Continuous everything

Integration, Delivery, Deployment

Agenda

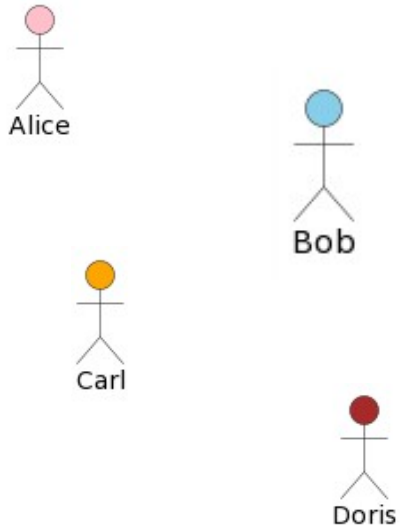
- First pieces of software product lines
  - Dev. Machine
  - Share for collaboration
  - Share for dependencies
    - SemVer
    - Build versus Run dependencies
    - Remote Build
- Virtualisation
  - Hypervisors
  - Containers
  - Dockefile
- Continuous Integration
  - Pipelines
  - Push versus Merge Request
  - Stages
  - Gitlab Pages
- Conclusion

Software product lines

First pieces of …

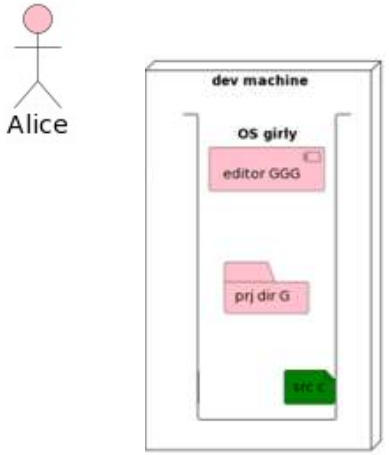
Story telling

- Let’s see the major tools of a software factory (in agile context) with the needs of a growing team
  - SCM (Source Code manager)
  - Dependency Manager
  - Continuous X Engine

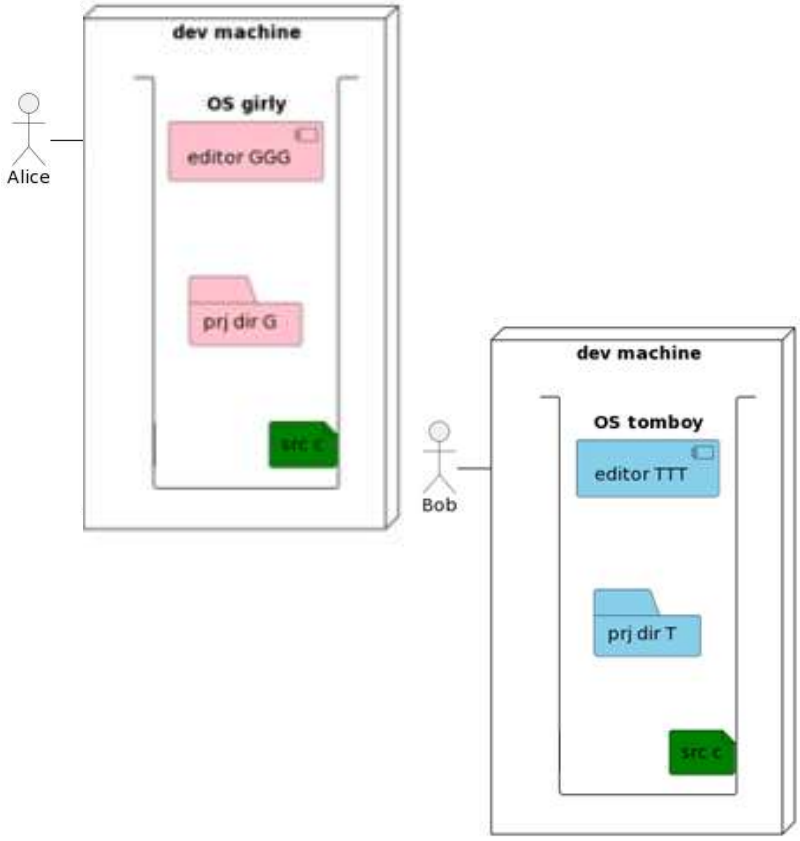


One dev : intro. To Dev. Machine

- Dev. Machine embeds
  - Directories with source code
  - Editor(s) of source code
  - Possibly build tools (or tasks runner(s))

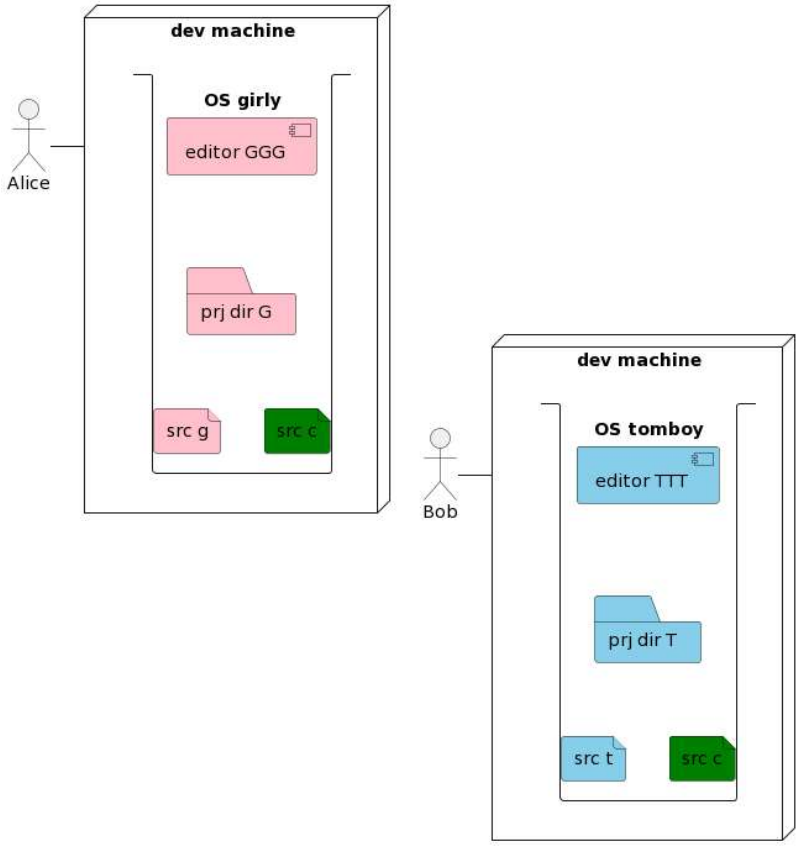


Second developper - Share for collaboration



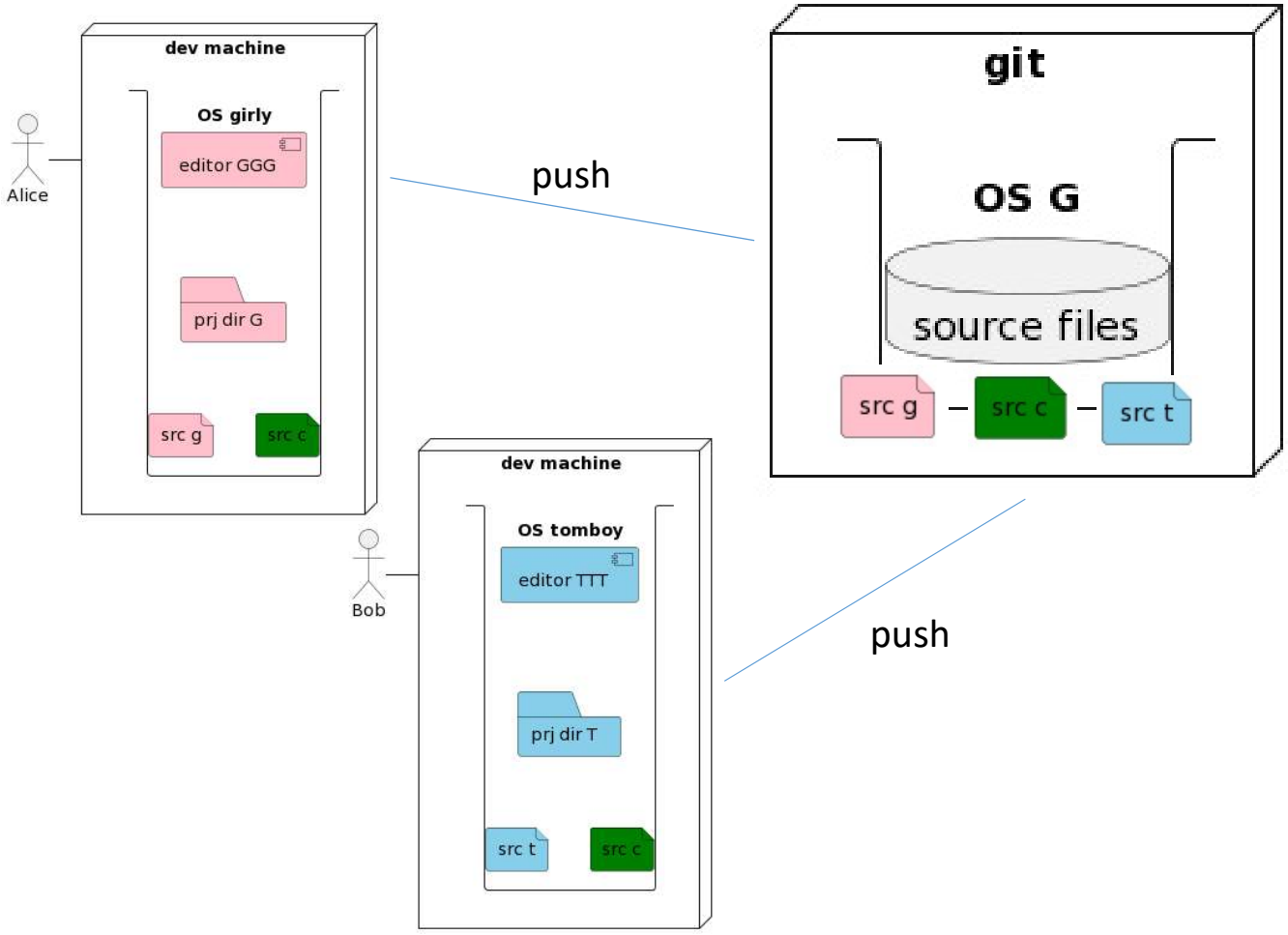
- A second dev. is joining (Bob)
  - He needs the existing file (src c)
  - He can copy it from Alice project's directory

