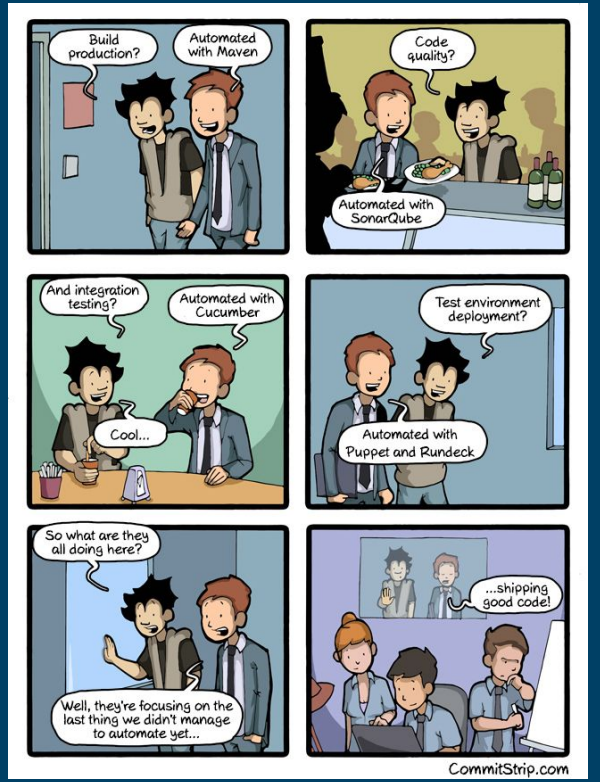
Lesson 07
Continuous integration

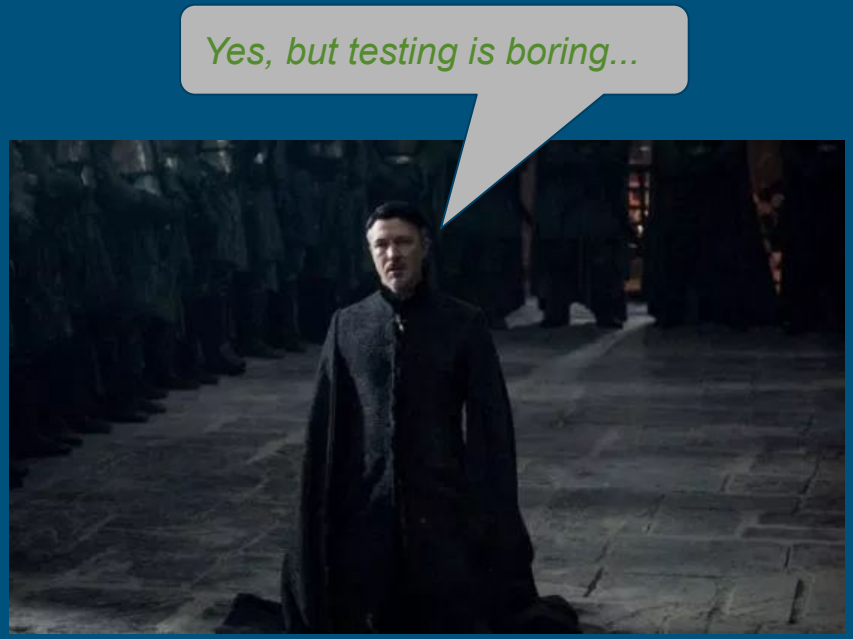
DEVOPS - ITI 4
HEI 2021-2022

What is continuous integration ?



