#DevCon

Édition 2018

OpenFaaS & GitLab CI-CD

TAM - SA @ **GitLab**

🦫 @k33g_org 🦊 @k33g | pcharriere@gitlab.com

https://gitlab.com/k33g/q/issues

📷 @GDGLyon - 🏫 @ScalaIO_FR - 📝 @BlendWebMix

Agenda

FaaS - OpenFaaS

1ère fonction

GitLab Runners - CI-CD

OpenFaaS & GitLab CI

Déploiement / Review App / Multi pipelines

FaaS

“ServerLess 🤪”

FaaS

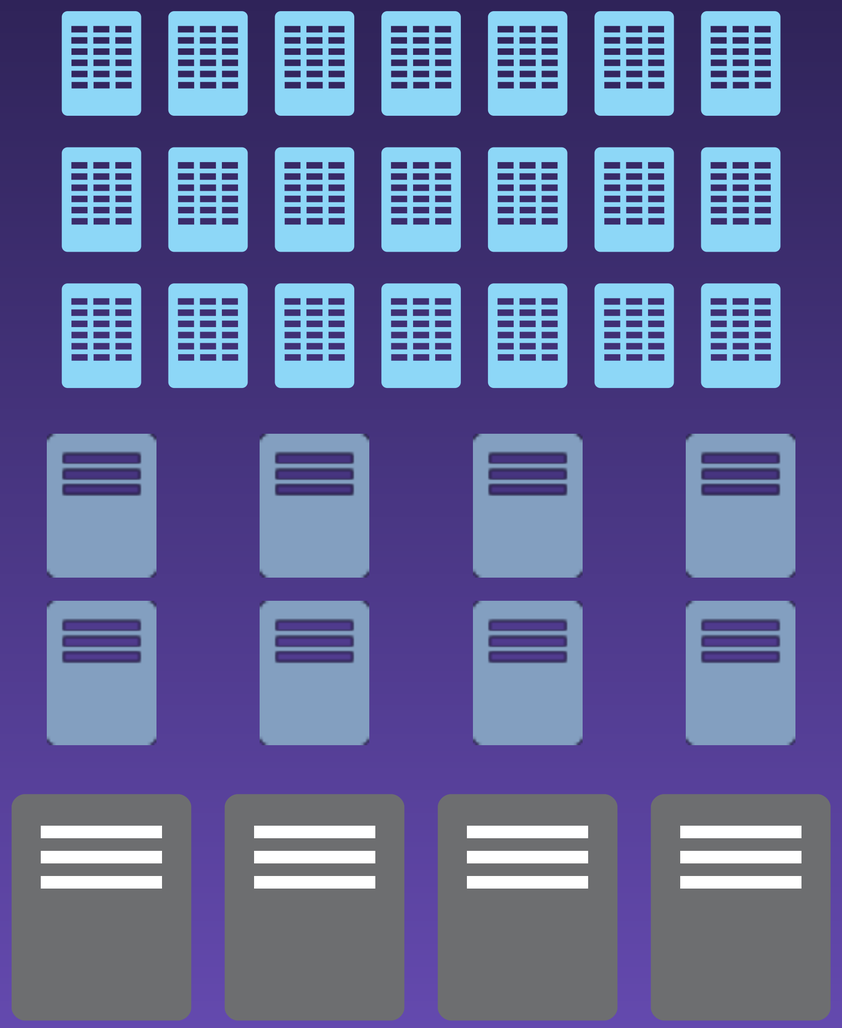
Développer (Tester) - Exécuter (Déployer…) - Gérer

les fonctionnalités d'une application

Sans la complexité

de l'infrastructure

“Une” architecture FaaS



Function Platform
(FaaS)

Container Platform
(CaaS)

Infrastructure Platform
(IaaS)

OpenFaaS

Framework ❤️

OpenFaaS?