Share for collaboration



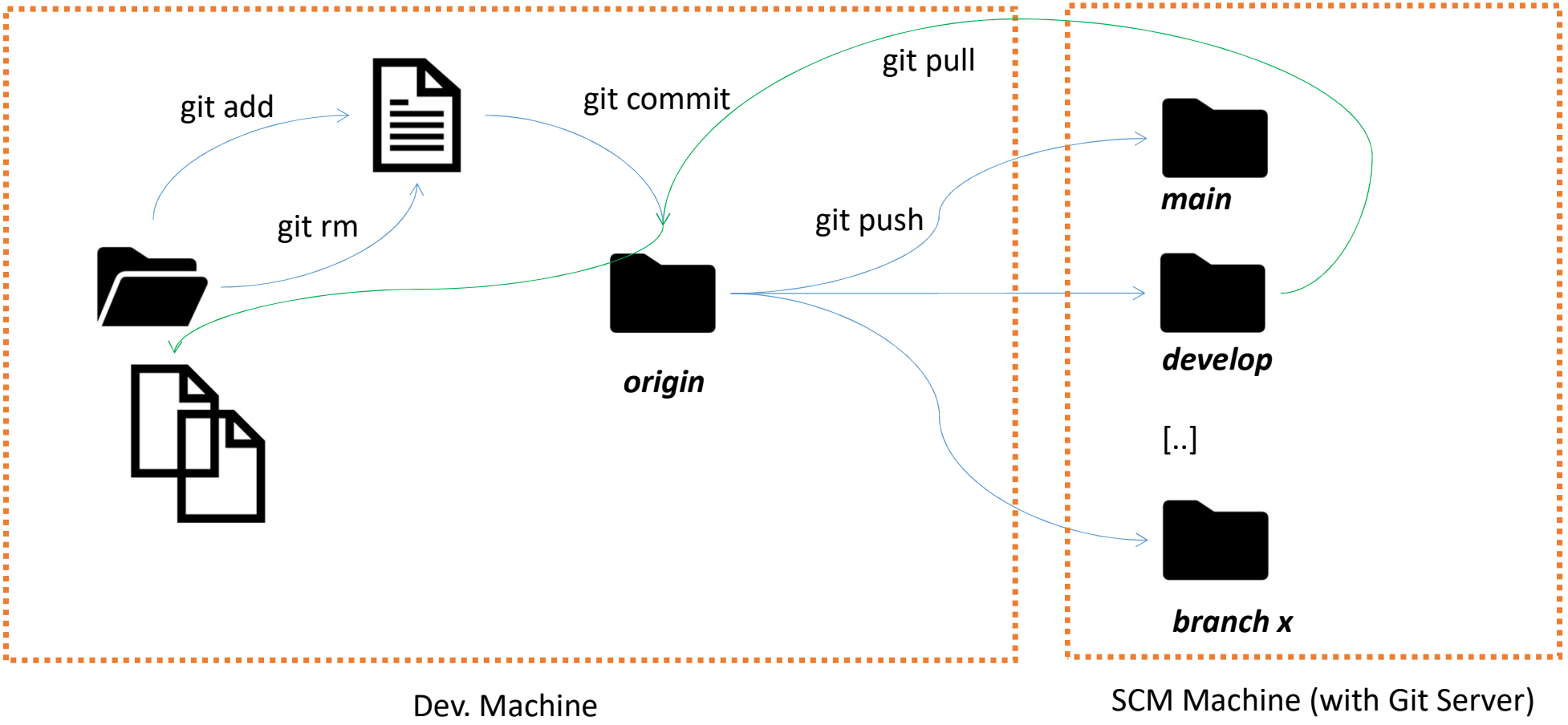
- A second dev. is joining (Bob)
  - He needs the existing file (src c)
  - He can copy it from Alice project's directory
- Life goes on
  - Alice creates src g
  - Bob creates src t
  - Maybe one of them change src c
- HOW TO SYNC !?

Share for collaboration

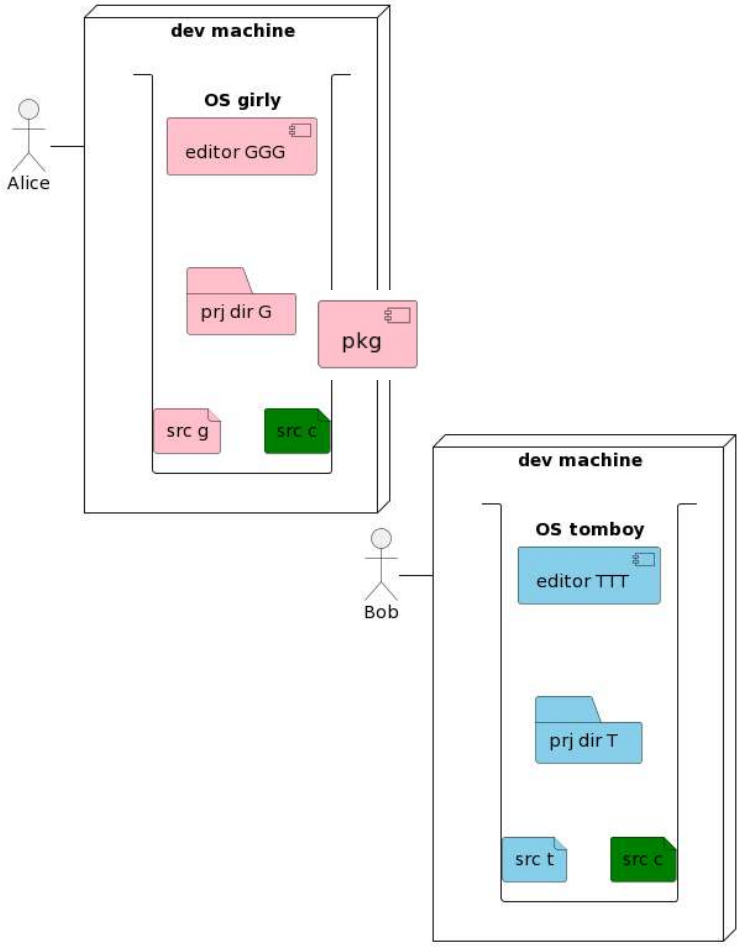


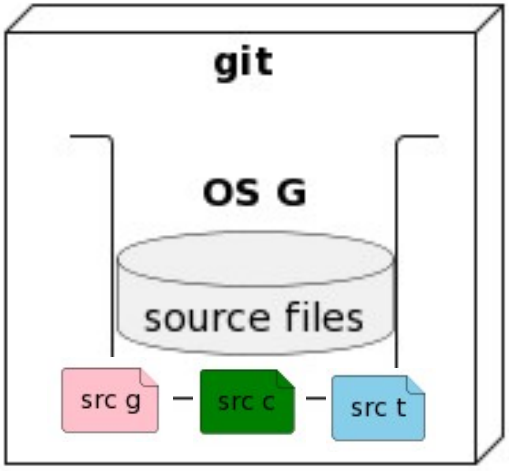
- Let SCM come into play
  - Source Code Manager
    - State of the art
      - Git
    - Older
      - SVN, CVS

