UCLouvain

Plateforme technologique de Calcul Intensif et Stockage de Masse

EURO BELGIUM

CI/CD with GitHub

Olivier Mattelaer

Program of this lecture

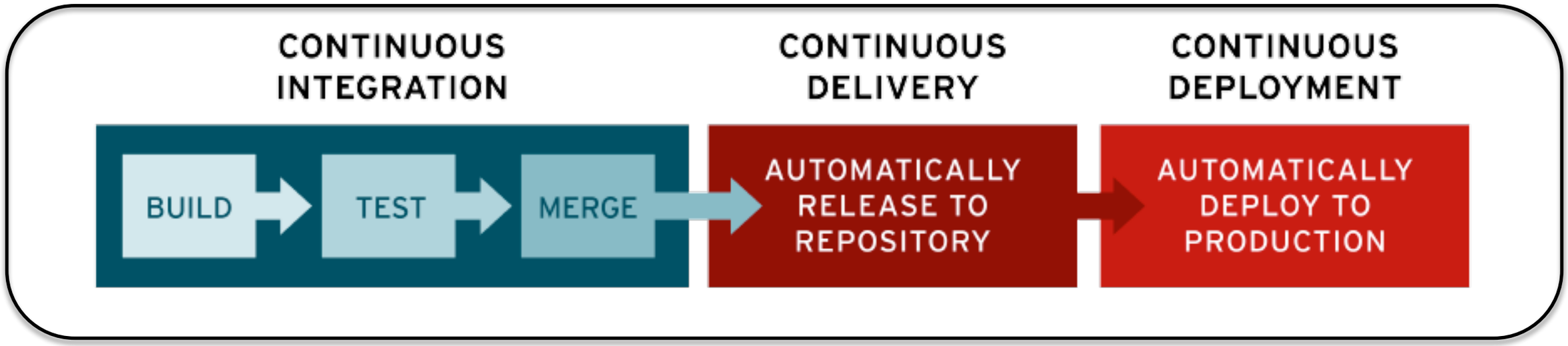
- What is CI/CD (GitHub actions)
- What is not CI/CD
- How to trigger workflow
- Runner (where to run workflow)
- Security consideration
- Running HPC workflow from GitHub action
- Other useful tricks

What is CI/CD

- CI: Continuous Integration
  - ➡ automation process for developers. Successful CI means new code changes to an app are regularly built, tested, and merged to a shared repository.

- CD: Continuous Delivery (or Deployement)
  - ➡ Continuous Delivery:
    - ➡ automatically bug tested and uploaded to a repository. Goal: ensure that it takes minimal effort to deploy new code
  - ➡ Continuous Deployment:
    - ➡ automatically releasing a developer's changes from the repository to production, where it is usable by customers.

What is CI/CD



- CI/CD is really a process, often visualised as a pipeline, that involves adding a high degree of ongoing automation and continuous monitoring to app development.

Goal / Assumptions

- Focus on Continuous integration
- Focus on the framework to run your workflow
- A lot of CI/CD will use the philosophy of "hardware as code"
  - ✦ Ansible/chef/vagrant/…
  - ✦ Will not be cover here (ask if interested)
- Main goal: How to run CI/CD on HPC cluster

- Assumptions:
  - ➡ Repository on GitHub (and therefore using git)
  - ➡ HPC cluster with slurm
  - ➡ No sudo access / no job on frontend

Git hook

- Git hook can run script before/after git command:
  - ➡ Locally
    - ✦ Pre-commit, pre-rebase, pre-push, post-merge
  - ➡ Server side
    - ✦ Need on premise server

- Some “basic” CI are possible at this level
  - formatting/ smart commit message / test
- Hook are not within the repo and can be bypassed: not a source of truth
- Local run: “work for me”

Hook advantages

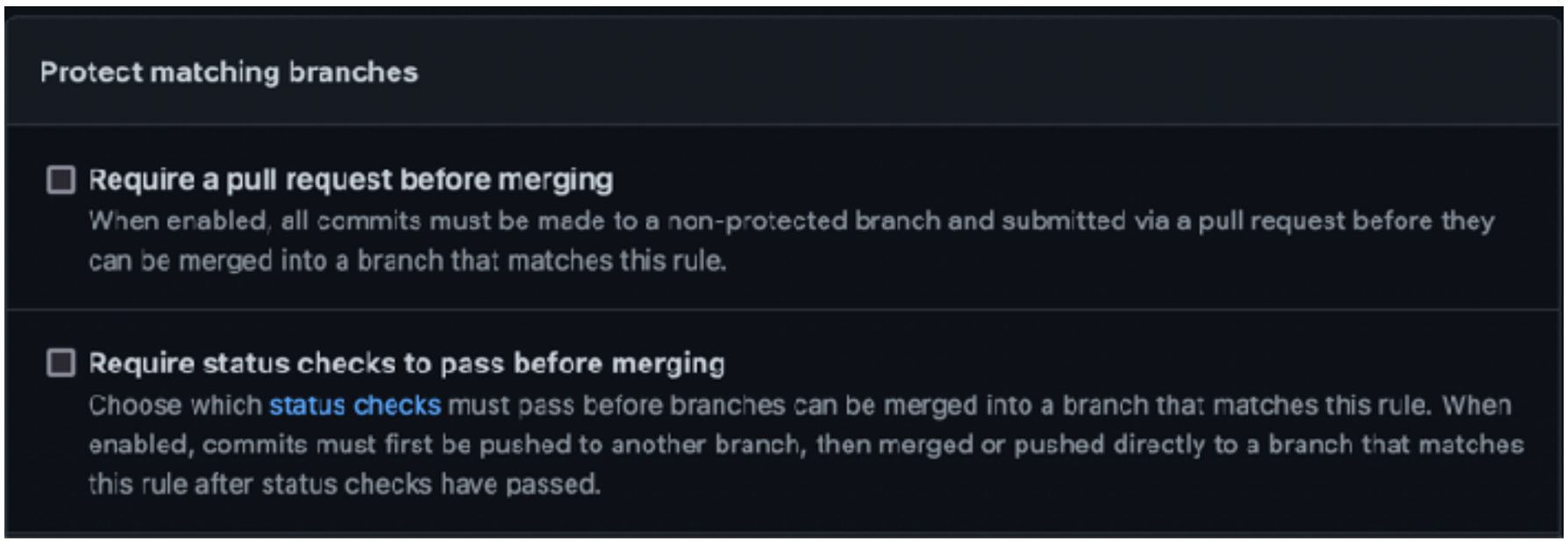
- On server side you can enforce policy by refusing commit