1 framework faas on-premises

no vendor lock-in

facile à utiliser

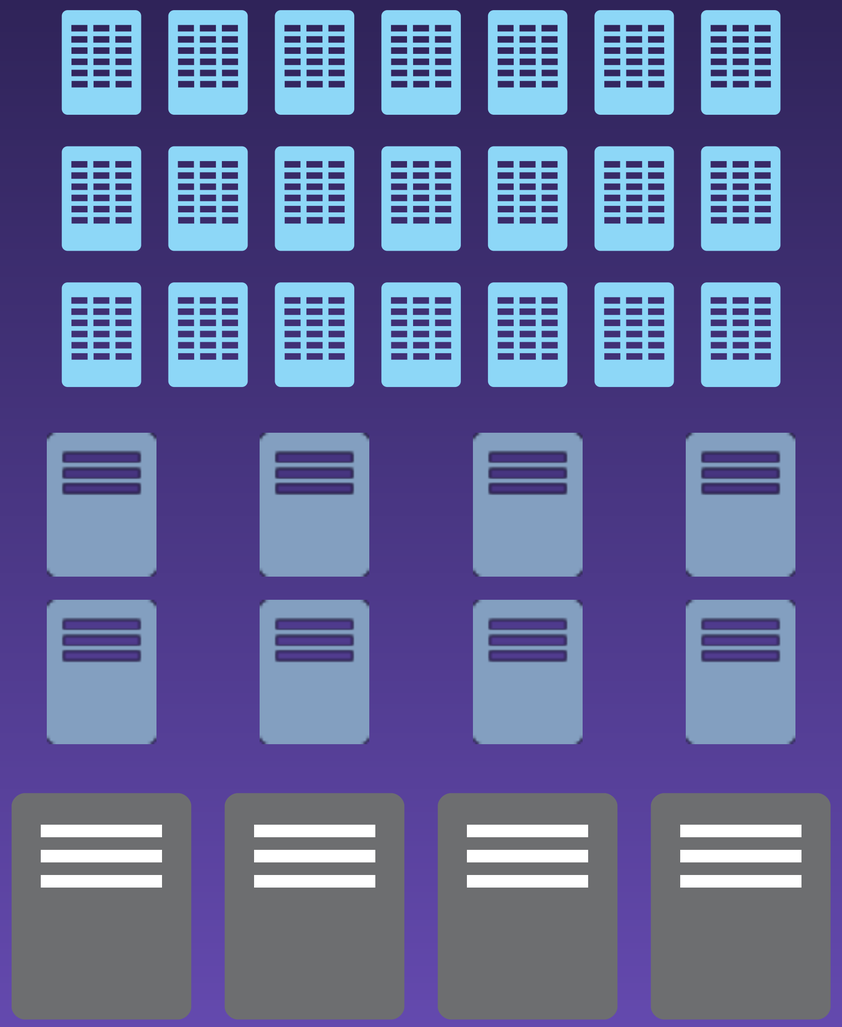
multi langages

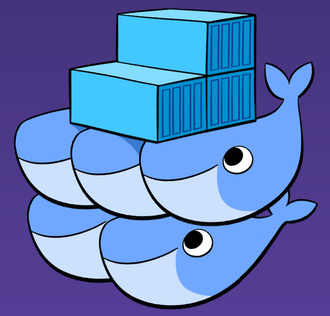
une communauté sympathique

Alex Ellis
@alexellisuk

https://www.openfaas.com

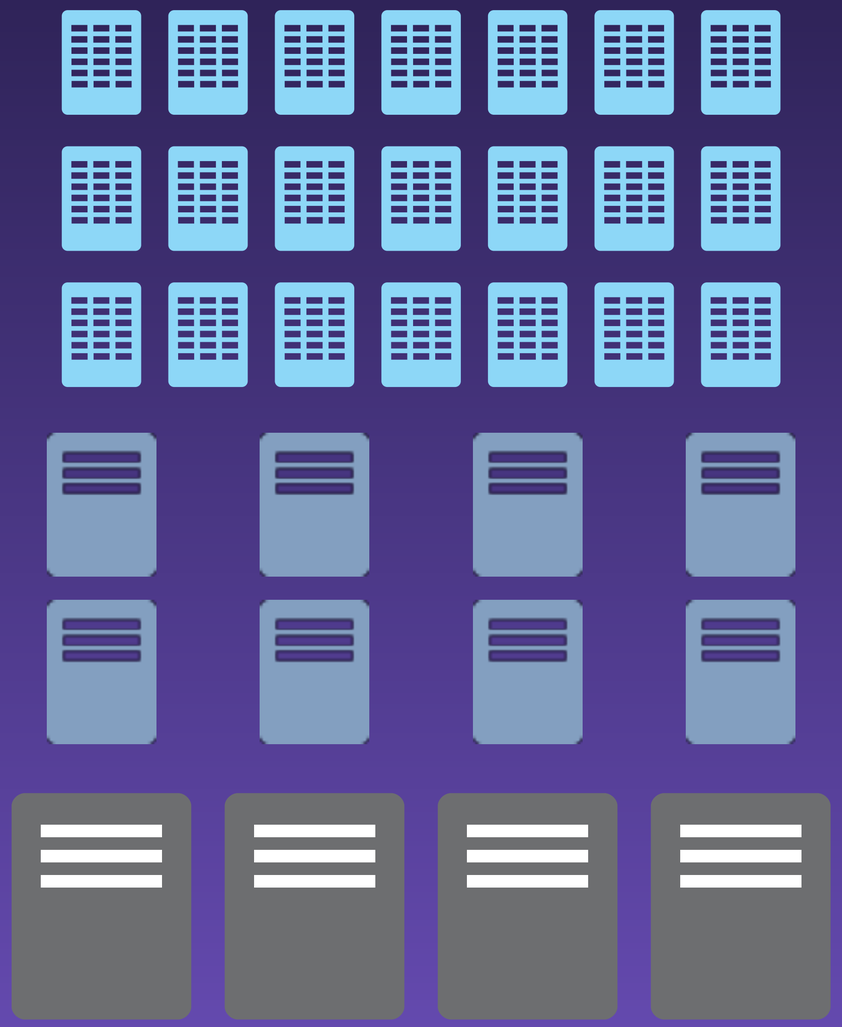
Une architecture OpenFaaS





My Laptop

Une autre architecture OpenFaaS



Your servers

Installation d'OpenFaaS (version Docker Swarm)

```
apt-get install -y docker-ce
# Add entries to hosts file:
echo "" >> /etc/hosts
echo '#{REGISTRY_IP} #{REGISTRY_DOMAIN}' >> /etc/hosts
echo "" >> /etc/hosts
docker swarm init --advertise-addr #{OPENFAAS_IP}
# Add unsecure registry
echo "" >> /etc/docker/daemon.json
echo '{' >> /etc/docker/daemon.json
echo '  "insecure-registries" : ["#{REGISTRY}"]' >> /etc/docker/daemon.json
echo '}' >> /etc/docker/daemon.json
echo "" >> /etc/docker/daemon.json

service docker restart

git clone https://github.com/openfaas/faas && \
cd faas && \
./deploy_stack.sh
```

Installation de la CLI OpenFaaS

```
apt-get install -y docker-ce
```

```
curl -sSL https://cli.openfaas.com | sudo -E sh
```

Demo Time - 1st f(x)