Share – zoom on git



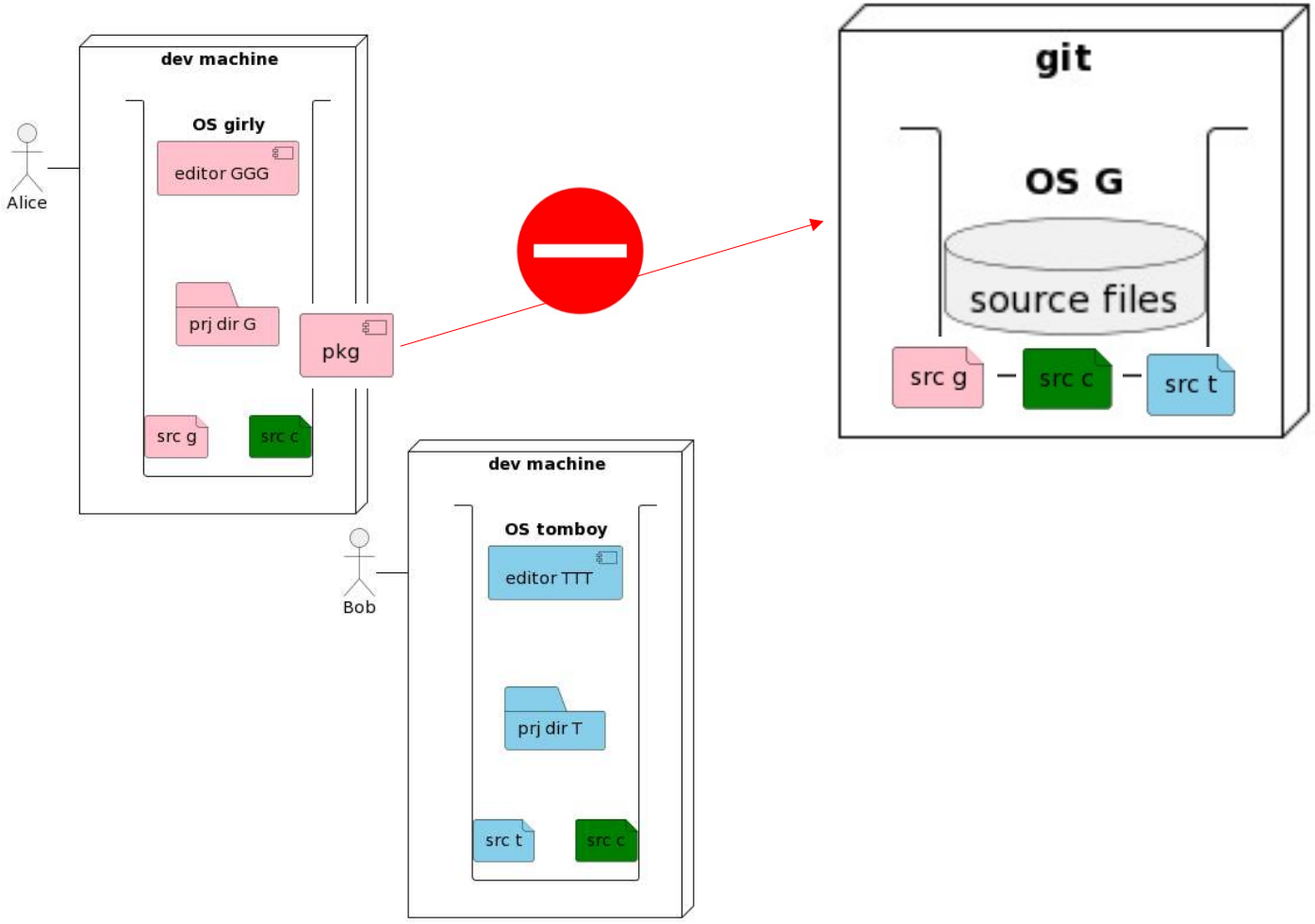
Share dependencies





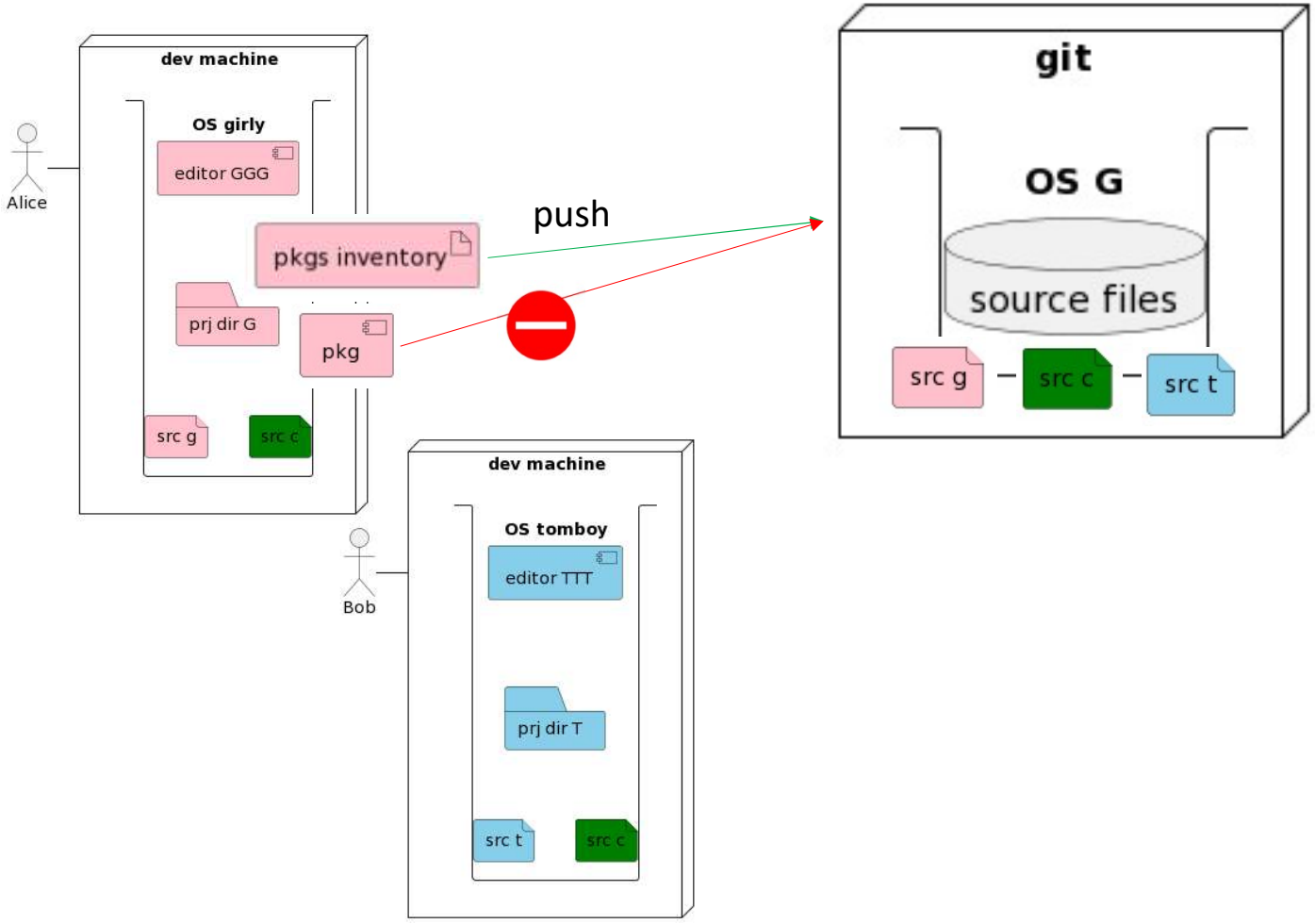
- Alice a new dependency (an external package : ie JavaFX libraries)

Share dependencies



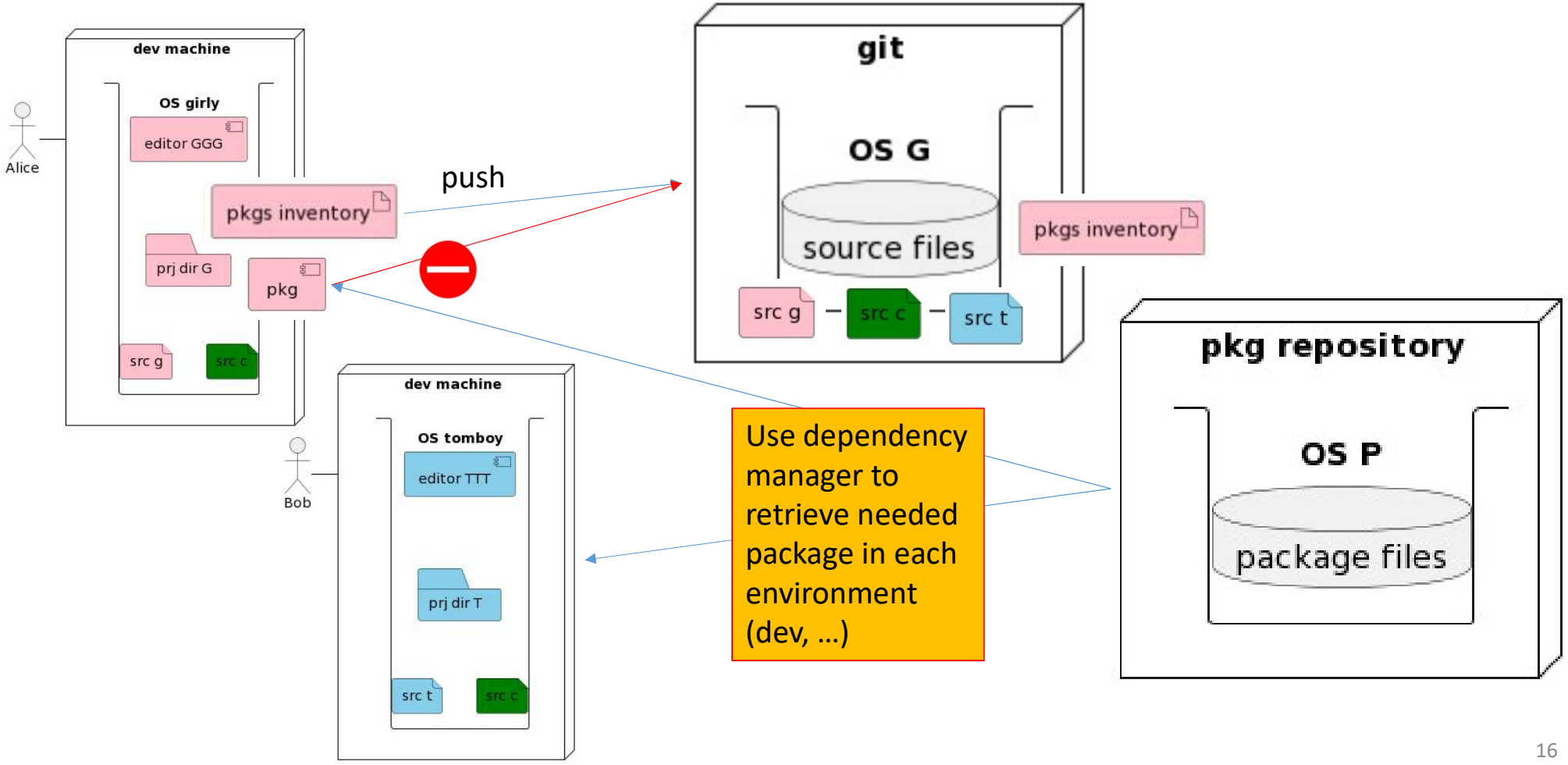
- Not in the scope of git that hosts only the content produced by the team
- NO

Share dependencies



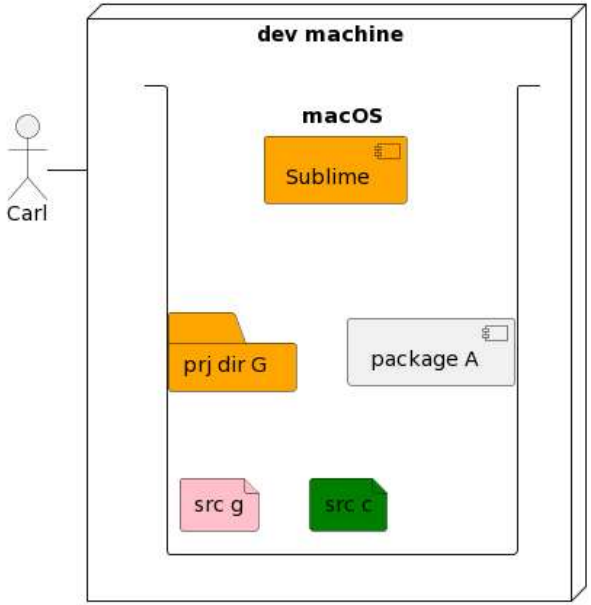
- But the list of package is a production of the team => do put it in git.

