

Technological foundations of software development

Integrate and deploy your software continuously

ICM – Computer Science Major – Course unit on Technological foundations of computer science

M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development

Maxime Lefrançois <https://maxime-lefrancois.info>

online: <https://ci.mines-stetienne.fr/cps2/course/tfsd/>

Objectives of the session

This session is designed to get you familiar with methods and tools for continuous software integration and deployment.
In particular, we will cover Gitlab CI/CD, and Github Actions

Technological foundations of software development

Integrate and deploy your software continuously

Part 1: DevOps

ICM – Computer Science Major – Course unit on Technological foundations of computer scienceopment

M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development

Maxime Lefrançois <https://maxime-lefrancois.info>

online: <https://ci.mines-stetienne.fr/cps2/course/tfsd/>

DevOps

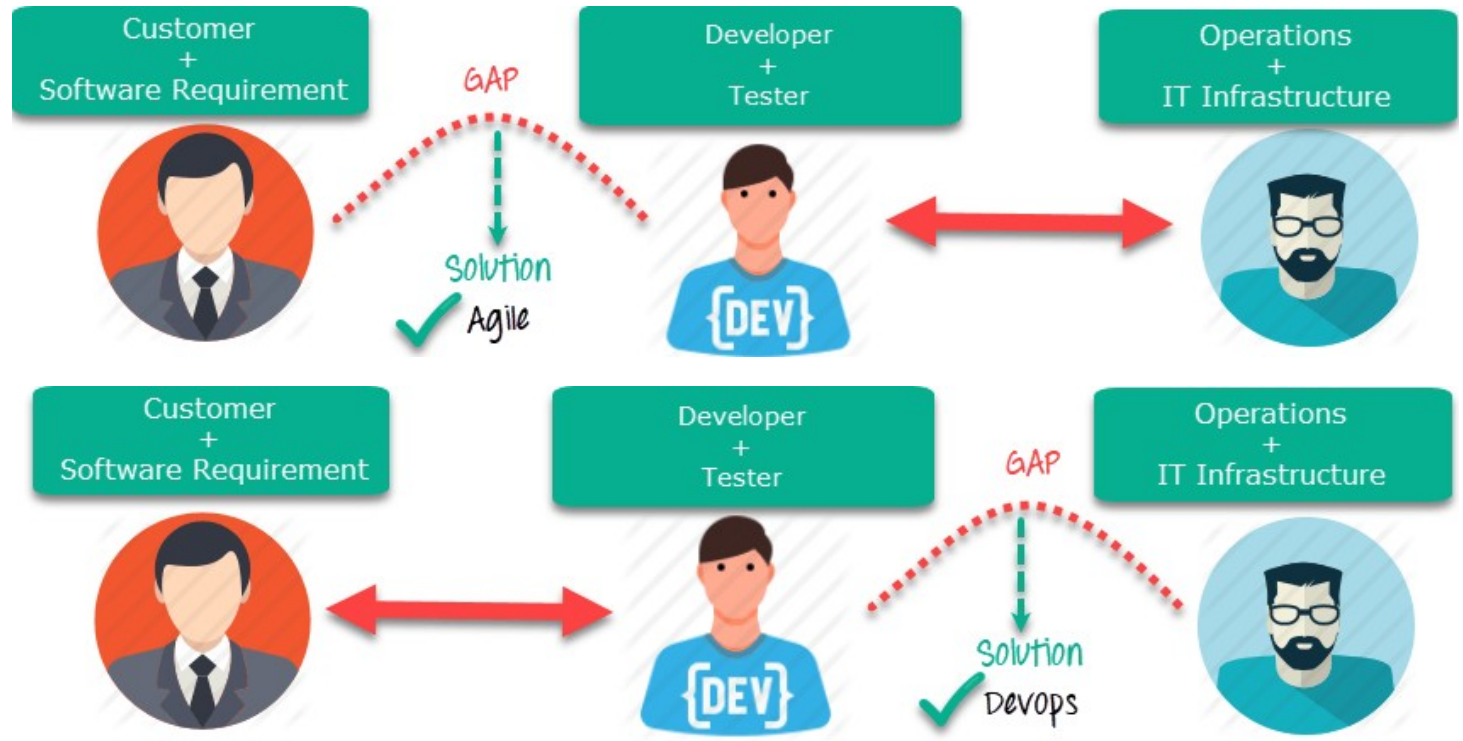
DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology.