GitLab + GitLab CI

Cycle de vie projet

GitLab

Plan

Create

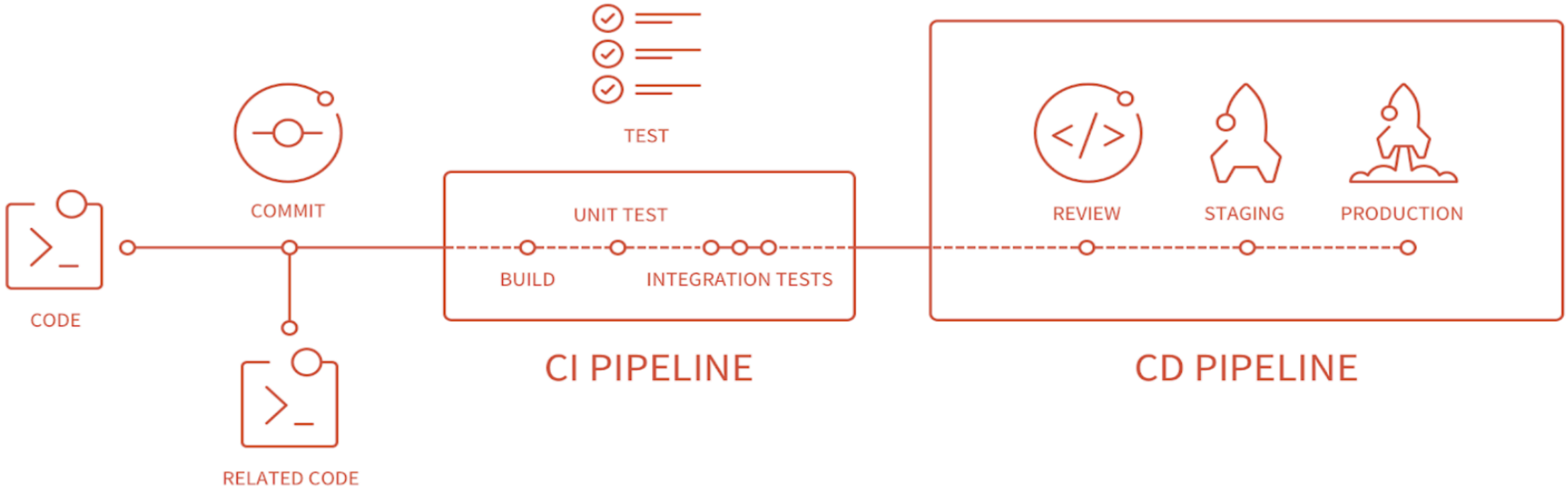
GitLab

Manage | Plan | Create | Verify | Package | Release | Configure | Monitor | Secure