Share dependencies



Share dependencies – Python example

- When joining a dev. Project, two operations
  - Get the source code
    - Example command
    - > git clone https://gitlab.insa-rouen.fr/delestre/mgpi-tp-cicd
  - Get the dependencies
    - Example command
    - > pip install -r requirements.txt

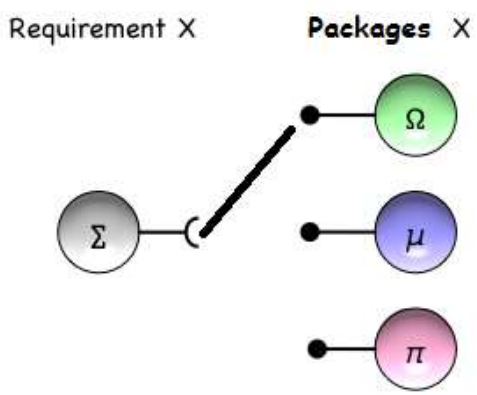


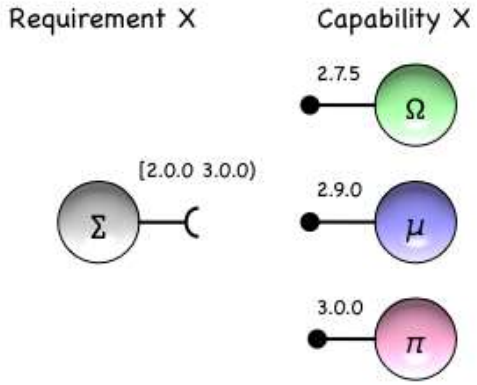
Share dependencies - Package Managers

	Tool	Dependencies file	Elements	
Java	maven	pom.xml	package	https://repo.maven.apache.org/maven2/
JavaScript (NodeJS)	npm	package.json	package	https://registry.npmjs.org/
Ruby	bundle	.Gemspec	gem	https://rubygems.org/
Python	pip	requirements.txt	module, package	

Share dependencies - semver

~~MAJOR.MINOR.PATCH~~
BREAKING.FEATURE.FIX

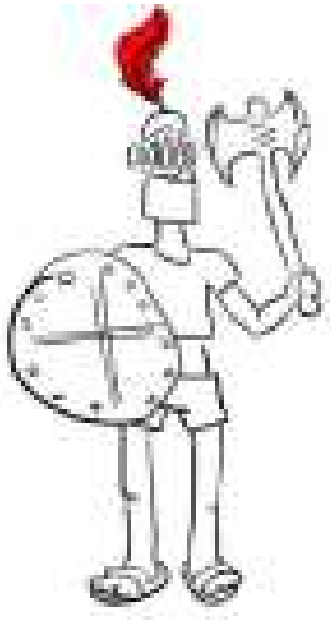




1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards-compatible manner, and
3. PATCH version when you make backwards-compatible bug fixes.

Image-credit : https://www.osgi.org/developer/modularity/

Share dependencies – semver – EXA7124B



http://prodageo.insa-rouen.fr/casi4publi/fboodle/EXA7124B.enonce.pdf

Share dependencies – prod vs dev

- Some dependencies are for machines **running** the deliverables
  - Utilities packages
  - Frameworks
  - Application Servers
- Others are for machines **building** the deliverables
  - Tools
    - Linters : **static code** analysis tool used to flag programming errors, bugs, stylistic errors and suspicious constructs.
    - Source generators : generate code based on a higher-level language specification (REST server from Swagger spec, …)

Share dependencies – lint example

Explicit dependency

```
"pylint": {
    "hashes": [
        "sha256:ea82cd6a1e11062dc86d555d07c021b0fb65afe39becbe6fe692efd6c4a67443",
        "sha256:ec4a87c33da054ab86a6c79afa6771dc8765cb5631620053e727fcf3ef8cbed7"
    ],
    "index": "pypi",
    "version": "==2.15.8"
},
```

Dev. Console

```
$ pylint --rcfile=.pylintrc huffman

huffman/compteur.py:15:22: C0201: Consider iterating the dictionary
directly instead of calling .keys() (consider-iterating-dictionary)
```

Source code not compliant

```
    def incrementer(self, element):
        """ méthode qui incrémente le nom d'un élément """
        if element in self._stat.keys():
            self._stat[element] = self._stat[element] + 1
huffman/compteur.py                                  15,22         17%
```

Pylint doc

consider-iterating-dictionary / C0201

Message emitted:

Consider iterating the dictionary directly instead of calling .keys()

Description:

Emitted when the keys of a dictionary are iterated through the ``.keys()`` method or when ``.keys()`` is used for a membership check. It is enough to iterate through the dictionary itself, ``for key in dictionary``. For membership checks, ``if key in dictionary`` is faster.

Problematic code:

```
FRUITS = {"apple": 1, "pear": 5, "peach": 10}


for fruit in FRUITS.keys():  # [consider-iterating-dictionary]
    print(fruit)
```

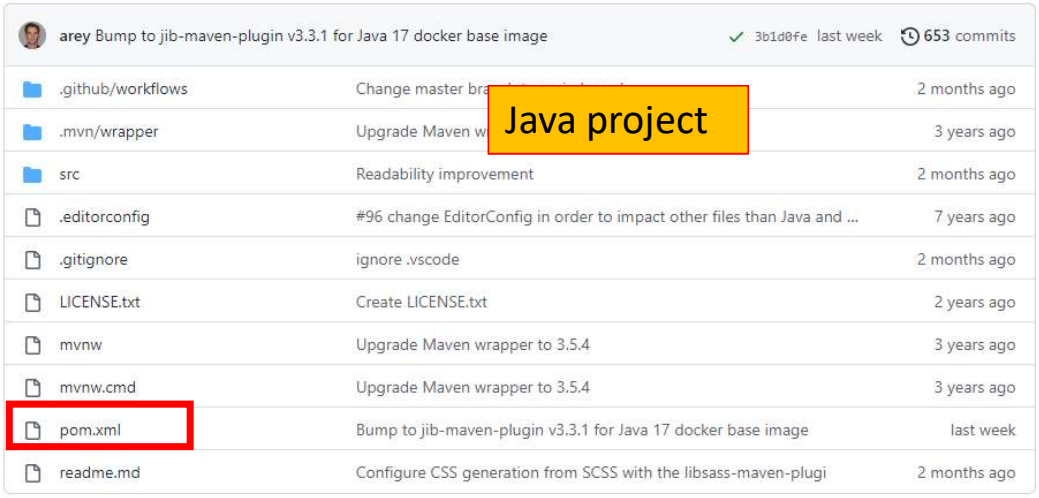
Correct code:

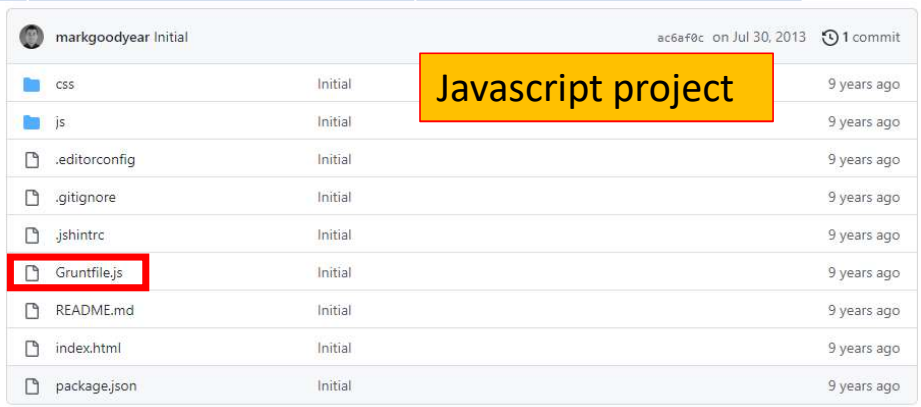
```
FRUITS = {"apple": 1, "pear": 5, "peach": 10}


for fruit in FRUITS:
    print(fruit)
```

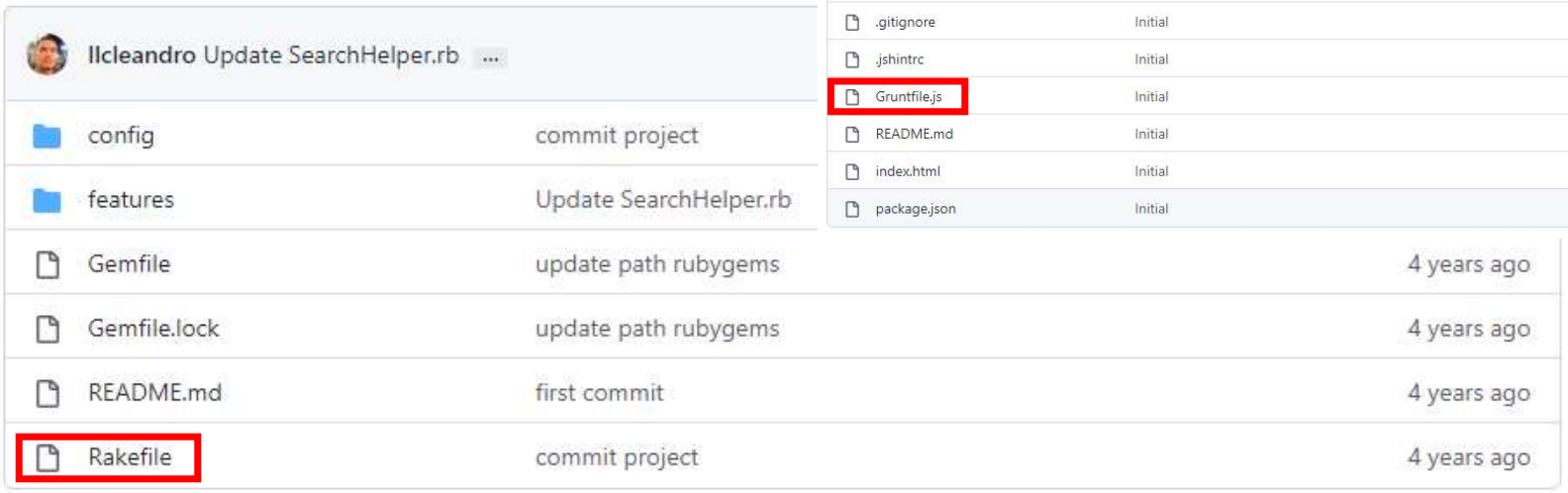
Source : https://pylint.pycqa.org/

Task Runners

	Tool	Default sc
Java	maven	pom.xml
JavaScript (NodeJS)	grunt	Gruntfile.js
Ruby	rake	Rakefile
Python		



