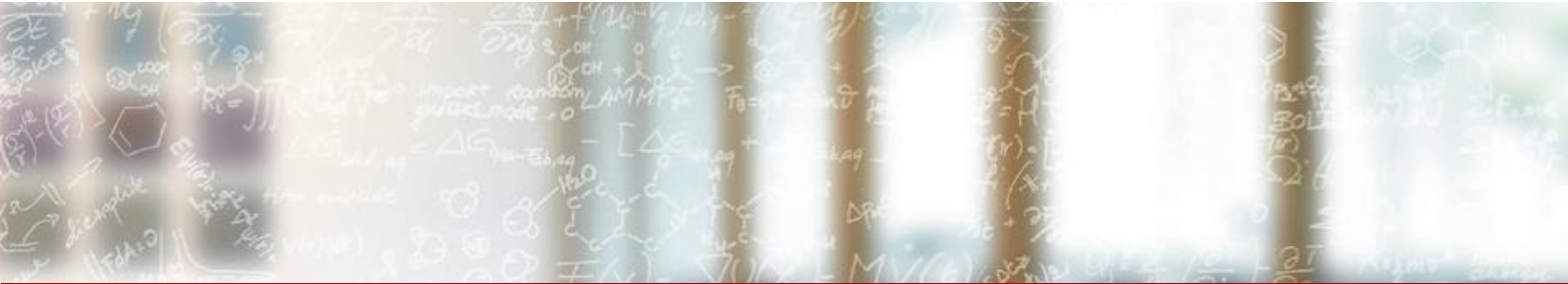




**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



# Jenkins service for CI/CD

CSCS User Lab Day – Meet the Swiss National Supercomputing Center

Manitaras Theofilos-Ioannis, CSCS

September 2, 2022

# Outline

- Continuous Integration & Jenkins in a Nutshell
- Accessing & using the CSCS CI Jenkins instance
- Best practices for Jenkins at CSCS
- Triggering Jenkins Jobs
- Conclusions



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich

# Continuous Integration & Jenkins in a Nutshell

---

# Continuous Integration (CI) & Jenkins



## Jenkins

- CI is a software development practice where code changes are continuously integrated in a shared code basis
- Each addition/change is tested via an automated pipeline
- Adoption of CI allows catching errors sooner
- Code conflicts are reduced due to the "continuous" nature of the practice
- Jenkins is an open source automation server which automates the non-human part of the software development process
- It offers a large number of plugins which enhance its capabilities.
- Compatible with all the popular software repositories, e.g GitHub, GitLab, BitBucket etc.

# Typical CI Workflow

1. Each developer owns a separate fork of the project
2. A change to the code base is submitted via a pull/merge request to the main/develop repository branch
3. An automated CI pipeline is triggered to ensure that the introduced change passes the test suite
4. The result of the CI is reported, notifying the developers if needed
5. The proposed change is reviewed and polished
6. The change is accepted and merged to the main branch
7. The software is redeployed to production (optional)



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich

# Accessing & using the CSCS CI Jenkins instance

---

# Gaining access to the CSCS CI

1. The PI of a project requests access to the service by opening an issue at [help@cscs.ch](mailto:help@cscs.ch)
2. A Jenkins folder with the corresponding name is created
3. A new user belonging to the project is created
4. A Jenkins agent(node) is associated to the folder
5. The pipelines run on the target system (Daint) on behalf of the above user


## Logging to the Jenkins Web interface (1/2)

- The CSCS Jenkins instance is not available in public Internet
- Local port forwarding has to be performed via ssh
- The end user has to forward a local port to **lisone.cscs.ch:443** via **ela.cscs.ch**
- For Linux/Mac users the forwarding can be performed from the shell:  
`ssh -L 7000:lisone.cscs.ch:443 ela.cscs.ch`
- The Jenkins web interface can now be accessed using a web browser through **https://ci.cscs.ch:7000/**
- The above local port (7000) is chosen by the end user



