

François-Emmanuel GOFFINET

# Publication en CI/CD sur AWS

Une étude de cas  
DevOps  
Infrastructure as Code

---

## Table des matières

Projet de publication en CI/CD sur AWS	1.1
Présentation AWS	1.2
Ops vers Dev en laC	1.3
Hébergement AWS S3	1.4
Exécution sur AWS EC2	1.5
Pipeline : Gitlab-ci	1.6
Instances Gitlab Runner	1.7
Exercices	1.8

---

# Projet de publication en CI/CD sur AWS

François-Emmanuel GOFFINET

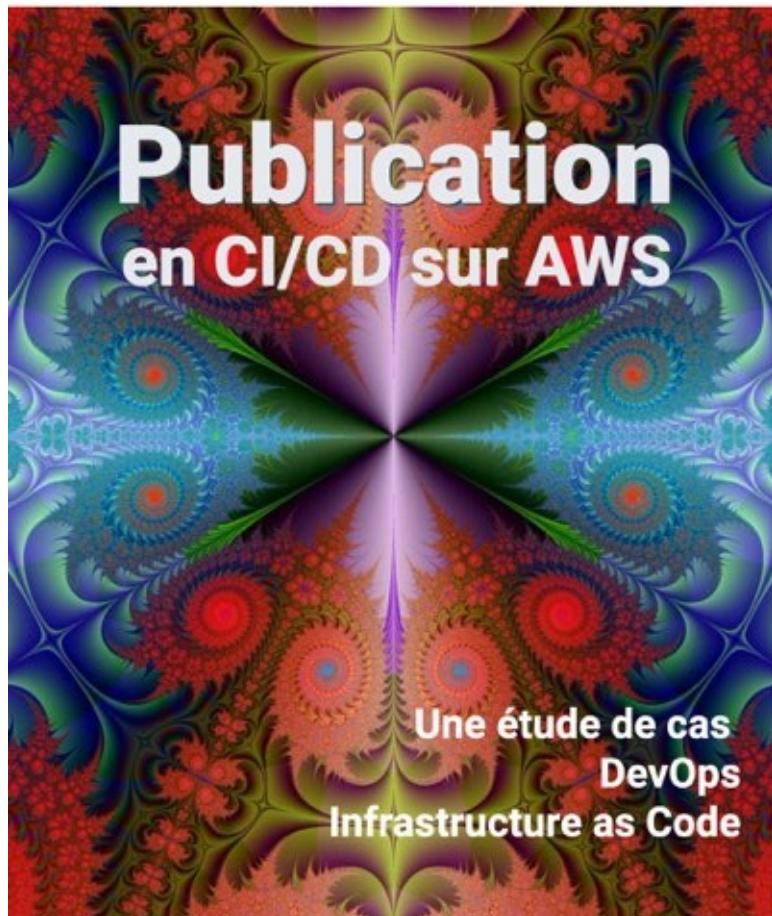


Figure 1 : Projet de publication en CI/CD sur AWS

Auteur : Francois-Emmanuel Goffinet

Date de fabrication : Wed Jan 16 2019 20:20:01 GMT+0000 (Coordinated Universal Time)

- [PDF](#)
- [EPUB](#)
- [MOBI](#)

Ce document est publié avec la méthode qu'il décrit. Il poursuit uniquement un but éducatif.

Sources du projet : <https://gitlab.com/goffinet/gitbook-gitlab/>

- 
- 1. Objectifs du document et résultat attendu
  - 2. Scénario et énoncé
  - 3. Modèles et références
    - 3.1. Autres références dans ce cadre
    - 3.2. Autres Frameworks de publication
  - 4. Principes DevOps
    - 4.1. La Première Voie
    - 4.2. La Seconde Voie
    - 4.3. La Troisième Voie

- 4.4. Culture DevOps
- 5. Dev vers Ops en pipeline CI/CD
- 8. Ops vers Dev en IaC sur AWS
- 6. Méthodes de configuration de l'infrastructure

## 1. Objectifs du document et résultat attendu

Ce document a pour objectif premier de fournir un énoncé dans le cadre d'une étude de cas DevOps qui met en oeuvre des outils d'intégration continue à partir d'une infrastructure Amazon Web Services AWS.

Les illustrations proposées sont à :

- évaluer
- adapter
- améliorer

Le résultat attendu est le suivant : fournir une solution sous forme de code informatique (repo git) avec une documentation la plus complète.

La méthode de travail est celle qui devrait le mieux correspondre aux principes DevOps.

Ce document d'énoncé n'a pas pour objectif de fournir une solution aboutie. Il pourrait toutefois évoluer dans cette direction.

## 2. Scénario et énoncé

Scénario : CI /CD avec Git, Gitlab, sur AWS EC2, Hébergement d'un site Web statique sur AWS S3 en CDN et en HTTPS.

Générer un livre numérique (ebook) en différents formats (HTML, PDF, MOBI, EPUB) en *intégration continue* et en *livraison continue* à partir d'un contenu écrit en Markdown sur une infrastructure dans le nuage. Le résultat attendu est la livraison continue du document sur un site Web aux normes actuelles (IPv6, HTTPS et CDN) offrant les différents formats de lecture. La solution est fournie sous forme de code. Le résultat est évalué sur base d'une vision défendue de la culture DevOps.

Les tâches "Dev" participent à la fabrication du produit. Les tâches "Ops" visent à délivrer le produit fini et de maintenir l'infrastructure nécessaire à la chaîne de production.

Il est demandé de développer la proposition de base en la testant, en la corrigeant, en la discutant, en ajoutant des étapes de retour dans les processus.

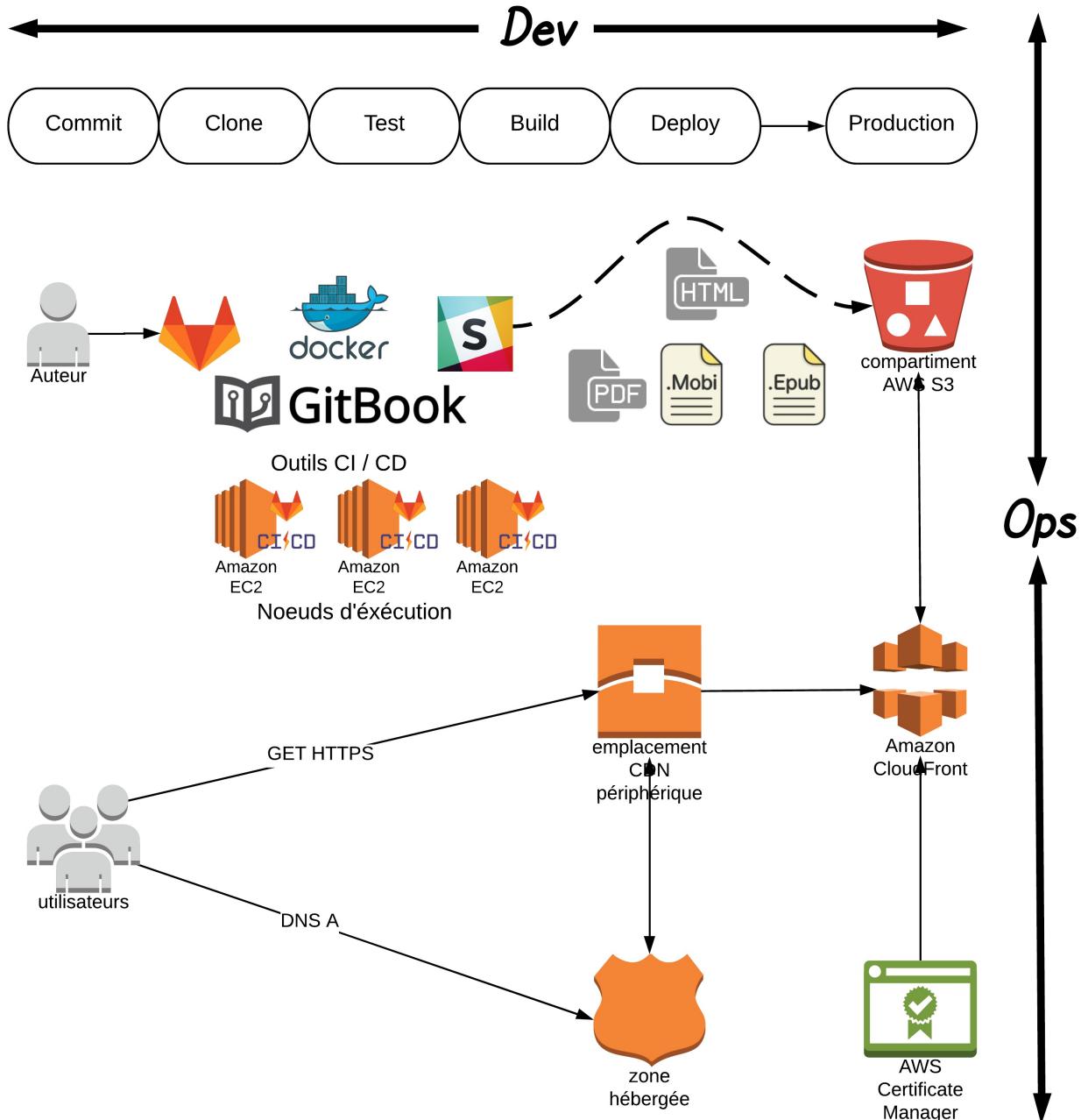


Figure 2 : Projet CI/CD de publication

On trouvera dans ce document des points de départs et des outils à évaluer, à adapter et à améliorer.

- Hébergement du code source : git
- Application : Gitbook-cli Toolchain
- Pipeline : Gitlab-ci
  - avec des noeuds d'exécution gitlab-runner (Docker)
- Hébergement Web :
  - AWS IAM
  - AWS S3,
  - AWS Route53,
  - AWS Cloudfront,
  - AWS Certificate Manager (ACM)
- Infrastructure as Code
  - Bash
  - Python

- Ansible
- Terraform
- Cloudformation

### 3. Modèles et références

- [Gitbook.com Legacy](#) et [Calibre](#)
- [Softcover](#)
- [Netlify](#) : *Build, deploy, and manage modern web projects. An all-in-one workflow that combines global deployment, continuous integration, and automatic HTTPS. And that's just the beginning.*

#### 3.1. Autres références dans ce cadre

- [Pandoc](#), un convertisseur open-source
- [Book publishing toolchain based on AsciiDoc](#)
- [Latex](#)
- [Jekyll](#)
- [MkDocs-Material](#)
- [Hugo](#)

#### 3.2. Autres Frameworks de publication

- [Wordpress](#)
- [Drupal](#)
- [Plone](#)
- [Ghost](#)

### 4. Principes DevOps

Les trois voies (ou manières) décrivent les valeurs et les philosophies qui encadrent les processus, les procédures, les pratiques de DevOps, ainsi que les étapes normatives.

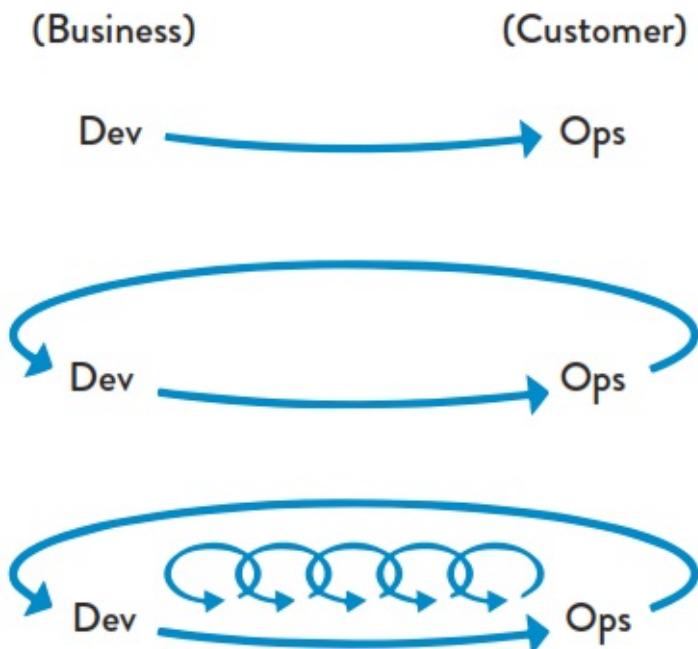


Figure 3 : Les trois manières: les principes sur lesquels se base DevOps

1. Le premier concerne le flux de travail de gauche à droite, du développement aux opérations informatiques.

2. Le second concerne le flux constant de retours rapides de droite à gauche à toutes les étapes du flux de valeur.
3. La troisième méthode consiste à créer une culture qui favorise deux choses : l'expérimentation continue et la compréhension du fait que la répétition et la pratique sont les conditions préalables à la maîtrise.

## 4.1. La Première Voie

La première méthode met l'accent sur le rendement de l'ensemble du système, par opposition au rendement d'un silo de travail ou d'un service particulier - qui peut être aussi important qu'une division (p. ex., Développement ou Opérations de TI) ou aussi petit qu'un contributeur individuel (p. ex., un développeur, un administrateur système).

L'accent est mis sur tous les flux de valeur de l'entreprise qui sont rendus possibles par les IT. En d'autres termes, cela commence lorsque les besoins sont identifiés (par l'entreprise ou le service informatique, par exemple), sont intégrés dans le développement, puis transférés dans les opérations informatiques, où la valeur est ensuite fournie au client sous la forme d'un service.

Les résultats de la mise en pratique de la Première Voie comprennent le fait de ne jamais transmettre un défaut connu aux centres de travail en aval, de ne jamais permettre à l'optimisation locale de créer une dégradation globale, de toujours chercher à augmenter le débit et de toujours chercher à obtenir une compréhension profonde du système (selon Deming).

## 4.2. La Seconde Voie

La deuxième méthode consiste à créer des boucles de rétroaction de droite à gauche. L'objectif de presque toutes les initiatives d'amélioration des processus est de raccourcir et d'amplifier les boucles de rétroaction afin que les corrections nécessaires puissent être apportées continuellement.

Les résultats de la deuxième voie comprennent la compréhension et la réponse à tous les clients, internes et externes, la réduction et l'amplification de toutes les boucles de rétroaction et l'intégration des connaissances là où on en a besoin.

## 4.3. La Troisième Voie

La troisième voie consiste à créer une culture qui favorise deux choses : l'expérimentation continue, la prise de risques et l'apprentissage de l'échec d'une part, et la compréhension que la répétition et la pratique sont les conditions préalables à la maîtrise d'autre part.

Nous avons besoin de ces deux éléments de la même façon. L'expérimentation et la prise de risques sont les garants de la volonté d'amélioration continue, même si cela signifie aller plus loin dans la zone dangereuse que on ne l'a jamais fait. Et on doit maîtriser les compétences qui peuvent aider à sortir de la zone dangereuse lorsque on est allé trop loin.

Les résultats de la troisième voie comprennent le temps alloué pour l'amélioration du travail quotidien, la création de rituels qui récompensent l'équipe pour avoir pris des risques et l'introduction de failles dans le système pour accroître la résilience.

Source : [The Three Ways: The Principles Underpinning DevOps](#)

## 4.4. Culture DevOps

- [Elements Of The First Way: And The DevOps Implications...](#)
- [DevOps Culture \(Part 1\)](#)
- [DevOps Culture \(Part 2\)](#)

## 5. Dev vers Ops en pipeline CI/CD

Dev = Application, frameworks, stack, toolchain

Outil	Fournisseur	Alternatives
Application Toolchain	<a href="#">Gitbook-cli</a> (basé NPM), calibre	...
Source Control Management	<a href="#">Git</a> , <a href="#">Gitlab</a>	Github, Gitlab Hosted (AWS), AWS CodeCommit, ...
CI / CD Tool	<a href="#">Gitlab-ci</a> , <a href="#">Gitlab-runner</a>	Jenkins, AWS CodePipeline, AWS CodeDeploy, Bitbucket
Container Repository	<a href="#">Gitlab Hub</a>	Docker Hub,
	versions, <a href="#">markdown-lint</a> , <a href="#">npm</a> (gitbook), <a href="#">pip</a>	

	(aws cli)	
Deploy	aws-cli s3	Python API, aws cli, openstack-cli, Ansible, ...

## 8. Ops vers Dev en IaC sur AWS

Ops = Infrastructure

- Hébergement
- Noeuds d'exécution

Service	Fournisseur	Alternatives
Name Registry	AWS Registrar	OVH, Gandi
DNS	AWS Route53	Cloudflare, OVH
TLS / HTTPS	AWS Certificate Manager	Let's Encrypt
CDN	Cloudfront	Cloudflare, OVH
Hosting / Storage	AWS S3	...
Build Node	AWS EC2 Gitlab-runner hosted	OVH, Scaleway, Azure, GCP, ...