— Contributors, Wikipedia <https://en.wikipedia.org/wiki/DevOps>

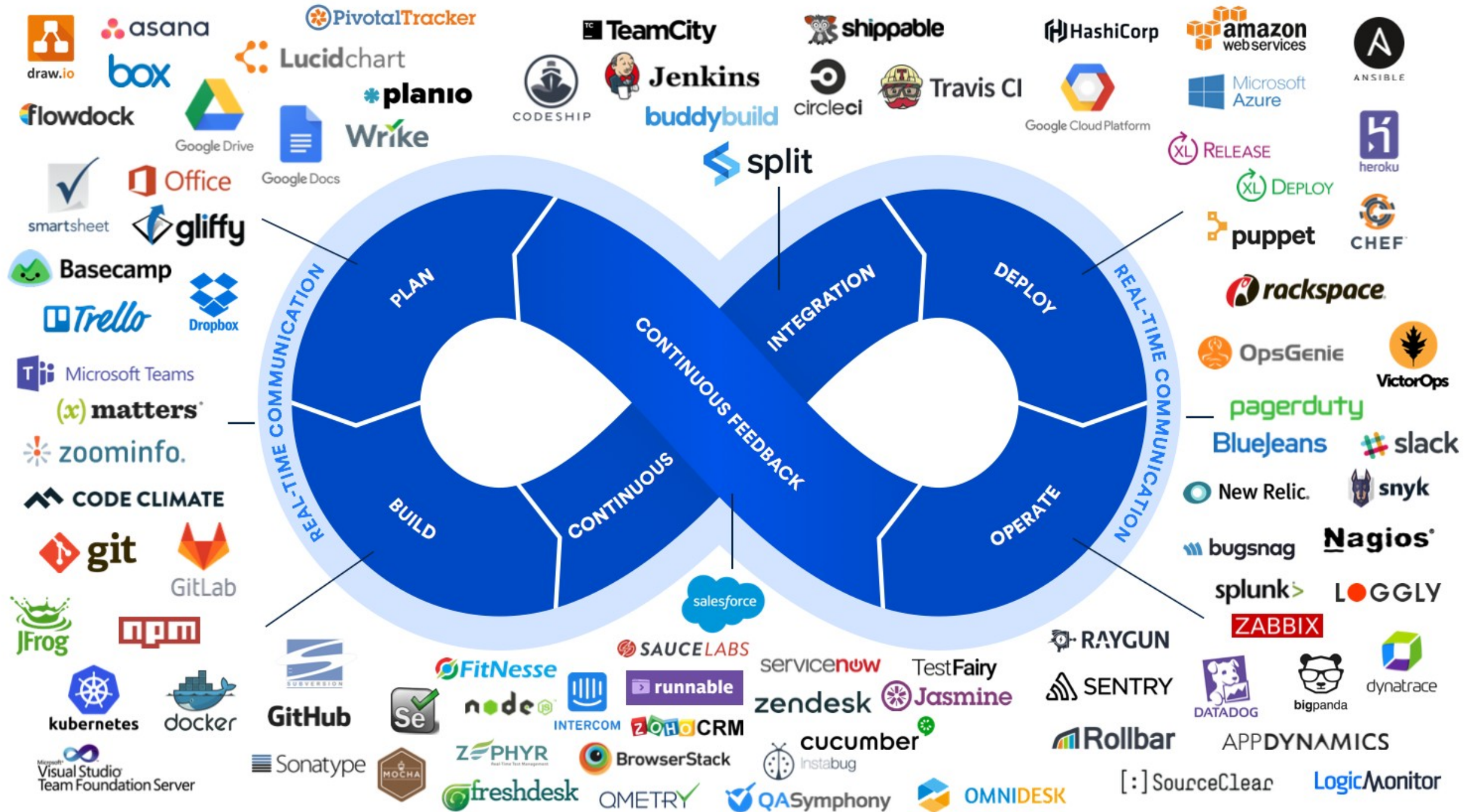


There are many variants

Agile vs DevOps



- DevOps is a practice of bringing development and operations teams together whereas Agile is an iterative approach that focuses on collaboration, customer feedback and small rapid releases.
- DevOps focuses on constant testing and delivery while the Agile process focuses on constant changes.
- DevOps requires relatively a large team while Agile requires a small team.
- DevOps leverages both shifts left and right principles, on the other hand, Agile leverage shift-left principle.
- The target area of Agile is Software development whereas the Target area of DevOps is to give end-to-end business solutions and fast delivery.
- DevOps focuses more on operational and business readiness whereas Agile focuses on functional and non-function readiness.



Definitions

Continuous Integration

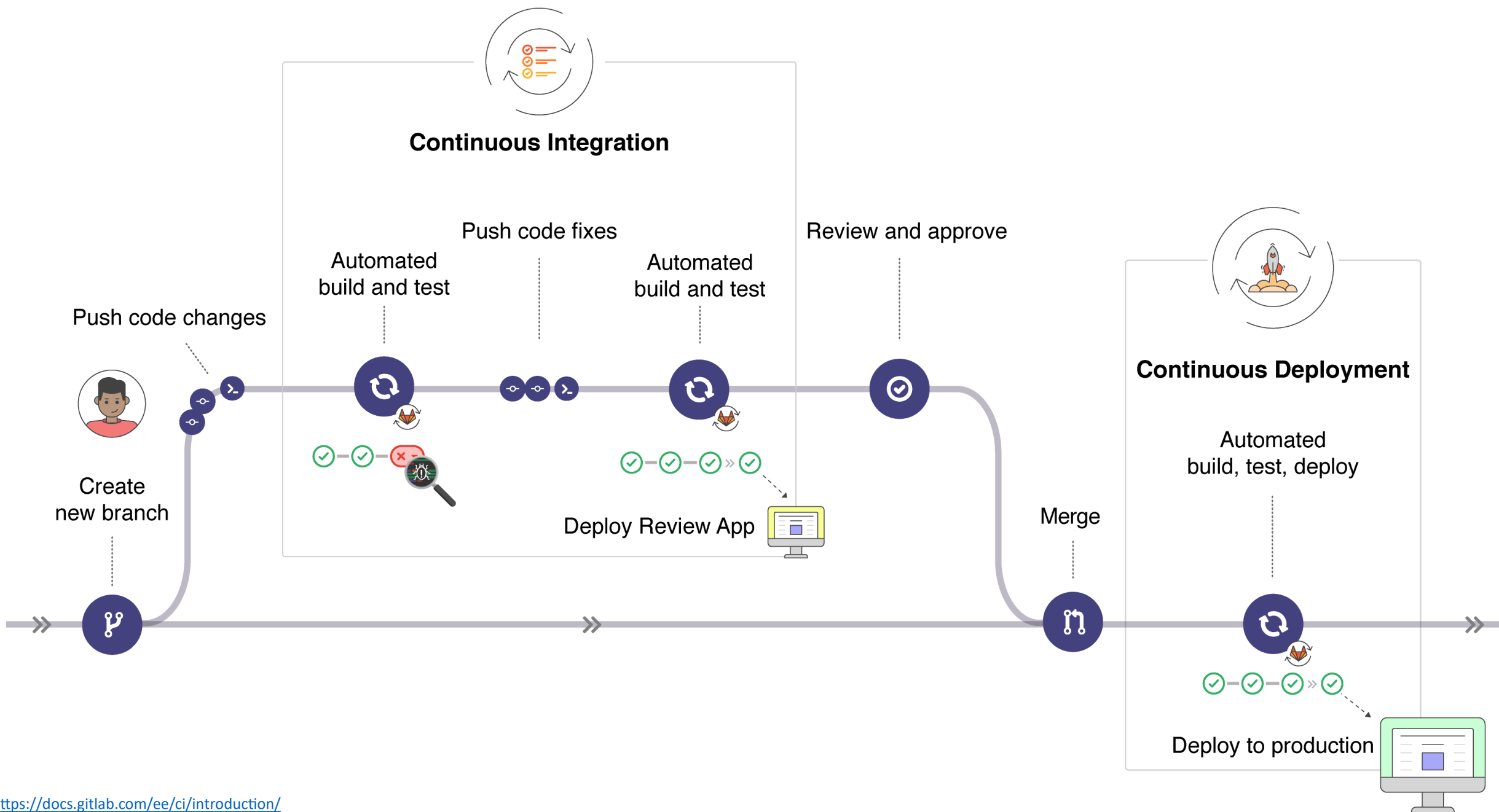
for every change submitted to an application - even to development branches - it's built and tested automatically and continuously, ensuring the introduced changes pass all tests, guidelines, and code compliance standards you established for your app.