# Logging to the Jenkins Web interface (2/2)

Use your CSCS credentials to login



**Welcome to Jenkins!**

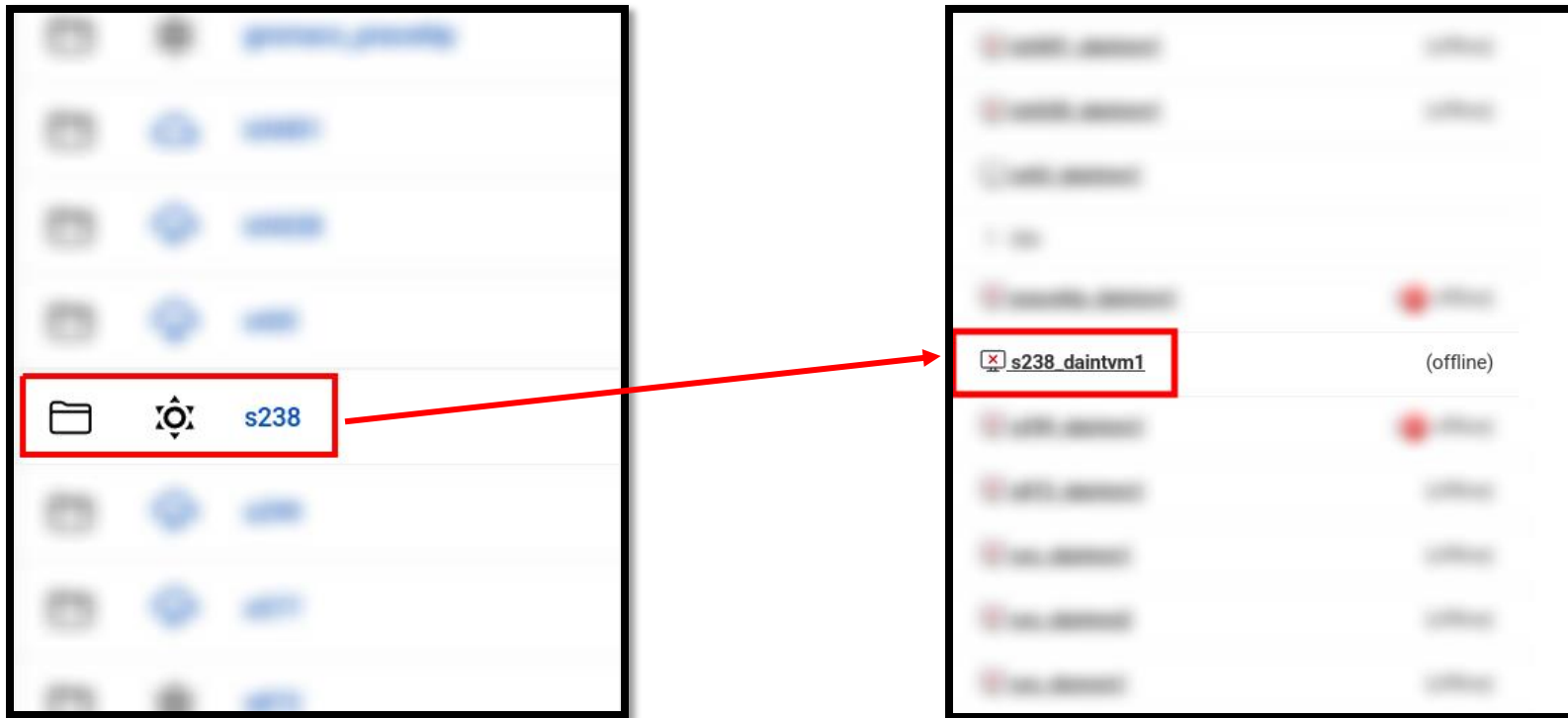
Username

Password

☐ Keep me signed in

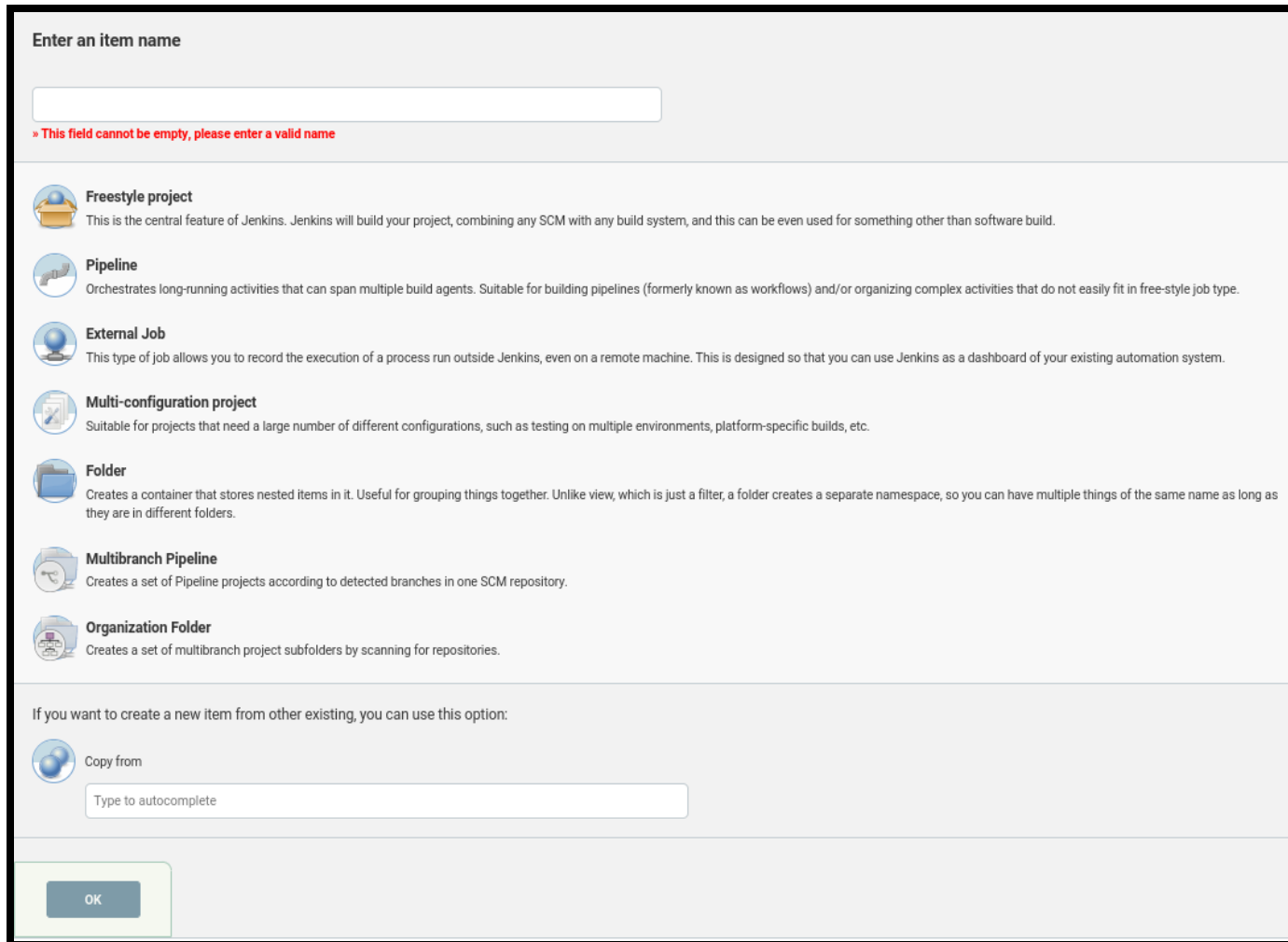
# Dedicated Folde – Agents per Project

Dedicated folder and Jenkins agent



# Creating a new Jenkins project

Creating a new project is straightforward by selecting **New Item**:



The screenshot shows the Jenkins 'New Item' page. At the top, there is a section titled 'Enter an item name' with a text input field. Below the field, a red error message states: '» This field cannot be empty, please enter a valid name'. The main area lists several project types, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- External Job**: This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

Below the list, there is a section titled 'If you want to create a new item from other existing, you can use this option:' with a 'Copy from' icon and a text input field labeled 'Type to autocomplete'. At the bottom left, there is an 'OK' button.