Remember

Unit testing



Maven

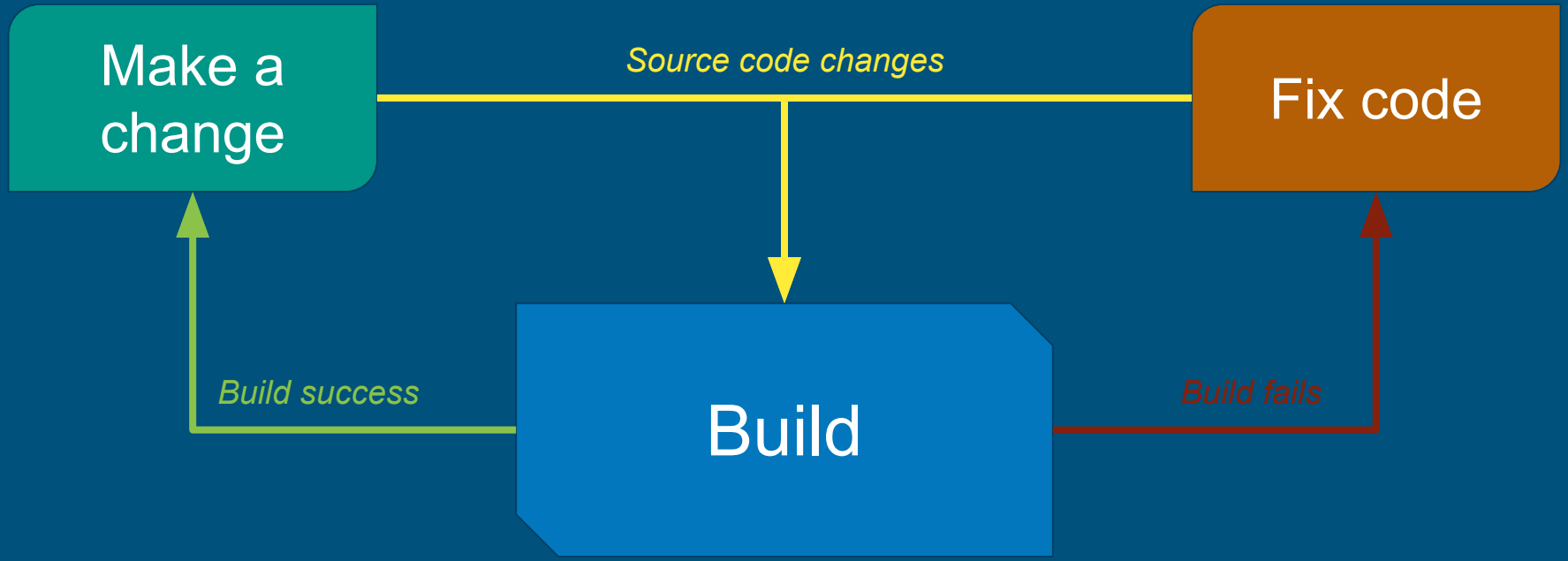
Overview

Definition

Continuous integration is a set of practices used in software engineering to verify at each source code change that the result of the changes does not produce regression in the developed application.

https://fr.wikipedia.org/wiki/Int%C3%A9gration_continue

Diagram


Make a change
Source code changes
Fix code
Build success
Build
Build fails

Goals

Improve Developer productivity

- Free developers from recurring tasks

- Encourage behaviors that help reduce the number or error and bugs

Find and fix bugs quicker

- Bugs are detected and fixed early (when least costly)

