

Continuous Integration and continuous delivery (CI/CD)

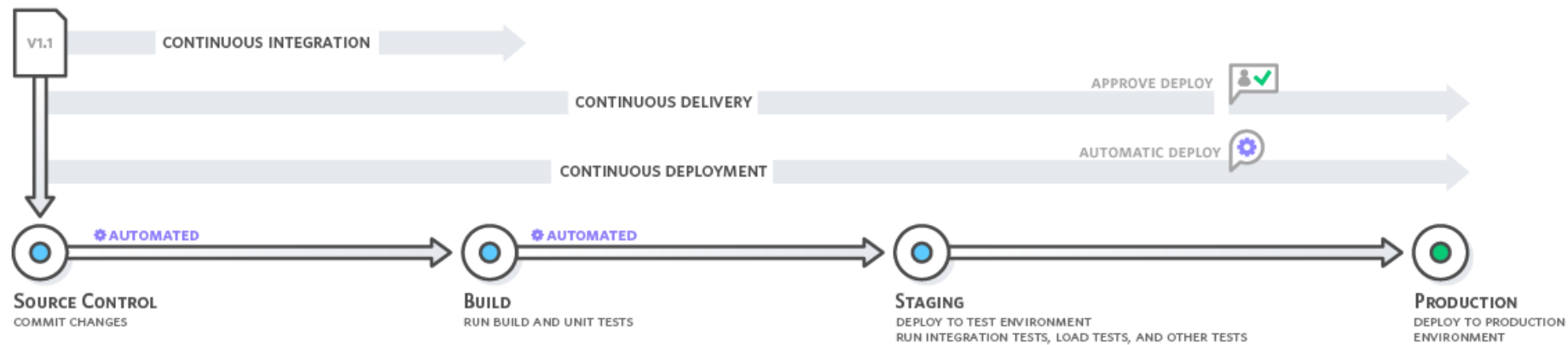
Continuous integration

- Integrate changes in the main codebase as often as possible
 - Rely on automation to guarantee proper checks are executed upon integration. Typically
 - Commits or pull requests trigger
 - Compilation/Build
 - Unit and integration test
 - Code quality check
 - Merge in the main repository