Hook advantages

- On server side you can enforce policy by refusing commit

Github solution

- Forbid direct commit on the master
- Activate GitHub actions for merge request
- Forbid merge that does not pass the CI



GitHub actions

When to run

- After a push
  - ➡ After merge
- Merge-request
- Timer based
  - ➡ Nightly build
- Manual trigger

Where to run

- On the cloud
  - Pre-image runner
  - Limited amount for free
- On premisse
  - Openstack
  - Demo later
  - Via deamon

- You can have multiple runner for different (set of) tasks

Demo #1:

Running hello-world on GitHub machine

Examples on https://github.com/oliviermattelaer/cicdtuto

Naming convention

Workflow

- A series of job
- Related to trigger

Job

- A series of actions
- One job is running on one runner

Actions/steps

- One job is running on one runner
- Same environment per job

Runner

- Code waiting for job
- Execute one job at a time
- Have name/tag

Naming convention

Workflow

- A series of job
- Related

Can be reused:
Reusable workflow

Job

- A series of actions
- One job is running on one runner

Actions/steps

- One job is running on one runner
- Same environment per job

Runner

- Code waiting for job
- Execute one job at a time
- Have name/tag

Naming convention

Workflow

- A series of job
- Related

Can be reused:
Reusable workflow

Job

- A series of actions
- One job is running on one runner

Actions/steps

- One job is runni
one runne
- S nt

Can be reused:
Composite action

Runner

- Code waiting for job
- Execute one job at a time
- Have name/tag

Demo #2:

SSH to a cluster

Examples on https://github.com/oliviermattelaer/cicdtuto

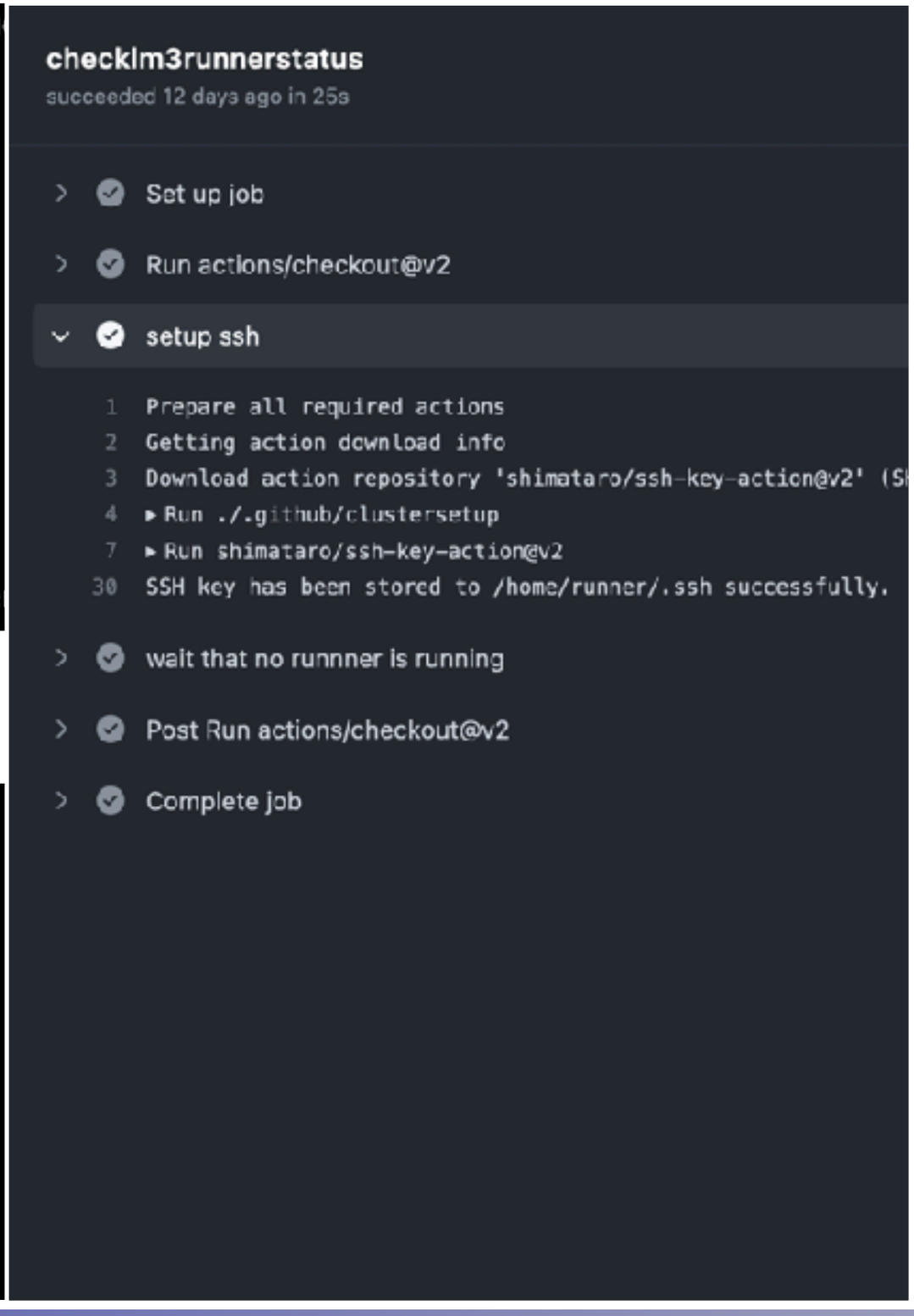
Secret/security

- A security risk does exists.
  - Restriction on who can trigger a runner to execute
    - GitHub at bit too permissive?
  - Use secret and/or local runner state
  - Isolate the runner
    - virtual machine
    - separate login/quota/…
  - Dedicated CODEOWNER for .github directory
  - Limit direct shell usage and avoid shell injection

Composite action

- Workflow :

```yaml
# A workflow run is made up of one or more jobs that can run sequ
jobs:
    # This workflow contains a single job called "build"
    checklm3runnerstatus:
    # The type of runner that the job will run on
      runs-on: ubuntu-latest
      steps:
        - uses: actions/checkout@v2
        - name: setup ssh
          uses: ./.github/clustersetup
          with:
           sshkey: ${{ secrets.CECI_KEY }}
        - name: wait that no runnner is running
          run: ssh lemaitre3 srun --dependency=singleton -J runne
```



- action.yml :

```yaml
description: "setup openssh for cluster connection"

inputs:
  sshkey:
    description: "ssh key to connect to cluster"
    required: true
runs:
  using: "composite"
  steps:
  - name: Install SSH key
    uses: shimataro/ssh-key-action@v2
    with:
      key: ${{ inputs.sshkey }}
      name: id_rsa.ceci # optional
```

Composite action

- A set of task defined as a function
- Will be part of a job
  - ➡ Run within a Single runner
  - ➡ Will be seen as a single action/step
- Can be external (github actions)
  - ➡ Prefer to use hash to tag version
- Can be local
  - ➡ Each one needs its own directory
  - ➡ Need to clone the repo first
- Secret environment are not accessible
  - ➡ Need to use parameter

Demo #3:

Setup your own runner

Examples on https://github.com/oliviermattelaer/cicdtuto

Setup a runner (1/2)

Setup a runner (2/4)

Runners / Create self-hosted runner

Adding a self-hosted runner requires that you download, configure, and execute the GitHub Actions Runner. By downloading and configuring the GitHub Actions Runner, you agree to the GitHub Terms of Service or GitHub Corporate Terms of Service, as applicable.

Runner image

○ macOS | ◉ Linux | ○ Windows

Architecture

x64

Download

```
# Create a folder
$ mkdir actions-runner && cd actions-runner
# Download the latest runner package
$ curl -o actions-runner-linux-x64-2.286.1.tar.gz -L https://github.com/actions/runner/releases
/download/v2.286.1/actions-runner-linux-x64-2.286.1.tar.gz
# Optional: Validate the hash
$ echo "7b1509c353ea4e6561b2ed2e916dcbf2a0d8ce21172edd9f8c846a20b6462cd6  actions-runner-linux-
x64-2.286.1.tar.gz" | shasum -a 256 -c
# Extract the installer
$ tar xzf ./actions-runner-linux-x64-2.286.1.tar.gz
```

Configure

```
# Create the runner and start the configuration experience
```

Setup a runner (3/4)

Configure

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/oliviermattelaer/mg5amc_test --token
AH[redacted]
# Last s
$ ./run.
```

```
[omatt@nic5-login1 action_runner]$ ./config.sh --url https://github.com/oliviermattelaer/mg5amc_test --token AH
--------------------------------------------------------------------------------
|        ____ _ _   _   _       _          _       _   _                      |
|       / ___(_) |_| | | |_   _| |__      / \   ___| |_(_) ___  _ __  ___      |
|       | |  _| | __| |_| | | | | '_ \    / _ \ / __| __| |/ _ \| '_ \/ __|     |
|       | |_| | | |_|  _  | |_| | |_) |  / ___ \ (__| |_| | (_) | | | \__ \     |
|        \____|_|\__|_| |_|\__,_|_.__/  /_/   \_\___|\__|_|\___/|_| |_|___/     |
|                                                                              |
|                       Self hosted runner registration                        |
|                                                                              |
--------------------------------------------------------------------------------

# Authentication


√ Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]

Enter the name of runner: [press Enter for nic5-login1]

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip] nic5

√ Runner successfully added
√ Runner connection is good

# Runner settings

Enter name of work folder: [press Enter for _work]

√ Settings Saved.