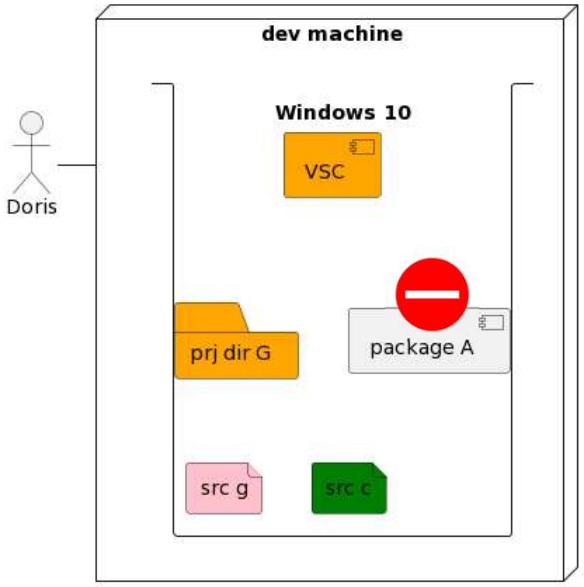


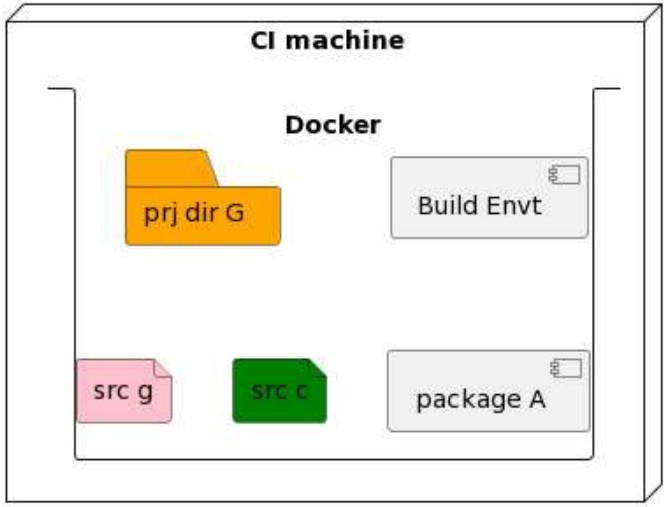


Help in :
- Running tests
- Init. DB

Remote build

- Doris is joining the team with a Windows OS.
- Some of the dev dependencies are not available on this OS !

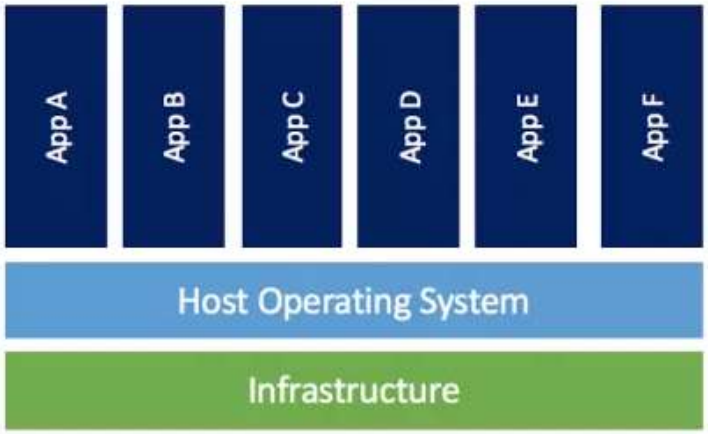




Virtualisation

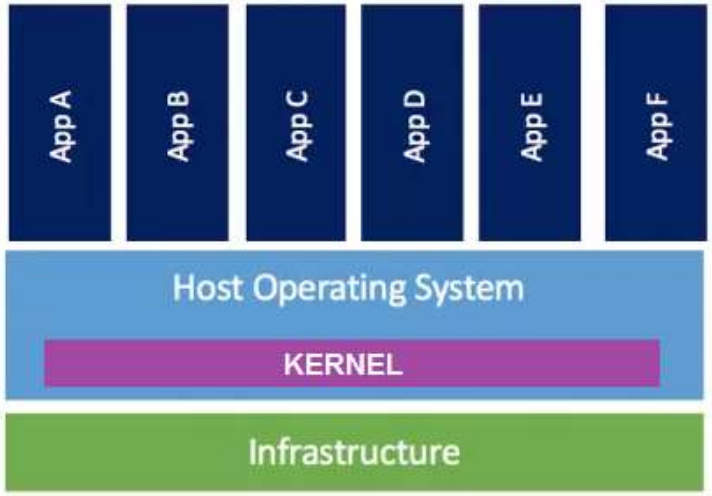
Piece of knowledge to understand the software factory line

Without virtualisation



- Applications are setup in a « Host Operating System » that gives access to resources
  - File System
  - Network
  - …
- Host Operating System adapt a physical infrastructure.

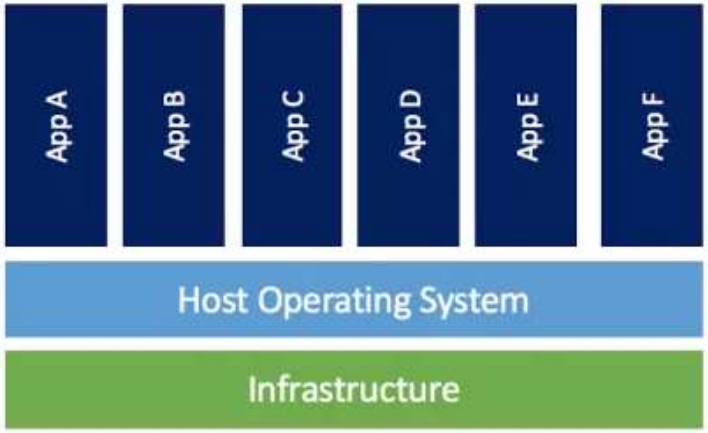
Kernel concept

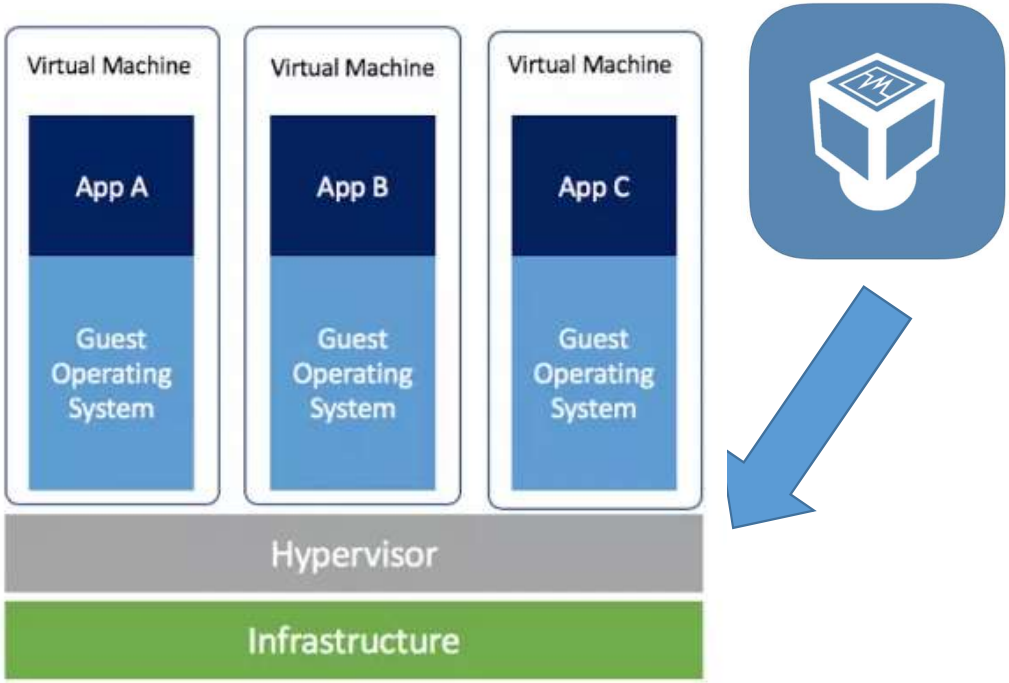


- The kernel is the important program in the operating system

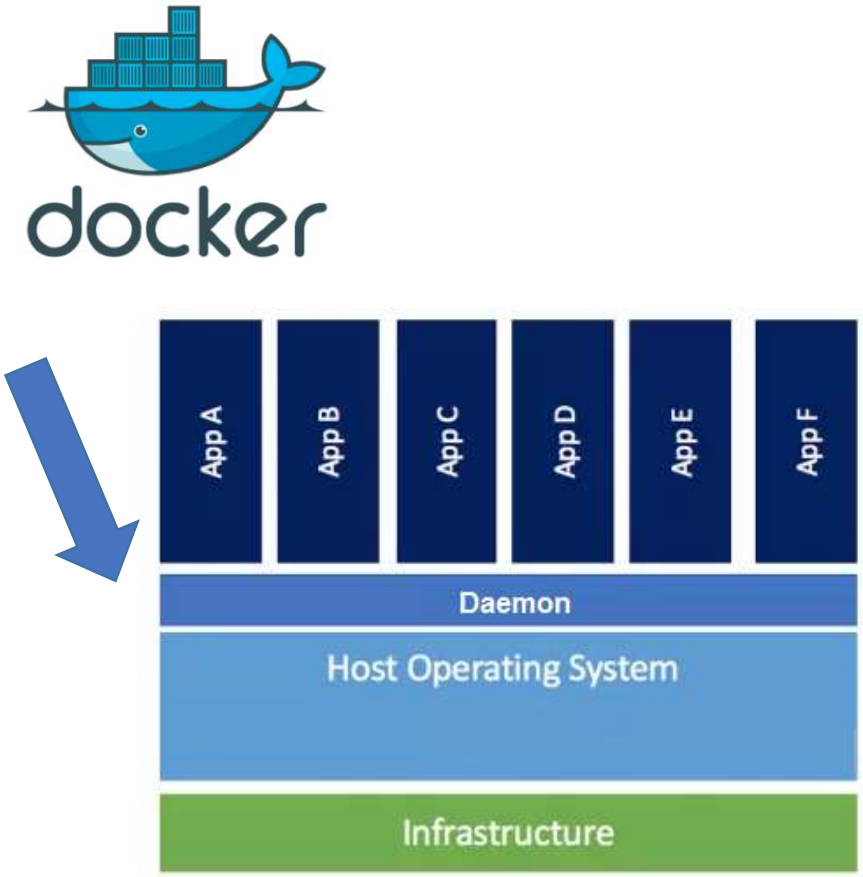
BASIS FOR COMPARISON	KERNEL	OPERATING SYSTEM
Basic	Kernel is an important part of the operating system.	Operating System is a system program.
Interface	Kernel is an interface between software and hardware of the computer.	Operating System is an interface between user and hardware of the computer.