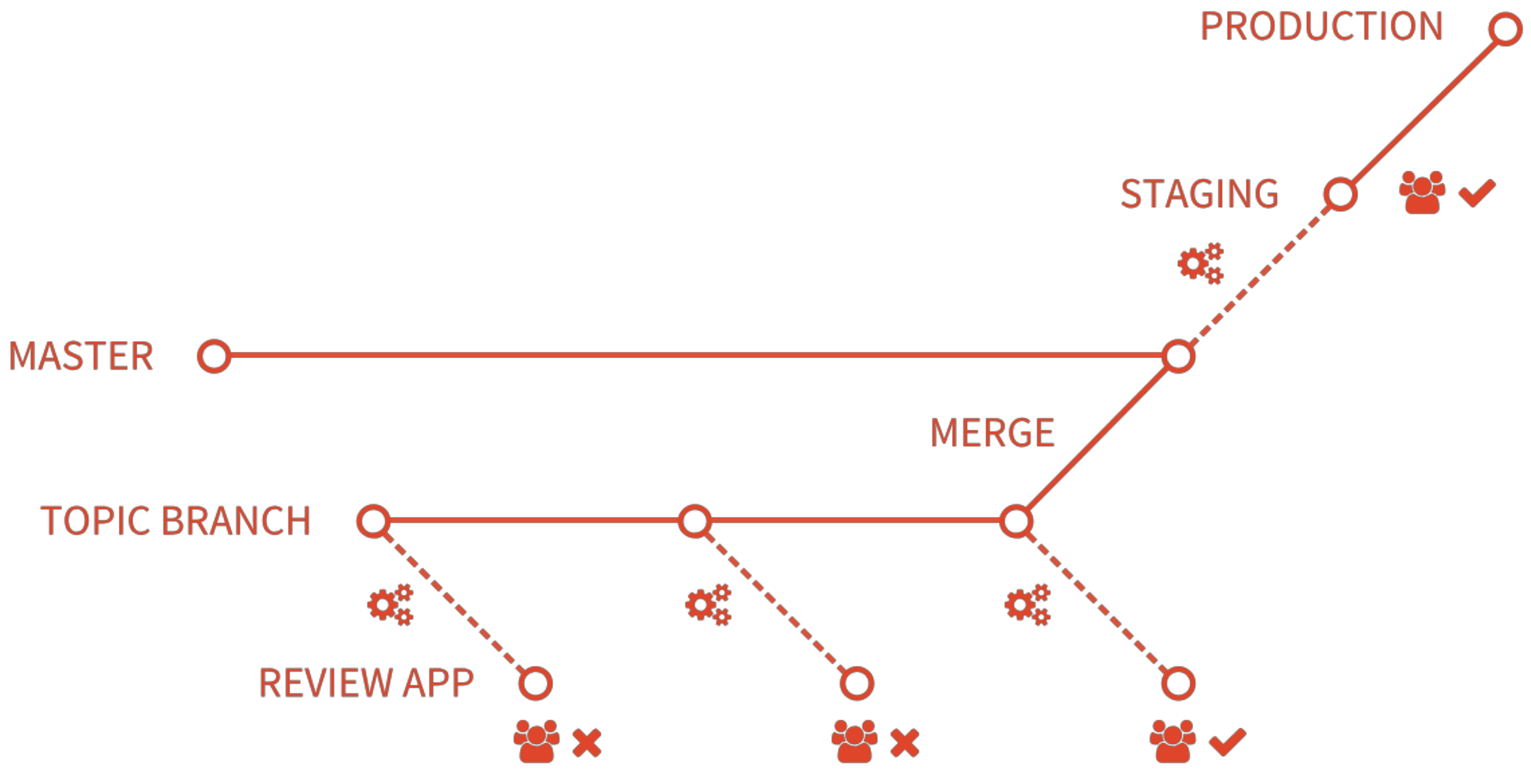
Continuous Integration

Continuous Delivery
Review Apps

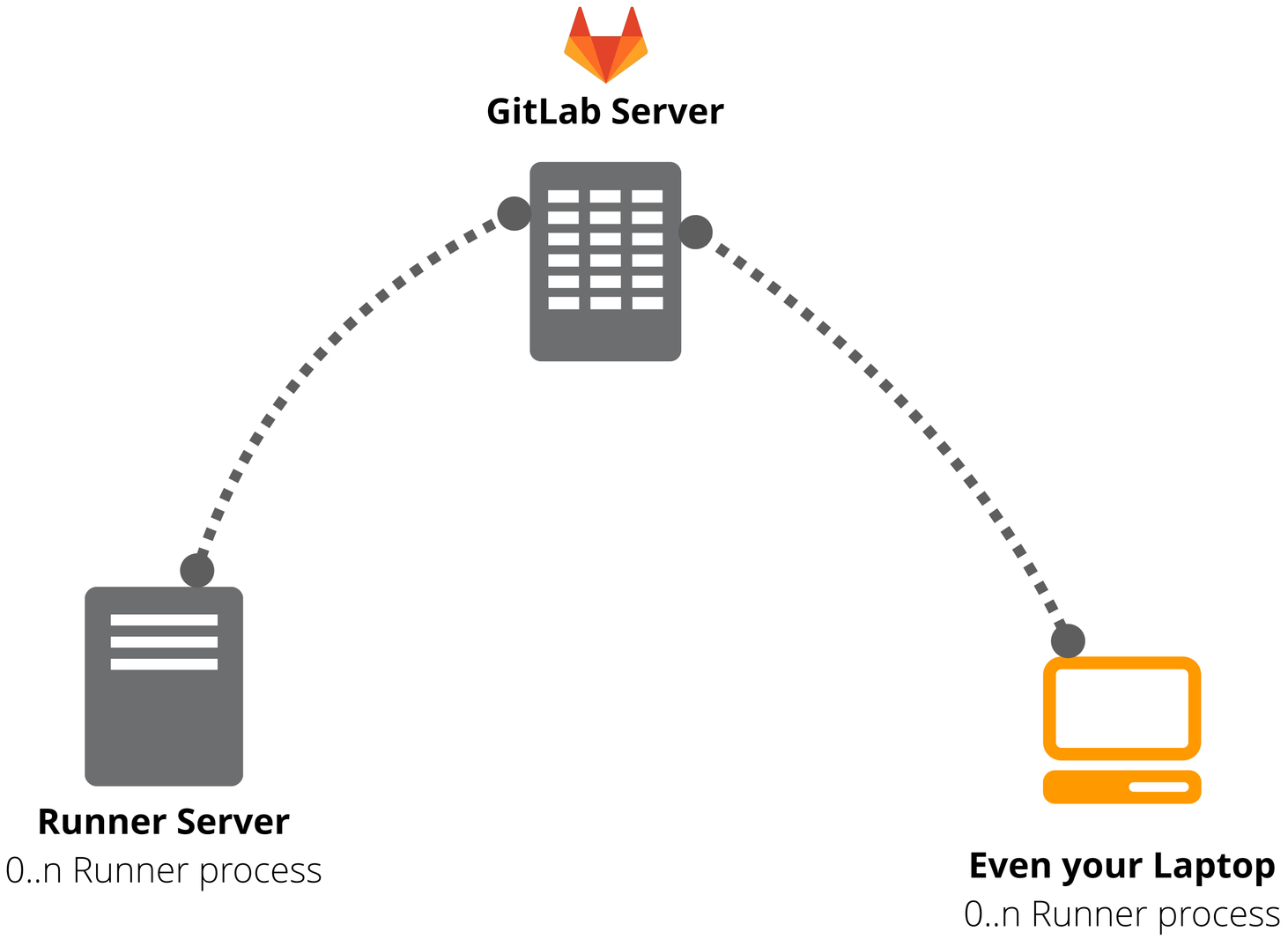