## 6. Méthodes de configuration de l'infrastructure

- Manuelle dans la console
- Manuelle ou scriptée avec aws cli
- En python avec le module Boto
- Ansible
- Cloudformation
- Terraform

# Présentation de Amazon Web services

- [1. AWS : Présentation des services](#)
- [2. Magic Quadrant for Cloud Infrastructure as a Service](#)
- [3. AWS /cli](#)
  - Installation d'AWS Command Line Interface
  - Configuration de l'AWS CLI
  - Boto 3
- [4. Régions et AZ \(zone de disponibilité\)](#)
  - Europe/Moyen-Orient/Afrique

## 1. AWS : Présentation des services

- [Amazon EC2](#) : Serveurs virtuels dans le cloud
- [Amazon Simple Storage Service \(S3\)](#) : Stockage adaptatif dans le cloud
- [Amazon DynamoDB](#) : Base de données NoSQL gérée
- [Amazon RDS](#) : Service de base de données relationnelle géré pour MySQL, PostgreSQL, Oracle, SQL Server et MariaDB
- [AWS Lambda](#) : Exécution de code à la demande sans avoir à se soucier des serveurs
- [Amazon VPC](#) : Contrôle, accès, services réseau
- [Amazon Lightsail](#) : Lancement et gestion de serveurs virtuels privés
- [Amazon Simple Notification Service \(SNS\)](#) : Pub/Sub, notifications Push sur mobile et SMS
- [AWS IAM](#) : service d'accès
- [AWS ROUTE53](#) : Service DNS
- [AWS Certificate Manager ACM](#) : Autorité de certification TLS, gestionnaire de certificats TLS
- [AWS Cloudfront](#) : CDN mondial
- [AWS Elastic Beanstalk](#) : Exécuter et gérer des applications web
- [Amazon EC2 Auto Scaling](#) : Mettre à l'échelle la capacité de calcul pour répondre à la demande
- [Elastic Load Balancing \(ELB\)](#) : Distribution de trafic entrant vers plusieurs cibles
- [AWS CloudFormation](#) : Infrastructure en tant que code
- [AWS OpsWorks](#) : Automatisations des opérations grâce à Chef et Puppet
- [AWS CodeDeploy](#) : Automatisation des déploiements de code
- [Amazon Elastic Container Registry \(ECR\)](#) : Registre de conteneurs
- [Amazon Elastic Container Service \(ECS\)](#) : Service de conteneurs

Illustration de la fonction Lambda sur Netlify :

Create your own URL shortener with Netlify's Forms and Functions : <https://linkylinky.netlify.com/> dont l'application : <https://shortener.eu/>.

## 2. Magic Quadrant for Cloud Infrastructure as a Service

AWS Named as a Leader in Gartner's Infrastructure as a Service (IaaS) Magic Quadrant for 7th Consecutive Year

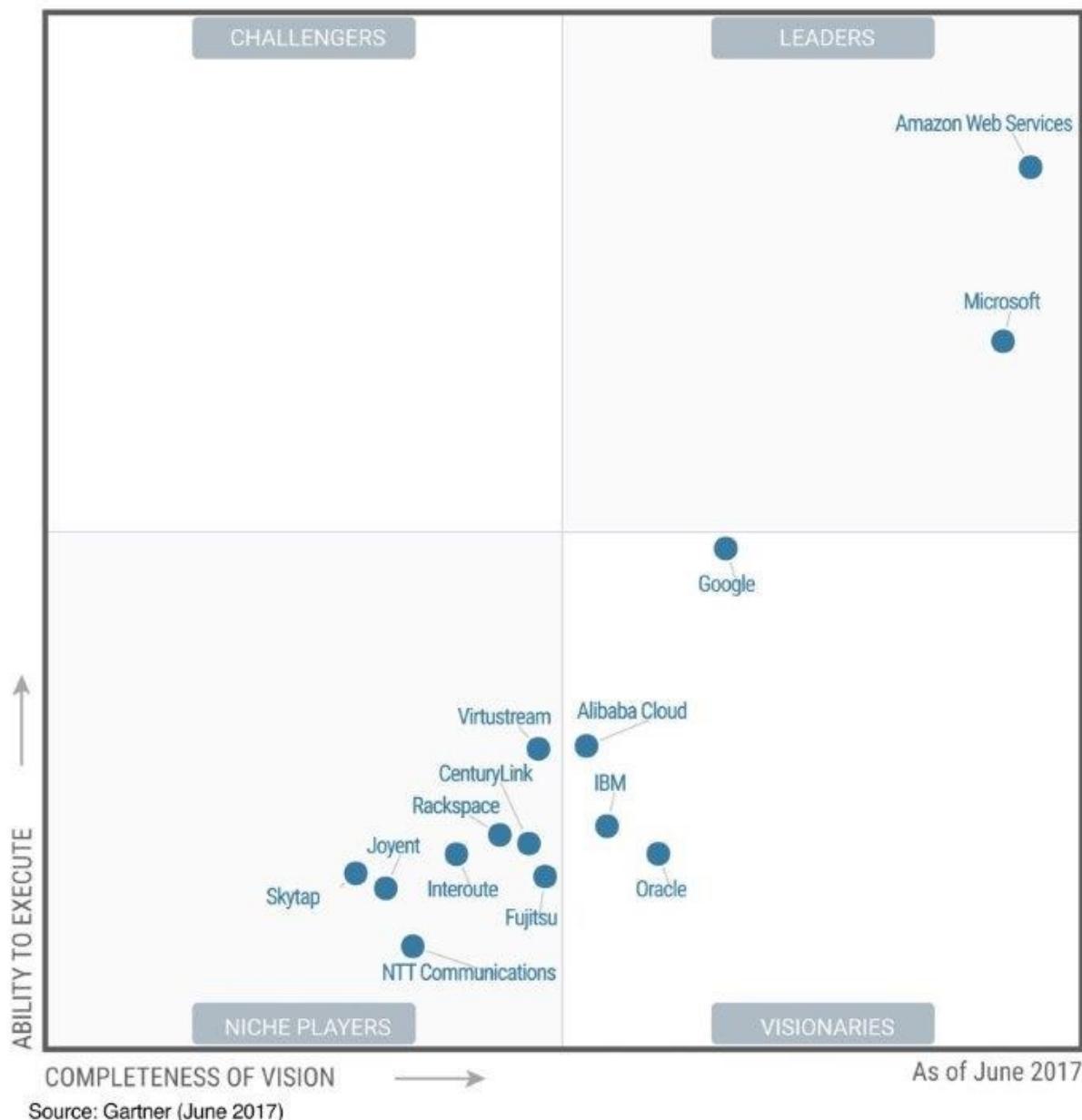
**Figure 1. Magic Quadrant for Cloud Infrastructure as a Service, Worldwide**

Figure 4 : AWS Named as a Leader in Gartner's Infrastructure as a Service (IaaS) Magic Quadrant for 7th Consecutive Year, 2017

Source : Magic Quadrant for Cloud Infrastructure as a Service, Worldwide

### 3. AWS /cli

[https://docs.aws.amazon.com/fr\\_fr/cli/latest/userguide/aws-cli.pdf](https://docs.aws.amazon.com/fr_fr/cli/latest/userguide/aws-cli.pdf)

#### Installation d'AWS Command Line Interface

##### Installation d'AWS Command Line Interface

L'AWS CLI est un outil à code source libre qui vous permet d'interagir avec les services AWS à l'aide des commandes du shell de ligne de commande. Avec une configuration minimale, vous pouvez commencer à utiliser toutes les fonctionnalités fournies par la console AWS Management Console depuis l'invite de commande de votre programme terminal préféré.

## Configuration de l'AWS CLI

Configuration de l'AWS CLI

### Boto 3

```
pip install boto3
```

## 4. Régions et AZ (zone de disponibilité)

Le cloud AWS gère 60 zones de disponibilité dans 20 régions géographiques autour du monde.

- <https://aws.amazon.com/fr/about-aws/global-infrastructure/>
- [https://docs.aws.amazon.com/fr\\_fr/AWSEC2/latest/UserGuide/using-regions-availability-zones.html](https://docs.aws.amazon.com/fr_fr/AWSEC2/latest/UserGuide/using-regions-availability-zones.html)

Les régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à faible latence, à débit élevé et à forte redondance.

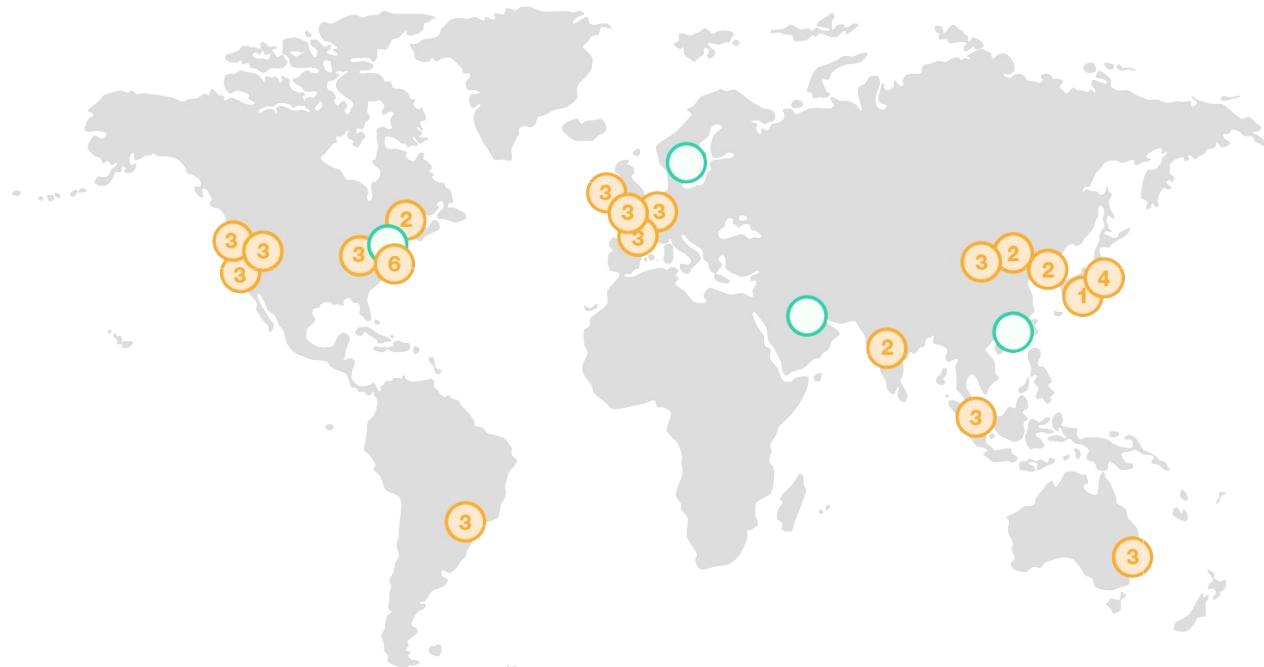


Figure 5 : Infrastructure mondiale

Amazon Infrastructure sur [Wiki Leaks](#).

### Europe/Moyen-Orient/Afrique

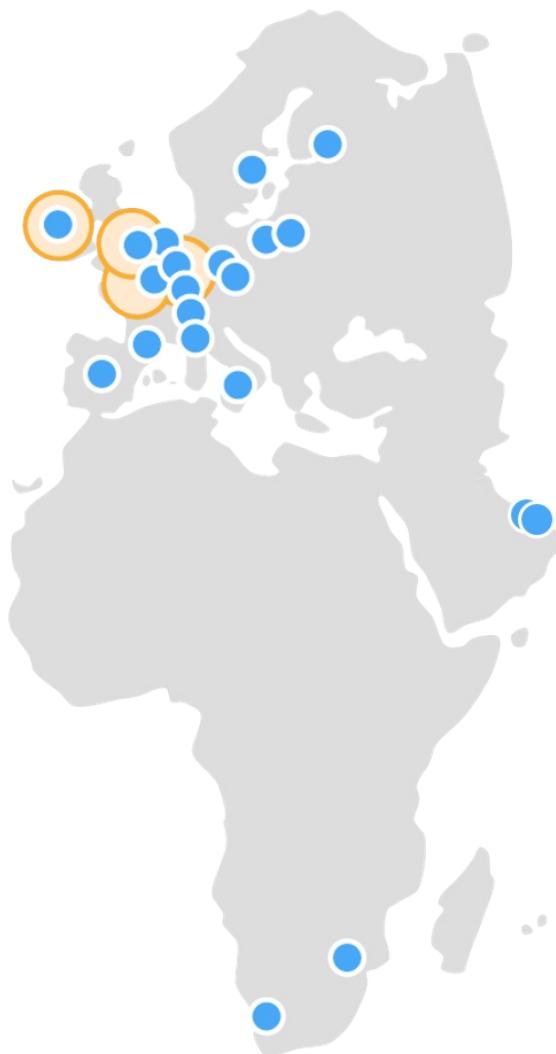


Figure 6 : Europe/Moyen-Orient/Afrique

#### Région Europe (Irlande)

- Zones de disponibilité EC2 : 3
- Lancée en 2007

#### Région Europe (Francfort)

- Zones de disponibilité EC2 : 3
- Lancée en 2014

#### Région Europe (Londres)

- Zones de disponibilité EC2 : 3
- Lancée en 2016

#### Région Europe (Paris)

- Zones de disponibilité EC2 : 3
- Lancée en 2017

### Emplacements de réseaux périphériques AWS

Emplacements périphériques : Amsterdam, Pays-Bas (2) ; Berlin, Allemagne ; Le Cap, Afrique du Sud ; Copenhague, Danemark ; Dubai, Émirats arabes unis ; Dublin, Irlande Francfort, Allemagne (8) ; Fujairah, Émirats arabes unis, Helsinki, Finlande ; Johannesburg, Afrique du Sud ; Londres, Angleterre (7) ; Madrid, Espagne (2) ; Manchester, Angleterre ; Marseille, France ; Milan, Italie ; Munich, Allemagne ; Oslo, Norvège ; Palerme, Italie ; Paris, France (4) ; Prague, République tchèque ; Stockholm, Suède (3) ; Vienne, Autriche ; Varsovie, Pologne ; Zurich, Suisse

Caches périphériques régionaux : Francfort, Allemagne, Londres, Angleterre

```
aws ec2 describe-regions --output table
-----+-----+-----+
|       DescribeRegions | |
+-----+-----+
| | Regions           | |
| +-----+-----+ |
| | Endpoint          | RegionName | |
| +-----+-----+ |
| | ec2.ap-south-1.amazonaws.com | ap-south-1 | |
| | ec2.eu-west-3.amazonaws.com | eu-west-3 | |
| | ec2.eu-west-2.amazonaws.com | eu-west-2 | |
| | ec2.eu-west-1.amazonaws.com | eu-west-1 | |
| | ec2.ap-northeast-2.amazonaws.com | ap-northeast-2 | |
| | ec2.ap-northeast-1.amazonaws.com | ap-northeast-1 | |
| | ec2.sa-east-1.amazonaws.com | sa-east-1 | |
| | ec2.ca-central-1.amazonaws.com | ca-central-1 | |
| | ec2.ap-southeast-1.amazonaws.com | ap-southeast-1 | |
| | ec2.ap-southeast-2.amazonaws.com | ap-southeast-2 | |
| | ec2.eu-central-1.amazonaws.com | eu-central-1 | |
| | ec2.us-east-1.amazonaws.com | us-east-1 | |
| | ec2.us-east-2.amazonaws.com | us-east-2 | |
| | ec2.us-west-1.amazonaws.com | us-west-1 | |
| | ec2.us-west-2.amazonaws.com | us-west-2 | |
| +-----+-----+ |
```