- Easily repeatable testing

Deliver updates faster

- Deliveries take less time thanks to automation
- More regular delivery

Build

Build is not only compilation

Step 1 : Compile

- Ensure code actually compiles

- On every target platform

Step 2 : Test

- Verify that the features run as expected

- Check that there is no regression

Step 3 : Deploy

- Generate a new resource

- Upload the resource on the remote repository

Always further

Continuous Delivery

- Extends Continuous Integration

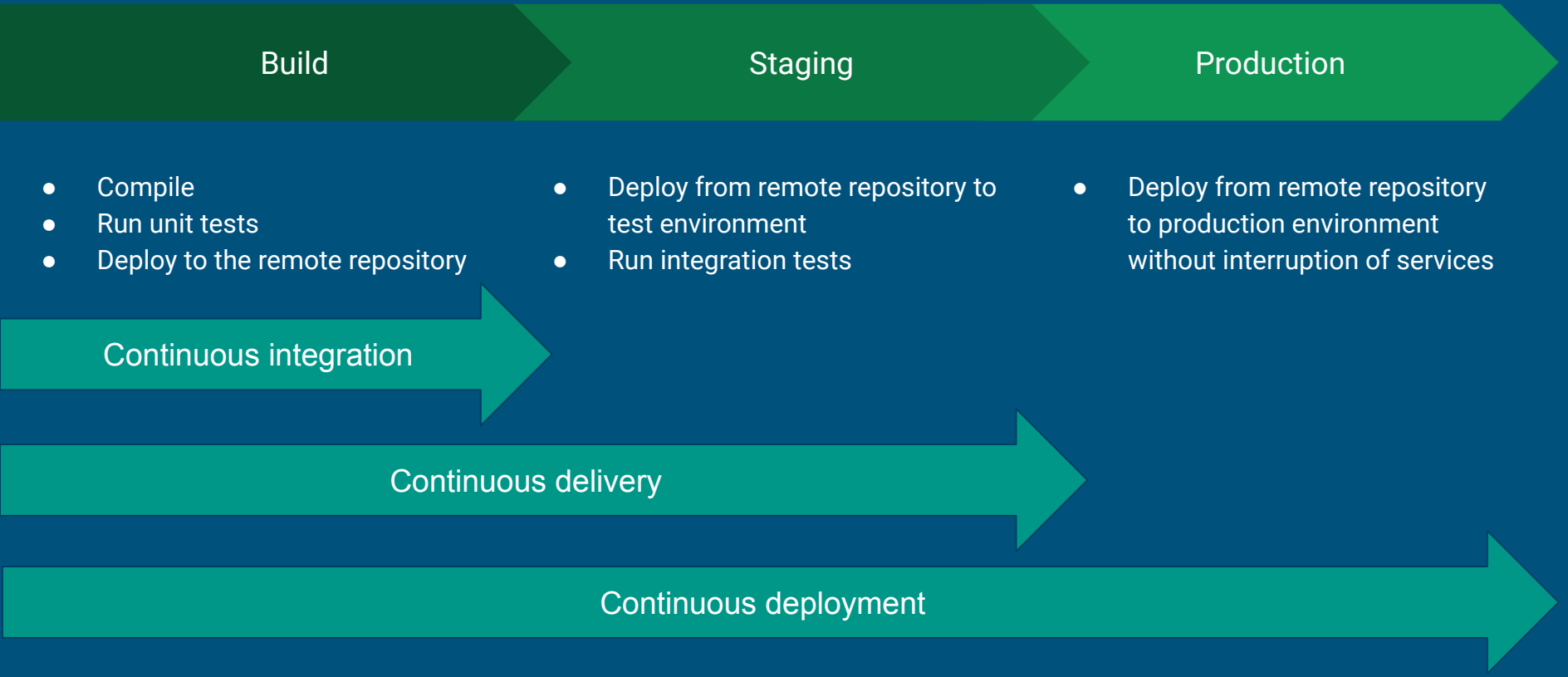
- Adds :
  - Automatic deployment of the application to a test environment
  - Automation of admission tests

Continuous Deployment

- Extends Continuous Integration

- Adds :
  - Automatic deployment of the application to a test environment
  - Automation of admission tests
  - Automatic deployment of the application to a production environment

Sum up


Build
Staging
Production
Compile
Run unit tests
Deploy to the remote repository
Deploy from remote repository to test environment
Run integration tests
Deploy from remote repository to production environment without interruption of services
Continuous integration
Continuous delivery
Continuous deployment

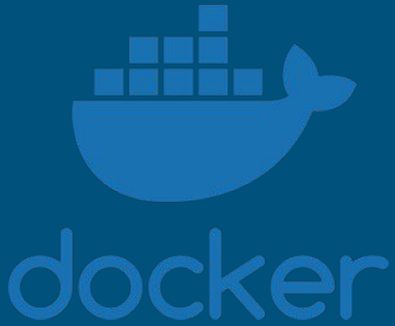
How to use Gitlab-ci ?

Overview

Presentation

- Continuous integration service
- Submitted by GitLab
- Using a yml file to manage this steps

maven



JUnit

Job

- Task to execute

- With constraints to state when it should be run

Stages

- Group of jobs
- All of the jobs in a stage are executed in parallel
- In error if at least one job fails