Project life cycle



Feature Flow



GitLab Runners


GitLab Server
Runner Server
0..n Runner process
Even your Laptop
0..n Runner process

.gitlab-ci.yml

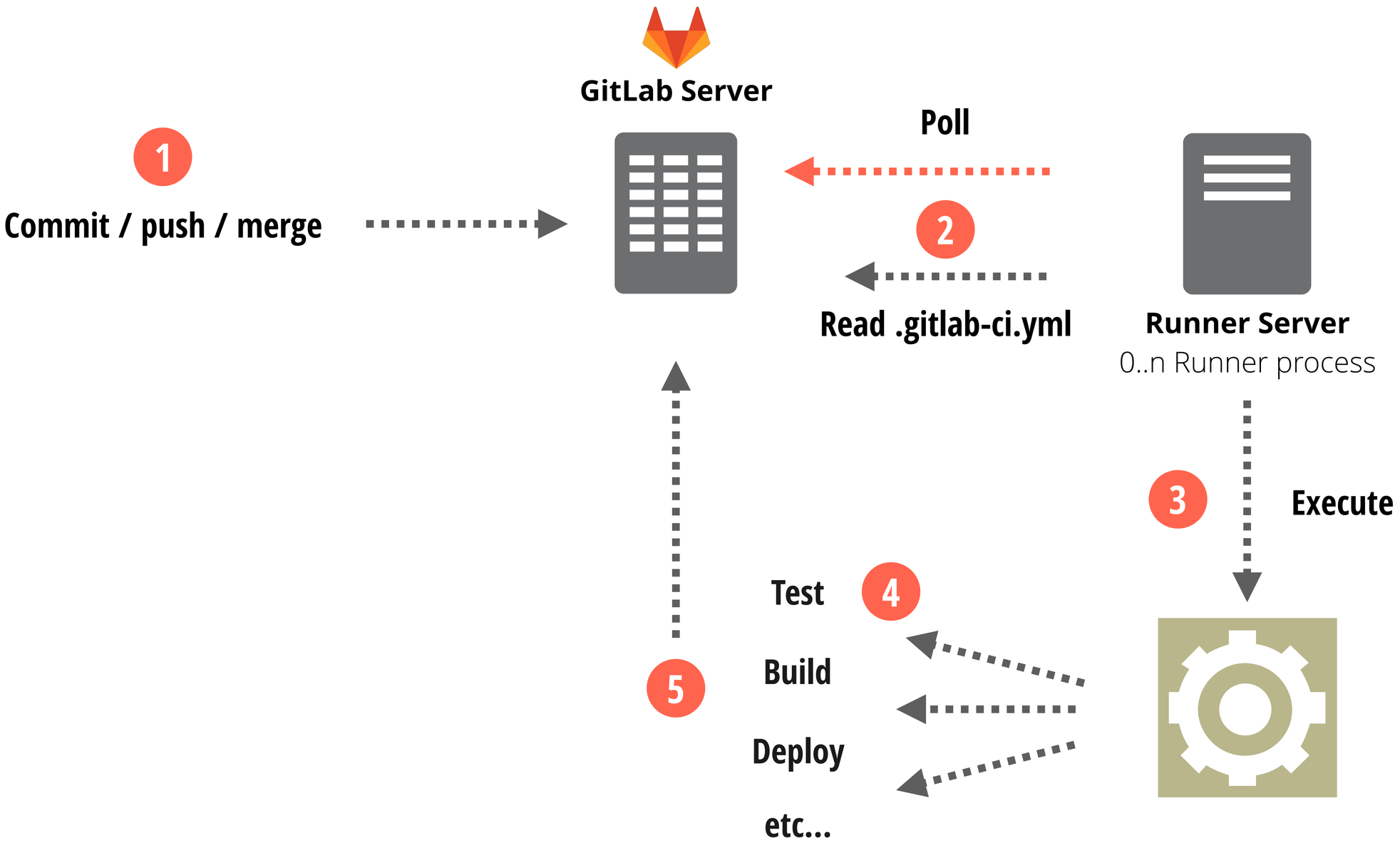
> Ajouter un fichier `.gitlab-ci.yml` à la racine du projet