# Ops vers Dev en IaC

- 1. Ops vers Dev
- 2. AWS
  - 2.1. Hébergement S3
  - 2.2. Instances d'exécution EC2
- 3. Infrastructure as Code
  - 3.1. Ansible
  - 3.2. Cloudformation
- 4. Alternatives
  - 4.1. Google Cloud Platform (GCP)
  - 4.2. Microsoft Azure
  - 4.3. API OpenStack
  - 4.4. API Tierce
- 5. Introduction à AWS DevOps
  - 5.1. Fondamentaux
  - 5.2. Technologie Cloud
  - 5.3. Philosophie et outils DevOps
  - 5.4. La philosophie culturelle de DevOps
  - 5.5. Explications à propos des bonnes pratiques concernant DevOps
  - 5.6. Explications à propos de l'intégration continue
    - Pourquoi l'intégration continue est-elle nécessaire
  - 5.7. Comment fonctionne l'intégration continue
  - 5.8. Avantages de l'intégration continue
    - Améliorer la productivité des développeurs
    - Trouver et corriger plus rapidement les bogues
    - Livrer plus rapidement des mises à jour
  - 5.9. Explications à propos de la livraison continue
    - Livraison continue vs. déploiement continu
    - Avantages de la livraison continue
- 6. Produits AWS CI/CD
  - 6.1. Microservices
    - Produits AWS microservices
  - 6.2. Infrastructure en tant que code
    - Gestion de configuration
  - 6.3. Consignation et supervision
  - 6.4. Communication et collaboration
  - 6.5. Exemple de configuration d'un pipeline de CI/CD sur AWS

## 1. Ops vers Dev

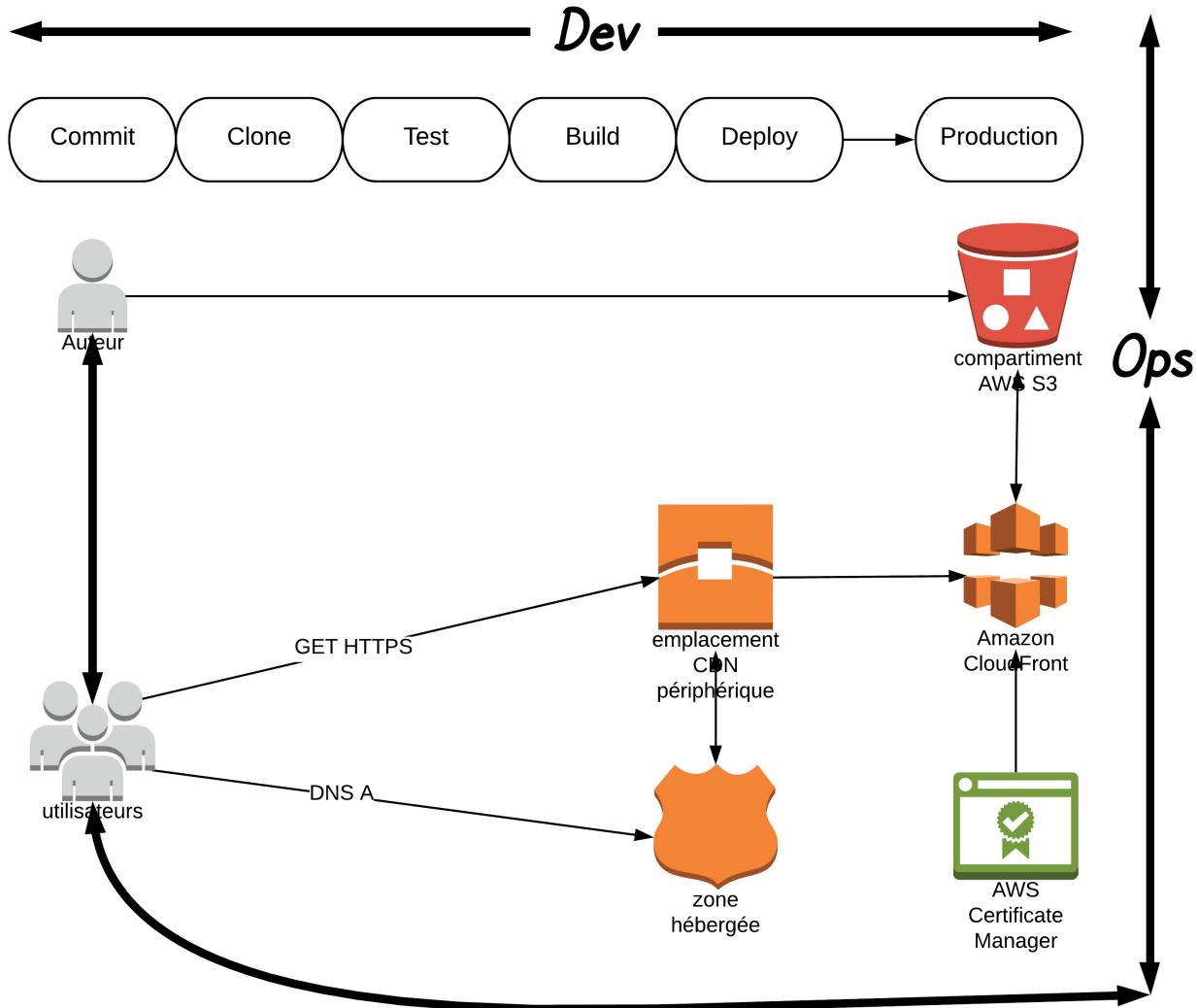


Figure 7 : Ops vers Dev

## 2. AWS

AWS Amazon Web Services est le leader mondial des technologies en nuage (voir plus loin).

### 2.1. Hébergement S3

- Découverte du service AWS S3
- Hébergement d'un site Web
- Domaine, HTTPS, distribution Cloudfront

### 2.2. Instances d'exécution EC2

- Automatisation d'instances EC2.
- Instances de conteneur [Amazon Elastic Container Service \(ECS\)](#) / [Amazon Elastic Container Registry \(ECR\)](#)

## 3. Infrastructure as Code

Comparaison des outils IaC : Cloudformation | Ansible | Terraform

- [Comparing CloudFormation, Terraform and Ansible - Simple example](#)
- [Comparing CloudFormation, Terraform and Ansible - Part #2](#)

- Pourquoi ansible n'est pas un bon choix pour créer son infra AWS ?

### 3.1. Ansible

[Amazon Web Services Guide](#)

### 3.2. Cloudformation

...

## 4. Alternatives

### 4.1. Google Cloud Platform (GCP)

...

### 4.2. Microsoft Azure

...

### 4.3. API OpenStack

OVH, par exemple.

### 4.4. API Tiers

Scaleway pour ses offres "Compute" mais utilise l'API de S3 pour son offre "Bucket".

## 5. Introduction à AWS DevOps

### 5.1. Fondamentaux

- <https://aws.amazon.com/fr/training/course-descriptions/cloud-practitioner-essentials/>
- <https://aws.amazon.com/fr/devops/>

### 5.2. Technologie Cloud

- [The NIST Definition of Cloud Computing](#)
- [NIST Cloud Computing Standards Roadmap](#)

### 5.3. Philosophie et outils DevOps

- <https://aws.amazon.com/fr/devops/what-is-devops/>

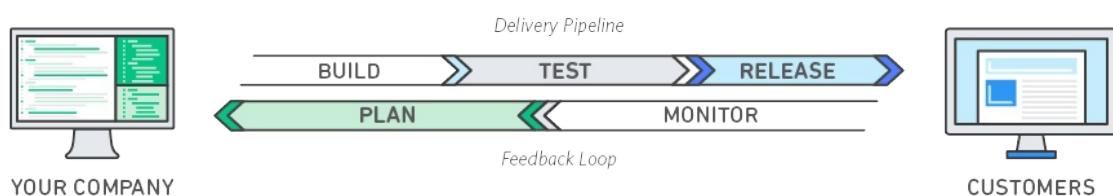


Figure 8 : DevOps Feedback

Illustration :

- [Netlify](#)

Formation : [Apprendre le développement continu avec des générateurs de site statique](#)

## 5.4. La philosophie culturelle de DevOps

La transition vers le DevOps implique un changement de culture et d'état d'esprit.

Pour simplifier, le DevOps consiste à éliminer les obstacles entre deux équipes traditionnellement isolées l'une de l'autre : l'équipe de développement et l'équipe d'exploitation. Certaines entreprises vont même jusqu'à ne pas avoir d'équipes de développement et d'exploitation séparées, mais des ingénieurs assurant les deux rôles à la fois.

- Avec le DevOps, les deux équipes travaillent en collaboration pour optimiser la productivité des développeurs et la fiabilité des opérations.
- Elles cherchent à communiquer fréquemment, à gagner en efficacité et à améliorer la qualité des services offerts aux clients.
- Elles assument l'entièr responsabilité de leurs services, et vont généralement au-delà des rôles ou postes traditionnellement définis en pensant aux besoins de l'utilisateur final et à comment les satisfaire.
- Les équipes d'assurance qualité et de sécurité peuvent également s'intégrer davantage à ces équipes.

Les organisations adoptant un modèle axé sur le DevOps, quelle que soit leur structure organisationnelle, rassemblent des équipes qui considèrent tout le cycle de développement et d'infrastructure comme faisant partie de leurs responsabilités.

## 5.5. Explications à propos des bonnes pratiques concernant DevOps

Il existe quelques pratiques clés pouvant aider les organisations à innover plus rapidement, par le biais de l'automatisation et de la simplification des processus de développement de logiciel et de gestion d'infrastructure. La plupart de ces pratiques sont rendues possibles par l'utilisation des outils appropriés.

Une pratique fondamentale consiste à réaliser des **mises à jour très fréquentes, mais de petite taille**. Ainsi, les entreprises innovent plus rapidement pour leurs clients. Ces mises à jour sont généralement de nature plus incrémentielle que les mises à jour occasionnelles associées aux pratiques de publication traditionnelles. Le recours à des mises à jour petites, mais fréquentes, limite les risques associés à chaque déploiement. Les équipes ont plus de facilité à détecter les bogues, car il est possible d'identifier le dernier déploiement ayant provoqué l'erreur. Bien que la cadence et la taille des mises à jour soient variables, les entreprises utilisant un modèle de DevOps déplacent des mises à jour beaucoup plus fréquemment que les entreprises utilisant des pratiques de développement de logiciel traditionnelles.

Les organisations peuvent également utiliser **une architecture de microservices** pour rendre leurs applications plus flexibles et favoriser des innovations plus rapides. L'architecture de microservices réduit de grands systèmes complexes pour obtenir des projets simples et indépendants. Les applications sont divisées en plusieurs composants (services) individuels, chaque service étant associé à une mission ou fonction spécifique et exploité indépendamment des autres services et de l'application dans son ensemble. Cette architecture réduit les coûts de coordination liés aux mises à jour d'applications, et quand chaque service est tenu par de petites équipes agiles, les entreprises avancent plus rapidement.

Cependant, l'alliance des microservices et d'une fréquence de publication plus élevée entraîne une augmentation considérable du nombre de déploiements, ce qui peut poser des problèmes opérationnels. Les pratiques de DevOps comme **l'intégration continue et la livraison continue** résolvent donc ces problèmes et permettent aux entreprises de livrer rapidement de manière fiable et sûre. Les pratiques d'automatisation de l'infrastructure, comme l'infrastructure en tant que code et la gestion de configuration, aident à préserver la souplesse et la réactivité des ressources informatiques face aux modifications fréquentes. En outre, le recours à la supervision et à la journalisation aide les ingénieurs à suivre les performances des applications et de l'infrastructure de manière à réagir rapidement en cas de problème.

Ensemble, ces pratiques aident les entreprises à livrer rapidement des mises à jour plus fiables à leurs clients. Voici une vue d'ensemble des pratiques de DevOps les plus importantes.

## 5.6. Explications à propos de l'intégration continue

L'intégration continue est une méthode de développement de logiciel DevOps avec laquelle les développeurs intègrent régulièrement leurs modifications de code à un référentiel centralisé, suite à quoi des opérations de création et de test sont automatiquement menées.

L'intégration continue désigne souvent l'étape de création ou d'intégration du processus de publication de logiciel, et implique un aspect automatisé (un service d'IC ou de création) et un aspect culturel (apprendre à intégrer fréquemment). Les principaux objectifs de l'intégration continue sont de trouver et de corriger plus rapidement les bogues, d'améliorer la qualité des logiciels et de réduire le temps nécessaire pour valider et publier de nouvelles mises à jour de logiciels.

### Pourquoi l'intégration continue est-elle nécessaire

Autrefois, les développeurs au sein d'une même équipe avaient tendance à travailler séparément pendant de longues périodes et à n'intégrer leurs modifications au référentiel centralisé qu'après avoir fini de travailler. Cela a rendu la fusion de changement de codes difficile et chronophage, et a également entraîné des bogues pendant une longue période, sans correction. La combinaison de ces différents facteurs empêchait de livrer rapidement des mises à jour aux clients.

## 5.7. Comment fonctionne l'intégration continue

Avec l'intégration continue, les développeurs appliquent régulièrement leurs modifications sur un référentiel partagé, avec un système de contrôle des versions comme Git. Avant d'envoyer leur code, les développeurs peuvent choisir d'exécuter des tests sur des unités locales pour le vérifier davantage avant son l'intégration. Un service d'intégration continue crée et exécute automatiquement des tests unitaires sur les nouveaux changements de codes pour détecter immédiatement n'importe quelle erreur.

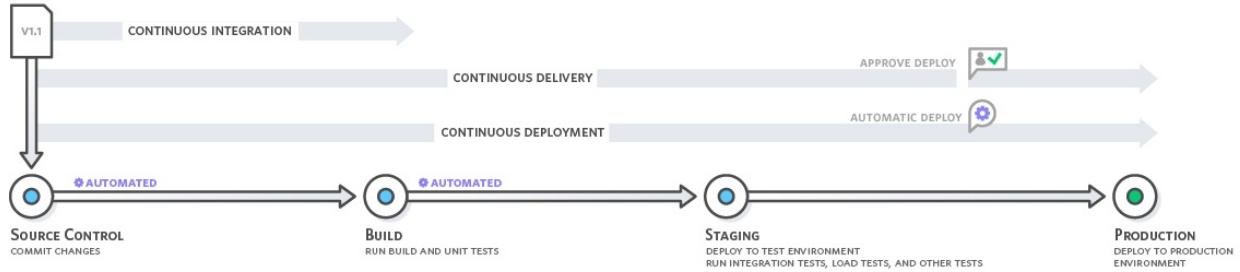


Figure 9 : CI

L'intégration continue désigne les étapes de création et de test d'unité du processus de publication de logiciel. Chaque révision appliquée déclenche un processus de création et de test automatisé.

Avec la livraison continue, les modifications de code sont automatiquement appliquées, testées et préparées à la production. La livraison continue étend le principe de l'intégration continue en déployant tous les changements de code dans un environnement de test et/ou un environnement de production après l'étape de création.

## 5.8. Avantages de l'intégration continue

### Améliorer la productivité des développeurs



Figure 10 : Améliorer la productivité des développeurs

L'intégration continue aide votre équipe à gagner en productivité, en limitant le nombre de tâches manuelles devant être accomplies par les développeurs et en encourageant les comportements qui contribuent à réduire le nombre d'erreurs et de bogues dans les versions publiées auprès des clients.

### Trouver et corriger plus rapidement les bogues



Figure 11 : Trouver et corriger plus rapidement les bogues

Avec des tests plus fréquents, votre équipe peut découvrir et corriger plus rapidement les bogues avant qu'ils ne prennent de l'ampleur.

## Livrer plus rapidement des mises à jour



Figure 12 : Livrer plus rapidement des mises à jour

L'intégration continue aide votre équipe à livrer plus rapidement et plus fréquemment des mises à jour après des clients.

### 5.9. Explications à propos de la livraison continue

La livraison continue est une méthode de développement de logiciels DevOps avec laquelle les changements de code sont automatiquement générés, testés et préparés pour une publication dans un environnement de production. Cette pratique étend le principe de l'intégration continue en déployant tous les changements de code dans un environnement de test et/ou un environnement de production après l'étape de création. Une bonne livraison continue permet aux développeurs de toujours disposer d'un artefact prêt au déploiement après avoir suivi un processus de test normalisé.

La livraison continue permet aux développeurs d'automatiser les tests au-delà des simples tests d'unité, afin de vérifier différents aspects d'une mise à jour d'application avant de la déployer auprès des clients. Il peut s'agir de tests d'interface, de charge, d'intégration, de fiabilité de l'API, etc. De cette manière, les développeurs peuvent vérifier de façon plus complète les mises à jour et détecter les éventuels problèmes à corriger avant le déploiement. Avec le cloud, l'automatisation de la création et de la réplication de plusieurs environnements de test est facile et économique, alors qu'une telle opération serait difficile à mettre en œuvre avec une infrastructure sur site.

