3IN017 – TECHNOLOGIES DU WEB

Mise en production

8 avril 2025

Gilles Chagnon

Plan

Risques de la mise en production

Pour le moment...

SCIENCES SORBONNE UNIVERSITÉ

Environnement de développement

- http ://localhost :3000
- Accessible uniquement en local
- Sur le port 3000
- Pas de chiffrement
- Sur votre machine
- Recharge automatiquement les sources

Objectifs

- Écoute sur http(s) ://monapp.example.com
- Ports normaux pour le Web : 80 et 443
- Sur un serveur dans un datacenter
- Déployer le code
- Chiffrement https : certificats

Single point of failure (SPOF)

SCIENCES SORBONNE UNIVERSITÉ

Ce qui peut casser

- Votre code NodeJS
- Base de données
- Matériel : serveur, alimentation, câbles
- Lien avec internet : FAI, backbone
- DNS
- Routage IP

Risques

- Erreurs humaines : programmation, configuration
- Panne accidentelle
- Piratage

Haute disponibilité

SCIENCES SORBONNE UNIVERSITÉ

Éviter les SPOF

- Pas seulement dimensionner les systèmes
- Accepter que tout va finir par tomber en panne

Redondances

- Multiplier les systèmes
- Matériel : alimentation**S**, câble**S**
- Serveur**S** web, bases de données
- Lien**S** Internet, DNS, *etc*

Gérer le passage à l'échelle

Tests de montée en charge avant : simuler des interactions utilisateur

Les solutions ?

- de plus gros serveurs, disques, connexions ?
- plus de machines ? bases de données distribuées (NoSQL, plusieurs serveurs Web. . .)

Même si vous avez de la chance, cela ne suffira pas (course entre le canon et la cuirasse. . .). Et ne permet pas de gérer efficacement les baisses de charge : il faut une infrastructure dynamique.

Cloud computing

SCIENCES SORBONNE UNIVERSITÉ

Serveurs dédiés : location classique, une machine, au mois ou à l'année

Cloud : (al)location dynamique de ressources

- À la demande ou instantanée
- Avec une API
- Paiement en fonction du temps et de la consommation

Fournisseurs

- International : Google, Amazon (AWS), Microsoft (Azure)...
- France : Gandi, OVHCloud...

(attention aux contraintes du RGPD pour les données personnelles concernant les citoyens européens)

Hébergement

Serveurs physiques / serveurs virtuels

SCIENCES SORBONNE UNIVERSITÉ

Serveurs physiques (hébergement dédié)

- 1 machine physique, 1 serveur
- Plusieurs services, pas d'isolation
- Un seul OS

Serveurs virtuels (hébergements mutualisés ou non)

- 1 machine physique, plusieurs serveurs
- Plusieurs services, plusieurs clients
- Tous isolés
- Un OS par serveur virtuel

Containers

Pour simplifier l'administration (passage à l'échelle, mises à jour)

Outils de virtualisation

SCIENCES SORBONNE UNIVERSITÉ

Virtualisation complète

- Un OS complet
- N'importe quel OS avec n'importe quel OS
- Vmware, Xen, Virtualbox, KVM...

Outils cloud

- Gestion des virtualiseurs
- Création, lancement, arrêt des machines virtuelles
- Migrations d'un serveur physique à un autre

Virtualisation légère avec les *containers*

- Pas un OS complet
- Linux hôte : que des Linux virtuels, avec le même noyau

Containers

SCIENCES SORBONNE UNIVERSITÉ

Virtualisation complète

- Pas de noyau
- Mais un OS complet qui vient en surcouche de l'OS de la machine physique
- Besoin d'administration et de mises à jour

Container

- Juste le nécessaire (pas d'OS)
- Des images jetables
- Image pré-construites
- Images autonomes
- Pas de mise à jour : on reconstruit l'image

Docker

Système de containers

- Basé sur les *namespaces* et les *capabilities* de Linux en étendant LXC (*Linux Containers*)
- Isolation entre containers : système de fichiers, utilisateurs, réseau

Environnement

- Construction des images
- Publication et distribution des images

Outils et pratiques de développement et de mise en production

Intégration continue (*Continuous integration*)

SCIENCES SORBONNE UNIVERSITÉ

Définition

- Automatisation des tâches de *build*
- Automatisation des tests
- À chaque *commit*, tests lancés automatiquement

Objectif

Détecter au plus tôt les problèmes

En pratique

- Il faut lancer du code...
- Besoin d'isolation : utilisation intensive des containers

Outils

- Travis
  - Propriétaire, lié à Github (mais utilisable ailleurs)
  - `https://docs.travis-ci.com/`
- Gitlab
  - CI intégrée avec la gestion des dépôts git
  - À peu près libre
  - Déployé à Sorbonne Université : `https://gitlabsu.sorbonne-universite.fr/`
  - `https://docs.gitlab.com/ee/ci/quick_start/`
- Woodpecker
  - Libre
  - Grosse utilisation des containers
  - `https://woodpecker-ci.org/docs/usage/intro`
- Jenkins
  - ne se limite pas à l'intégration continue
  - basé sur java
  - `https://www.jenkins.io/doc/pipeline/tour/getting-started/`

Livraison continue (*(Continuous deployment*)

SCIENCES SORBONNE UNIVERSITÉ

Construction d'artefacts

- Binaires, installeurs, paquets
- Images docker (envoyées ensuite sur un registry)
- Sites web (pages github par exemple)
- Déploiement des sources (`git pull`)

Objectifs

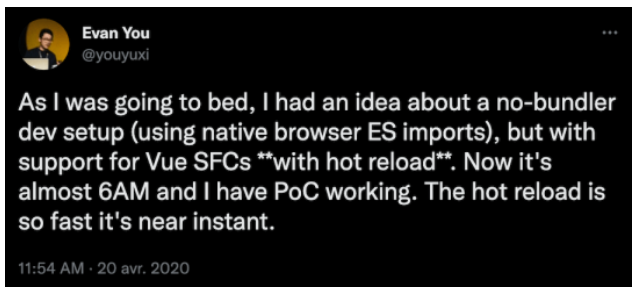
- Accélérer la mise en production
- Réduction des risques : mise à jour incrémentale
- Plus (+) d'interactions entre client et développeur (méthodes agiles)

Pour le développement et la mise en production

Exemples pour le Web : Webpack et Vite, des *bundlers*

- gestion des dépendances
- automatisation...

Webpack + ancien, Vite plus récent (créé en 2020 par le créateur de VueJS) et plus rapide pour l'intégration de nombreux fichiers JS volumineux.



Nous avons déjà utilisé Vite, mais on aurait pu utiliser Webpack (avec `npx create-react-app`).

Conclusion

Difficultés

- Passage à l'échelle
- Tolérances aux pannes
- Robustesse

Outils

- Cloud
- Isolation et containerisation : Docker
- *Continuous integration/delivery*