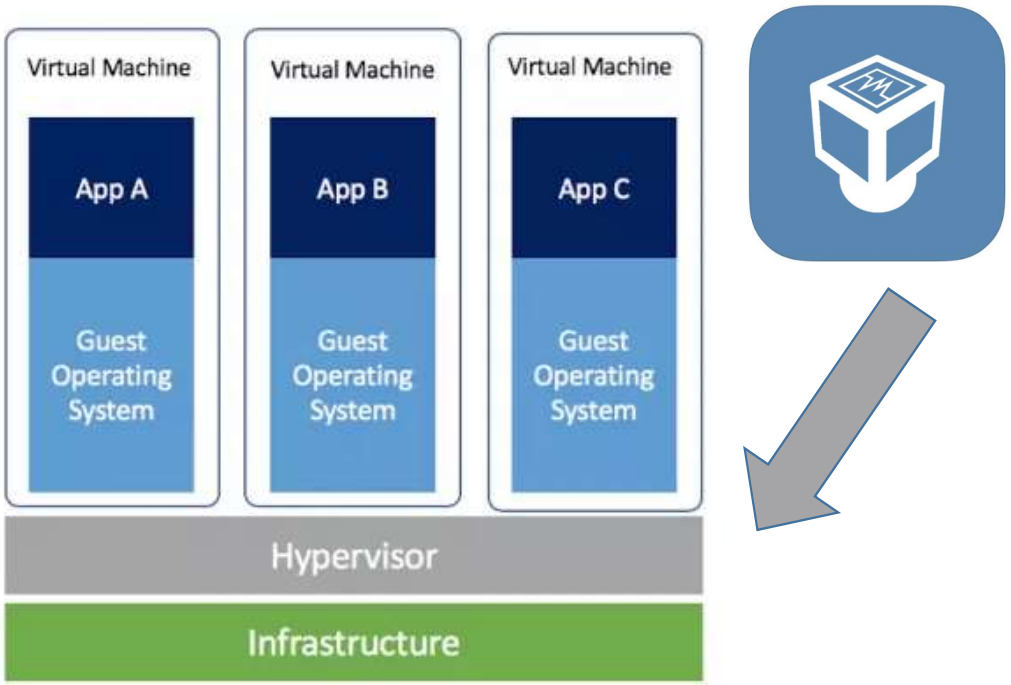
Physical Machine vs Hypervisor



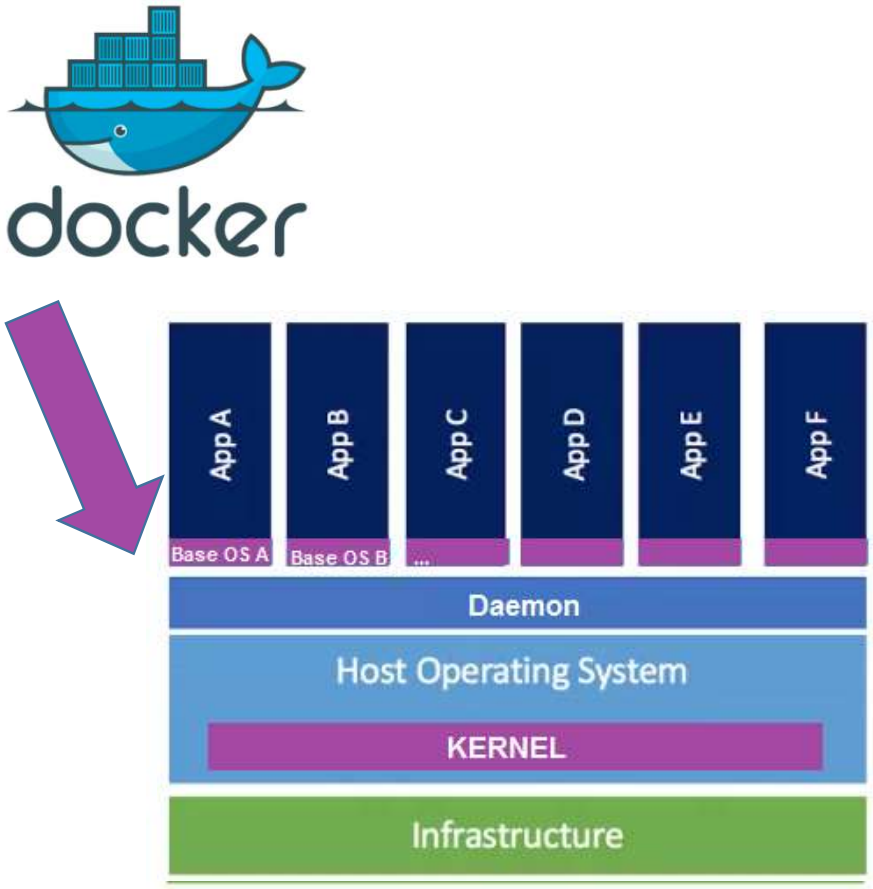


Containers versus Hypervisor



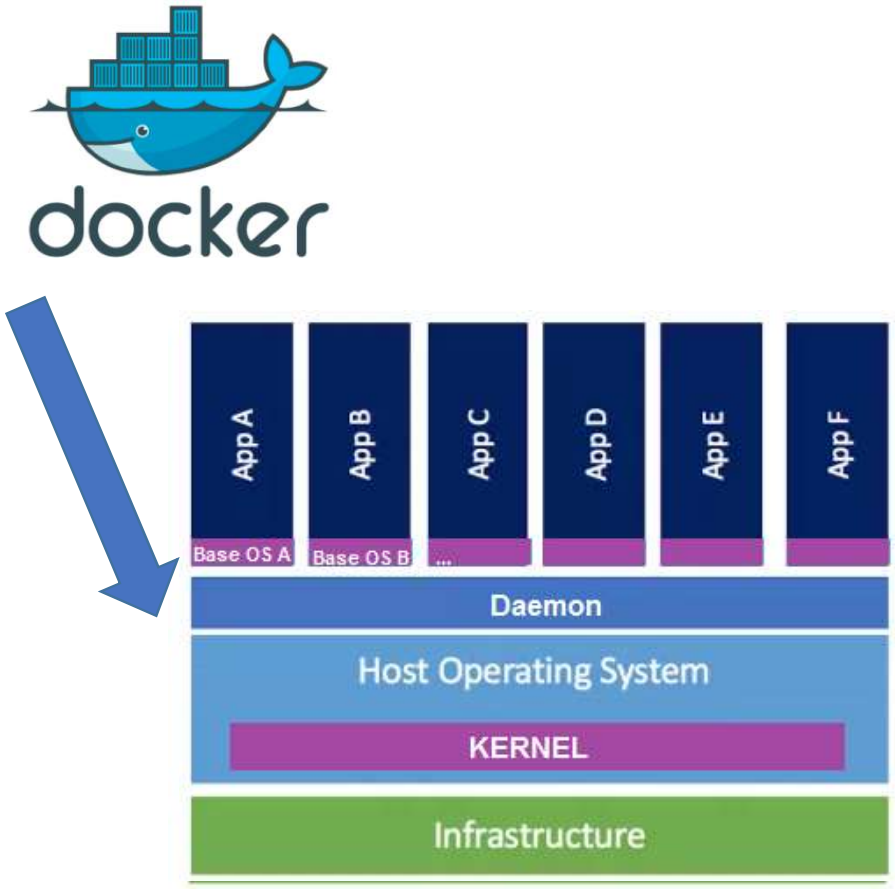


Containers : « Base OS », « kernel »



- Container embeds
  - One or several application(s)
  - A *Base OS*
- « *Base OS* » interact with « *kernel* » of « *Host Operating System* »
- Note : in specific cases, some apps are allowed to interact directly with kernel (by-passing « Base OS »).

Containers : « daemon »

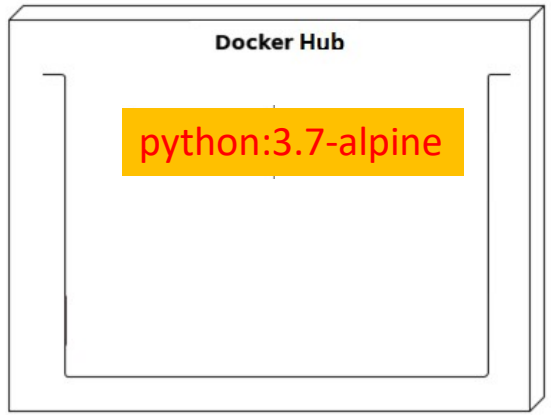


- « daemon » is in charge of managing container
  - Starting
  - Stopping
  - …

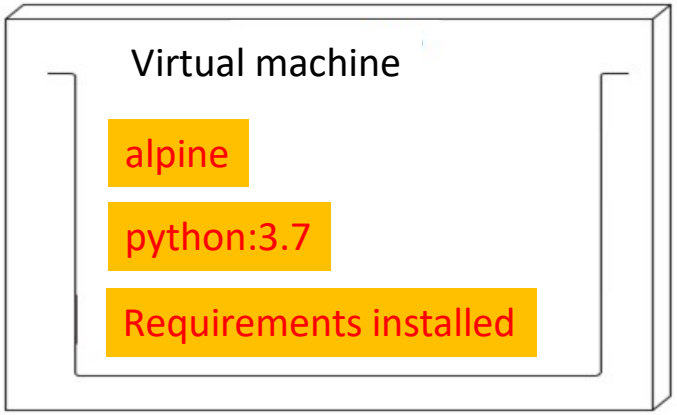
Dockerfile : installation fully scripted

