible_0"
      - "Gitlab_Runner_1"
    config:
      - executor: "ssh"
        run_untagged: false
        tags:
          - "ssh-executor"
        environment:
          SSH_HOST: "192.168.0.47"
          SSH_PORT: 22
          SSH_USER: "ubuntu"
          SSH_IDENTITY_FILE: "ssh_runner_key"
          SSH_LOCAL_ID_FILE: "ssh_runner_2"
```

Add SSH-Runner to Gitlab-CI

Heads up

Some aspects to be aware of, when using SSH-Runner:

- **!** No built-in security; CI runs as user whose credentials are used
  $\Rightarrow$ be sure to not use personal accounts.
- **!** No automatic clean-up; previous jobs may affect later jobs by changing files, variables
- **!** Only a single OS usable; the one of the machine you run on

Conclusions

Conclusions

- GitHub2Lab
  - ✓ Github-Action to combine large community & visibility with Gitlab-CI
  - ✓ Use own GL instances and own runners (e.g. on own hardware)
  - ✓ Easy setup
  - ✓ Can be combined with other Github-Actions, so usage of both Github-Action & Gitlab-CI is possible

Conclusions

- GitHub2Lab
  - ✓ Github-Action to combine large community & visibility with Gitlab-CI
  - ✓ Use own GL instances and own runners (e.g. on own hardware)
  - ✓ Easy setup
  - ✓ Can be combined with other Github-Actions, so usage of both Github-Action & Gitlab-CI is possible
- Gitlab-Runners with SSH-Executors
  - ✓ Possibility to use hardware without needing docker
  - ✓ low entry-barrier but few comfort-features for runners
  - ✓ Easy to set up (using Ansible)

Thank you for your attention!

If you are interested in the shown codes or have questions, feel free to use them and to contact me! `j.fritz@fz-juelich.de`

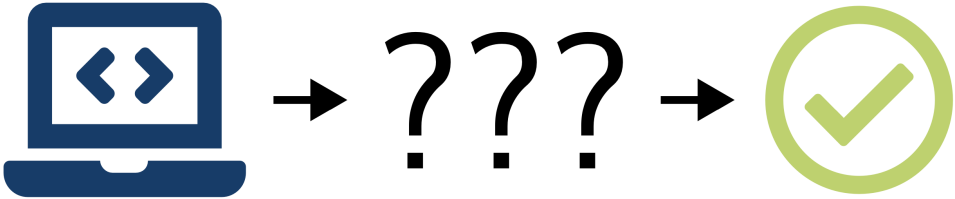
See Short-Talk CI in HPC: Working hard or hardly working?
by Ivo Kabadshow about how to get Gitlab-Runner up-and-running after Break in LaBRI

Thank you for your attention!

Link to GitHub2Lab:

https://github.com/jakob-fritz/github2lab_action

Link to Ansible for Gitlab-Runners:
https://jugit.fz-juelich.de/ATML-SE/cx/gitlab-runners-via-ansible



CONTINUOUS INTEGRATION FOR HPC

JLESC, March 23, 2023 | Jakob Fritz, Ivo Kabadshow, Robert Speck | Jülich Supercomputing Center, Forschungszentrum Jülich

JÜLICH Forschungszentrum