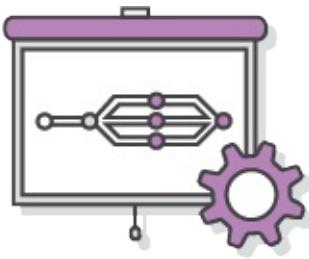
### Livraison continue vs. déploiement continu

Grâce à la livraison continue, chaque modification de code est appliquée, testée puis envoyée vers un environnement de test ou de préparation hors production. Plusieurs procédures de test peuvent avoir lieu en parallèle avant un déploiement de production. La différence entre la livraison continue et le déploiement continu réside dans la présence d'une approbation manuelle pour mettre à jour et produire. Avec le déploiement continu, la production se fait automatiquement, sans approbation explicite.

### Avantages de la livraison continue

En plus des avantages l'intégration continue, la livraison continue a pour avantage :

- Automatiser le processus de publication de logiciel



*Figure 13 : Automatiser le processus de publication de logiciel*

La livraison continue permet à votre équipe de créer, tester et préparer automatiquement les modifications de code en vue d'une mise en production, afin d'améliorer la rapidité et l'efficacité de vos projets de livraison de logiciel.

## 6. Produits AWS CI/CD

Dans une solution Amazon AWS, on configure un flux de travail d'intégration continue avec [AWS CodePipeline](#), qui vous permet de créer un flux de travail qui produit du code dans [AWS CodeBuild](#) à chaque fois que vous validez un changement.

### 6.1. Microservices

L'architecture de microservices est une approche de conception qui consiste à diviser une application en un ensemble de petits services. Chaque service est exécuté par son propre processus et communique avec les autres services par le biais d'une interface bien définie et à l'aide d'un mécanisme léger, typiquement une interface de programmation d'application (API) HTTP. Les microservices sont conçus autour de capacités métier. Chaque service est dédié à une seule fonction. Vous pouvez utiliser différents frameworks ou langages de programmation pour écrire des microservices et les déployer indépendamment, en tant que service unique ou en tant que groupe de services.

#### Produits AWS microservices

- [Amazon Container Service \(ECS\)](#) : Service de conteneurs logiciels.
- [AWS Lambda](#) : exécution de code à la demande.

### 6.2. Infrastructure en tant que code

L'infrastructure en tant que code est une pratique qui implique la mise en service et la gestion de l'infrastructure à l'aide de code et de techniques de développement de logiciel, notamment le contrôle des versions et l'intégration continue. Le modèle axé sur les API du cloud permet aux développeurs et aux administrateurs système d'interagir avec l'infrastructure de manière programmatique et à n'importe quelle échelle, au lieu de devoir installer et configurer manuellement chaque ressource. Ainsi, les ingénieurs peuvent créer une interface avec l'infrastructure à l'aide d'outils de code et traiter l'infrastructure de la même manière qu'un code d'application. Puisqu'ils sont définis par du code, l'infrastructure et les serveurs peuvent être rapidement déployés à l'aide de modèles standardisés, mis à jour avec les derniers patchs et les dernières versions, ou dupliqués de manière répétable.

Le produit [AWS CloudFormation](#) est un outil qui permet de gérer une infrastructure en tant que code.

### Gestion de configuration

Les développeurs et les administrateurs système utilisent du code pour automatiser la configuration du système d'exploitation et de l'hôte, les tâches opérationnelles et bien plus encore. Le recours au code permet de rendre les changements de configuration répétables et standardisés. Les développeurs et les administrateurs système ne sont plus tenus de configurer manuellement les systèmes d'exploitation, les applications système ou les logiciels de serveur.

Le produit Amazon [EC2 Systems Manager](#) permet de configurer et de gérer des systèmes Amazon EC2.

Le produit [AWS OpsWorks](#) offre des fonctions de gestion des configuration.

### 6.3. Consignation et supervision

Les entreprises suivent les métriques et les journaux pour découvrir l'impact des performances de l'application et de l'infrastructure sur l'expérience de l'utilisateur final du produit.

En capturant, catégorisant et analysant les données et les journaux générés par les applications et l'infrastructure, les organisations comprennent l'effet des modifications ou des mises à jour sur les utilisateurs, afin d'identifier les véritables causes de problèmes ou de changements imprévus.

La supervision active est de plus en plus importante, car les services doivent aujourd'hui être disponibles 24 heures sur 24 et la fréquence des mises à jour d'infrastructure augmente sans cesse. La création d'alerte et l'analyse en temps réel de ces données aident également les entreprises à suivre leurs services de manière plus proactive.

Le produit [Amazon CloudWatch](#) permet de surveiller les métriques et journaux d'une infrastructure.

Le produit [AWS CloudTrail](#) permet d'enregistrer et de journaliser les appels d'API AWS.

## 6.4. Communication et collaboration

L'instauration d'une meilleure collaboration et communication au sein de l'entreprise est un des principaux aspects culturels du DevOps. Le recours aux outils de DevOps et l'automatisation du processus de livraison de logiciel établit la collaboration en rapprochant physiquement les workflows et les responsabilités des équipes de développement et d'exploitation.

Partant de cela, ces équipes instaurent des normes culturelles importantes autour du partager des informations et de la facilitation des communications par le biais d'applications de **messagerie**, de **systèmes de suivi des problèmes** ou des projets et de **wikis**. Cela permet d'accélérer les communications entre les équipes de développement et d'exploitation et même d'autres services comme le marketing et les ventes, afin d'aligner chaque composant de l'entreprise sur des objectifs et des projets communs.

## 6.5. Exemple de configuration d'un pipeline de CI/CD sur AWS

<https://aws.amazon.com/fr/getting-started/projects/set-up-ci-cd-pipeline/>

Dans ce projet, on découvre comment configurer un pipeline d'intégration et de livraison continues (CI/CD) sur AWS. Un pipeline aide à automatiser les étapes du processus de publication de logiciel, comme lancer les versions automatiques et les déployer ensuite sur les instances Amazon EC2.

On utilisera AWS CodePipeline pour créer, tester et déployer le code chaque fois que celui-ci est modifié, en fonction des modèles de processus de publication que l'on a définis.

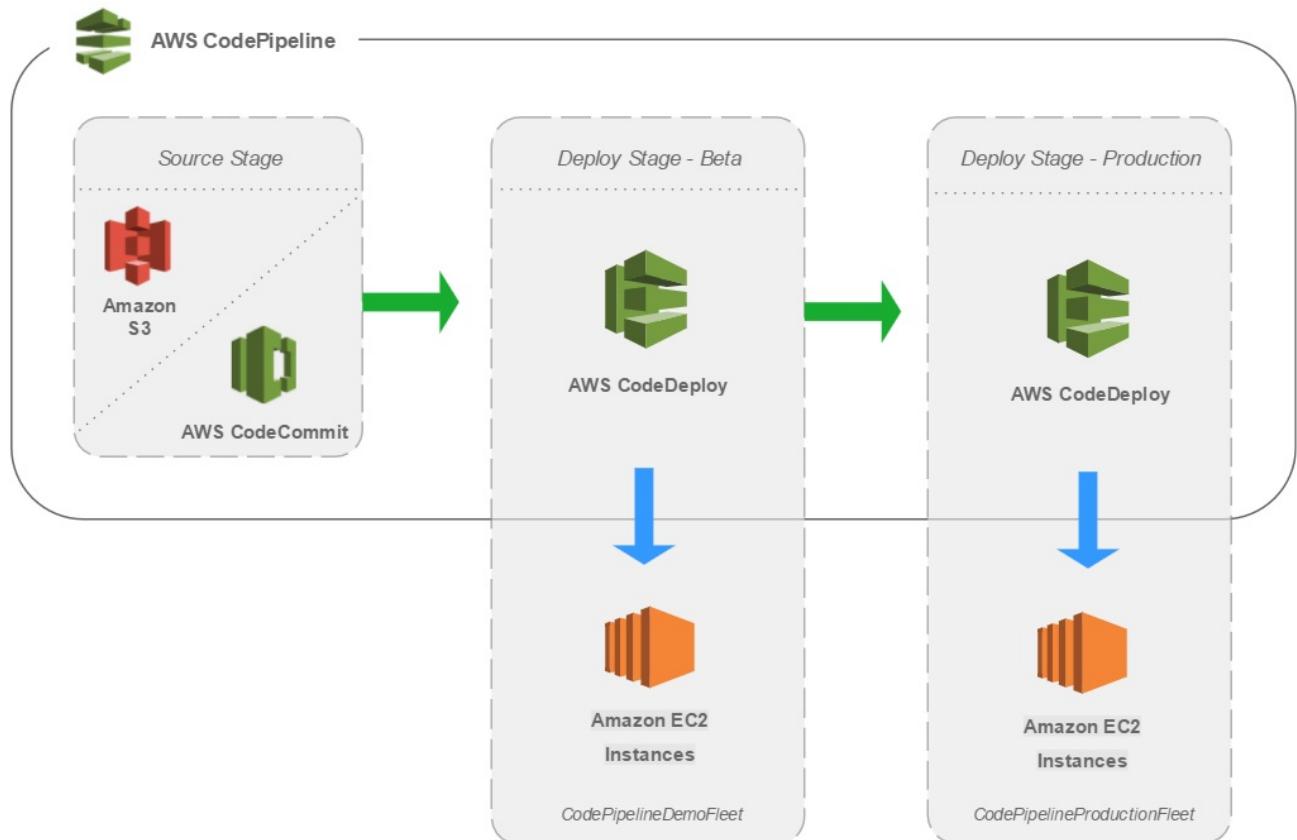


Figure 14 : Mise en place d'un pipeline CI/CD sur AWS

# Hébergement d'un site Web statique sur AWS S3

- [1. Idée de base](#)
  - Phases
  - Phase Infra
  - [1.1. Livrable Web aux normes actuelles](#)
  - [1.2. Avantages des sites Web statiques](#)
  - [1.3. Coûts estimés](#)
  - [1.4. Aspects dynamiques](#)
  - Droits nécessaires
- [2. Expérimentation dans la console AWS](#)
  - [2.1. Création d'un bucket avec un utilisateur autorisé](#)
  - [2.2. Configuration d'un site web statique](#)
  - [2.3. Configuration d'un site web statique grâce à un domaine personnalisé](#)
  - [2.4. accélérez votre site web avec Amazon CloudFront](#)
- [3. Expérimentation avec aws-cli](#)
  - Étape 1
  - Étape 2
  - Étape 3
    - Note sur le `hosted_zone_id`
  - Étape 4
- [4. Outils d'approvisionnement, de gestion de configuration, d'orchestration, IaC](#)
  - [4.1. Bash avec aws cli et jq](#)
  - [4.2. Ansible](#)
  - [4.3. Cloudformation](#)
  - [4.3. Terraform](#)
- [5. Expérimentation Ansible](#)
  - [5.1. Livre de jeu](#)
  - [2. Rôle d'hébergement](#)
  - [5.3. Optimisation](#)
- [6. En Python](#)
- [7. Template Cloudformation](#)

## 1. Idée de base

Basé sur le *White Paper "Hosting Static Websites on AWS"*

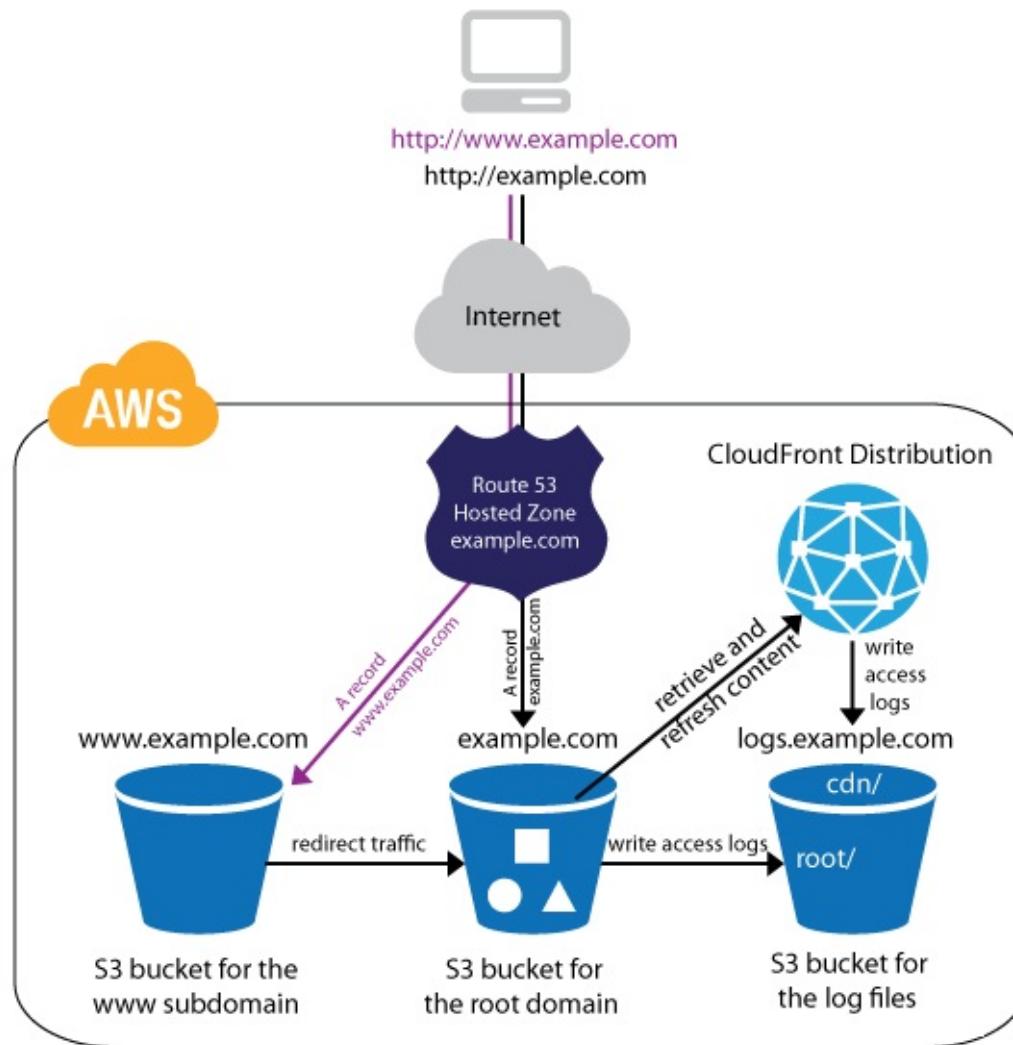


Figure 15 : Hébergement d'un site Web statique sur AWS

On trouvera ici une proposition à évaluer, à adapter et à améliorer sous forme IaC (Infrastructure as Code).

- Stockage / hébergement : AWS S3
- DNS : AWS Route 53
- CDN / Logging : AWS Cloudfront
- HTTPS : AWS Certificate Manager
- Credits Management : AWS IAM

## Phases

- Déploiement sur un Bucket S3 existant
- Infrastructure nécessaire : Nom, DNS, Distribution, certificat, utilisateur et policies.
- Construction

## Phase Infra

- Création du Bucket S3 avec une propriété 'w ebsite'. Existe-t-il ?
- Une distribution Cloudfront associée au Bucket S3 existe-t-elle ?
- Le nom de domaine est-il en gestion Route 53 ? Est-il associé ?
- Le certificat HTTPS existe-t-il pour le domaine ?

### 1.1. Livrable Web aux normes actuelles

- IPv6
- HTTP 2.0
- CDN
- HTTPS
- Robustesse du stockage

## 1.2. Avantages des sites Web statiques

### 1.3. Coûts estimés

Coûts estimés mensuels.

AWS Service Charges		\$15.15
▶ Budgets		\$0.00
▶ CloudFront		\$0.00
▶ Data Transfer		\$0.00
▶ Key Management Service		\$0.00
▼ Registrar		\$12.00
▼ Global		\$12.00
Amazon Registrar DomainRegistration		\$12.00
Registration of aws-fr.com	1 Quantity	\$12.00
▼ Route 53		\$0.50
▼ Global		\$0.50
Amazon Route 53 DNS-Queries		\$0.00
\$0.40 per 1,000,000 queries for the first 1 Billion queries	681 Queries	\$0.00
Amazon Route 53 HostedZone		\$0.50
\$0.50 per Hosted Zone for the first 25 Hosted Zones	1 HostedZone	\$0.50
Amazon Route 53 Intra-AWS-DNS-Queries		\$0.00
Queries to Alias records are free of charge	179 Queries	\$0.00
▶ Simple Storage Service		\$0.02
Taxes		
TVA à percevoir		\$2.63

Figure 16 : Coûts estimés mensuels

[AWS Tableau de bord Gestion de la facturation et des coûts](#)

## 1.4. Aspects dynamiques

Il y a lieu de réfléchir aux aspects dynamiques à donner au livrable.