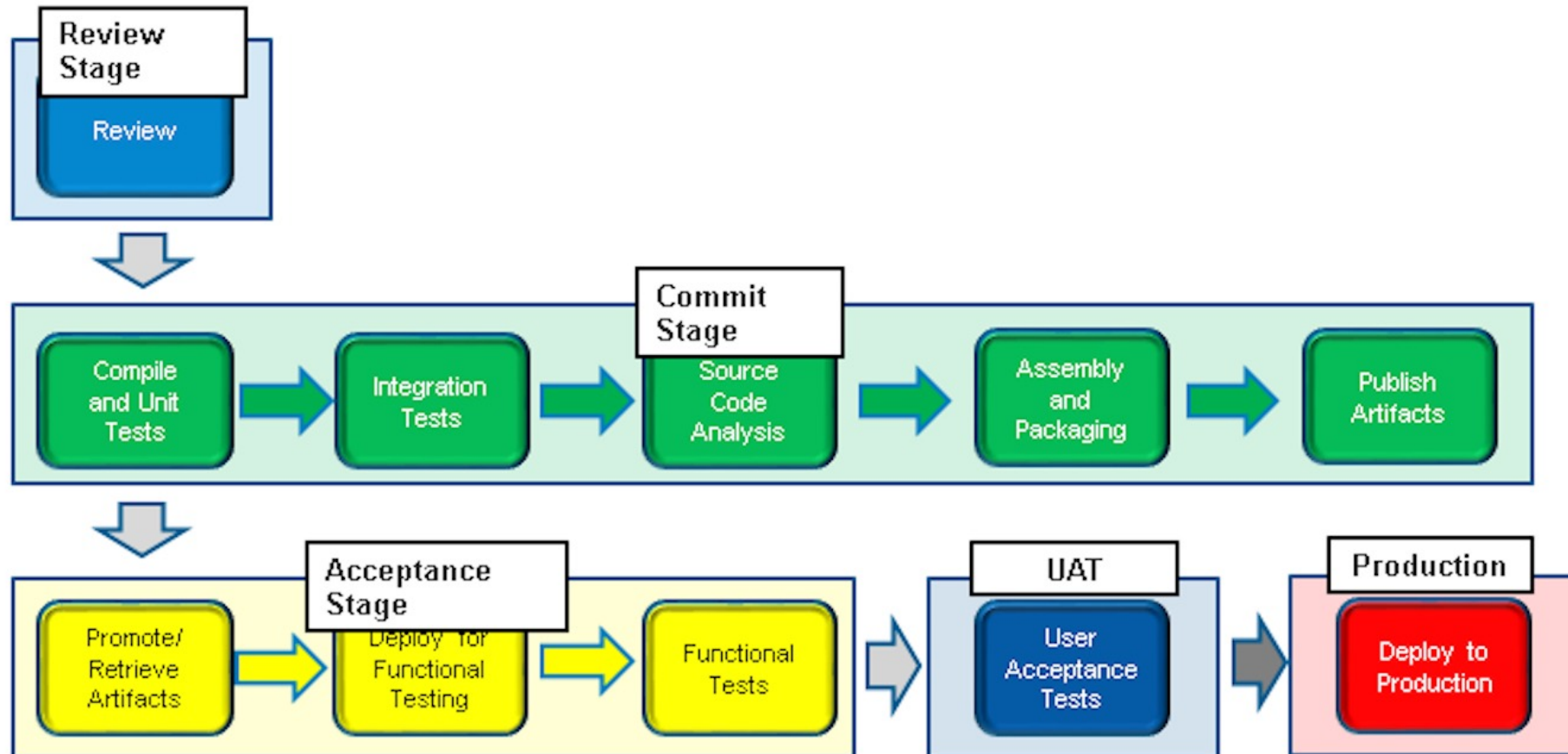


Continuous delivery and continuous deployment

- Continuous delivery: automatically prepare and track a release to production
- Continuous deployment: automatically deploy in the operational environment



A CI/CD pipeline













<https://www.oreilly.com/content/configuring-a-continuous-delivery-pipeline-in-jenkins/>