[omatt@nic5-login1 action_runner]$
```

Setup a runner (4/4)

Configure

```
# Create the runner and start the configuration experience
$ ./config.sh --url https://github.com/oliviermattelaer/mg5amc_test --token
AH[redacted]
# Last step, run it!
$ ./run.sh
```

```
[omatt@nic5-login1 action_runner]$ ./run.sh

√ Connected to GitHub

Current runner version: '2.287.1'
2022-02-07 13:27:30Z: Listening for Jobs
```

Running on HPC facility

Consideration

- Scheduler support
  - ➡ No daemon support
- Portability
  - ➡ Very difficult to be 100% portable
    - ✦ Name of the queue, type of the hardware,…
- User identification
  - ➡ Who runs?

Gitlab solution

Custom executor

- ECP continuous integration
- Used and develop for spack (code installation on HPC cluster)
- Support of nersc / alcf / olcf
  - ➡ Part of the exascale doe project
- Support of various scheduler
  - ➡ But portability is still tricky with the need of many site customisation
- Solution needs support from local admin (due to a matching policy between local user and gitlab user)

Simple solution

Tmux/screen

- Setup the runner on the frontend on a tmux/screen session
  - ➡ Cluster policy?
  - ➡ No local heavy job

Ssh + srun

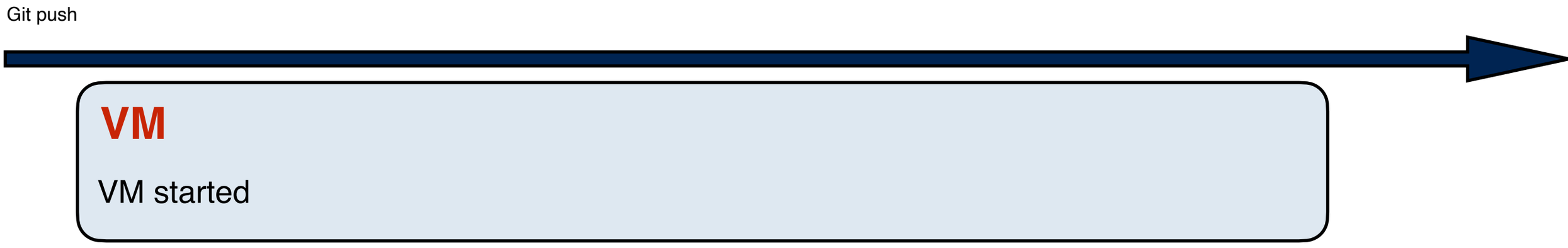
- Run from a virtual machine
  - Execute srun command via ssh

Issue:

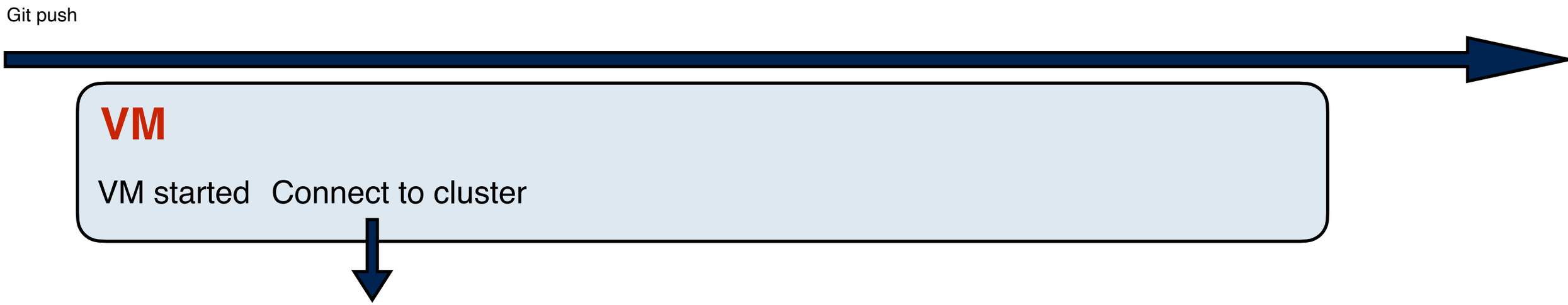
- Each “srun” needs to wait for the allocation
- Issue with frontend reboot/…

Solution: Runnner on demand

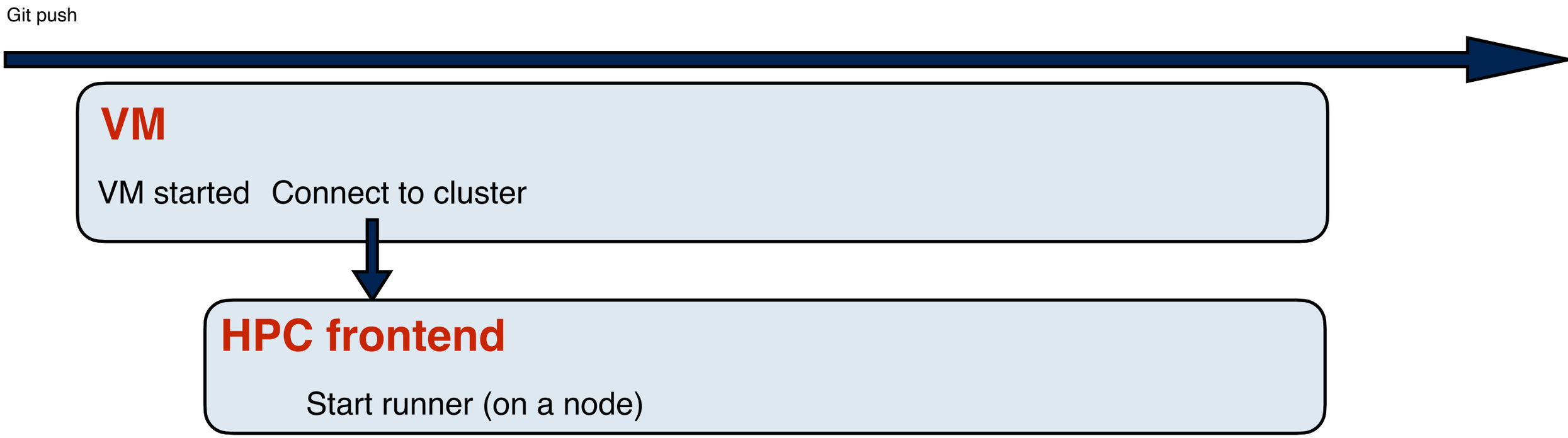
Solution: Runnner on demand



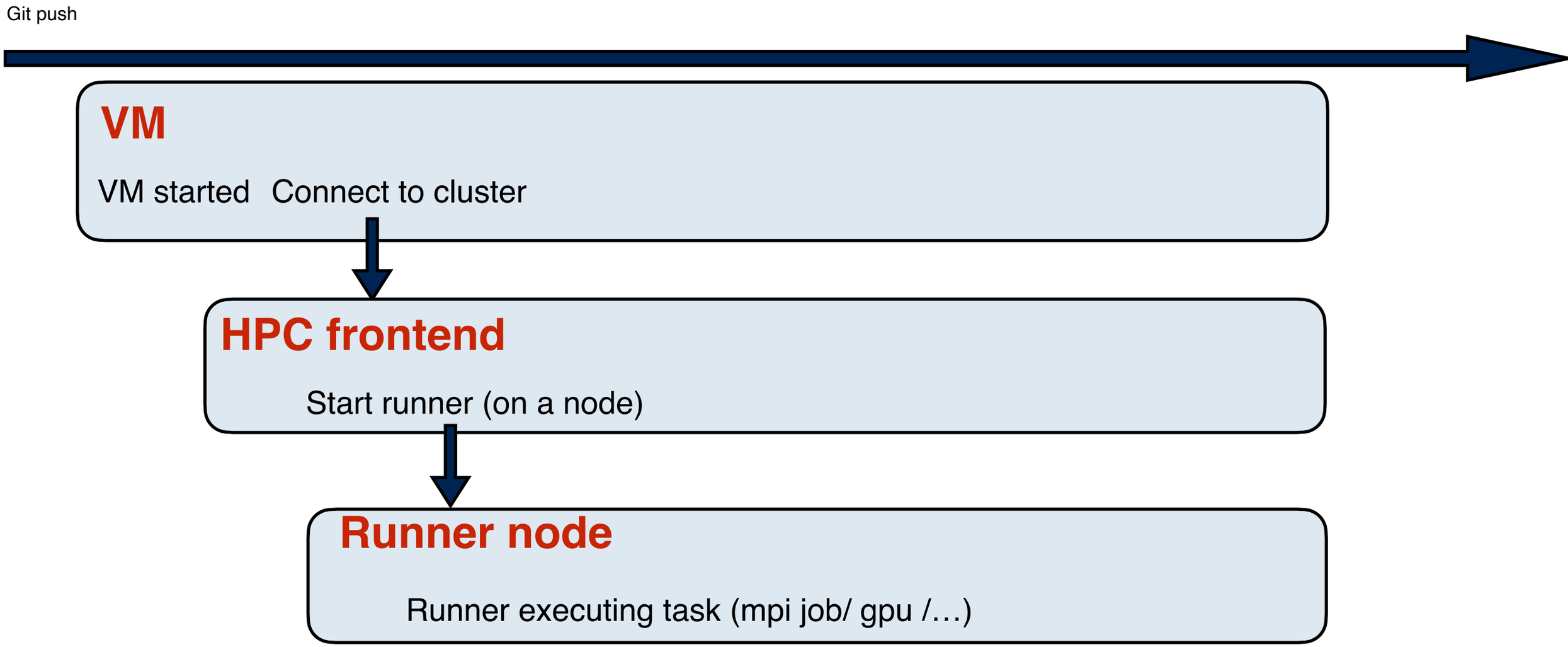
Solution: Runnner on demand



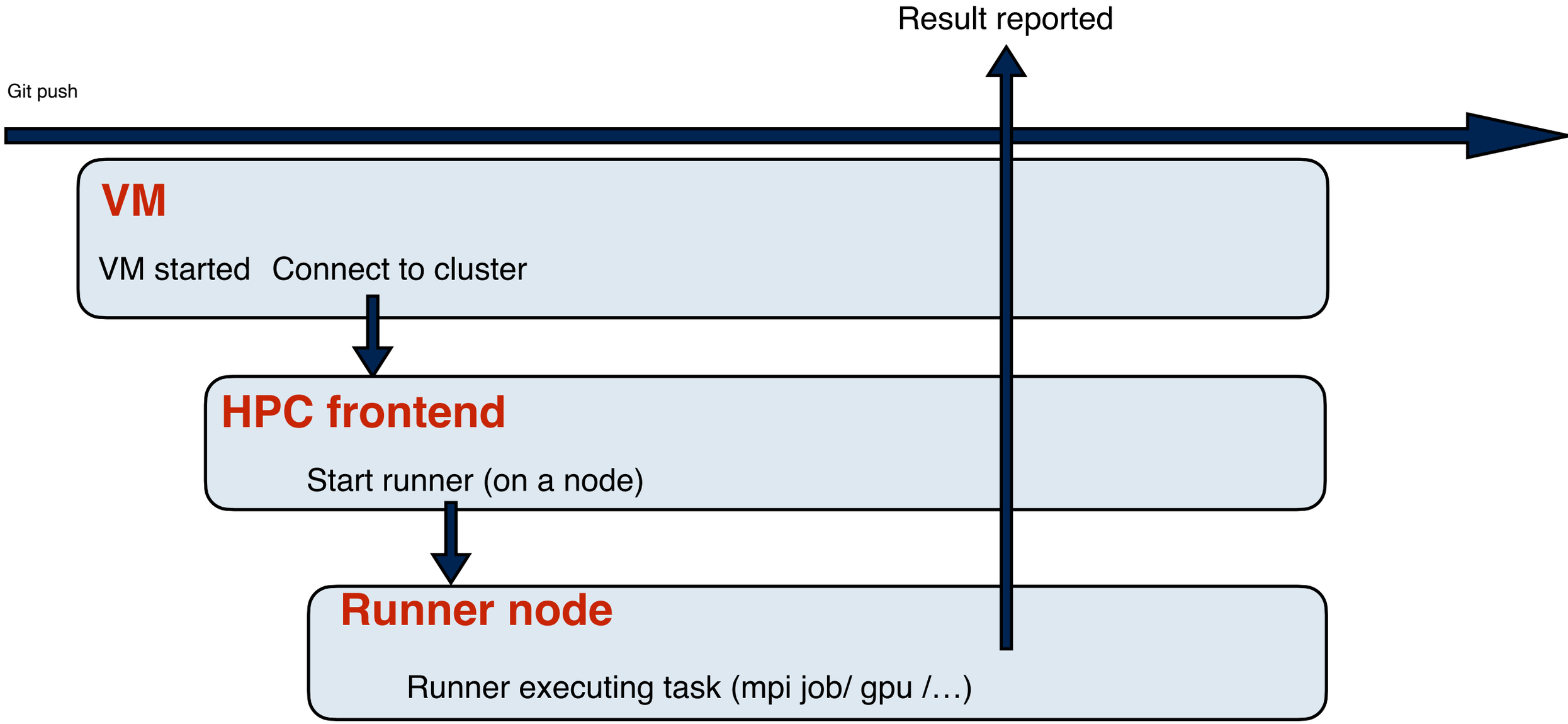
