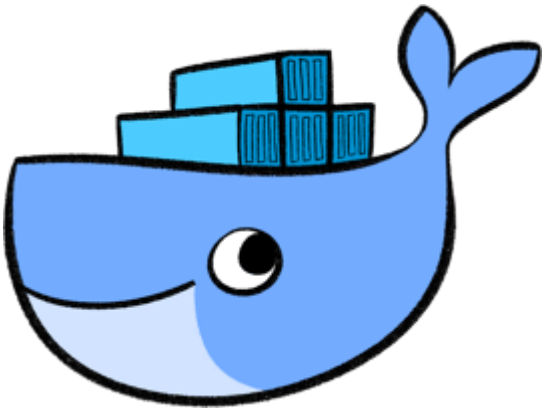
SIMPLON.CO

# ATELIER PRATIQUE

## DOCKER

## TRAVIS CI/CD

## HEROKU

Travis CI

# Compétences :

## Description de la compétence – processus de mise en œuvre

À partir de **l'architecture de l'application répartie** et éventuellement d'un **processus d'intégration continue** et des différents composants assemblés ou indépendants, élaborer **le diagramme de déploiement** correspondant.
Déployer l'exécutable obtenu par assemblage des différents composants ou les exécutables des composants indépendants, sur le ou les environnements de qualification, y compris dans le Cloud, afin d'obtenir une application logicielle opérationnelle et signée selon les exigences de sécurité.

## Contexte(s) professionnel(s) de mise en œuvre

La préparation et le déploiement d'une application logicielle répartie se réalise en fin de processus de développement ou tout au long du processus avec un outil d'intégration continue.
Dans le cas d'architecture en microservice, les composants sont déployés indépendamment.
Le déploiement de l'application s'effectue sur un ou des environnements de qualification interne ou externe en mode Web ou en mode Cloud.
Selon le contexte, la taille de la Direction des Systèmes d'information, le concepteur développeur est amené à exécuter ou non le déploiement.

## Critères de performance

- Le déploiement est formalisé à partir d'un diagramme
- Les composants assemblés ou indépendants sont déployés sur les environnements de qualification
- L'application déployée fournit les services demandés
- Le code de l'application est signé, tout ou partie, selon les exigences de sécurité

## Savoir-faire techniques, savoir-faire organisationnels, savoir-faire relationnels, savoirs

Réaliser un diagramme de déploiement
Déployer l'application ou le microservice
Prendre en compte les dépendances vis-à-vis des composants externes du composant à déployer
Prendre en compte les évolutions de versions de l'ensemble des composants externes
Gérer la sécurité de l'application en termes de signature numérique des exécutables, selon les exigences
de sécurité identifiées
Planifier et suivre les tâches de déploiement
Connaissance des bases de la cryptographie
Connaissance des concepts liés aux architectures réparties
Connaissance des différents types de serveurs
Connaissance des diagrammes UML concernant les composants et le déploiement
Connaissance du processus d'intégration continue
Connaissance d'un outil de signature de code
Connaissance des règles de mise en production associées à l'application

SIMPLON.CO

# Learning/Training

https://www.youtube.com/watch?v=caXHwYC3tq8
https://guillaumebriday.fr/comprendre-et-mettre-en-place-docker
**https://www.wanadev.fr/23-tuto-docker-comprendre-docker-partie1/**
**https://devhints.io/docker-compose**

https://blog.octo.com/pourquoi-utiliser-docker-en-tant-que-dev/
https://guillaumebriday.fr/comprendre-et-mettre-en-place-docker

https://hn.werick.codes/how-to-host-your-node-app-in-a-docker-container-on-heroku-cjwzf5w6q000r3ws1p647ks7m

https://www.valentinog.com/blog/redux/
https://www.rockyourcode.com/docker-compose-postgres-and-travis-ci/

https://alligator.io/react/testing-react-redux-with-jest-enzyme/

https://auth0.com/blog/use-docker-to-create-a-node-development-environment/
**https://buddy.works/guides/how-dockerize-node-application**

Avantages de Docker

- Vous pouvez lancer un environnement de développement complet sur n'importe quel ordinateur prenant en charge Docker
- L'environnement de travail de l'application reste cohérent sur l'ensemble du flux de travail. Cela signifie que l'application fonctionne *exactement de la même manière* pour le développeur, le testeur et le client, que ce soit sur un serveur de développement, de transfert ou de production.

# Atelier 1

**Veille et initiation à Docker**

1. **Veille :**

- **Philosophie de docker**
- **containerisation vs virtualisation**
- **containers**
- **volumes**
- **images et Dockerfile**
- **docker-compose**
- **intégration continue**
- **travisCI**

2. **Katacoda :**

https://www.katacoda.com/courses/docker/deploying-first-container

# Atelier 2

1. **Dockeriser une application NodeJS / Express**

https://nodejs.org/de/docs/guides/nodejs-docker-webapp/

2. **Dockeriser une application ReactJs**

. Create Dockerfile for React app
. Build docker image

```
docker build -t myimage .
```

. Run image and add ENV variables to container

```
docker run --name myapp -p 8080:5000 \
-e REACT_APP_ENVIRONMENT=docker \
--rm myimage
```

. List des containers :

```
docker ps
```

. Test that your app is running on port 8080 in local browser at http://localhost:8080

3. **Docker-compose**
   Orchestrer les deux services préalablement dockerisés avec docker-compose

**Démarrer : docker-compose up -d**
**Eteindre : docker-compose down**

Bonus **:**
**Imaginez-vous, sans installer la LAMP stack (Linux, Apache2, Mysql, PHP), il est possible de mettre en place une application wordpress avec une image Docker.**

SIMPLON.CO

# Atelier 3

**Déploiement automatisé, sur Heroku de Docker avec Travis CI/CD (intégration continue )**

Reprenons le repository de l'authentification avec Redux, ExpressJs et PostgreSQL, dockerisons l'application en automatisant son déploiement avec Travis et Heroku

https://openclassrooms.com/fr/courses/4087056-testez-et-suivez-letat-de-votre-application-php/4419481-tp-mettre-en-place-un-outil-dintegration-continue-travis
https://reactic.io/blog/technologie-de-virtualisation/docker/avantages-docker.php

**Bonus :**
Implémenter une nouvelle feature en TDD et déployer l'application :

- Sur la page private afficher la liste des utilisateurs inscrits.