- Interaction avec les utilisateurs : commentaires, support en ligne
- Vente en ligne des artefacts, vente en ligne sur Amazon (production de livres papier)
- Protection des pages, fidélisation
- URL signées
- ...

## Droits nécessaires

À définir précisément.

- S3
- Cloudfront
- Route53
- ACM

## 2. Expérimentation dans la console AWS

### Mise en route sur Amazon Simple Storage Service

Inspiration : un tuto parmi d'autres [Hosting a HTTPS static website on Amazon S3 w/ CloudFront and Route 53](#) avec des captures d'écran.

#### 2.1. Crédation d'un bucket avec un utilisateur autorisé

- IAM
- [Comment créer un compartiment S3 ?](#)
- [Utilisation des commandes s3 de haut niveau avec l'AWS Command Line Interface](#)
- [s3](#)

#### 2.2. Configuration d'un site web statique

##### Inspiration

- Crédation d'un compartiment et configuration de celui-ci comme site web
- Ajout d'une stratégie de compartiment permettant de rendre disponible publiquement le contenu de votre compartiment
- Chargement d'un document Web, Chargement d'un document d'index
- Test de votre site web

#### 2.3. Configuration d'un site web statique grâce à un domaine personnalisé

##### Inspiration

- enregistrez un domaine
- créez et configurez des compartiments et chargez des données
  - créez deux compartiments
  - configurez des compartiments pour l'hébergement de site Web
  - configurez la redirection de site web
  - configurez la "journalisation" pour le trafic du site web (facultatif)
    - testez le point de terminaison et la redirection
- ajoutez des enregistrements d'alias pour example.com et www.example.com
- test

#### 2.4. accélérez votre site web avec Amazon CloudFront

##### Inspiration

- Crédation d'une distribution CloudFront
- Mise à jour des jeux d'enregistrements pour votre domaine et votre sous-domaine
- (Facultatif) Vérification des fichiers journaux

### 3. Expérimentation avec aws-cli

On suppose qu'un certificat HTTPS est disponible et que l'on dispose de son ARN. On suppose que la zone Route 53 existe.

Se référer à [Hosting a Static Website with Hugo and AWS](#) pour la correction.

#### Étape 1

Créer un compartiment et activer l'hébergement d'un site web statique sur le compartiment.

Le point de terminaison de votre compartiment est `bucketname.s3-website-region.amazonaws.com`.

Ajoutez une stratégie de compartiment qui autorise un accès en lecture publique sur le compartiment que vous avez créé.

```
BUCKET_NAME="test1.aws-fr.com"
BUCKET_REGION="eu-west-3"
aws s3 mb s3://${BUCKET_NAME} --region ${BUCKET_REGION}
echo '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow Public Access to All Objects",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::${BUCKET_NAME}/*"
    }
  ]
}' > /tmp/policy.json

aws s3api put-bucket-policy --bucket ${BUCKET_NAME} --policy file:///tmp/policy.json
aws s3 website s3://${BUCKET_NAME} --index-document index.html --error-document error.html

mkdir website
cat < EOF >> website/index.html
<html xmlns="http://www.w3.org/1999/xhtml" >
<meta charset="UTF-8">
<head>
  <title>My Website Home Page</title>
</head>
<body>
  <h1>Welcome to my website</h1>
  <p>Now hosted on Amazon S3!</p>
</body>
</html>
EOF
aws s3 sync --acl public-read --sse --delete website/ s3://${BUCKET_NAME}
curl http://${BUCKET_NAME}.s3-website.eu-west-3.amazonaws.com/
```

```
aws s3 rm s3://${BUCKET_NAME} --recursive
aws s3 rb s3://${BUCKET_NAME}
```

#### Étape 2

Créez une distribution web CloudFront. Assurez-vous de configurer les paramètres suivants :

Pour Nom du domaine d'origine, indiquez le point de terminaison. Pour Méthodes HTTP autorisées, sélectionnez GET, HEAD, OPTIONS. Pour Autres noms de domaine (CNAME), entrez le CNAME que vous souhaitez utiliser pour votre site web.

Si vous souhaitez utiliser le protocole SSL pour votre site web, vous pouvez choisir l'option Demander ou importer un certificat avec ACM pour demander un certificat. Pour plus d'informations, consultez Utilisation de noms de domaines alternatifs et de HTTPS.

```
aws acm request-certificate --domain-name example.com --subject-alternative-names a.example.com b.example.com *.c.example.com
```

!!! à vérifier et corriger

```
aws configure set preview.cloudfront true

cat < EOF >> /tmp/distconfig.json
{
  "CallerReference": "'${BUCKET_NAME}'-'`date +%s`'",
  "Aliases": {
```

```

        "Quantity": 0
    },
    "DefaultRootObject": "",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "OriginPath": "/",
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                },
                "Id": "S3-${BUCKET_NAME}",
                "DomainName": "${BUCKET_NAME}.s3.amazonaws.com"
            }
        ]
    },
    "DefaultCacheBehavior": {
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "TargetOriginId": "S3-${BUCKET_NAME}",
        "ViewerProtocolPolicy": "allow-all",
        "ForwardedValues": {
            "Headers": {
                "Items": [
                    "Access-Control-Request-Headers",
                    "Origin"
                ],
                "Quantity": 2
            },
            "Cookies": {
                "Forward": "none"
            },
            "QueryString": true
        },
        "MaxTTL": 31536000,
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "AllowedMethods": {
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "Quantity": 2
            },
            "Quantity": 2
        },
        "MinTTL": 0
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Bucket": "",
        "Prefix": "",
        "Enabled": false,
        "IncludeCookies": false
    },
    "WebACLId": "",
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "SSLv3"
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    }
}
EOF

aws cloudfront create-distribution --distribution-config file:///tmp/distconfig.json > /tmp/distconfig_result.json

```

```
cat /tmp/distconfig_result.json | jq .Distribution.DomainName
```

Exemple réel :

ETag indispensable ? Mise en variable de `"ViewerCertificate.ACMCertificateArn"` et `"ViewerCertificate.Certificate"`.

```
{
    "ETag": "E2908GS64VRLST",
    "DistributionConfig": {
        "Comment": "Created by Ansible s3-website-hosting role",
        "CacheBehaviors": {
            "Quantity": 0
        },
        "IsIPV6Enabled": true,
        "Logging": {
            "Bucket": "",
            "Prefix": "",
            "Enabled": false,
            "IncludeCookies": false
        },
        "WebACLId": "",
        "Origins": {
            "Items": [
                {
                    "OriginPath": "",
                    "CustomOriginConfig": {
                        "OriginSslProtocols": {
                            "Items": [
                                "TLSv1",
                                "TLSv1.1",
                                "TLSv1.2"
                            ],
                            "Quantity": 3
                        },
                        "OriginProtocolPolicy": "http-only",
                        "OriginReadTimeout": 30,
                        "HTTPPort": 80,
                        "HTTPSPort": 443,
                        "OriginKeepaliveTimeout": 5
                    },
                    "CustomHeaders": {
                        "Quantity": 0
                    },
                    "Id": "S3-${BUCKET_NAME}",
                    "DomainName": "${BUCKET_NAME}.s3-website-us-east-1.amazonaws.com"
                }
            ],
            "Quantity": 1
        },
        "DefaultRootObject": "index.html",
        "PriceClass": "PriceClass_100",
        "Enabled": true,
        "DefaultCacheBehavior": {
            "FieldLevelEncryptionId": "",
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "TargetOriginId": "S3-${BUCKET_NAME}",
            "ViewerProtocolPolicy": "redirect-to-https",
            "ForwardedValues": {
                "Headers": {
                    "Quantity": 0
                },
                "Cookies": {
                    "Forward": "none"
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                },
                "QueryString": false
            },
            "MaxTTL": 2592000,
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "AllowedMethods": {
                "Items": [
                    "HEAD",
                    "GET",
                    "PUT",
                    "DELETE",
                    "OPTIONS",
                    "PATCH"
                ],
                "Quantity": 5
            }
        }
    }
}
```

```
        "GET"
    ],
    "CachedMethods": {
        "Items": [
            "HEAD",
            "GET"
        ],
        "Quantity": 2
    },
    "Quantity": 2
},
"MinTTL": 0,
"Compress": true
},
"CallerReference": "${BUCKET_NAME}`date +%"`",
"ViewerCertificate": {
    "SSLSupportMethod": "sni-only",
    "ACMCertificateArn": "arn:aws:acm:us-east-1:733718180495:certificate/e60e1dd7-6329-4598-bc85-6003b2237cf5",
    "MinimumProtocolVersion": "TLSv1.1_2016",
    "Certificate": "arn:aws:acm:us-east-1:733718180495:certificate/e60e1dd7-6329-4598-bc85-6003b2237cf5",
    "CertificateSource": "acm"
},
"CustomErrorResponses": {
    "Items": [
        {
            "ErrorCode": 403,
            "ResponsePagePath": "/index.html",
            "ResponseCode": "200",
            "ErrorCachingMinTTL": 300
        },
        {
            "ErrorCode": 404,
            "ResponsePagePath": "/index.html",
            "ResponseCode": "200",
            "ErrorCachingMinTTL": 300
        }
    ],
    "Quantity": 2
},
"OriginGroups": {
    "Items": [],
    "Quantity": 0
},
"HttpVersion": "http2",
"Restrictions": {
    "GeoRestriction": {
        "RestrictionType": "none",
        "Quantity": 0
    }
},
"Aliases": {
    "Items": [
        "${BUCKET_NAME}"
    ],
    "Quantity": 1
}
}
```

## Étape 3

Mettez à jour les enregistrements DNS pour que votre domaine pointe le CNAME de votre site web vers votre nom de domaine de distribution CloudFront. Le nom de domaine de votre distribution est disponible dans la console CloudFront dans un format similaire à celui-ci : d1234abcd.cloudfront.net.

Attendez que vos modifications de DNS soient propagées et que les entrées précédentes du DNS expirent.

[Comment créer des jeux d'enregistrements de ressources d'alias dans Route 53 à l'aide de l'interface de ligne de commande AWS CLI ?](#)

[Jeu d'enregistrement de ressources d'alias pour une distribution CloudFront](#)

```
"myDNS" : {
    "Type" : "AWS::Route53::RecordSetGroup",
    "Properties" : {
        "HostedZoneId" : { "Ref" : "myHostedZoneID" },
        "RecordSets" : [
            {
                "Name" : { "Ref" : "myRecordSetDomainName" },
                "Type" : "A",
                "AliasTarget" : {
                    "HostedZoneId" : "Z2FDTNDATAQYW2",
```

```

        "DNSName" : { "Ref" : "myCloudFrontDistributionDomainName" }
    }
}
}
}
```

## Note sur le hosted\_zone\_id

```

# Specify the region to create the AWS resources in
DEFAULT_REGION = "us-east-1"

# A mapping of hosted zone IDs to AWS regions.
# Apparently this data is not accessible via API
# http://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region
# https://forums.aws.amazon.com/thread.jspa?threadID=116724
S3_HOSTED_ZONE_IDS = {
    'us-east-1': 'Z3AQBSTGFYJSTF',
    'us-west-1': 'Z2F56UZL2M1ACD',
    'us-west-2': 'Z3BJ6K6RIION7M',
    'ap-south-1': 'Z11RGJOFQNVJUP',
    'ap-northeast-1': 'Z2M4EHUR26PTZW',
    'ap-northeast-2': 'Z3W0307B5YMIYP',
    'ap-southeast-1': 'Z300J2DXBE1FTB',
    'ap-southeast-2': 'Z1WCIGYICN2BYD',
    'eu-central-1': 'Z21DNDUVLTQW6Q',
    'eu-west-1': 'Z1BKCTXD74EZPE',
    'sa-east-1': 'Z7KQH4QJS5SS0',
    'us-gov-west-1': 'Z31GFT0UA1I2HV',
}
}
```

## Étape 4

Utilisateur S3 pour les mise à jour

Create the user

```
aws iam create-user --user-name S3-user
```

Create the policy

```
aws iam create-policy --policy-name S3-user-write --policy-document file://iam.json
```

Attach the iam policy to the user (policy-arn will be in output from previous command)

```
aws iam attach-user-policy --usr-name S3-user --policy-arn arn:aws:iam::938109129012:policy/S3-user-write
```

You probably want the access and secret key for your user to use somewhere:

```
aws iam create-access-key --user-name S3-user
```

## 4. Outils d'approvisionnement, de gestion de configuration, d'orchestration, IaC

### 4.1. Bash avec aws cli et jq

- A collection of bash shell scripts for automating various tasks with Amazon Web Services using the AWS CLI and jq.

### 4.2. Ansible

- Ansible route53\_zone - add or delete Route53 zones
- ACM ?
- Ansible Role - Sets up a website in a S3 bucket fronted by Cloudfront for HTTPs and custom domains

### 4.3. Cloudformation

- Cloudformation template for creating static website

### 4.3. Terraform

- Terraform scripts to setup an S3 based static website, with a CloudFront distribution and the required Route53 entries.
- Terraform module to easily provision CloudFront CDN backed by an S3 origin <https://cloudposse.com/>

## 5. Expérimentation Ansible

Voir le document local [ansible-aws/roles/s3-website-hosting](#)

### 5.1. Livre de jeu

On crée un livre de jeu adapté avec les bonnes variables.

```
#s3-website.yml
- hosts: localhost
  vars:
    website_root: 'website/'
    s3_website_bucket_name: 'test6.aws-fr.com'
    s3_website_alias_domain_names:
      - 'test6.aws-fr.com'
    s3_website_certificate_arn: 'arn:aws:acm:us-east-1:733718180495:certificate/e60e1dd7-6329-4598-bc85-6003b2237cf5'
    s3_website_create_dns_record: true
    s3_website_root_object: 'index.html'
  roles:
    - s3-website-hosting
```

## 2. Rôle d'hébergement

On a ajouté une tâche de synchronisation d'un site de test.

```
#roles/s3-website-hosting/tasks/main.yml
---
- import_tasks: s3-cloudfront-route53.yml
- import_tasks: sync-s3.yml
```

On a ajouté des entrées `ignore_errors` pour être en mesure de jouer le rôle.

```
#grep 'name:' roles/s3-website-hosting/tasks/s3-cloudfront-route53.yml
- name: Name of the website s3 bucket that will be used
- name: Create S3 bucket for website hosting
- name: S3 bucket details
- name: Configure S3 bucket for website hosting
- name: Output Website domains
- name: Search CloudFront distribution based on alias domain names given (task fails if cloudfront still needs creating)
- name: Assign first found element to variable
- name: Extract distribution config if Cloudfront distribution was found
- name: Output infos of existing CloudFront distribution (confirm if correct one was matched)
- name: Wait to give time to read above message
- name: Set caller reference to pre-existing one or generate new one for creating new Cloudfront distribution
- name: Output caller reference to be used to identify Cloudfront dist
- name: Create Cloudfront Website distribution
- name: Output result diff
- name: Save Cloudfront Domain and ID in variables
- name: Output Cloudfront domain
- name: Output Cloudfront ID
- name: Create DNS alias for CloudFront distribution on Route53
```

Cette liste de tâche a été ajoutée.

```
#roles/s3-website-hosting/tasks/sync-s3.yml
---
- name: synchronize files
  s3_sync:
    bucket: "{{ website_bucket.name }}"
    file_root: "{{ website_root }}"
    file_change_strategy: force
    permission: public-read
    delete: yes
```

```
- name: create a batch of invalidations using a distribution_id for a reference
  cloudfront_invalidation:
    distribution_id: "{{ cloudfront_website_distribution.id }}"
    target_paths:
      - /*
```

### 5.3. Optimisation

- Crédit du certificat
- Restriction des droits

## 6. En Python

- [S3 Website Using cf , Route 53 and Python Scripts](#)

## 7. Template Cloudformation

- [Cloudformation template for creating static website](#)

# AWS EC2

- [Introduction](#)
- [1. AMI](#)
  - [1.1. Trouver une AMI Linux](#)
- [2. Création d'une instance EC2 avec aws cli](#)
  - [2.1. VPC](#)
  - [2.2. Clé d'accès](#)
  - [2.3. Lancer une instance t2.micro](#)
  - [2.4. Instances EC2](#)
  - [2.5. Connexion à l'instance](#)
  - [2.6. Opérations sur l'instance \(Ansible\)](#)
  - [2.7. Une application simple](#)
  - [2.8. Terminer une instance](#)
- [3. Déploiement avec Cloud-init](#)
- [4. AWS EC2 avec Ansible](#)
- [5. AWS CloudFormation](#)
  - [5.1. Fonctionnement AWS CloudFormation](#)
  - [5.2. Notes](#)
- [6. LightSail](#)
- [7. Elastic Beanstack](#)

## Introduction

Pour lancer une instance EC2, on besoin :

1. AMI : une image de référence
2. VPC : un switch virtuel auquel va se connecter l'instance EC2 avec des règles d'accès entrant (au minimum TCP22)
3. Clé : une clé privée SSH pour se connecter à l'instance. Il est nécessaire de prendre connaissance du nom d'utilisateur
4. Script Cloud-init : un script à lancer au démarrage de l'instance pour un approvisionnement automatique

## 1. AMI

Une Amazon Machine Image (AMI) fournit les informations requises pour lancer une instance, qui est un serveur virtuel dans le cloud. Vous devez spécifier une AMI source lorsque vous lancez une instance. Lorsque vous avez besoin de plusieurs instances configurées de manière identique, il est possible de lancer plusieurs instances à partir d'une même AMI. Lorsque vous avez besoin d'instances configurées de manière différente, vous pouvez utiliser différentes AMI pour lancer ces instances.