CI/CD best tools

<https://thectoclub.com/tools/best-ci-cd-tools/>

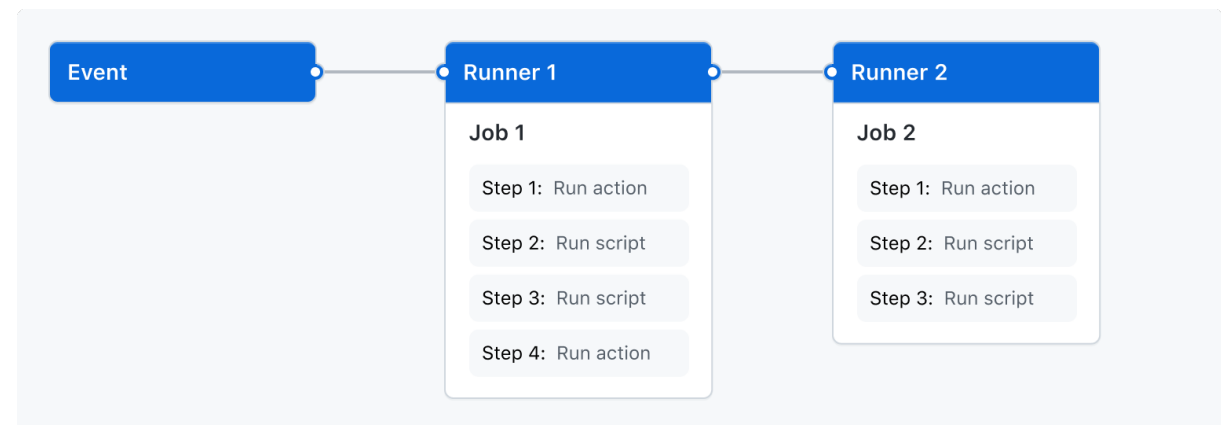


POLITECNICO
MILANO 1863

Tools										
	GitLab CI/CD	GitHub Actions	Azure DevOps	Spinnaker	Argo CD	CircleCI	OpenShift Pipelines	Travis CI	Jenkins	Terraform
	Website	Website	Website	Website	Website	Website	Website	Website	Website	Website
Price	From \$29/user/month	Pricing upon request	From \$52/user/month	Free	Free	From \$15/month for 5 users	Free	From \$34/user/month	Free To Use	Pricing upon request
Best for	Best maturity feedback	Best for small teams	Best for Azure development	Best for custom integrations	Best for Kubernetes development	Best for enterprise development	Best open-source option	Best for on-premise deployments	Best for scaling companies	Best repeatable code
Trial Info	Free plan available	Free plan available	Free plan available	Free	Free plan available	Free plan available	Free plan available	Free trial available	Free	Free plan available
Pros	<ul style="list-style-type: none"> ☆ Pipeline templates ☆ Supports DevSecOps ☆ Detailed maturity feedback 	<ul style="list-style-type: none"> ☆ Actions are isolated, minimizing conflicts and compatibility issues ☆ Wide range of events to link to actions ☆ Easy to use 	<ul style="list-style-type: none"> ☆ Robust repository management ☆ Includes project management solutions for scrum and agile ☆ Combines CI/CD with DevOps 	<ul style="list-style-type: none"> ☆ Supports canary analysis ☆ Active developer community for support ☆ Very configurable 	<ul style="list-style-type: none"> ☆ User-friendly UI ☆ Supports GitOps ☆ Kubernetes native 	<ul style="list-style-type: none"> ☆ Scalable ☆ SSH debugging ☆ Detailed metrics with insights 	<ul style="list-style-type: none"> ☆ Serverless architecture ☆ Kubernetes native ☆ Flexible configuration options ☆ Straightforward setup 	<ul style="list-style-type: none"> ☆ Provides preconfigured customizable build images ☆ Multipurpose GitHub integration 	<ul style="list-style-type: none"> ☆ Highly scalable ☆ Extensible with hundreds of plugins ☆ Active developer community for support 	<ul style="list-style-type: none"> ☆ IAC features that work across most platforms ☆ Robust automation capabilities ☆ Strong code management features
Cons	<ul style="list-style-type: none"> ⊖ No standalone version ⊖ Significantly underpowered free tier 	<ul style="list-style-type: none"> ⊖ Poor support for actions originating outside the core development team ⊖ Built entirely around repositories 	<ul style="list-style-type: none"> ⊖ Limited customization options ⊖ Poor integration with third-party services 	<ul style="list-style-type: none"> ⊖ Relies heavily on third-party tools ⊖ CD only 	<ul style="list-style-type: none"> ⊖ Limited to Kubernetes environments ⊖ Requires significant Kubernetes expertise 	<ul style="list-style-type: none"> ⊖ Visit WebsiteOpens new window ⊖ Support teams often take long to respond ⊖ Expensive 	<ul style="list-style-type: none"> ⊖ Requires extensive configuration ⊖ Doesn't work as well in non-Kubernetes environments 	<ul style="list-style-type: none"> ⊖ Not as configurable as other options ⊖ Reporting is too tight 	<ul style="list-style-type: none"> ⊖ It's very dependent on plugins ⊖ Dated UI 	<ul style="list-style-type: none"> ⊖ Relies heavily on third-party tools for full functionality ⊖ HCL takes a while to learn

Github actions – main concepts

- Workflow: configurable automated process that will run one or more jobs.
- Event: triggers the execution of workflows.
- Job: a set of steps in a workflow.
- Step: can be an action (reusable in different contexts) or a shell script.
- Runner: a VM where a job is run.