> `.gitlab-ci.yml` définit un ensemble de **jobs** (commandes) et de contraintes par **stages** ({} == **pipeline**)

> Le **pipeline** est déclenché à chaque commit ou push et s'exécute dans le runner

> Et `.gitlab-ci.yml` explique au(x) Runner(s) ce qu'il faut faire

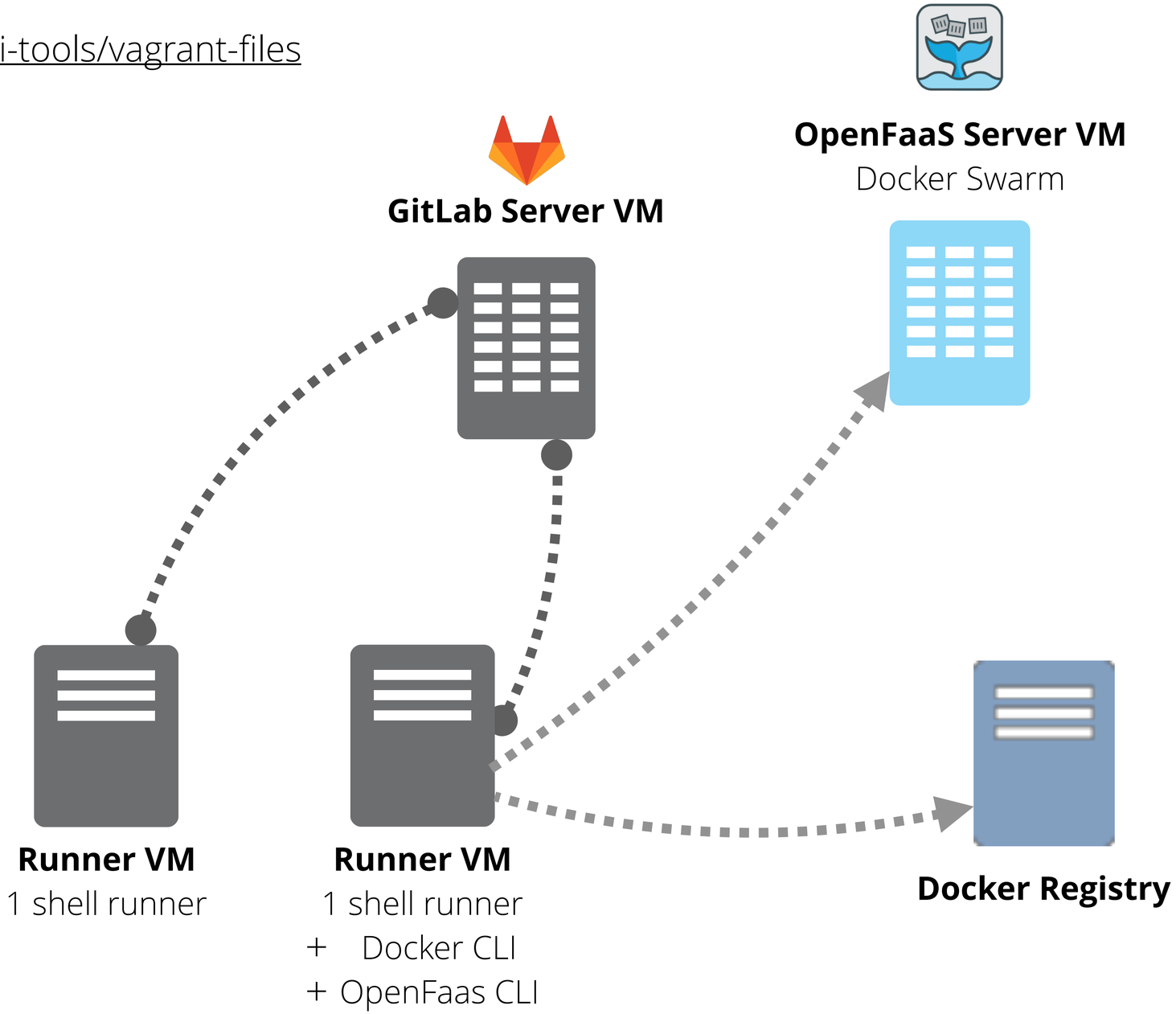
```yaml
stages:
  - ▶️play
  - 🦄tests
  - 📦generate

1️⃣addition:
  only:
    - master
  stage: ▶️play
  tags:
    - mini-runner
  script:
    - npm start add 40 2

✅🆘tests:
  stage: 🦄tests
  tags:
    - mini-runner
  script:
    - npm test
```