Une AMI comprend les éléments suivants :

- Un modèle d'image pour le volume racine de l'instance (par exemple, un système d'exploitation, un serveur d'applications et des applications)
- Les autorisations de lancement qui contrôlent les comptes AWS qui peuvent utiliser l'AMI pour lancer les instances

Un "mappage" de périphérique de stockage en mode bloc qui spécifie les volumes à attacher à l'instance lorsqu'elle est lancée

[https://docs.aws.amazon.com/fr\\_fr/AWSEC2/latest/UserGuide/AMIs.html](https://docs.aws.amazon.com/fr_fr/AWSEC2/latest/UserGuide/AMIs.html)

Cette commande effectue une recherche d'une AMI Linux Amazon (Amazon Linux AMI) pour une instance "x86\_64 HVM GP2" :

```
aws ec2 describe-images --filters "Name=description,Values=Amazon Linux AMI * x86_64 HVM GP2" --query 'Images[*].[CreationDate, Description, ImageId]' --output text | sort -k 1 | tail
```

```
2018-01-03T19:01:53.000Z Amazon Linux AMI 2017.09.1.20180103 x86_64 HVM GP2 ami-8715a2fa
2018-01-08T18:42:47.000Z Amazon Linux AMI 2017.09.1.20180108 x86_64 HVM GP2 ami-fe03b483
2018-01-15T19:12:53.000Z Amazon Linux AMI 2017.09.1.20180115 x86_64 HVM GP2 ami-8ee056f3
2018-01-18T23:08:21.000Z Amazon Linux AMI 2017.09.1.20171120 x86_64 HVM GP2 ami-27e85e5a
2018-03-07T06:59:00.000Z Amazon Linux AMI 2017.09.1-testlongids.20180307 x86_64 HVM GP2 ami-08f0e11237ddeb2f0
2018-03-07T06:59:52.000Z Amazon Linux AMI 2017.09.1.20180307 x86_64 HVM GP2 ami-4f55e332
2018-04-13T00:25:52.000Z Amazon Linux AMI 2018.03.0.20180412 x86_64 HVM GP2 ami-cae150b7
```

```
2018-05-08T18:10:54.000Z Amazon Linux AMI 2018.03.0.20180508 x86_64 HVM GP2 ami-969c2deb
2018-06-22T22:24:50.000Z Amazon Linux AMI 2018.03.0.20180622 x86_64 HVM GP2 ami-d50bbaa8
2018-08-11T02:29:44.000Z Amazon Linux AMI 2018.03.0.20180811 x86_64 HVM GP2 ami-0ebc281c20e89ba4b
```

Retenons l'AMI `ami-0ebc281c20e89ba4b` comme la plus récente.

```
export AWS_IMAGE="ami-0ebc281c20e89ba4b"
```

## 1.1. Trouver une AMI Linux

Exemple : Rechercher l'IMA Amazon Linux 2 actuelle

```
aws ec2 describe-images --owners amazon \
--filters 'Name=name,Values=amzn2-ami-hvm-2.0.??????-x86_64-gp2' 'Name=state,Values=available' \
--output json | jq -r '.Images | sort_by(.CreationDate) | last().ImageId'
```

Exemple : Rechercher l'IMA Amazon Linux actuelle

```
aws ec2 describe-images --owners amazon \
--filters 'Name=name,Values=amzn-ami-hvm-????.?.?.?????-x86_64-gp2' 'Name=state,Values=available' \
--output json | jq -r '.Images | sort_by(.CreationDate) | last().ImageId'
```

Exemple : Rechercher l'IMA Ubuntu Server 16.04 LTS actuelle

```
aws ec2 describe-images --owners 099720199477 \
--filters 'Name=name,Values=ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-????????' 'Name=state,Values=available' \
--output json | jq -r '.Images | sort_by(.CreationDate) | last().ImageId'
```

Exemple : Rechercher l'IMA Red Hat Enterprise Linux 7.5 actuelle

```
aws ec2 describe-images --owners 309956199498 \
--filters 'Name=name,Values=RHEL-7.5_HVM_GA*' 'Name=state,Values=available' \
--output json | jq -r '.Images | sort_by(.CreationDate) | last().ImageId'
```

Exemple : Rechercher l'IMA SUSE Linux Enterprise Server 15 actuelle

```
aws ec2 describe-images --owners amazon \
--filters 'Name=name,Values=suse-sles-15-v?????-hvm-ssd-x86_64' 'Name=state,Values=available' \
--output json | jq -r '.Images | sort_by(.CreationDate) | last().ImageId'
```

## 2. Création d'une instance EC2 avec aws cli

### 2.1. VPC

Un Virtual Private Cloud (VPC) est un réseau virtuel dédié logiquement isolé des autres réseaux virtuels dans le cloud AWS. On peut lancer des ressources AWS, comme des instances Amazon EC2, dans un VPC, spécifier une plage d'adresses IP pour le VPC, ajouter des sous-réseaux, associer des groupes de sécurité et configurer des tables de routage.

Un sous-réseau est une plage d'adresses IP dans le VPC.

```
aws ec2 describe-vpcs --output table
-----
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
```

```
|||| CidrBlockState ||||  
||+-----+-----+-----+  
|| State | associated ||||  
||+-----+-----+-----+||
```

```
export AWS_VPC="vpc-476c332e"
```

```
export AWS_VPC=$(aws ec2 describe-vpcs --query 'Vpcs[*].VpcId' --output text)
```

```
aws ec2 create-security-group \  
--group-name demo-lab \  
--description "Demo Lab Security Group" \  
--vpc-id $AWS_VPC
```

```
aws ec2 authorize-security-group-ingress \  
--group-name demo-lab \  
--protocol tcp \  
--port 22 \  
--cidr 0.0.0.0/0
```

```
aws ec2 authorize-security-group-ingress \  
--group-name demo-lab \  
--protocol tcp \  
--port 80 \  
--cidr 0.0.0.0/0
```

```
aws ec2 authorize-security-group-ingress \  
--group-name demo-lab \  
--protocol tcp \  
--port 3000 \  
--cidr 0.0.0.0/0
```

```
aws ec2 describe-security-groups \  
--group-names demo-lab \  
--output table
```

```
-----  
| DescribeSecurityGroups |  
+-----+-----+-----+  
|| SecurityGroups |||  
||+-----+-----+-----+  
|| Description | GroupId | GroupName | OwnerId | VpcId |||  
||+-----+-----+-----+-----+  
|| Demo Lab Security Group | sg-04edf90e659364a62 | demo-lab | 733718180495 | vpc-476c332e |||  
||+-----+-----+-----+-----+  
|| IpPermissions |||  
||+-----+-----+-----+  
|| FromPort | IpProtocol | ToPort |||  
||+-----+-----+-----+  
|| 80 | tcp | 80 |||  
||+-----+-----+-----+  
|| IpRanges |||  
||+-----+-----+-----+  
|| CidrIp |||  
||+-----+-----+-----+  
|| 0.0.0.0/0 |||  
||+-----+-----+-----+  
|| IpPermissions |||  
||+-----+-----+-----+  
|| FromPort | IpProtocol | ToPort |||  
||+-----+-----+-----+  
|| 22 | tcp | 22 |||  
||+-----+-----+-----+  
|| IpRanges |||  
||+-----+-----+-----+  
|| CidrIp |||  
||+-----+-----+-----+  
|| 0.0.0.0/0 |||  
||+-----+-----+-----+  
|| IpPermissionsEgress |||  
||+-----+-----+-----+  
||
```

```
||| IpProtocol ||| | | |
||+-----+||| +|||
||| -1 ||| |||
||+-----+||| +|||
||| IpRanges ||| +|||
||+-----+||| +|||
||| CidrIp ||| +|||
||+-----+||| +|||
||| 0.0.0.0/0 ||| +|||
||+-----+||| +|||
```

On retiendra l'ID du groupe de sécurité

```
export AWS_SGID="sg-04edf90e659364a62"
```

## 2.2. Clé d'accès

```
aws ec2 create-key-pair --key-name demo-lab-key --query 'KeyMaterial' --output text > ~/.ssh/demo-lab-key.pem
```

```
aws ec2 describe-key-pairs --key-name demo-lab-key
{
    "KeyPairs": [
        {
            "KeyName": "demo-lab-key",
            "KeyFingerprint": "75:96:3c:ae:00:4f:27:88:af:35:52:a1:b9:cd:6d:0e:ff:2c:f4:58"
        }
    ]
}
```

Restreindre les droits

```
chmod 400 ~/.ssh/demo-lab-key.pem
```

## 2.3. Lancer une instance t2.micro

```
aws ec2 run-instances \
--instance-type t2.micro \
--key-name demo-lab-key \
--security-group-ids $AWS_SGID \
--image-id $AWS_IMAGE
```

```
{
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": "",
            "StateReason": {
                "Message": "pending",
                "Code": "pending"
            },
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2018-10-28T12:34:52.000Z",
            "PrivateIpAddress": "172.31.35.250",
            "ProductCodes": [],
            "VpcId": "vpc-476c332e",
            "CpuOptions": {
                "CoreCount": 1,
                "ThreadsPerCore": 1
            },
            "StateTransitionReason": "",
            "InstanceId": "i-03f6971fdaff8558",
            "ImageId": "ami-0ebc281c20e89ba4b",
            "PrivateDnsName": "ip-172-31-35-250.eu-west-3.compute.internal",
            "KeyName": "demo-lab-key",
            "SecurityGroups": [
                {
                    "Status": "authorized"
                }
            ],
            "NetworkInterfaces": [
                {
                    "MacAddress": "enx03f6971fdaff",
                    "AssociatePublicIpAddress": true,
                    "Description": "Amazon VPC interface card",
                    "DeviceIndex": 0,
                    "DryRun": false,
                    "IamInstanceProfile": null,
                    "Ipv6Addresses": [
                        {
                            "Ipv6Address": "2607:2000:1000:1000::1"
                        }
                    ],
                    "Primary": true,
                    "PrivateDnsName": "ip-172-31-35-250.eu-west-3.compute.internal",
                    "PrivateIpAddress": "172.31.35.250",
                    "SecondaryPrivateIpAddressCount": 0,
                    "SubnetId": "subnet-00000000",
                    "Status": "in-use"
                }
            ],
            "RootDeviceType": "ebs",
            "RootDeviceName": "/dev/xvda",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "VirtualName": "/dev/sda1",
                    "Ebs": {
                        "VolumeSize": 8,
                        "DeleteOnTermination": true,
                        "VolumeType": "standard"
                    }
                }
            ],
            "EnvironnementVariables": [
                {
                    "Name": "AWS_INSTANCE_ID",
                    "Value": "i-03f6971fdaff8558"
                },
                {
                    "Name": "AWS_INSTANCE_TYPE",
                    "Value": "t2.micro"
                },
                {
                    "Name": "AWS_REGION",
                    "Value": "eu-west-3"
                },
                {
                    "Name": "AWS_AVAILABILITY_ZONE",
                    "Value": "eu-west-3a"
                },
                {
                    "Name": "AWS_PARTITION",
                    "Value": "aws"
                },
                {
                    "Name": "AWS_ACCOUNT_ID",
                    "Value": "123456789012"
                }
            ],
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "demo-lab-key"
                }
            ],
            "PrivateDnsName": "ip-172-31-35-250.eu-west-3.compute.internal",
            "PublicDnsName": "ip-172-31-35-250.eu-west-3.compute.amazonaws.com",
            "PublicIpAddress": "54.172.31.35",
            "PrivateIpAddress": "172.31.35.250",
            "State": "pending",
            "Type": "t2.micro"
        }
    ]
}
```

```

        "GroupName": "demo-lab",
        "GroupId": "sg-04edf90e659364a62"
    },
],
"ClientToken": "",
"SubnetId": "subnet-074deb4a",
"InstanceType": "t2.micro",
"NetworkInterfaces": [
{
    "Status": "in-use",
    "MacAddress": "0e:3e:0a:fc:59:56",
    "SourceDestCheck": true,
    "VpcId": "vpc-476c332e",
    "Description": "",
    "NetworkInterfaceId": "eni-092f4bd7687ea0b89",
    "PrivateIpAddresses": [
        {
            "PrivateDnsName": "ip-172-31-35-250.eu-west-3.compute.internal",
            "Primary": true,
            "PrivateIpAddress": "172.31.35.250"
        }
    ],
    "PrivateDnsName": "ip-172-31-35-250.eu-west-3.compute.internal",
    "Attachment": {
        "Status": "attaching",
        "DeviceIndex": 0,
        "DeleteOnTermination": true,
        "AttachmentId": "eni-attach-0f2b364177e190547",
        "AttachTime": "2018-10-28T12:34:52.000Z"
    },
    "Groups": [
        {
            "GroupName": "demo-lab",
            "GroupId": "sg-04edf90e659364a62"
        }
    ],
    "Ipv6Addresses": [],
    "OwnerId": "733718180495",
    "SubnetId": "subnet-074deb4a",
    "PrivateIpAddress": "172.31.35.250"
},
],
"SourceDestCheck": true,
"Placement": {
    "Tenancy": "default",
    "GroupName": "",
    "AvailabilityZone": "eu-west-3c"
},
"Hyperervisor": "xen",
"BlockDeviceMappings": [],
"Architecture": "x86_64",
"RootDeviceType": "ebs",
"RootDeviceName": "/dev/xvda",
"VirtualizationType": "hvm",
"AmiLaunchIndex": 0
},
],
"ReservationId": "r-091126bd779f312ce",
"Groups": [],
"OwnerId": "733718180495"
}
]

```

## 2.4. Instances EC2

Amazon Elastic Compute Cloud (Amazon EC2) est un service Web qui fournit une capacité de calcul sécurisée et redimensionnable dans le cloud. Destiné aux développeurs, il est conçu pour faciliter l'accès aux ressources de cloud computing à l'échelle du Web.