Solution: Runnner on demand



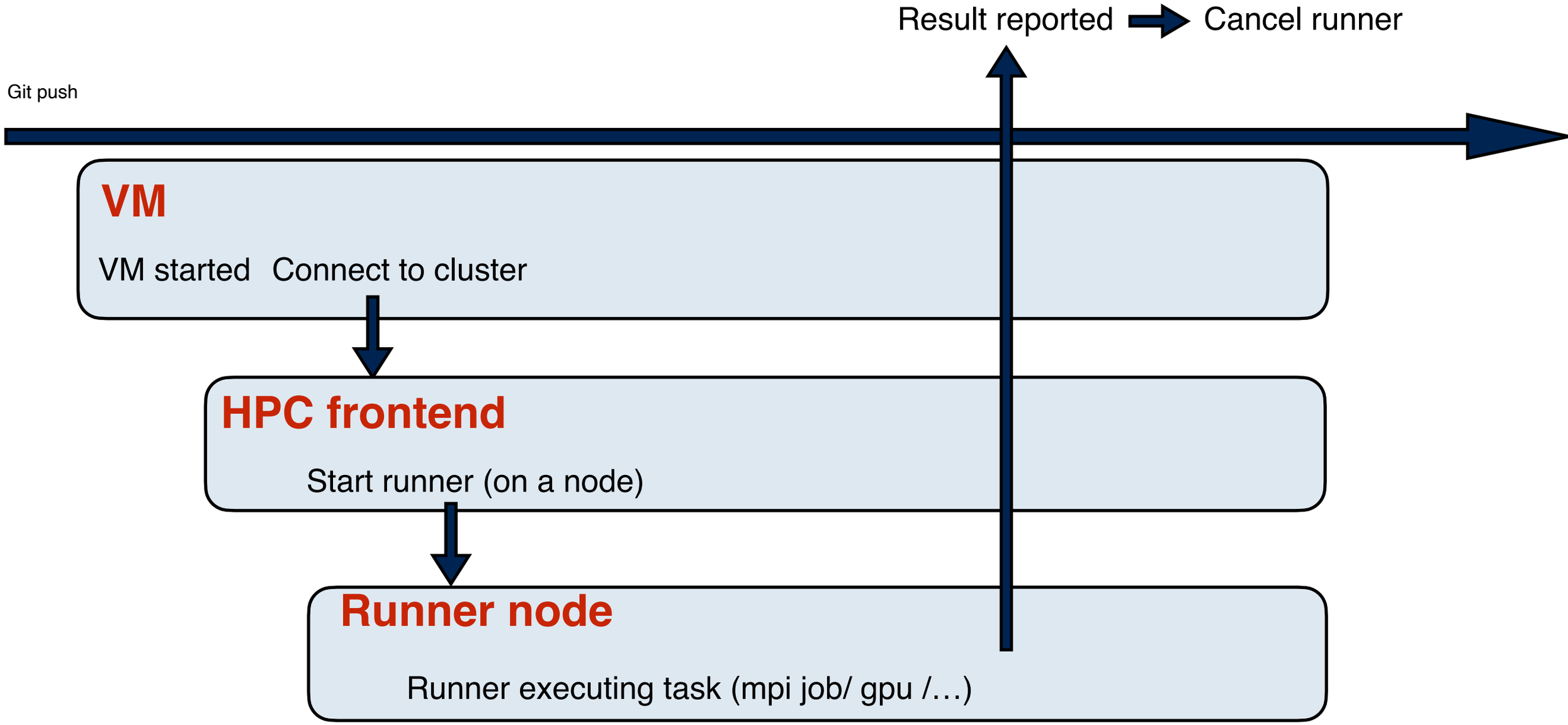
Solution: Runnner on demand



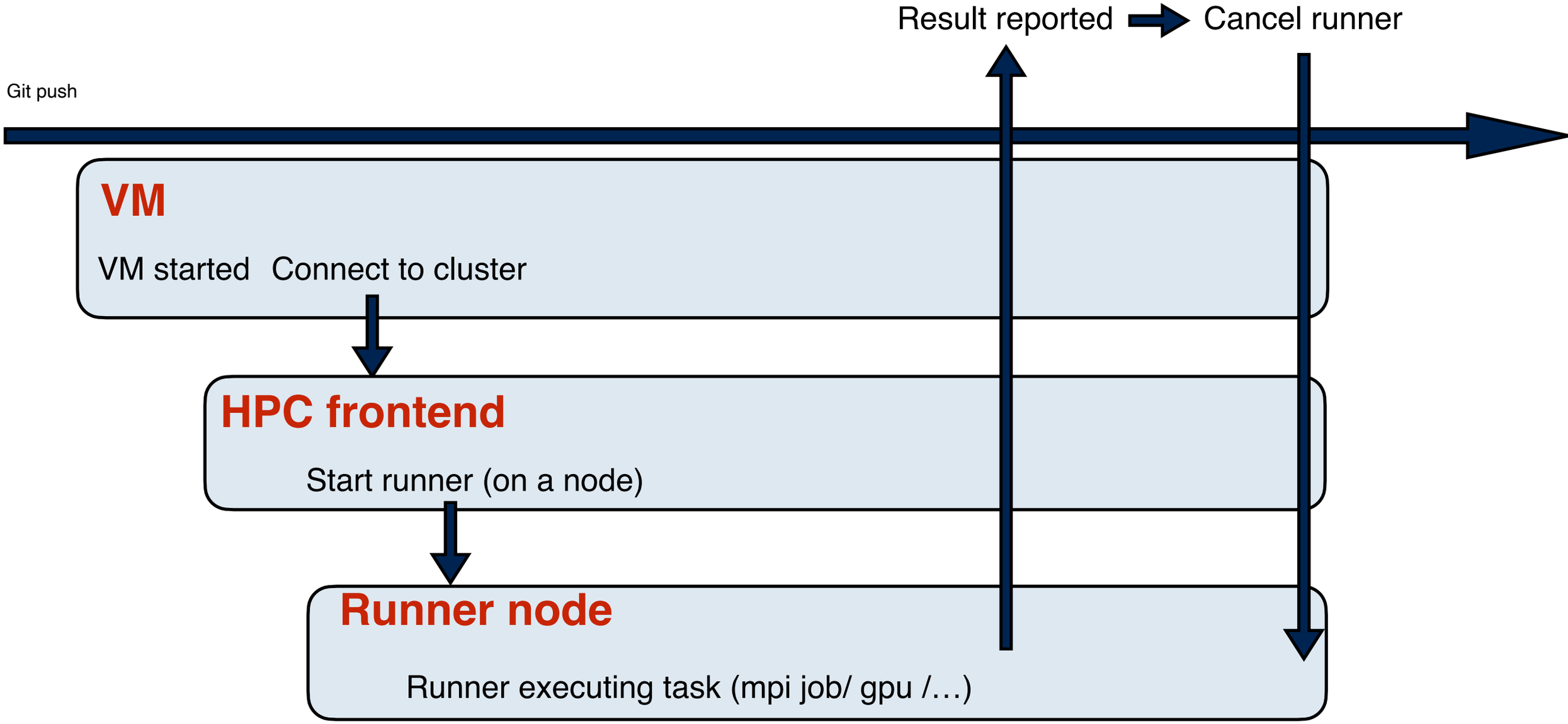
Solution: Runnner on demand



Solution: Runnner on demand



Solution: Runnner on demand



Job dependency

You can chain job

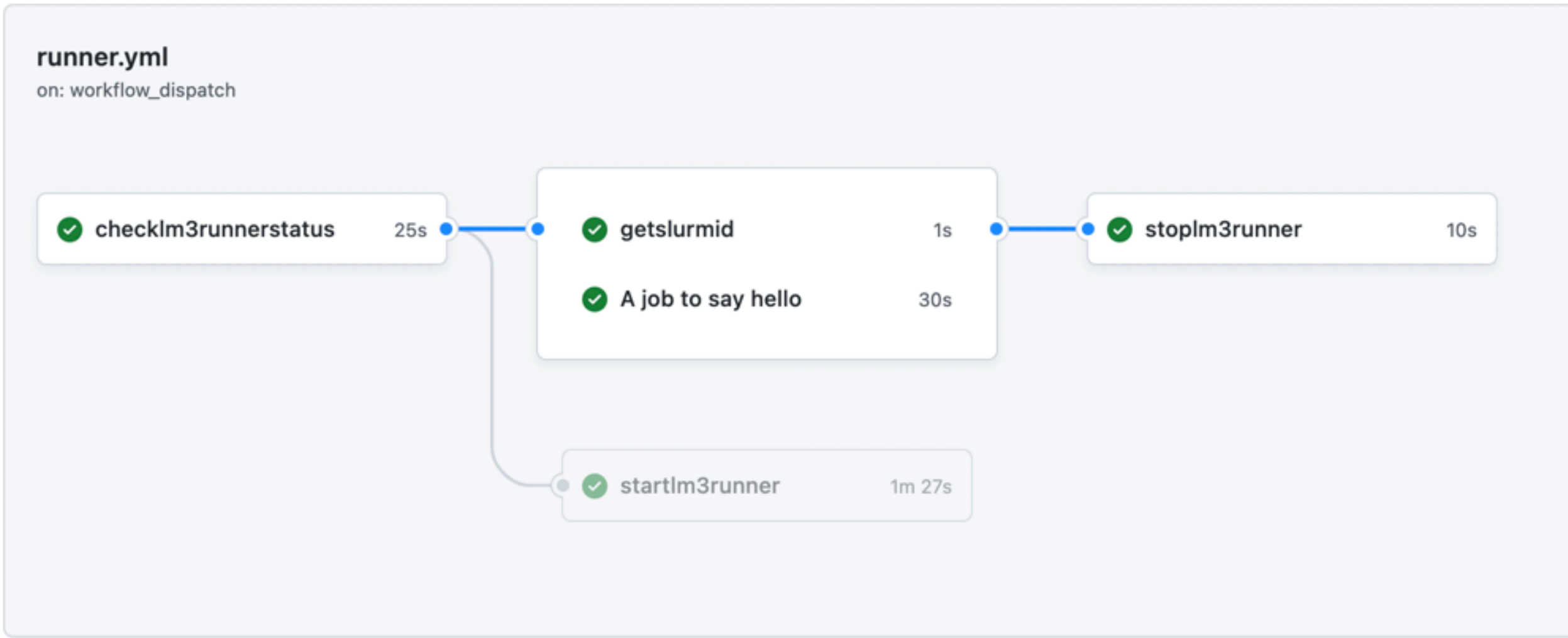
```yaml
getslurmid:
  needs: checklm3runnerstatus
  runs-on: lemaitre3
  outputs:
    output1: ${{ steps.step1.outputs.test }}
  steps:
   - id: step1
     run: echo "::set-output name=test::${SLURM_JOB_ID}"
```