Dockerfile

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```



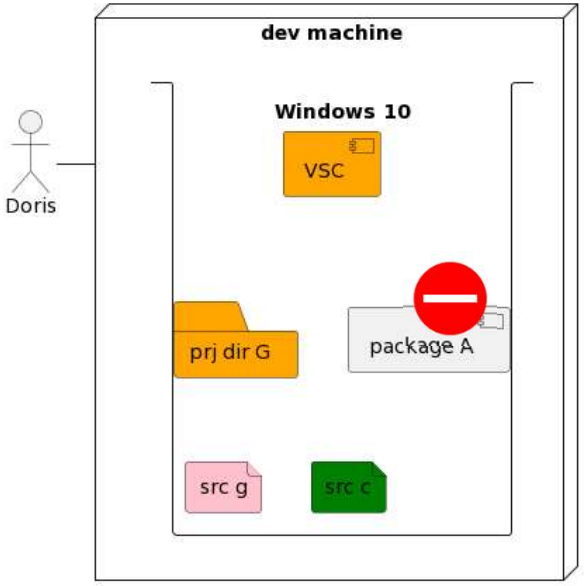
> docker compose up

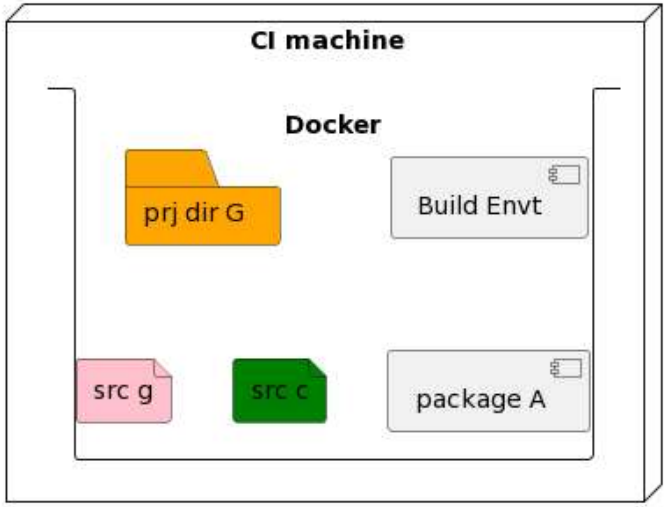


Continuous Integration

Remote build

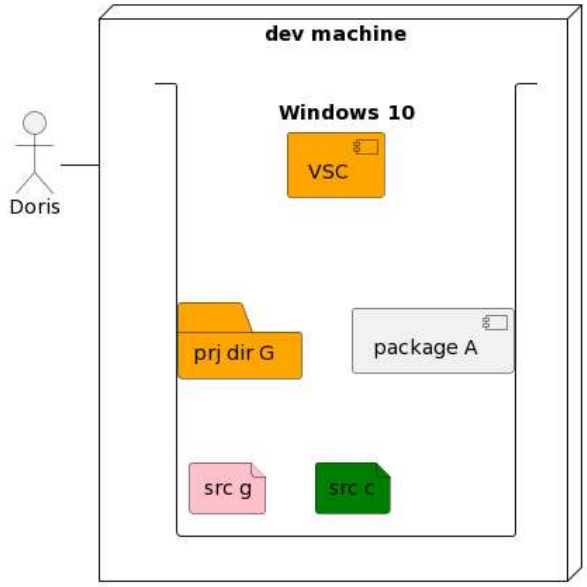
- Doris is joining the team with a Windows OS.
- Some of the dev dependencies are not available on this OS !

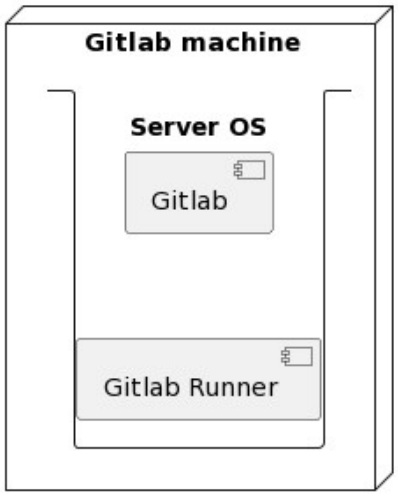


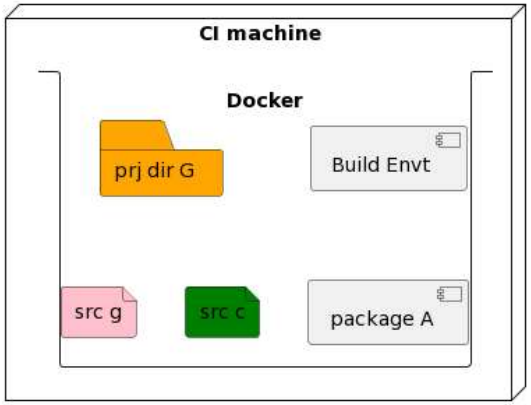


Pipeline

- Based on an event, Gitlab Runner will trigger the **CREATION** of a virtual machine to run the job(s) of the CI platform !
- Event (CI_PIPELINE_SOURCE)
  - push
  - merge_request_event
  - scheduled







First pipeline : failed

Edit | Visualize | Validate NEW | View merged YAML

Browse templates | Help

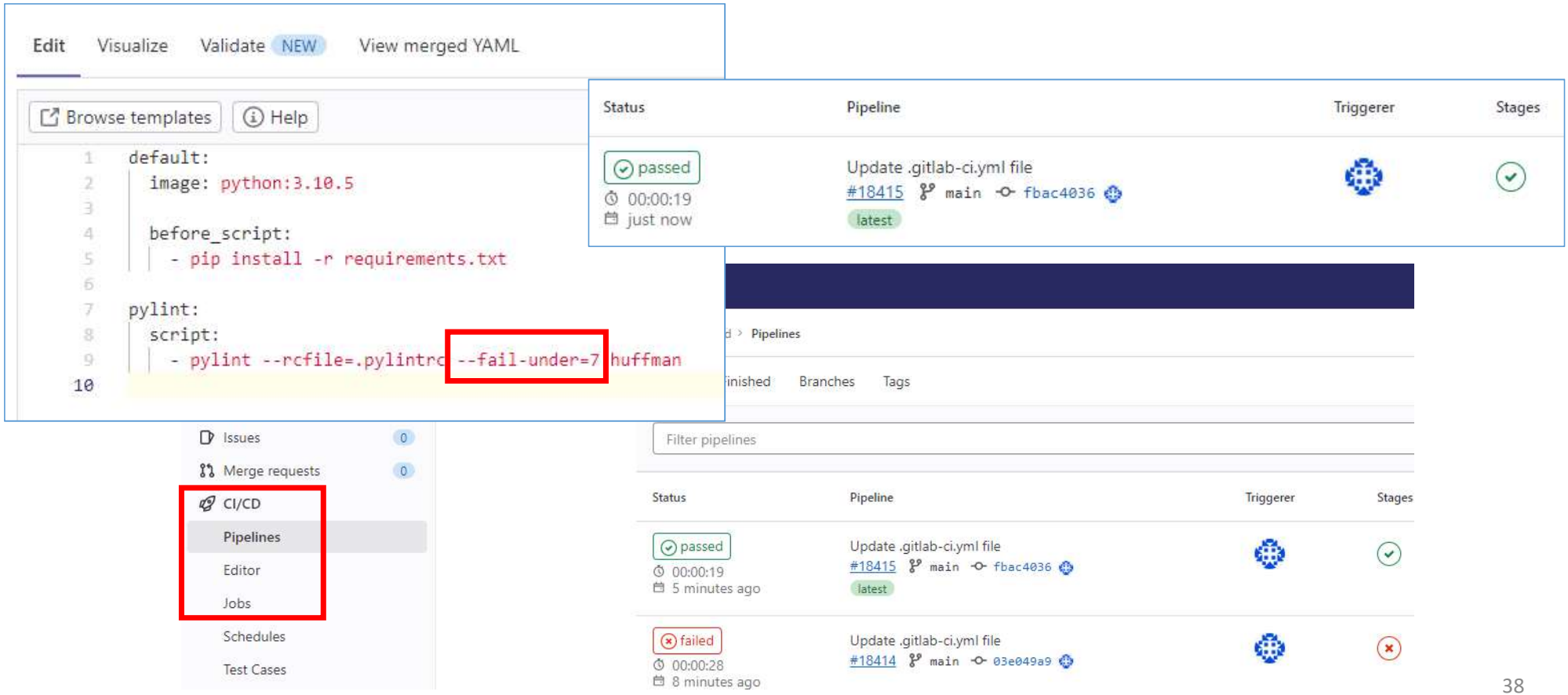
```yaml
default:
  image: python:3.10.5

  before_script:
    - pip install -r requirements.txt

pylint:
  script:
    - pylint --rcfile=.pylintrc huffman
```

failed
00:00:28
3 minutes ago

Update .gitlab-ci.yml file
#18414 main 03e049a9

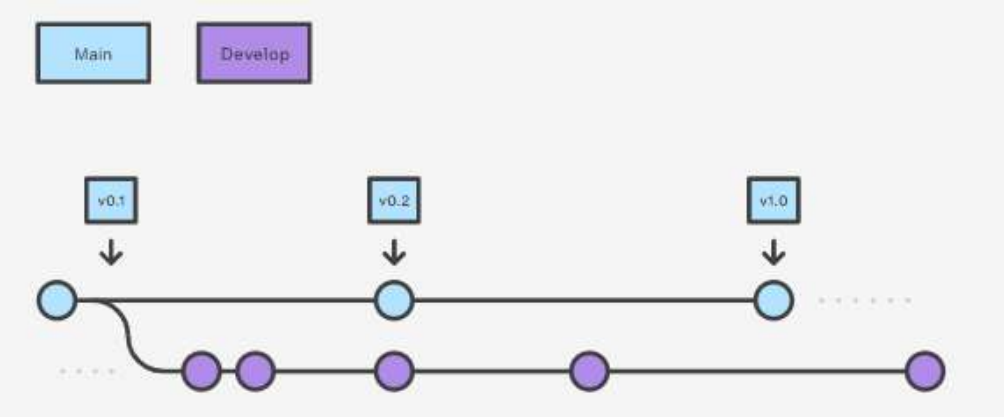
```
ompteur' inherits from object, can be safely removed from bases in python3 (useless-object-inheritance)
er iterating the dictionary directly instead of calling .keys() (consider-iterating-dictionary)
er iterating the dictionary directly instead of calling .keys() (consider-iterating-dictionary)
89 huffman/compteur.py:40:0: C0206: Consider iterating with .items() (consider-using-dict-items)
90 huffman/compteur.py:44:54: W0108: Lambda may not be necessary (unnecessary-lambda)
91 huffman/compteur.py:48:54: W0108: Lambda may not be necessary (unnecessary-lambda)
92 huffman/compteur.py:53:29: R1718: Consider using a set comprehension (consider-using-set-comprehension)
93 huffman/compteur.py:55:16: R1718: Consider using a set comprehension (consider-using-set-comprehension)
94 huffman/compteur.py:59:15: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
95 ----------------------------------
96 Your code has been rated at 7.63/10
98 Cleaning up project directory and file based variables
100 ERROR: Job failed: exit code 1
```