# Pipeline as Code – The Jenkinsfile

The Jenkinsfile is at the core of the Jenkins pipeline-as-code approach serving a dual purpose:

1. Describes via a domain specific language (DSL) the actions performed when a build is triggered
2. Existence of a Jenkinsfile signifies that the project is using Jenkins for its CI

Jenkinsfiles come in two alternative flavors:

1. **Scripted Pipeline:** a Groovy based script describes the actions performed by the Jenkins build
2. **Declarative Pipeline:** an easier to follow declarative syntax based on simple constructs specifying Jenkins actions

# Structure of a Jenkinsfile

The structure of a Jenkinsfile is different, depending on the syntax flavor:

## Scripted

```
stage("Stage 1") {  
    node("mynode1") {  
        <step 1>  
        .  
        .  
        .  
        <step n>  
    }  
}  
  
stage("Stage N") {  
    node("mynode2") {  
        <step 1>  
        .  
        .  
        .  
        <step n>  
    }  
}
```

## Declarative

```
pipeline {  
    stages {  
        stage("Stage 1") {  
            agent { node { label 'mynode1' } }  
            steps {  
                <step 1>  
                .  
                .  
                <step n>  
            }  
        }  
        stage("Stage 2") {  
            agent { node { label 'mynode2' } }  
            steps {  
                <step 1>  
                .  
                .  
                <step n>  
            }  
        }  
    }  
}
```

# Nodes/Agents

- The Jenkins **Built-In-Node** is the system running the jenkins instance and is not intended to run any actual jobs
- A **Node** in Jenkins terms is any system capable of executing jobs, e.g a remote machine, a VM, a container.
- An **Agent** is the declarative pipeline equivalent of a node other than the build-in node, executing Jenkins Jobs
- In the CSCS setup, the nodes/agents consist of lightweight VMs which allow for batch job submission but have limited functionality otherwise



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich

# Best practices for Jenkins at CSCS

---

# Best Practices

- Adopt the pipeline-as code modern approach of Jenkins 2 including the **Jenkinsfile** in your git repository
- Use the **--wait** option to submit batch jobs using **sbatch**:  
`sbatch -wait <batch_script>`
- For single node jobs use the **cscsci** partition offering higher priority and is suitable for ci
- Copy the build output/error on **SCRATCH** to be able to access via your user
- Make use of artifacts to store output/error and produced binaries