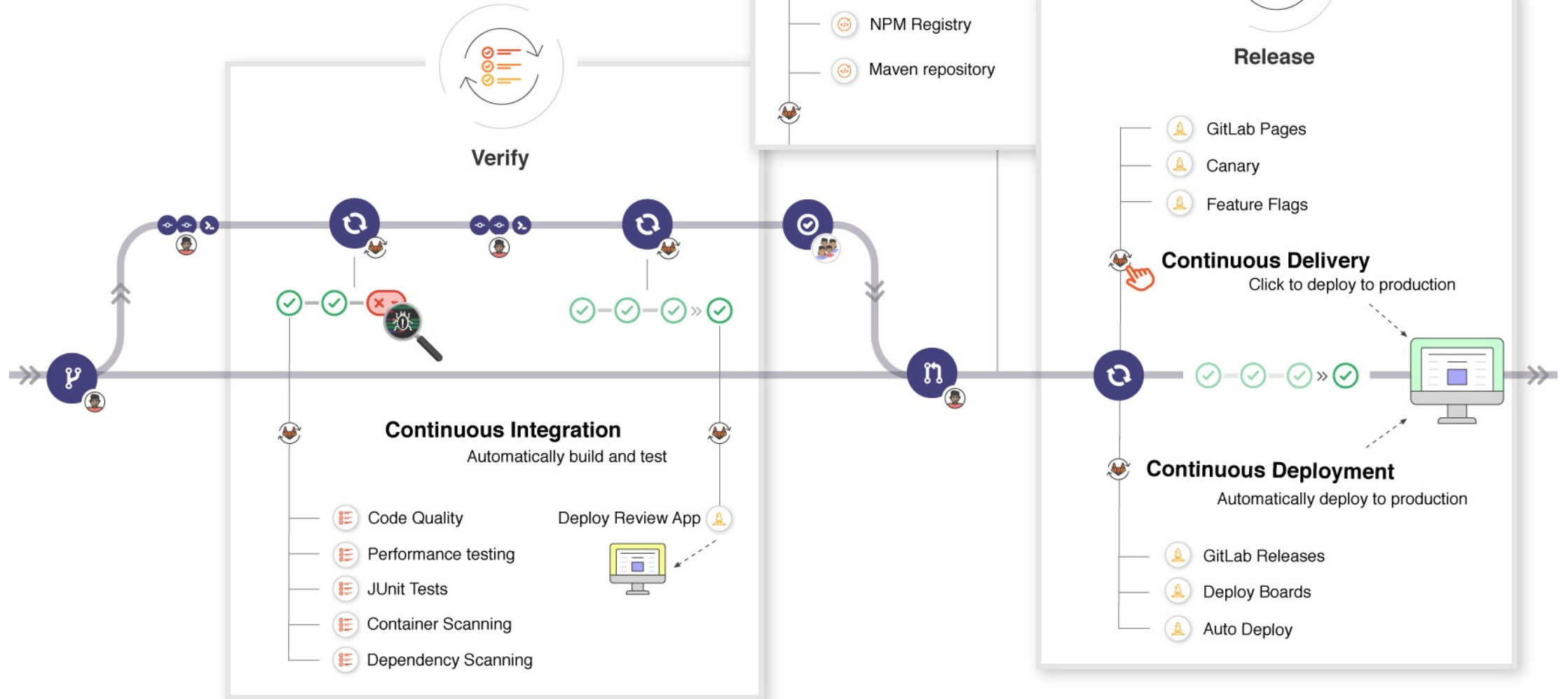
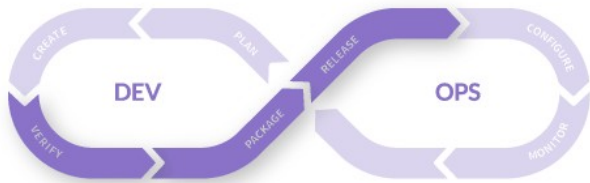
Continuous Delivery

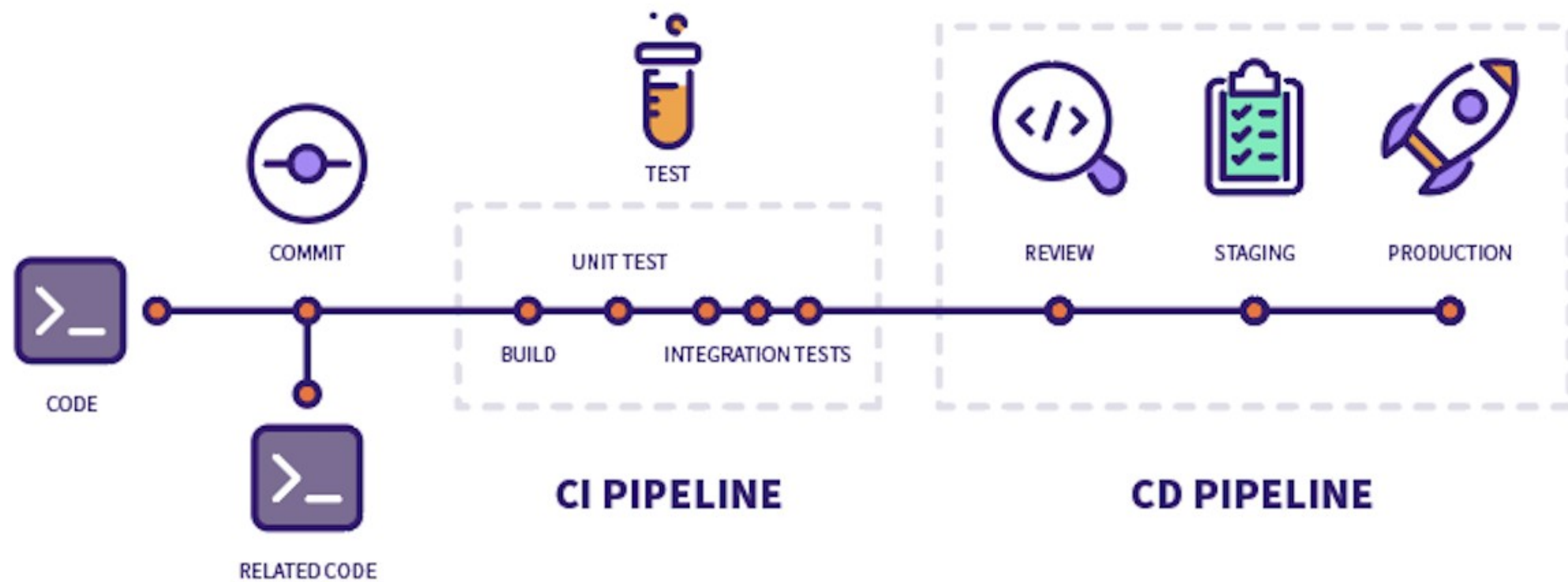
not only built and tested at every code change pushed to the codebase, but, as an additional step, it's also deployed continuously, though the deployments are triggered manually

Continuous Deployment

instead of deploying your application manually, you set it to be deployed automatically. It does not require human intervention at all to have your application deployed



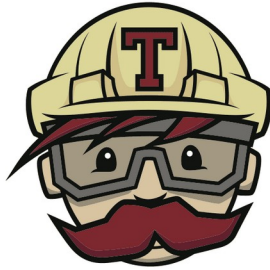




Many tools for CI/CD ...