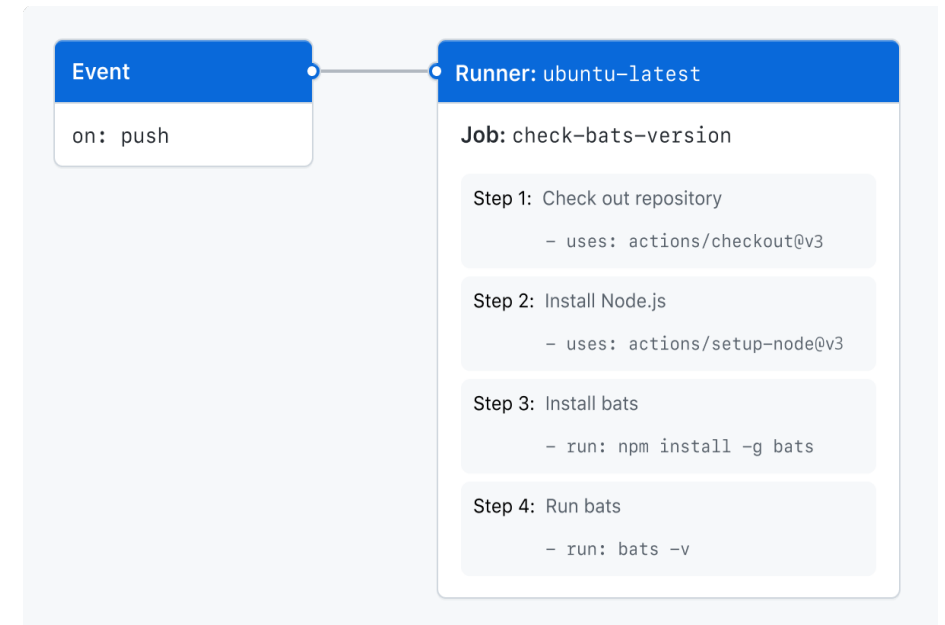


Workflow

- Defined in the `.github/workflows/` directory within a repository

Example

```
name: learn-github-actions
run-name: ${ github.actor } is learning GitHub Actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '20'
      - run: npm install -g bats
      - run: bats -v
```



An example of CI workflow using cmake

<https://github.com/actions/starter-workflows/ci/cmake-single-platform.yml>

```
name: CMake on a single platform
on: [push, pull_request]
env:
  BUILD_TYPE: Release
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Configure CMake
        run: cmake -B ${{github.workspace}}/build -DCMAKE_BUILD_TYPE=${{env.BUILD_TYPE}}
      - name: Build
        run: cmake --build ${{github.workspace}}/build --config ${{env.BUILD_TYPE}}
      - name: Test
        working-directory: ${{github.workspace}}/build
        run: ctest -C ${{env.BUILD_TYPE}}
```


An example of CI workflow using Maven

<https://github.com/actions/starter-workflows/ci/maven.yml>

```
name: Java CI with Maven
on:
  push:
    branches: [ $default-branch ]
  pull_request:
    branches: [ $default-branch ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: maven
      - name: Build with Maven
        run: mvn -B package --file pom.xml
```

Triggering events

- Events that occur in your workflow's repository
- Events that occur outside of GitHub and trigger a repository_dispatch event on GitHub
- Scheduled times
- Manual
- Complete list <https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows>

Triggering events

- Defined after the on keyword

```
on: push
```

- Multiple events can be specified

```
on: [push, pull_request]
```

```
on:
  push:
    branches:
      - main
  pull_request:
    branches: [ $default-branch ]
  label:
    types:
      - created
```

Filter

Activity type

- NB: A workflow instance starts for every event occurred

Actions

- Building blocks that can be used in a workflow
- Can be defined in
 - Your repository
 - Any public repository
 - Directory of publicly available actions: GitHub Marketplace
<https://github.com/marketplace/actions/>
- Are pieces of code written in JavaScript or in any other language if made available as a container
- Each action must have a metadata file to define the inputs, outputs and main entrypoint: `action.yaml` or `action.yml`
- Publicly available actions should have a readme describing it

Workflow execution

- The occurrence of an event with the specified action type and fulfilling the filter triggers the execution of a workflow
- A runner is spooned for each job in the workflow

```
jobs:  
  build1:  
    runs-on: ubuntu-latest  
  ...  
  build2:  
    runs-on: macos-latest  
  ...
```

```
jobs:  
  test:  
    runs-on: [self-hosted, linux]
```

- Steps within each job are executed sequentially

A note on YAML

- YAML (YAML Ain't Markup Language) is a data serialization language designed to be human-friendly
- Is based on three main syntactical elements
 - indentation typically used for structure
 - colons often to separate key/value pairs
 - dashes often to create lists
- <https://yaml.org/spec/1.2.2/>

Jenkins

<https://www.jenkins.io/>



POLITECNICO
MILANO 1863

- History

- Kohsuke Kawaguchi, a Java developer, working at SUN Microsystems, was tired of building the code and fixing errors repetitively. In 2004, created an automation server called Hudson that automates build and test task.
- In 2011, Oracle who owned Sun Microsystems had a dispute with Hudson open-source community, so they forked Hudson and renamed it as Jenkins.
- Both Hudson and Jenkins continued to operate independently. But in short span of time, Jenkins acquired a lot of projects and contributors while Hudson remained with only 32 projects. With time, Jenkins became more popular, and Hudson is not maintained anymore.

Jenkins architecture