First pipeline : success



Some gouvernance on the pipeline

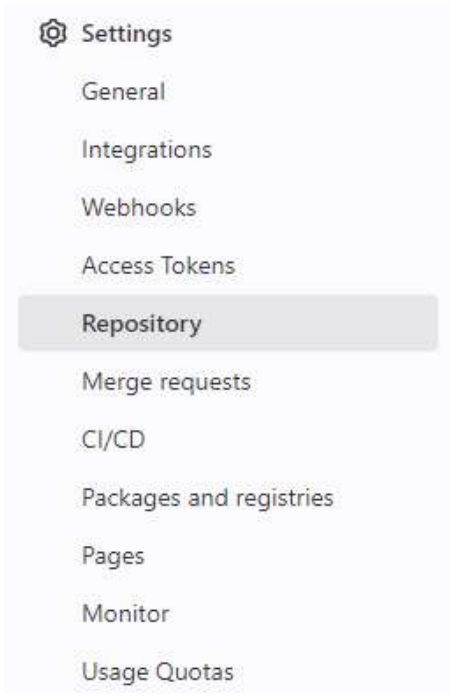
- With 4 persons working on the project, Alice wants to have control on the source code introduced in the **main** branch
  - => she set a new branch : **develop**
- By default, a repository is created with **one branch** named **main**
  - Note : **master** has been eradicated



Merge Request versus Push

- Push
  - The developper **decides** to add the source code to a branch
- Merge Request
  - Other designated people (Alice) **do decide**
  - Possible to insertion additional control operation

Gitlab – restrict access to a branch



Branch	Allowed to merge	Allowed to push	Allowed to force push	Code owner approval	
main default	Maintainers	No one			Unprotect

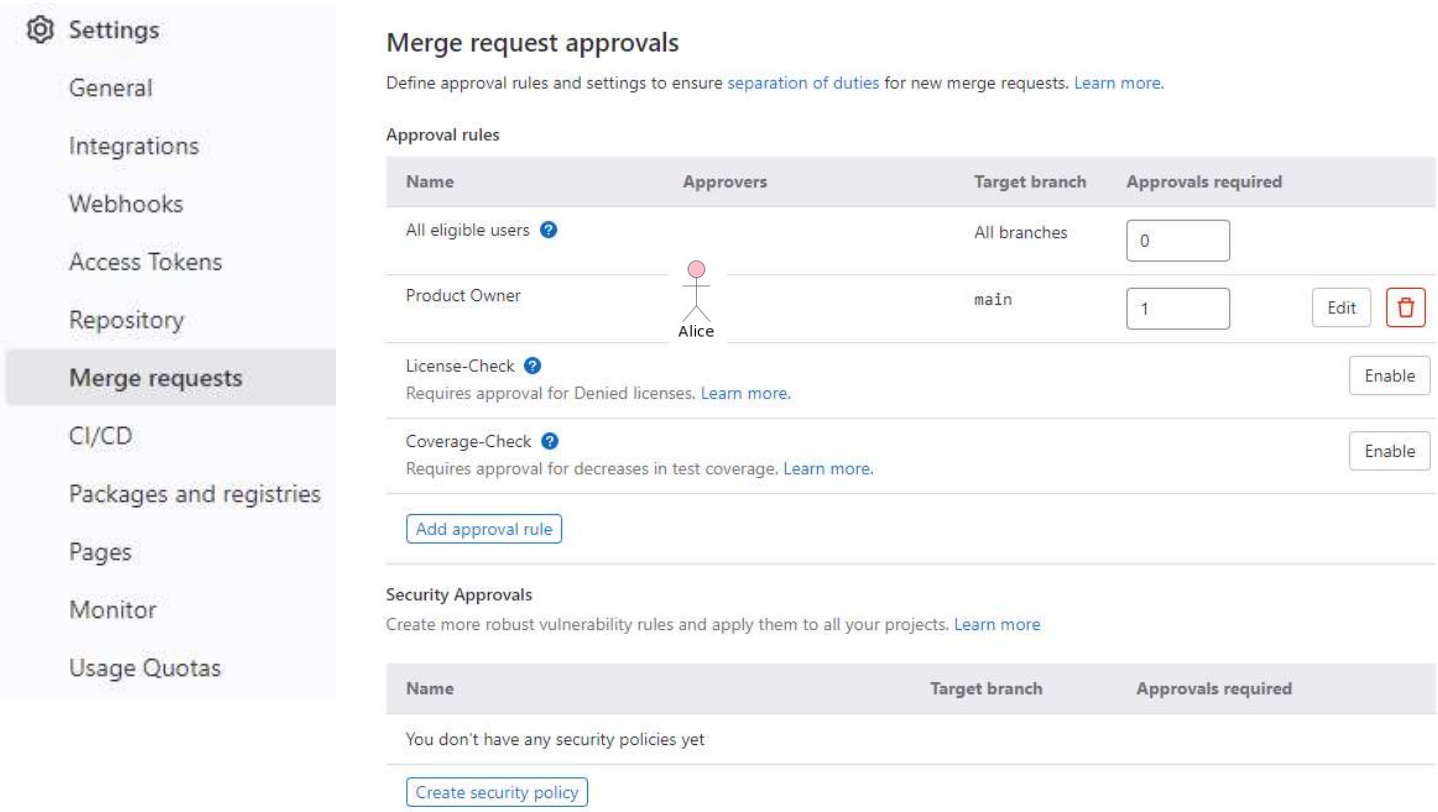
```
C:\Apps\laragon\www\cicd (main)
λ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 555 bytes | 555.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: GitLab: You are not allowed to push code to protected branches on this project.
To https://gitlab.insa-rouen.fr/prodageo/cicd.git
 ! [remote rejected] main -> main (pre-receive hook declined)
error: failed to push some refs to 'https://gitlab.insa-rouen.fr/prodageo/cicd.git'
```

Sync with the develop branch

- Bob can continue to work
- He switched to the develop branch
- And now, push is possible.

```
C:\Apps\laragon\www\cicd (main)
λ git checkout develop
Switched to a new branch 'develop'
Branch 'develop' set up to track remote branch 'develop' from 'origin'.

C:\Apps\laragon\www\cicd (develop)
λ git push origin develop
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 307 bytes | 307.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for develop, visit:
remote:   https://gitlab.insa-rouen.fr/prodageo/cicd/-/merge_requ
remote:
To https://gitlab.insa-rouen.fr/prodageo/cicd.git
   fbac403..855db7d  develop -> develop
```

Gitlab – approval permissions rules



Gitlab – operations specific to Merge

Browse templates | Help

```
default:
  image: python:3.10.5

  before_script:
    - pip install -r requirements.txt

pylint:
  script:
    - pylint --rcfile=.pylintrc --fail-under=7 huffman

job1:
  script:
    - echo "This job runs in merge request pipelines"
  rules:
    - if: $CI_PIPELINE_SOURCE == 'merge_request_event'
```

Pipelines - Stages

```
stages:
  - tests
  - static analysis
  - deploy
```

Stage tests

j1a j1b j1c

Stage static analysis

j2a j2b

Stage deploy

j3a j3b j3c

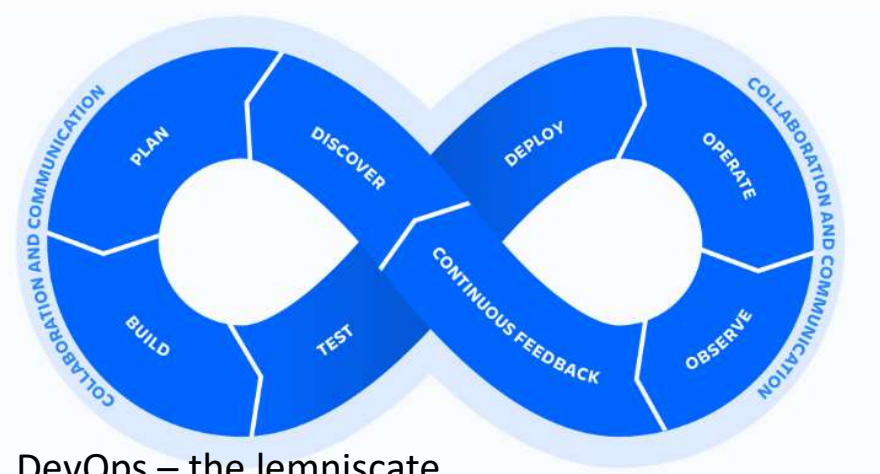
Pipelines - Gitlab Pages

```yaml
image: ruby:2.7

pages:
  script:
    - gem install bundler
    - bundle install
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
```

- A special job (pages:deploy) is triggered after the pages job.
- Every files (HTML, …) generated within **public** directory are published on the URL indicated in Settings > Pages
  - https://account.pages.insa-rouen.fr/reponame
    - account : name of the person or group
    - Reponame : name of the repository (project)

Conclusion : Continous Everything and DevOps

- Continuous
  - Integration : unit test, lint, …
    - Prepare and test the build
  - Delivery
    - Package the build in artefacts (deliverables like files, …)
  - Deployment
    - Copy and install the artefacts of the target environment (be production, pre-production, demo, …)



DevOps – the lemniscate
Source : https://www.atlassian.com/devops