Pipeline

- Group of stages

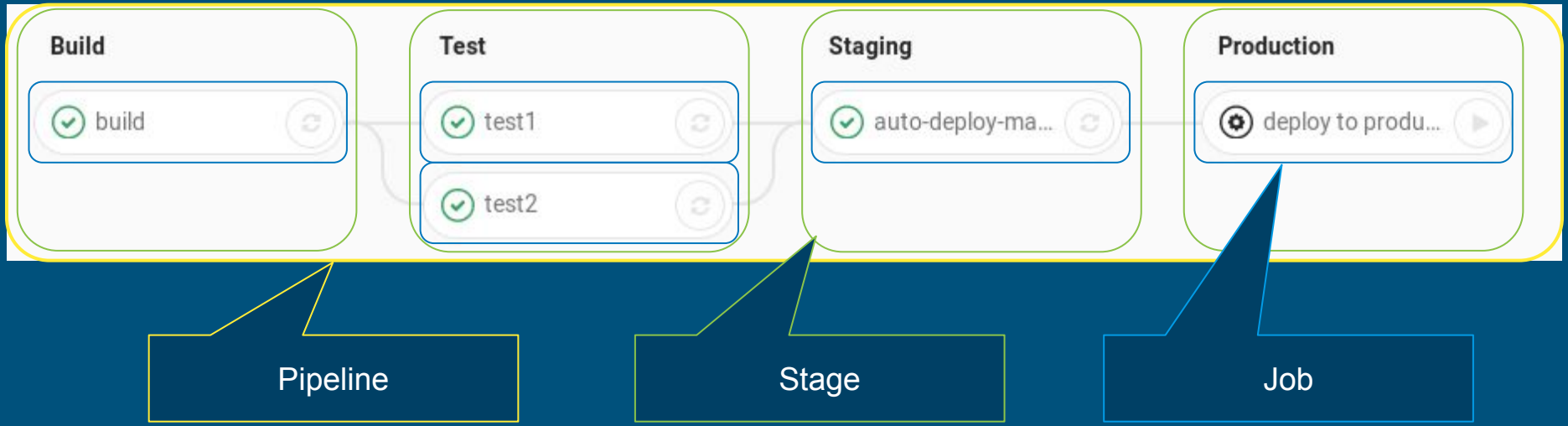
- Run stages sequentially (One after another)

Runner

- Element to run a job

- 2 types :
  - Shared runners
  - Specific runnes

Sum up


Build
build
Test
test1
test2
Staging
auto-deploy-ma...
Production
deploy to produ...
Pipeline
Stage
Job

.gitlab-ci.yml

stages

- Defines stage that can be used by job

- Defines the order of the stages

```
stages:
  - stage1
  - stage2
  - stage3
```

variables

- Defines variable whose values will be passed in job environment

- 2 types :
  - global
  - per-job

```
variables:
 MY_VARIABLE_KEY: MY_VARIABLE_VALUE
```

before_script

- Defines commands to run before each job

- 2 types :
  - global
  - per-job

```
before_script:
  - my command
```

after_script

- Defines command to run after each job (even failed ones)

- 2 types :
  - global
  - per-job

```
after_script:
  - my command
```

script

- Defines scripts which is executed by Runner

- Required in each job

```
job:
 script:
 - execute-script-for-job1
```

image

- Defines docker image to use to execute job

- 2 types :
  - global
  - per-job

```
job:
 image: my-image:version
```

stage

- Defines a job stage (defined by *stages*)

- Only used in a job

```
job:
 stage: stage1
```

when

- Defines the condition to execute job
- Only used in a job
- Conditions are :
  - on_success : execute job only when all jobs from prior stages succeed (default).
  - on_failure : execute job only when at least one job from prior stages fails.
  - always : execute job regardless of the status of jobs from prior stages.
  - manual : execute job manually.

```
job:
 when: manual
```

tags

- Selects specific runners with this tag

- Only used in a job

```
job:
 tags:
  - docker
```

cache

- Define a storage for temporary resources
- 2 types :
  - global
  - per-job
- Cache is available to all job with the same key

```
cache:
 key: my-key
 paths:
 - myDirectory/*
```

artifacts

- Specifies a list of files and directory which should be attached to the job
- Files will be available for download in the GitLab UI
- Only used in a job

```
job:
 artifacts:
  paths:
  - myDirectory/*
  expire_in: 1 week
```

dependencies

- Define other jobs that a job depends on so that you can pass artifacts between them
- Only used in a job

```
job_01:
...

job_02:
 dependencies:
    - job_01
```

To go further

https://docs.gitlab.com/ee/ci/yaml/

Question

What’s happen ?

```
image: maven:latest

stages:
  - build
  - test
  - run

variables:
  MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

cache:
  key: my-key
  paths:
    - .m2/repository/
    - target/
```

```
build:
  stage: build
  script:
    - mvn compile

test:
  stage: test
  script:
    - mvn test

run:
  stage: run
  script:
    - mvn package
    - mvn exec:java
-Dexec.mainClass="hei.devops.y2019.lesson07.Application"
```

Thank you for your attention

Links :

- Sources
  - http://www.commitstrip.com/
  - https://giphy.com/
  - https://en.wikipedia.org/wiki/Continuous_integration
  - https://codeship.com/continuous-integration-essentials
  - https://www.youtube.com/watch?v=_zCyLT33moA
  - https://aws.amazon.com/devops/continuous-integration
  - https://docs.gitlab.com/ee/ci/