```yaml
stoplm3runner:
  runs-on: ubuntu-latest
  needs: [getslurmid,actualwork]
  if: always()
  steps:
   - run: echo ${{needs.getslurmid.outputs.output1}}
   - uses: actions/checkout@v2
   - name: setup ssh
     uses: ./.github/clustersetup
     with:
       sshkey: ${{ secrets.CECI_KEY }}
   - name: stop runner
     run: ssh lemaitre3 scancel ${{needs.getslurmid.outputs.output1}}
```

Full workflow

Trick: avoid race condition

- The runner is started with --dependency=singleton:runner
- A first job is checking that no runner is running



Demo #4:

Submitting workflow on HPC cluster

Examples on https://github.com/oliviermattelaer/cicdtuto

Useful capabilities

- Artefacts:
  - ➡ Allow to retrieve file
  - ➡ Pass them further away in the pipeline
- Reusable workflow
  - ➡ Still not fully mature
    - ✦ Recent progress in the correct direction
- Expression
  - ➡ ${{ NAME }}
  - ➡ Advance information
    - ✦ Commit message:
      - ${{ github.event.head_commit.message }}
- Many others (matrix strategy/…)

Conclusion

CI/CD is a philosophy

- Need the content (i.e. the test suite/workflow/…)
- Need Leadership / Diplomacy
- Will grow/strengthen over time

CI/CD is easy

- Ressource are easily accessible (also on HPC)
- Good documentation / user support

CI/CD is efficient

- Speed up your development
- More stable code / bug finding