Circle CI
2011
[https://circleci.com/
YAML](https://circleci.com/YAML)



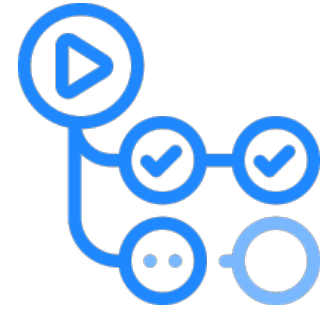
Travis CI
2011
[http://travis-ci.com/
YAML](http://travis-ci.com/YAML)



Jenkins
2011
[https://jenkins.io/
Groovy](https://jenkins.io/Groovy)



GitLab CI/CD
2013
[https://gitlab.com/
YAML](https://gitlab.com/YAML)



GitHub Actions
2018
[https://github.com/
features/actions](https://github.com/features/actions)
YAML

... and others

... all supporting Docker (2013)

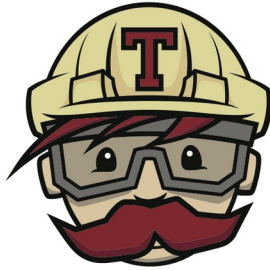


Circle CI

2011

[https://circleci.com/
YAML](https://circleci.com/YAML)

✔ Supports Docker



Travis CI

2011

[http://travis-ci.com/
YAML](http://travis-ci.com/YAML)

✔ Supports Docker



Jenkins

2011

[https://jenkins.io/
Groovy](https://jenkins.io/Groovy)

✔ Supports Docker

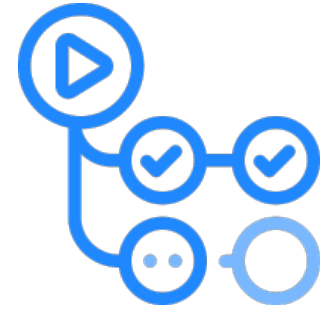


GitLab CI/CD

2013

[https://gitlab.com/
YAML](https://gitlab.com/YAML)

✔ Supports Docker



GitHub Actions

2018

[https://github.com/
features/actions](https://github.com/features/actions)
YAML

✔ Supports Docker

Technological foundations of software development

Intégrer et déployer son logiciel en continue

Part 2: Gitlab CI/CD

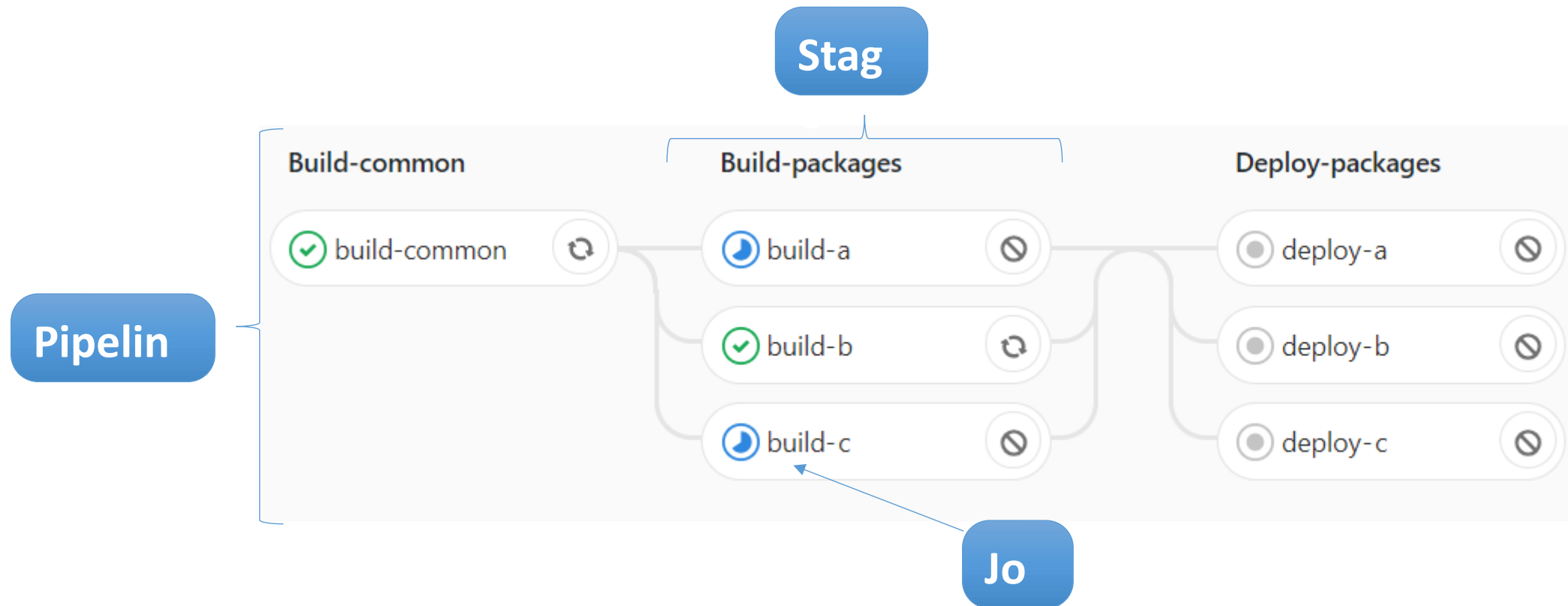
ICM – Computer Science Major – Course unit on Technological foundations of computer scienceopment

M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development

Maxime Lefrançois <https://maxime-lefrancois.info>

online: <https://ci.mines-stetienne.fr/cps2/course/tfsd/>

Pipelines



Pipelines

Stage

By default:

- Build
- Test
- Deploy

```
1 image: maximelefrancois86/asciidoc-rsync:2020
2
3 variables:
4   SSH_PRIVATE: <my_private_key>
5
6 build:
7   stage: build
8   script:
9     - mkdir -p public
10    - cp -r resources/* public/
11    - cp resources/.htaccess public/
12    - gem install tilt concurrent-ruby haml
13    - asciidoctor -o public/index.html index.adoc
14  artifacts:
15    paths:
16      - public
17    only:
18      - master
19
20 deploy:
21   stage: deploy
22   script:
23     - eval $(ssh-agent -s)
24     - ssh-add <(echo "$SSH_PRIVATE")
25     - mkdir -p ~/.ssh
26     - '[[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
27     - rsync -rtcv --delete public/ tb3-i2si@ci.mines-stetienne.fr:/var/www/html/i2si/socle
28   only:
29     - master
30
```

Pipelines

Stage

By default:

- Build
- Test
- Deploy

Job

YAML key names

```
1 image: maximelefrancois86/asciidoc-rsync:2020
2
3 variables:
4   SSH_PRIVATE: <my_private_key>
5
6 build:
7   stage: build
8   script:
9     - mkdir -p public
10    - cp -r resources/* public/
11    - cp resources/.htaccess public/
12    - gem install tilt concurrent-ruby haml
13    - asciidoctor -o public/index.html index.adoc
14  artifacts:
15    paths:
16      - public
17    only:
18      - master
19
20 deploy:
21   stage: deploy
22   script:
23     - eval $(ssh-agent -s)
24     - ssh-add <(echo "$SSH_PRIVATE")
25     - mkdir -p ~/.ssh
26     - '[[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
27     - rsync -rtcv --delete public/ tb3-i2si@ci.mines-stetienne.fr:/var/www/html/i2si/socle
28   only:
29     - master
30
```

Env variables

Variable

- At the project level
- Group level
- By default

https://docs.gitlab.com/ee/ci/variables/predefined_variables.html

<https://docs.gitlab.com/ee/ci/variables/README.html>

The image shows a GitLab interface with two overlapping windows. The background window displays the `.gitlab-ci.yml` file for the `tb3-i2si` project. The file content is as follows:

```
1 image: maximelefrancois86/asciidoc-rsync
2
3 variables:
4   SSH_PRIVATE: <my_private_key>
5
6 build:
7   stage: build
8   script:
9     - mkdir -p public
10    - cp -r resources/* public/
11    - cp resources/.htaccess public/
12    - gem install tilt concurrent-ruby h
13    - asciidoctor -o public/index.html i
14
15 artifacts:
16   paths:
17     - public
18   only:
19     - master
20
21 deploy:
22   stage: deploy
23   script:
24     - eval $(ssh-agent -s)
25     - ssh-add <(echo "$SSH_PRIVATE")
26     - mkdir -p ~/.ssh
27     - '[[ -f /.dockerenv ]] && echo -e "
28     - rsync -rtcv --delete public/ tb3-i2
29
30 master
```

The foreground window shows the `CI / CD` settings for the `tb3-i2si` group. The `Variables` section is active, showing a table of environment variables:

Type	Key	Value	Protected	Masked
Variable	SSH_PRIVATE	*****	×	×

Below the table, there are buttons for `Reveal values` and `Add Variable`. The `Runners` section is also visible, explaining that runners are processes that pick up and execute jobs for GitLab.

Environments

Environments allow control of the continuous deployment of your software, all within GitLab.

Introduction

There are many stages required in the software development process before the software is ready for public consumption.

For example:

1. Develop your code.
2. Test your code.
3. Deploy your code into a testing or staging environment before you release it to the public.

This helps find bugs in your software, and also in the deployment process as well.

GitLab CI/CD is capable of not only testing or building your projects, but also deploying them in your infrastructure, with the added benefit of giving you a way to track your deployments. In other words, you will always know what is currently being deployed or has been deployed on your servers.

It's important to know that:

- Environments are like tags for your CI jobs, describing where code gets deployed.
- Deployments are created when [GitLab CI/CD](#) is used to deploy versions of code to environments.

GitLab:

- Provides a full history of your deployments for each environment.
- Keeps track of your deployments, so you always know what is currently being deployed on your servers.

If you have a deployment service such as [Kubernetes](#) associated with your project, you can use it to assist with your deployments, and can even access a [web terminal](#) for your environment from within GitLab!

Configuration of the pipeline execution

- Scheduled execution
- Configuration in the `.gitlab-ci.yml` file
- Configuration for a Job
 - `tags`: *to choose the “runner”*
 - `stage`: *in which stage the job is run*
 - `variables`: *define job variables on a job level.*
 - `dependencies`: *list of jobs whose artifacts will be used*
 - `when`: *When to run job. Also available: `when:manual` and `when:delayed`.*
 - `allow_failure`: *false (by default), true*
 - `script`: *the shell script(s) to execute.*
 - `only`: *limit when jobs are created. Also available: `only:refs`, `only:kubernetes`, `only:variables`, and `only:changes`.*
 - `except`: *limit when jobs are not created. Also available: `except:refs`, `except:kubernetes`, `except:variables`, and `except:changes`.*
 - `image`: *docker image to use `name:tag`*
 - `services`: *use Docker services images. Also available: `services:name`, `services:alias`, `services:entrypoint`, and `services:command`.*

Configuration of the pipeline execution

- Scheduled execution
- Configuration in the `.gitlab-ci.yml` file
- Configuration for a Job
 - `only`: limit when jobs are created. Also available: `only:refs`, `only:kubernetes`, `only:variables`, and `only:changes`.
 - `except`: limit when jobs are not created. Also available: `except:refs`, `except:kubernetes`, `except:variables`, and `except:changes`.

`only` and `except` are two keywords that set a job policy to limit when jobs are created:

1. `only` defines the names of branches and tags the job runs for.
2. `except` defines the names of branches and tags the job does **not** run for.

There are a few rules that apply to the usage of job policy:

- `only` and `except` are inclusive. If both `only` and `except` are defined in a job specification, the ref is filtered by `only` and `except`.
- `only` and `except` allow the use of regular expressions (supported regexp syntax).
- `only` and `except` allow to specify a repository path to filter jobs for forks.