### Type d'instances

#### Types de virtualisation AMI Linux

#### Instances dédiées Amazon EC2

```
aws ec2 describe-instances
```

```
aws ec2 describe-instances --output table
```

```
aws ec2 describe-instances --output table
-----
|                               DescribeInstances                               |
+-----+
||                               Reservations                           ||
|+-----+-----+
||  OwnerId      | 733718180495          ||
||  ReservationId | r-091126bd779f312ce  ||
|+-----+-----+
|||                               Instances                            ||
||+-----+-----+
|||  AmiLaunchIndex | 0           ||
|||  Architecture   | x86_64        ||
|||  ClientToken    |             ||
|||  EbsOptimized   | False         ||
|||  EnaSupport     | True          ||
|||  Hypervisor     | xen           ||
|||  ImageId        | ami-0ebc281c20e89ba4b  ||
|||  InstanceId    | i-03f6971ffddaff8558  ||
|||  InstanceType   | t2.micro      ||
|||  KeyName        | demo-lab-key  ||
|||  LaunchTime     | 2018-10-28T12:34:52.000Z  ||
|||  PrivateDnsName | ip-172-31-35-250.eu-west-3.compute.internal  ||
|||  PrivateIpAddress | 172.31.35.250  ||
|||  PublicDnsName  | ec2-35-180-32-243.eu-west-3.compute.amazonaws.com  ||
|||  PublicIpAddress | 35.180.32.243  ||
|||  RootDeviceName | /dev/xvda    ||
|||  RootDeviceType | ebs          ||
|||  SourceDestCheck | True          ||
|||  StateTransitionReason |             ||
|||  SubnetId       | subnet-074deb4a  ||
|||  VirtualizationType | hvm          ||
|||  VpcId          | vpc-476c332e  ||
||+-----+-----+
|||                               BlockDeviceMappings                ||
|||+-----+-----+
||||  DeviceName      | /dev/xvda    ||
|||+-----+-----+
||||  Ebs                         ||
|||+-----+-----+
||||  AttachTime      | 2018-10-28T12:34:52.000Z  ||
||||  DeleteOnTermination | True          ||
||||  Status          | attached      ||
||||  VolumeId        | vol-043ff032d638e917a  ||
|||+-----+-----+
||||  CpuOptions        ||
|||+-----+-----+
||||  CoreCount        | 1           ||
||||  ThreadsPerCore  | 1           ||
|||+-----+-----+
||||  Monitoring        ||
|||+-----+-----+
||||  State            | disabled     ||
|||+-----+-----+
||||  NetworkInterfaces  ||
|||+-----+-----+
||||  Description      |             ||
||||  MacAddress       | 0e:3e:0a:fc:59:56  ||
||||  NetworkInterfaceId | eni-092f4bd7687ea0b89  ||
||||  OwnerId          | 733718180495  ||
||||  PrivateDnsName  | ip-172-31-35-250.eu-west-3.compute.internal  ||
||||  PrivateIpAddress | 172.31.35.250  ||
||||  SourceDestCheck  | True          ||
||||  Status           | in-use        ||
||||  SubnetId         | subnet-074deb4a  ||
||||  VpcId            | vpc-476c332e  ||
|||+-----+-----+
||||  Association        ||
|||+-----+-----+
||||  IpOwnerId        | amazon        ||
||||  PublicDnsName   | ec2-35-180-32-243.eu-west-3.compute.amazonaws.com  ||
||||  PublicIp          | 35.180.32.243  ||
|||+-----+-----+
||||  Attachment        ||
|||+-----+-----+
||||  AttachTime      | 2018-10-28T12:34:52.000Z  ||
||||  AttachmentId   | eni-attach-0f2b364177e190547  ||
||||  DeleteOnTermination | True          ||
||||  DeviceIndex     | 0           ||
||||  Status          | attached      ||
|||+-----+-----+
||||  Groups           ||
|||+-----+-----+
```

```
||||| GroupId      | sg-04edf90e659364a62      |||||
||||| GroupName    | demo-lab                    |||||
||||+-----+-----+-----+-----+-----+
|||||           PrivateIpAddresses      |||||
||||+-----+-----+-----+-----+-----+
||||| Primary       | True                      |||||
||||| PrivateDnsName | ip-172-31-35-250.eu-west-3.compute.internal |||||
||||| PrivateIpAddress | 172.31.35.250            |||||
||||+-----+-----+-----+-----+-----+
|||||           Association          |||||
||||+-----+-----+-----+-----+-----+
||||| IpOwnerId     | amazon                     |||||
||||| PublicDnsName | ec2-35-180-32-243.eu-west-3.compute.amazonaws.com |||||
||||| PublicIp       | 35.180.32.243             |||||
||||+-----+-----+-----+-----+-----+
|||||           Placement          |||||
||||+-----+-----+-----+-----+-----+
||||| AvailabilityZone | eu-west-3c                |||||
||||| GroupName      |                         |||||
||||| Tenancy        | default                   |||||
||||+-----+-----+-----+-----+-----+
|||||           SecurityGroups      |||||
||||+-----+-----+-----+-----+-----+
||||| GroupId       | sg-04edf90e659364a62      |||||
||||| GroupName     | demo-lab                  |||||
||||+-----+-----+-----+-----+-----+
|||||           State              |||||
||||+-----+-----+-----+-----+-----+
||||| Code          | 16                        |||||
||||| Name          | running                   |||||
||||+-----+-----+-----+-----+-----+
```

```
aws ec2 describe-instances --filters "Name=tag:Name,Values=demo-lab"
```

```
aws ec2 describe-instances --filters "Name=instance-type,Values=t2.micro" --query Reservations[].Instances[].InstanceId
```

```
[  
  "i-01295da4c589e3178"  
]
```

```
export AWS_INSTANCE="i-03f6971fddaff8558"
```

```
aws ec2 describe-instances \  
  --instance-ids $AWS_INSTANCE \  
  --query "Reservations[*].Instances[*].PublicDnsName"
```

## 2.5. Connexion à l'instance

```
ssh -i ~/.ssh/demo-lab-key.pem ec2-user@ec2-35-180-32-243.eu-west-3.compute.amazonaws.com
```

## 2.6. Opérations sur l'instance (Ansible)

```
sudo yum -y update  
pip install --upgrade pip  
sudo pip install ansible
```

```
# Configure and deploy Apache  
- hosts: localhost  
  connection: local  
  tasks:  
    - name: confirm using the latest Apache server  
      become: yes  
      become_method: sudo  
      yum:  
        name: httpd  
        state: latest  
    - name: reload service httpd, in all cases  
      become: yes  
      become_method: sudo
```

```

service:
  name: httpd
  state: reloaded
- name: check the service
  shell: curl -s 127.0.0.1
  register: check_apache
- debug:
    msg: "Is Apache installed ? {{ check_apache.stdout }}"
- name: Create index.html file for test
  become: yes
  become_method: sudo
  shell: echo 'True' > /var/www/html/index.html
  when: check_apache.stdout != 'True'

```

```
ansible-playbook apache.yml -v
```

## 2.7. Une application simple

```
sudo yum install --enablerepo=epel -y nodejs
```

Fichier hellworld.js

```

var http = require("http")

http.createServer(function (request, response) {

  // Send the HTTP header
  // HTTP Status: 200 : OK
  // Content Type: text/plain
  response.writeHead(200, {'Content-Type': 'text/plain'})

  // Send the response body as "Hello World"
  response.end('Hello World\n')
}).listen(3000)

// Console will print the message
console.log('Server running')

```

```
node hellworld.js
```

```

cat << EOF >> /etc/init/helloworld.conf
description "Hello world Deamon"

# Start when the system is ready to do networking.
start on started elastic-network-interfaces

# Stop when the system is on its way down.
stop on shutdown

respawn
script
  exec su --session-command="/usr/bin/node /home/ec2-user/helloworld.js" ec2-user
end script
EOF

```

Démarrage du service

```
sudo start helloworld
```

## 2.8. Terminer une instance

```
aws ec2 terminate-instances --instance-ids $AWS_INSTANCE
```

## 3. Déploiement avec Cloud-init

Exécution de commandes sur votre instance Linux lors du lancement

- Approvisionnement de stack Python ou Ansible

## 4. AWS EC2 avec Ansible

- roles ec2 aw s
- envoi de crédits
- suppression d'instances et de clés

## 5. AWS CloudFormation

AWS CloudFormation fournit un langage commun pour décrire et provisionner toutes les ressources d'infrastructure dans votre environnement cloud. CloudFormation vous permet d'utiliser un simple fichier texte pour modéliser et provisionner, de manière automatisée et sécurisée, toutes les ressources nécessaires pour vos applications à travers toutes les régions et tous les comptes. Ce fichier sert de source unique de vérité pour votre environnement cloud.

### 5.1. Fonctionnement AWS CloudFormation

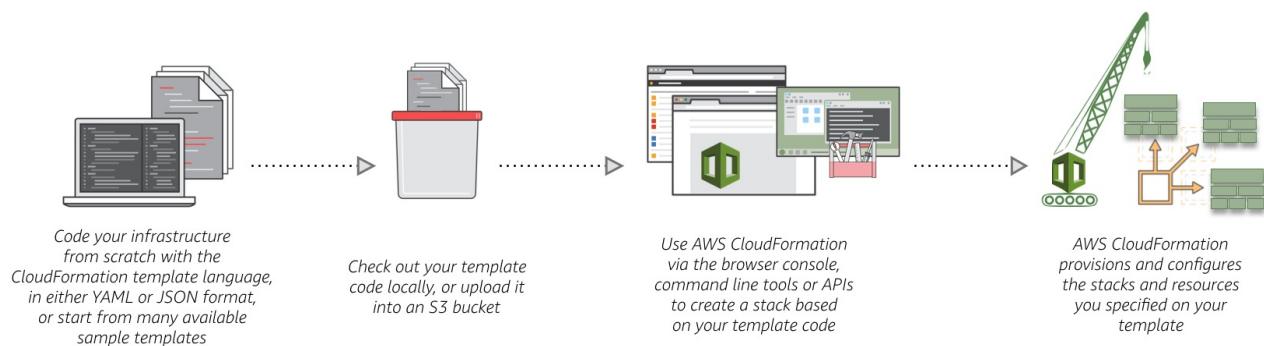


Figure 17 : Fonctionnement AWS CloudFormation

### 5.2. Notes

<https://console.aws.amazon.com/cloudformation/designer>

<https://templates.cloudonaut.io/en/stable/>

<https://github.com/widdix/learn-cloudformation/>

<https://medium.com/boltops/a-simple-introduction-to-aws-cloudformation-part-1-1694a41ae59d>

<https://github.com/tongueroo/cloudformation-examples/blob/master/templates/single-instance.yml>

## 6. LightSail

<https://lightsail.aws.amazon.com/ls/docs/en/articles/getting-started-with-amazon-lightsail>

## 7. Elastic Beanstack

[AWS Elastic Beanstalk - Didacticiels et exemples](https://aws.amazon.com/fr/elastic-beanstalk/)

# Pipeline : Gitlab-ci

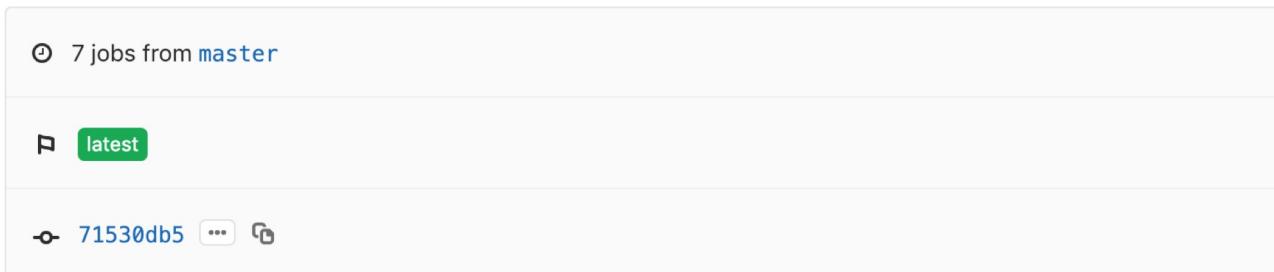
- 1. Idée
  - 1.1. Dev vers Ops
- 2. Boîte à outil
- 3. Le serveur d'intégration Gitlab / Gitlab-CI
  - 3.1. Mise en place d'un projet d'intégration continue sur Gitlab
  - 3.2. Exécution des jobs dans des conteneurs
- 4. Application : Toolchains
  - 4.1. Le Stack
  - 4.2. Gitbook-cli Toolchain
  - 4.3. Toolchains dans une image Docker
  - 4.4. Point de départ avec Netlify
  - 4.5. Autres générateurs et toolchains
- 4.6. Créer un contenu gitbook
- 5. Paramètres du pipeline
  - 5.1. Variables du pipeline
- 6. Test
- 7. Build
- 8. Deploy
- 9. Exécutions
  - 9.1. Noeud d'exécution Docker et registre d'images
  - 9.2. Serveur d'intégration
  - 9.3. Alternatives d'hébergement de site statique
- 10. Améliorations à proposer

## 1. Idée

gitbook-gitlab &gt; Pipelines &gt; #41109935

passed Pipeline #41109935 triggered 56 seconds ago by  goffinet

## mod



Pipeline Jobs 7

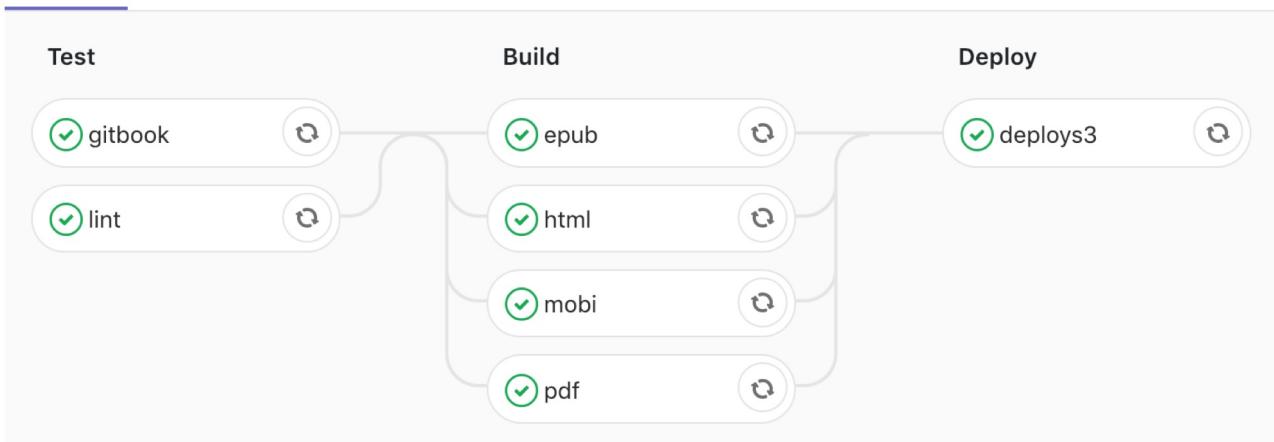


Figure 18 : Pipeline Gitlab-ci, Test, Build and Deploy

Inspiration : [AWS S3 + GitLab CI = automatic deploy for every branch of your static website](#)

Dans ce point de départ, le concepteur du contenu soumet son code au SCM.

Cette action enclenche une alerte auprès d'un serveur/service d'intégration continue qui exécute un pipeline divisé en quelques stages classiques (Test, Build, Deploy) interdépendants. Chaque stage comprend des jobs (ensembles de tâches) parallèles dont une exécution réussie fait passer à un stage suivant.

Les jobs sont habituellement exécutés dans des noeuds d'exécution partagés (sur Gitlab-ci) ou dédiés (gitlab-runner dans ce cas) habituellement dans le nuage. Ces exécuteurs sont des instances de calcul capables d'exécuter des images Docker. Nécessairement dans ce cadre des images Docker doivent être accessibles à travers un service de "repository" de conteneur. On peut aussi envisager la construction à la demande des images Docker, mais cette perspective est plus coûteuse. Il sera d'ailleurs conseillé d'embarquer l'outil de construction des artefacts.

Cette étape de la solution nécessite de s'intéresser aux outils d'intégration continue, à des toolchains différents basés node ou python, aux procédures spécifiques de tests, de construction et de déploiement. Le choix de docker comme exécuteur est certainement un choix judicieux qui pose questions sur son implémentation pour des raisons de performance.

Enfin, les étapes d'intégration continue et de la livraison continue sont annoncées sur le canal dédié d'un espace de travail Slack.