GitLab Server
1
Commit / push / merge
Poll
2
Read .gitlab-ci.yml
Runner Server
0..n Runner process
3
Execute
Test
4
Build
Deploy
etc...
5

Installation d'un Runner

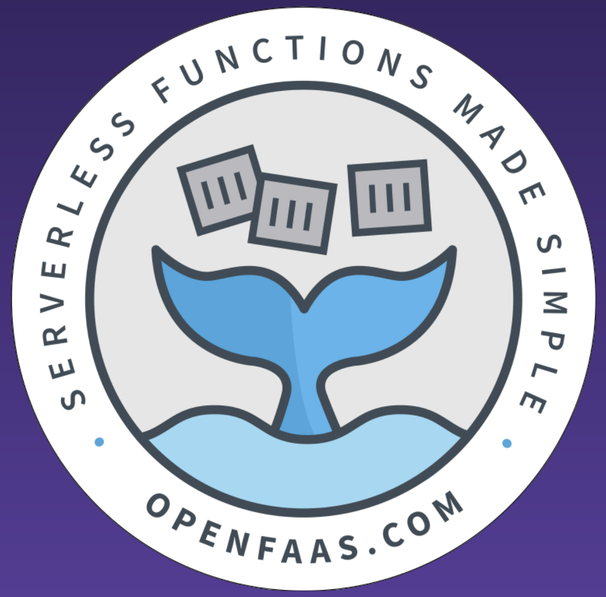
```
# Install GitLab Runner
curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh
| sudo bash
apt-get install -y gitlab-runner

# Add entries to hosts file:
echo "" >> /etc/hosts
echo '#{GITLAB_IP} #{GITLAB_DOMAIN}' >> /etc/hosts
echo "" >> /etc/hosts

# Registering GitLab Runner
gitlab-runner register --non-interactive \
  --url "http://#{GITLAB_DOMAIN}/" \
  --name "shell_runner_tests_nodejs" \
  --registration-token #{CI_REGISTRATION_TOKEN} \
  --tag-list "#{OPENFAAS_RUNNER_TAG}" \
  --executor shell
```

https://gitlab.com/tanuki-tools/vagrant-files



Demo(s) Time

CI-CD with OpenFaaS



https://www.openfaas.com

Merci 😍 & Questions