Configuration of the pipeline execution

- Scheduled execution
- Configuration in the `.gitlab-ci.yml` file
- Configuration for a Job

- `only`: limit when jobs are created. Also available: `only:refs`, `only:branches`
- `except`: limit when jobs are not created. Also available: `except:refs`, `except:branches`

`only` and `except` are two keywords that set a job policy to

1. `only` defines the names of branches and tags the job runs on
2. `except` defines the names of branches and tags the job does not run on

There are a few rules that apply to the usage of job policy:

- `only` and `except` are inclusive. If both `only` and `except` are used, the job runs on branches and tags that are in `only` and not in `except`.
- `only` and `except` allow the use of regular expressions
- `only` and `except` allow to specify a repository path to

In addition, `only` and `except` allow the use of special keywords:

Value	Description
<code>api</code>	For pipelines triggered by the pipelines API .
<code>branches</code>	When the Git reference for a pipeline is a branch.
<code>chat</code>	For pipelines created by using a GitLab ChatOps command.
<code>external</code>	When using CI services other than GitLab.
<code>external_pull_requests</code>	When an external pull request on GitHub is created or updated (See Pipelines for external pull requests).
<code>merge_requests</code>	For pipelines created when a merge request is created or updated. Enables merge request pipelines , merged results pipelines , and merge trains .
<code>pipelines</code>	For multi-project pipelines created by using the API with <code>CI_JOB_TOKEN</code> , or the <code>trigger</code> keyword.
<code>pushes</code>	For pipelines triggered by a <code>git push</code> event, including for branches and tags.
<code>schedules</code>	For scheduled pipelines .
<code>tags</code>	When the Git reference for a pipeline is a tag.
<code>triggers</code>	For pipelines created by using a trigger token .
<code>web</code>	For pipelines created by using Run pipeline button in the GitLab UI, from the project's CI/CD > Pipelines section.

Configuration of the pipeline execution

- Scheduled execution
- Configuration in the `.gitlab-ci.yml` file
- Configuration for a Job
 - `image`: *docker image to use name:tag*
 - `services`: *use Docker services images. Also available: `services:name`, `services:alias`, `services:entrypoint`, and `services:command`.*

Configuration de l'exécution du pipeline

- Scheduled execution
- Configuration in the `.gitlab-ci.yml` file
- Configuration for a Job
 - `before_script`
 - `after_script`
 - `cache`: *List of files that should be cached between subsequent runs. Also available: `cache:paths`, `cache:key`, `cache:untracked`, `cache:when`, and `cache:policy`.*
 - `variables`
 - `image`: defines the default image
 - `services` : defines the default services

Job artifacts: Output, use, and reuse job artifacts.

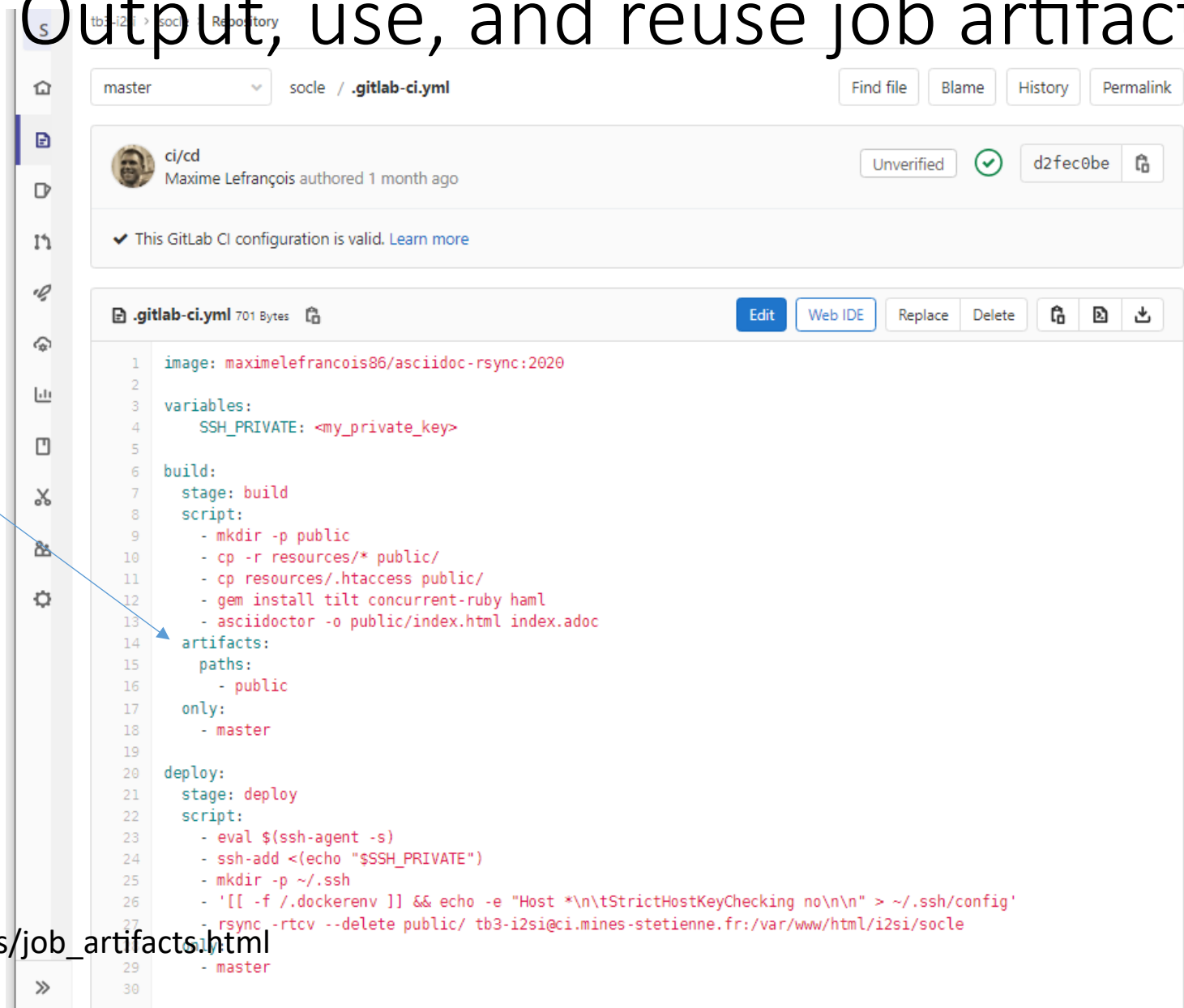
Artifact

Archive available on success.