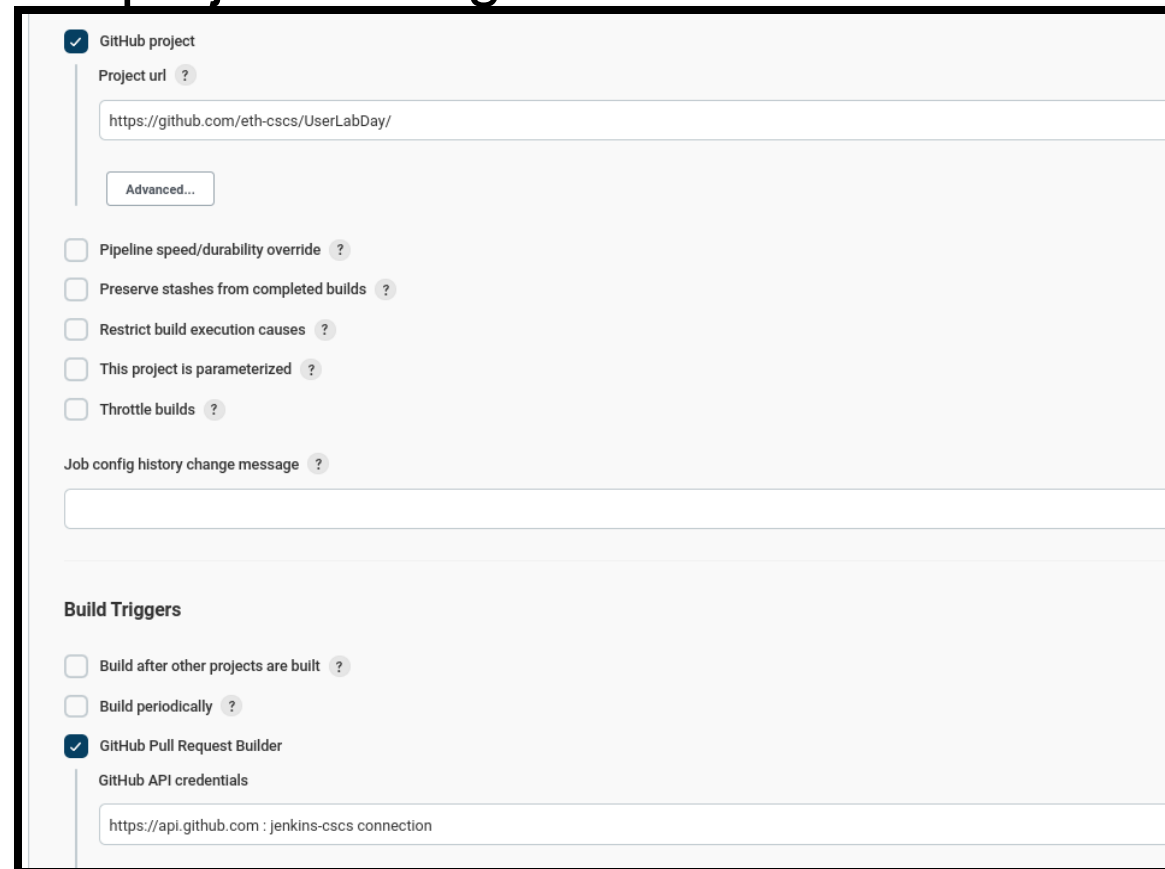


# Triggering Jenkins Jobs

---

# Enabling the ghprb for your project (1/2)

1. Invite the **jenkins-cscs** GitHub user in your project, setup by CSCS with both read & write privileges
2. Enable ghprb on your project configuration:



The screenshot shows the Jenkins project configuration page for a GitHub project. The 'GitHub project' section is checked. The 'Project url' is set to 'https://github.com/eth-cscs/UserLabDay/'. Below this is an 'Advanced...' button. A list of options follows: 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'Restrict build execution causes', 'This project is parameterized', and 'Throttle builds', all of which are unchecked. There is a 'Job config history change message' field. The 'Build Triggers' section includes 'Build after other projects are built', 'Build periodically', and 'GitHub Pull Request Builder', which is checked. Below the 'GitHub Pull Request Builder' is the 'GitHub API credentials' field, which contains the text 'https://api.github.com : jenkins-cscs connection'.

# New PR triggers the build

The screenshot shows a GitHub interface for the repository **eth-cscs / UserLabDay**. A new pull request titled **Add hostname command #4** is open, created by **teojgo**. The PR is currently open, showing 1 commit and 1 file changed. The status bar indicates **All checks have passed** with 1 successful check, including **CSCS CI** and **This branch has no conflicts with the base branch**. The PR is currently open, showing 1 commit and 1 file changed. The status bar indicates **All checks have passed** with 1 successful check, including **CSCS CI** and **This branch has no conflicts with the base branch**.

# Retriggering a build

## Add hostname command #4

[Open](#) teojgo wants to merge 1 commit into `master` from `feature/add_hostname`

Conversation 0

Commits 1

Checks 0

Files changed 1

+1 -0

teojgo commented 6 days ago

Member + 👤 ...

No description provided.

teojgo

Add hostname command

22ffbd5

teojgo self-assigned this 6 days ago

teojgo commented 16 seconds ago

Member + 👤 ...

retest this please

Add more commits by pushing to the `feature/add_hostname` branch on `eth-cscs/UserLabDay`.

Some checks haven't completed yet

1 pending check

CSCS CI

Pending — Build started for merge commit.

Details

✓

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

or view [command line instructions](#).

Reviewers

No reviews

Assignees

teojgo

Labels

None yet

Projects

None yet

Milestone

No milestone


Notifications

Unsubscribe

You're receiving notifications because you were assigned.

1 participant

teojgo

 CSCS

| 20

**ETH** zürich

# PR from non-whitelisted user

The screenshot displays a GitHub Pull Request (PR) interface. The main content area shows a list of comments:

- rsarm** (Member) commented 5 minutes ago: "No description provided."
- jenkins-cscs** (Collaborator) commented 4 minutes ago: "Can one of the admins verify this patch?"
- teoigo** (Member) commented 37 seconds ago: "test this please"

Below the comments, a message states: "Add more commits by pushing to the **test-branch** branch on **eth-cscs/UserLabDay**."

A summary section titled "Some checks haven't completed yet" (1 pending check) includes:

- CSCS CI** (Pending) — Build triggered for merge commit.
- This branch has no conflicts with the base branch** (Merging can be performed automatically).

At the bottom of the summary, there is a button "Merge pull request" and a link "or view [command line instructions](#)."