[Configuration](#) (sub-list):

- artifacts:paths,
- artifacts:exclude,
- artifacts:expose_as,
- artifacts:name,
- artifacts:untracked,
- artifacts:when,
- artifacts:expire_in, and
- [artifacts:reports](#).

https://docs.gitlab.com/ee/ci/pipelines/job_artifacts.html



Example

ETS Labs Projects Groups Snippets Help

develop-v1.1.1 saref4lift / .gitlab-ci.yml Find file Blame History Permalink

migrating to labs.etsi.org/rep/ Maxime Lefrançois authored 4 months ago Unverified 4327c235

✓ This GitLab CI configuration is valid. [Learn more](#)

.gitlab-ci.yml 2.04 KB Edit Web IDE

```
1 image: openjdk:8-jdk
2
3 variables:
4   SAREF_DEV_KEY: <my_private_key>
5   SAREF_PORTAL_KEY: <my_private_key>
6
7 before_script:
8   - eval $(ssh-agent -s)
9   - ssh-add <(echo "$SAREF_DEV_KEY")
10  - mkdir -p ~/.ssh
11  - '[[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config'
12
13 stages:
14   - test
15   - build
16   - deploy
17
18 test-develop:
19   stage: test
20   script:
21     - curl -L -o saref-pipeline.jar "https://labs.etsi.org/rep/saref/saref-pipeline/-/jobs/artifacts/m
22     - java -jar saref-pipeline.jar develop -s && touch target/success
23   allow_failure: true
24   except:
25     - /^prerelease-v/
26     - /^release-v/
27   artifacts:
28     when: always
29     paths:
30       - target/site
31       - target/success
```

Example

The screenshot displays the ETSI Labs Pipelines interface for the `saref4lift` repository. The page shows a list of pipeline runs with the following columns: Status, Pipeline, Triggerer, Commit, and Stages.

Status	Pipeline	Triggerer	Commit	Stages
passed	#386 latest	[Avatar]	<code>develop-v1...</code> <code>c7363cc3</code> Merge branch 'devel...	00:01:02 4 months ago
passed	#356 latest	[Avatar]	<code>release-v1...</code> <code>c7363cc3</code> Merge branch 'devel...	00:01:15 4 months ago
passed	#352 latest	[Avatar]	<code>prerelease-...</code> <code>c7363cc3</code> Merge branch 'devel...	00:01:12 4 months ago
failed	#351	[Avatar]	<code>develop-v1...</code> <code>a92652c2</code> TS published, doc sy...	00:00:01 4 months ago
passed	#332	[Avatar]	<code>prerelease-...</code> <code>4327c235</code> migrating to labs.etsi...	00:01:15 4 months ago
passed	#309	[Avatar]	<code>prerelease-...</code> <code>cb8278ca</code> typo	00:01:22 6 months ago
passed	#308	[Avatar]	<code>develop-v1...</code> <code>cb8278ca</code> typo	00:01:07 6 months ago

Example

The screenshot shows a web browser window with the URL `https://labs.etsi.org/rep/saref/saref4lift/-/jobs/763`. The browser's address bar and tabs are visible at the top. The main content area is a terminal window displaying a shell script execution. The script includes commands for setting environment variables, adding an SSH identity, creating a directory, and downloading a pipeline. The output shows the script's progress and a warning message about undefined terms in an ontology. The sidebar on the right contains job details for 'test-develop', including duration, timeout, runner, and a list of job artifacts. The bottom of the sidebar shows a dropdown menu with 'test' selected and a link to 'test-develop'.

```
16 $ eval $(ssh-agent -s)
17 Agent pid 42250
18 $ ssh-add <(echo "$SAREF_DEV_KEY")
19 Identity added: /dev/fd/63 (/dev/fd/63)
20 $ mkdir -p ~/.ssh
21 $ [[ -f /.dockerenv ]] && echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
22 $ curl -L -o saref-pipeline.jar "https://labs.etsi.org/rep/saref/saref-pipeline/-/jobs/artifacts/master/raw/target/saref-pipeline.jar?job=build"
23 % Total % Received % Xferd Average Speed Time
24 Time Time Current Dload Upload Total
25 Spent Left Speed
26 100 163 100 163 0 0 1646 0 --:--:-- -
27 100 58.4M 100 58.4M 0 0 28.4M 0 0:00:02
28 0:00:02 --:--:-- 34.2M
29 $ java -jar saref-pipeline.jar develop -s && touch target/success
30 ## WARN in SAREF Pipeline: .SAREF4LIFT v1.1.1 - Issues in TS 103 673 Clause 9.5
31 Themis found some problems. Violations are:
32 - id="testCase1", name="AlarmInTheWell subclassOf Alarm", message="The terms AlarmInTheWell are undefined in the ontology with URI https://saref.etsi.org/saref4lift"
33 - id="testCase2", name="AlarmVoiceCommunication subclassOf Alarm", message="The terms AlarmVoiceCommunication are undefined in the ontology with URI https://saref.etsi.org/saref4lift"
34 - id="testCase3", name="CarAlarm subclassOf hasValue some boolean", message="The terms CarAlarm are undefined in the ontology with URI https://saref.etsi.org/saref4lift"
35 - id="testCase4", name="AlarmInTheCar subclassOf Alarm", message="The terms AlarmInTheCar are undefined in the ontology with URI https://saref.etsi.org/saref4lift"
36 - id="testCase5", name="AlarmInTheMachinery subclassOf Alarm", message="The terms AlarmInTheMachinery are undefined in the ontology with URI https://saref.etsi.org/saref4lift"
```

test-develop

Duration: 1 minute 0 seconds

Timeout: 1h (from project)

Runner: build2.forge.etsi.org (#2)

Job artifacts

The artifacts were removed 3 months ago

Commit c7363cc3

Merge branch 'develop-v1.1.1' into prerelease-v1.1.1

✓ **Pipeline #386** for **develop-v1.1.1**

test

→ ✓ test-develop

Technological foundations of software development

Integrate and deploy your software continuously

Part 3: GitHub Actions

ICM – Computer Science Major – Course unit on Technological foundations of computer scienceopment

M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development

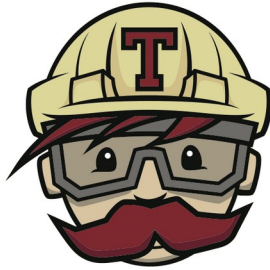
Maxime Lefrançois <https://maxime-lefrancois.info>

online: <https://ci.mines-stetienne.fr/cps2/course/tfsd/>

GitHub Actions: GitHub's response



Circle CI
2011
[https://circleci.com/
YAML](https://circleci.com/YAML)
Supports Docker



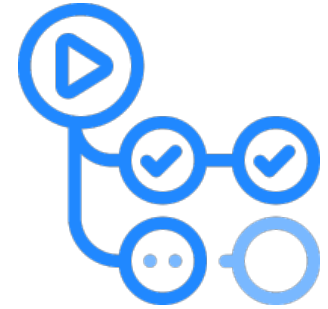
Travis CI
2011
[http://travis-ci.com/
YAML](http://travis-ci.com/YAML)
Supports Docker



Jenkins
2011
[https://jenkins.io/
Groovy](https://jenkins.io/Groovy)
Supports Docker



GitLab CI/CD
2013
[https://gitlab.com/
YAML](https://gitlab.com/YAML)
Supports Docker



GitHub Actions
2018
[https://github.com/
features/actions](https://github.com/features/actions)
YAML
Supports Docker

GitHub Flow with GitHub Actions



Find actions

The screenshot shows the GitHub Marketplace interface. The browser address bar displays `https://github.com/marketplace?type=actions`. The top navigation bar includes the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. The main content area is titled 'Marketplace / Search results' and features a search bar with the text 'Search for apps and actions' and a 'Sort: Best Match' button. On the left sidebar, under the 'Types' section, the 'Actions' tab is selected. Below this, a list of categories is shown, including API management, Chat, Code quality, Code review, Continuous integration, Dependency management, Deployment, IDEs, Learning, Localization, Mobile, Monitoring, Project management, Publishing, and Recently added. The main content area displays '11109 results filtered by Actions'. A section titled 'Actions' provides a description: 'An entirely new way to automate your development workflow.' Below this, a grid of action cards is shown, each with a play button icon, a title, a description, and a star count. The actions listed are: 'Setup Go environment' (519 stars), 'First interaction' (140 stars), 'Cache' (2.3k stars), 'Setup .NET Core SDK' (377 stars), 'Setup Node.js environment' (1.6k stars), 'Setup Java JDK' (565 stars), 'Download a Build Artifact' (470 stars), and 'Upload a Build Artifact' (1.3k stars).

GitHub Marketplace · Actions to i x +

← → ↺ `https://github.com/marketplace?type=actions` ☆ 🌐 ⚙️ 👤

🐱 Search or jump to... / Pull requests Issues Marketplace Explore 🔔 + 👤

Marketplace / Search results

Types

Apps

Actions x

Categories

API management

Chat

Code quality

Code review

Continuous integration

Dependency management

Deployment

IDEs

Learning

Localization

Mobile

Monitoring

Project management

Publishing

Recently added

Search for apps and actions

Sort: Best Match

Actions

An entirely new way to automate your development workflow.

11109 results filtered by Actions x

Actions

Setup Go environment
By actions ✓
Setup a Go environment and add it to the PATH
☆ 519 stars

First interaction
By actions ✓
Greet new contributors when they create their first issue or open their first pull request
☆ 140 stars

Cache
By actions ✓
Cache artifacts like dependencies and build outputs to improve workflow execution time
☆ 2.3k stars

Setup .NET Core SDK
By actions ✓
Used to build and publish .NET source. Set up a specific version of the .NET and authentication to private NuGet repository
☆ 377 stars

Setup Node.js environment
By actions ✓
Setup a Node.js environment by adding problem matchers and optionally downloading and adding it to the PATH
☆ 1.6k stars

Setup Java JDK
By actions ✓
Set up a specific version of the Java JDK and add the command-line tools to the PATH
☆ 565 stars

Download a Build Artifact
By actions ✓
Download a build artifact that was previously uploaded in the workflow by the upload-artifact action
☆ 470 stars

Upload a Build Artifact
By actions ✓
Upload a build artifact that can be used by subsequent workflow steps
☆ 1.3k stars

Example

The screenshot shows a GitHub repository page for `ns.hyperagents.org/deploy.yml`. The page header includes navigation links for Code, Issues (11), Pull requests, Actions, Projects, Security, and Insights. The repository path is `ns.hyperagents.org / .github / workflows / deploy.yml`. A button labeled "View runs" is visible. Below the header, a commit by `maximelefrancois86` is shown, triggered on the master branch, with the latest commit `6da708c` from 14 days ago. A section indicates 1 contributor. The main content area displays the workflow file `deploy.yml` with 28 lines (27 sloc) and 686 Bytes. The file content is as follows:

```
1 name: deploy
2 on:
3   push:
4     branches:
5       - master
6
7 jobs:
8   deploy:
9     runs-on: ubuntu-latest
10    steps:
11      - uses: actions/checkout@v2
12        with:
13          fetch-depth: 0
14      - uses: actions/setup-python@v2
15        with:
16          python-version: '3.x'
17          cache: 'pip'
18      - run: pip install -r scripts/requirements.txt
19      - run: python scripts/process_core.py
20      - name: rsync deployments
21        uses: burnett01/rsync-deployments@5.1
22        with:
23          switches: -a
24          path: public/
25          remote_path: /var/www/hmas/html
26          remote_host: ci.mines-stetienne.fr
27          remote_user: hyperagents
28          remote_key: ${ secrets.SSH_PRIVATE }
```

Example

Example

The screenshot shows a GitHub Actions workflow run for the repository 'HyperAgents / ns.hyperagents.org'. The workflow is named 'Update contributors' and is a deployment (#16). The run is successful, having completed 5 days ago in 22 seconds. The workflow consists of several jobs, all of which are marked as successful. The 'deploy' job is the final and most detailed one, showing a list of steps and their durations.

Update contributors
deploy #16

Summary

Jobs

- ✓ deploy

deploy
succeeded 5 days ago in 22s

Search logs

- > ✓ Set up job 3s
- > ✓ Build burnett01/rsync-deployments@5.1 6s
- > ✓ Run actions/checkout@v2 1s
- > ✓ Run actions/setup-python@v2 2s
- > ✓ Run pip install -r scripts/requirements.txt 2s
- > ✓ Run python scripts/process_core.py 2s
- > ✓ rsync deployments 4s
- > ✓ Post Run actions/setup-python@v2 1s
- > ✓ Post Run actions/checkout@v2 0s
- > ✓ Complete job 0s

Technological foundations of software development

Integrate and deploy your software continuously

ICM – Computer Science Major – Course unit on Technological foundations of computer scienceopment

M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development

Maxime Lefrançois <https://maxime-lefrancois.info>

online: <https://ci.mines-stetienne.fr/cps2/course/tfsd/>

... Your turn

Complete the TODO section:

https://ci.mines-stetienne.fr/cps2/course/tfsd/course-8.html#_todos