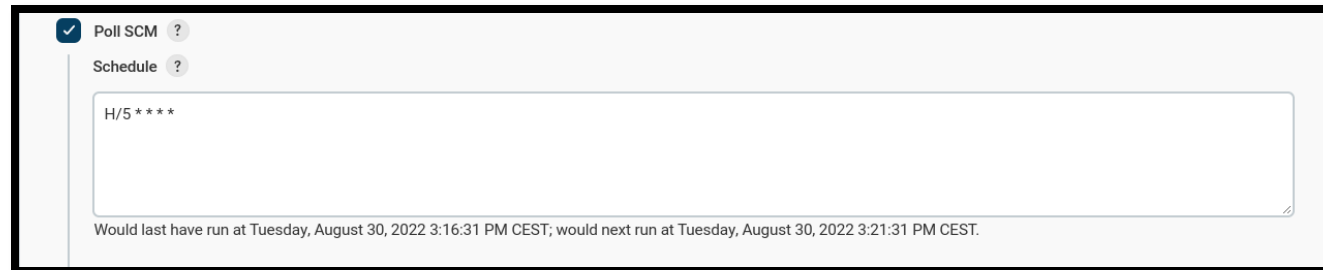
The right sidebar contains several sections:

- Reviewers**: No reviews
- Assignees**: No one—assign yourself
- Labels**: None yet
- Projects**: None yet
- Milestone**: No milestone
- Notifications**: You're receiving notifications because you commented. (Unsubscribe button)
- 3 participants**: (User avatars)
- Lock conversation** (Lock icon)

The bottom of the interface shows a "Write" button, a "Preview" button, and a rich text editor toolbar with icons for bold, italic, quote, code, link, list, and other formatting options.

# Polling a remote repository

- An alternative to ghprb since the CSCS Jenkins is polling the remote repository for changes
- The option can be enabled by selecting **Poll SCM** under Build Triggers:



The screenshot shows the Jenkins configuration page for Build Triggers. The 'Poll SCM' option is checked, and the 'Schedule' is set to 'H/5 \*\*\*'. Below the schedule field, a message indicates the last and next run times: 'Would last have run at Tuesday, August 30, 2022 3:16:31 PM CEST; would next run at Tuesday, August 30, 2022 3:21:31 PM CEST.'

# Conclusions

---

# Conclusions

- The Jenkins CSCS CI service can be used to test software on the actual HPC systems
- Access to the service is requested by a PI of a project
- Several best practices must be followed to effectively incorporate the CI on your project
- Ghprb and Polling are two of the proposed ways to trigger pipelines based on PRs/MRs



# Additional Resources

## Online Resources:

- ❑ [CSCS CI documentation](#)
- ❑ [Jenkins User Documentation](#)
- ❑ [Jenkins Handbook](#)

## Books:

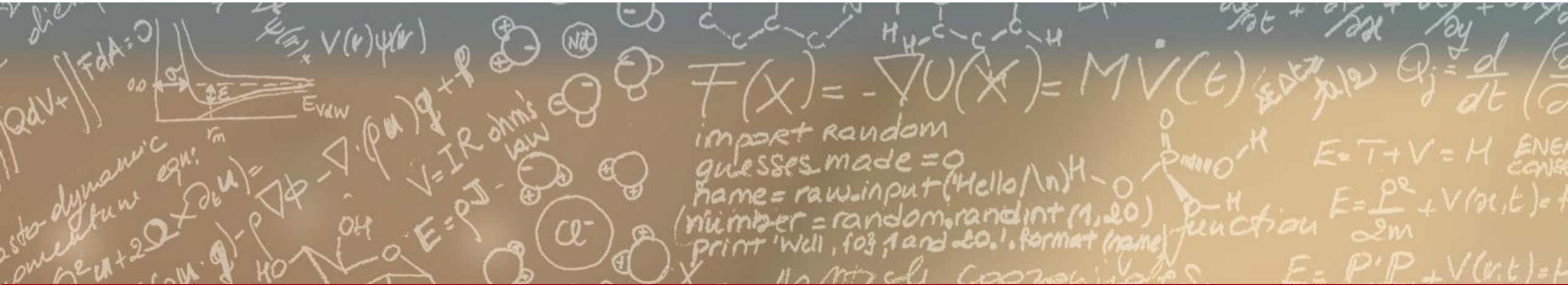
- ❑ [Jenkins 2 Up and Running](#)
- [CI/CD Pipeline Using Jenkins Unleashed](#)
- [Jenkins Administrator's Guide](#)



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



**Thank you for your attention.**