### 1.1. Dev vers Ops

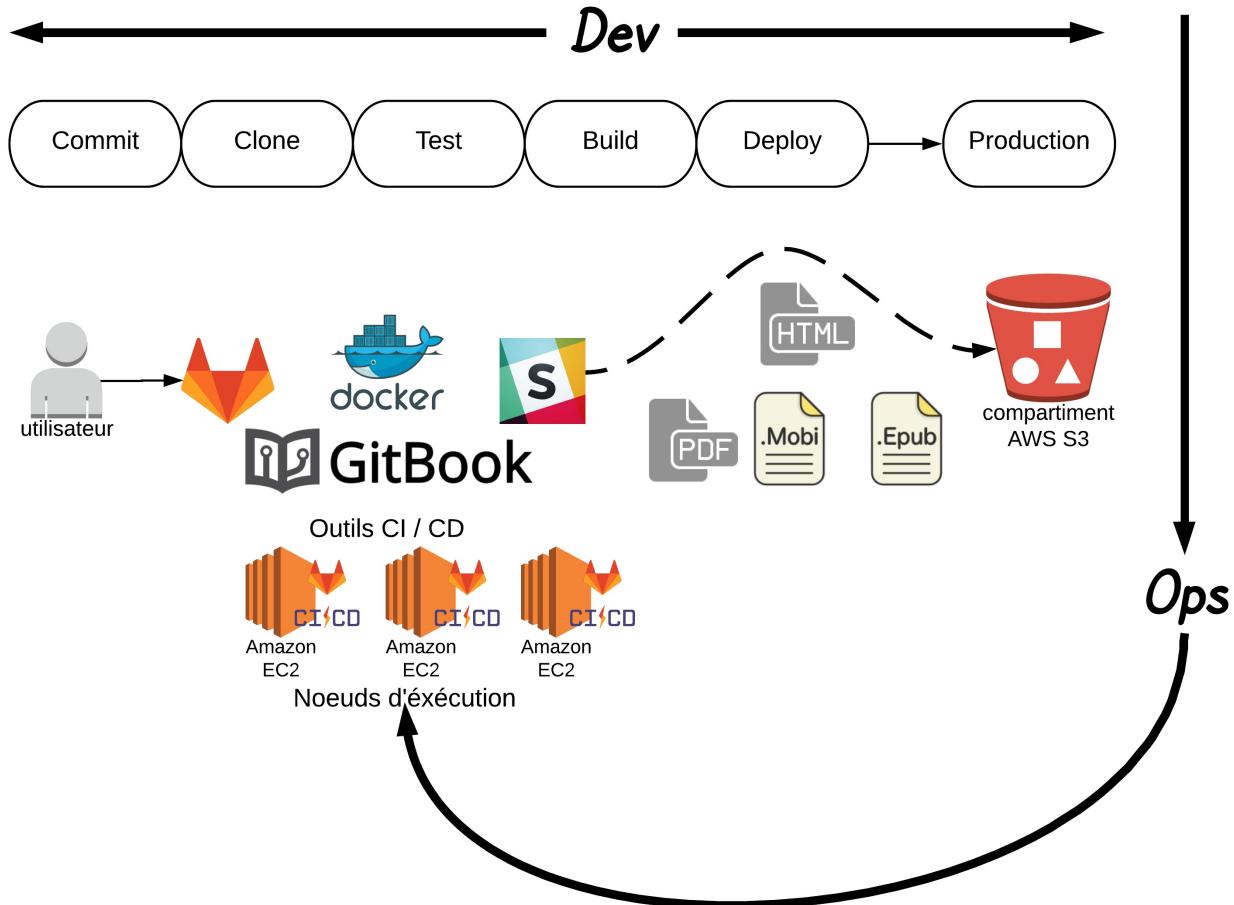


Figure 19 : Dev vers Ops

## 2. Boîte à outil

Ce tableau représente la proposition de départ.

Outil	Fournisseur	Alternatives
Source Control Management	Git, Gitlab	Github, Gitlab Hosted (AWS), AWS CodeCommit, ...
CI/CD Tool	Gitlab-ci, Gitlab-runner	Jenkins, AWS CodePipeline, AWS CodeDeploy, Bitbucket
Environnement CI/CD	Docker	local
Container Repository	Gitlab Hub	Docker Hub,
Test	versions, markdown-lint, npm (gitbook), pip (aws-cli)	...
Application Toolchain (build)	Gitbook-cli (basé NPM), calibre	...
Deploy	aws-cli s3	Python API, aws-cli, openstack-cli, Ansible, ...
Monitoring	Slack	AWS SNS, ...

Il est demandé de justifier le choix des outils sur base de différents critères :

- Coûts
- Centralisation, homogénéité
- Infrastructure as Code

- L'inter-opérabilité
- ...

Le Pipeline de base proposé est le suivant :

- Phase de test des outils et du code (test) : test des images docker et du code
- Phase de construction : usage des outils
- Phase de déploiement : déploiement auprès d'une infrastructure codée (ici sur AWS S3 en HTTPS/CDN)

## 3. Le serveur d'intégration Gitlab / Gitlab-CI

Gitlab fournit une solution d'intégration complète et facile à prendre en main.

Outil	Fonction
Git	outils de gestion de code standard
Gitlab	service SCM auto-hébergé ou hébergé
Gitlab-CI	service CI/CD auto-hébergé ou hébergé
Gitlab Runner	service d'exécution CI/CD auto-hébergé ou hébergé

### 3.1. Mise en place d'un projet d'intégration continue sur Gitlab

- Création du projet
- Fichier de configuration
- Clé SSH
- Variables privées
- Gitlab-ci
- Construction automatique des images Docker et hébergement de l'image sur un registre Docker
- Intégration d'un canal Slack

### 3.2. Exécution des jobs dans des conteneurs

Avec Gitlab-ci, il est habituel d'exécuter les jobs dans des conteneurs (voir plus bas).

[Migrating Netlify's Continuous Deployment infra to Kubernetes \(and everything we learned along the way\)](#).

Si vous connaissez Docker, vous savez probablement que la construction d'images spécialisées dans un but spécifique est généralement une bonne pratique. Par exemple, vous avez une image pour Nginx et son seul but est d'exécuter Nginx. Vous avez aussi une image pour Go, et son seul but est d'exécuter des commandes Go, et ainsi de suite. Dans les services traditionnels d'intégration continue, c'est très pratique - si vous voulez exécuter des tests pour votre application Rails, vous utilisez une image avec Ruby et Rails installés, et rien d'autre. Cela permet également de réduire la taille des images, ce qui est plus efficace lorsque vous devez les télécharger plusieurs fois à partir d'un registre de conteneurs.

La nature des projets que nos clients construisent sur Netlify rend cela moins pratique. Habituellement, un projet web implique une compilation JavaScript, nous avons donc besoin de Node. Il peut également avoir besoin d'un autre langage de programmation si vous utilisez un générateur de site ou des bibliothèques pour traiter les fichiers image. Avoir une image Docker spécialisée serait en fait un inconvénient pour nous car nos clients devraient les modifier pour chacun de leurs cas d'utilisation.

## 4. Application : Toolchains

### 4.1. Le Stack

L'application choisie n'utilise pas un compilateur C ou un framework JEE ou encore un célèbre framework Web PHP, Python, Node ou Angular.

L'application choisie est un générateur statique de sites Web [StaticGen : A List of Static Site Generators for JAMstack Sites](#).

### 4.2. Gitbook-cli Toolchain

**Gitbook Toolchain.** Gitbook-cli est un projet open-source à personnaliser soi-même et/ou à partir d'une librairie de "plug-ins" pour générer un site Web statique de type "documentation". Il génère un site Web statique et utilise le logiciel calibre pour produire des fichiers en format PDF, MOBI et EPUB.

Il est recommandé de s'exercer sur le Toolchain gitbook-cli en suivant et en comprenant de manière approfondie le document [GitBook Toolchain Documentation](#). Le détail de l'utilisation du toolchain est illustré dans le pipeline.

En bref au préalable,

- le toolchain gitbook-cli est écrit en nodejs (utilitaire npm)
- la commande `gitbook` transforme un dossier de contenu écrit en Markdown en différent formats via le logiciel
- des plugins doivent être installés à titre de dépendances ainsi que le logiciel calibre

### 4.3. Toolchains dans une image Docker

L'avantage d'une exécution par Docker est de disposer d'un environnement identique pour un développement local et automatique.

Il est demandé d'exécuter les opérations à partir d'une image Docker comme par exemple "[goffinet/gitbook](#)".

```
# Base image, default node image
FROM node:10-slim

# Environment configuration
ENV GITBOOK_VERSION="3.2.3"

# Install gitbook
RUN npm install --global gitbook-cli \
  && gitbook fetch ${GITBOOK_VERSION} \
  && npm install --global markdownlint-cli \
  && npm cache clear --force \
  && apt-get update \
  && apt-get install --no-install-recommends --no-install-suggests -y \
  #wget \
  #curl \
  #vim \
  #openssh-client \
  calibre \
  ttf-freefont \
  ttf-liberation fonts-liberation \
  && mkdir -p /root/.ssh \
  && rm -rf /var/lib/apt/lists/* \
  && rm -rf /tmp/*
```

### 4.4. Point de départ avec Netlify

A Step-by-Step Guide: [GitBook on Netlify](#)

Un livre de base est proposé ici : <https://github.com/goffinet/gitbooktest>

Netlify publie l'image qu'il utilise pour déployer ses services : [Netlify automated build image](#).

### 4.5. Autres générateurs et toolchains

Le projet pourrait prendre de la plus-value à partir d'un contenu écrit en Markdown pour **Jekyll** ou plus simplement pour **MkDocs-Material** qui au passage d'une moulinette fabriquerait le modèle "gitbook" pour générer le différents artefacts.

## 4.6. Créer un contenu gitbook

- création du projet
- construction et lancement du conteneur
- installation des plug-ins
- création du support

## 5. Paramètres du pipeline

Le pipeline de départ comporte trois stages : "Test", "Build", "Deploy". Il manque une phase "staging" qui teste les artefacts avant la phase "Deploy". Elle est à développer.

```
# This pipeline run three stages Test, Build and Deploy
stages:
- test
- build
- deploy
```

## 5.1. Variables du pipeline

Avec un serveur Gitlab, les variables publiques du pipeline peuvent être déclarées dans le fichier de configuration.

Le projet se déploie sur AWS S3 avec un utilisateur IAM déjà créé avec les bonnes autorisations. Il est préférable de définir les variables d'authentification AWS `AWS_ACCESS_KEY_ID` et `AWS_SECRET_ACCESS_KEY` dans l'interface du serveur Gitlab dans le menu `Project | Settings | CI / CD | Variables` en tant que variables protégées.

```
# Open variables for S3
# AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY are protected variables
# added in the gitlab interface : Project | Settings | CI / CD | Variables
variables:
AWS_DEFAULT_REGION: eu-west-3 # The region of our S3 bucket
BUCKET_NAME: gitbook-gitlab.aws-fr.com # gitbook-gitlab.aws-fr.com Your bucket name
CDN_DISTRIBUTION_ID: E1956KV4Y1NHX7
```

## 6. Test

Ce stage "Test" a pour objectif de tester le stack et le code qui servira à l'étape suivante "Build". Il comporte deux jobs parallèles "gitbook" et "lint".

Le job "gitbook" utilise l'image docker privée `registry.gitlab.com/goffinet/gitbook-gitlab:latest` qui comprend tous les outils qui seront utilisés pour la fabrication des documents. Il teste la présence de nodejs et de gitbook-cli.

```
# the 'gitbook' job will test the gitbook tools
gitbook:
stage: test
image: registry.gitlab.com/goffinet/gitbook-gitlab:latest
script:
- 'echo "node version: $(node -v)"'
- gitbook -V
- calibre --version
allow_failure: false
```

Le job "Lint" teste nodejs, markdown-lint et teste la qualité des documents Markdown `README.md` et du sous-dossier `content` de manière non contraignante (`allow_failure: true`).

```
# the 'lint' job will test the markdown syntax
lint:
stage: test
image: registry.gitlab.com/goffinet/gitbook-gitlab:latest
script:
- 'echo "node version: $(node -v)"'
- echo "markdownlint version:" $(markdownlint -V)
- markdownlint --config ./markdownlint.json README.md
- markdownlint --config ./markdownlint.json ./content/*.md
allow_failure: true
```

## 7. Build

On trouvera quatre tâches parallèles de construction dans le stage "Build". Dans cette proposition la prochaine étape "Deploy" s'enclenchera après la fin réussie de tous ces jobs de construction. Il est fort probable que la tâche de déploiement ne soit jamais enclenchée en cas d'erreur ou de délais dans une des tâches de construction.

```
# the 'html' job will build your document in html format
html:
stage: build
dependencies:
- gitbook
- lint
image: registry.gitlab.com/goffinet/gitbook-gitlab:latest
```

```

script:
  - gitbook install # add any requested plugins in book.json
  - gitbook build . book # html build
artifacts:
  paths:
    - book
  expire_in: 1 day
only:
  - master # this job will affect only the 'master' branch the 'html' job will build your document in pdf format
allow_failure: false

# the 'pdf' job will build your document in pdf format
pdf:
  stage: build
  dependencies:
    - gitbook
    - lint
  image: registry.gitlab.com/goffinet/gitbook-gitlab:latest
  before_script:
    - mkdir ebooks
  script:
    - gitbook install # add any requested plugins in book.json
    - gitbook pdf . ebooks/${CI_PROJECT_NAME}.pdf # pdf build
  artifacts:
    paths:
      - ebooks/${CI_PROJECT_NAME}.pdf
    expire_in: 1 day
only:
  - master # this job will affect only the 'master' branch the 'pdf' job will build your document in pdf format

# the 'epub' job will build your document in epub format
epub:
  stage: build
  dependencies:
    - gitbook
    - lint
  image: registry.gitlab.com/goffinet/gitbook-gitlab:latest
  before_script:
    - mkdir ebooks
  script:
    - gitbook install # add any requested plugins in book.json
    - gitbook epub . ebooks/${CI_PROJECT_NAME}.epub # epub build
  artifacts:
    paths:
      - ebooks/${CI_PROJECT_NAME}.epub
    expire_in: 1 day
only:
  - master # this job will affect only the 'master' branch

# the 'mobi' job will build your document in mobi format
mobi:
  stage: build
  dependencies:
    - gitbook
    - lint
  image: registry.gitlab.com/goffinet/gitbook-gitlab:latest
  before_script:
    - mkdir ebooks
  script:
    - gitbook install # add any requested plugins in book.json
    - gitbook mobi . ebooks/${CI_PROJECT_NAME}.mobi # mobi build
  artifacts:
    paths:
      - ebooks/${CI_PROJECT_NAME}.mobi
    expire_in: 1 day
only:
  - master # this job will affect only the 'master' branch

```

## 8. Deploy

La phase "Deploy" utilise une autre image `python:latest` qui installe le "stack" aw s-cl. Celle-ci sera utile au transfert des fichiers sur le Bucket AWS S3. On suppose que le Bucket est déjà configuré, que le domaine existe ainsi qu'un enregistrement ALIAS qui pointe vers une distribution Cloudfront associée au Bucket et à un certificat TLS. Ces "pré-supposés" devraient faire partie du projet d'intégration continue sous forme de code de déploiement (cloudformation, terraform, ansible, aw scli, python3 boto3)

```

deploys3:
  image: "python:latest" # We use python because there is a well-working AWS Sdk
  stage: deploy
  dependencies:

```

```

- html
- pdf
- mobi
- epub # We want to specify dependencies in an explicit way, to avoid confusion if there are different build jobs
before_script:
- pip install awscli # Install the SDK
script:
- aws s3 sync book s3://${BUCKET_NAME} --acl public-read --delete
- aws s3 sync ebooks s3://${BUCKET_NAME}/ebooks --acl public-read --delete
- aws cloudfront create-invalidation --distribution-id ${CDN_DISTRIBUTION_ID} --paths "/*" # The next time a viewer requests
the file, CloudFront returns to the origin to fetch the latest version of the file.
environment:
  name: production

```

## 9. Exécutions

### 9.1. Noeud d'exécution Docker et registre d'images

On remarquera qu'une implémentation DevOps en intégration continue demande des ressources locales ou dans le nuage.

Si ces instances d'exécution sont situées dans le nuage elle sont soit partagées (offertes dans les limites d'une offre payante ou gratuite), soit dédiée mais surtout à la charge du client.

Le pipeline est exécuté dans le nuage avec des conteneur docker à partir d'images stockées elle-même dans le nuage. Leur mise à disposition pose question. Quel serait la balance de coût et d'impact entre le téléchargement d'images et une construction locale ? Quel est le délai de construction en fonction des type d'instance. (exercice de test, résultats à inclure).

Quo qu'il en soit, dans notre cas d'étude, il est nécessaire de disposer de noeud d'exécution lié au serveur/service Gitlab. Faut-il dédier une instance statique (à quelle dimension), ne faut-il pas réfléchir à une infrastructure de noeuds d'exécution plus élastique ? A qui confier cette infrastructure, à AWS, à un autre, à Gitlab-ci ?

Gitlab-ci offre des noeuds d'exécution partagés gratuits, mais que l'on peut acheter avec services, que l'on peut déployer avec terraform sur Google Cloud Platform (GCP) et offre la possibilité d'intégrer facilement des noeuds externes à un serveur Gitlab ou au service.

La mise à disposition des noeuds d'exécution Docker doit être intégrée à la partie Ops sous forme IaC (Infrastructure as Code) Cloudformation, Ansible ou Terraform.

### 9.2. Serveur d'intégration

- Service Gitlab-ci avec gitlab-runner partagés
- Service Gitlab-ci avec gitlab-runner dédié chez AWS EC2 (un réutilisable dans ce PoC) ou autre (attention souci de "scalabilité")
- Solution sur l'AWS Market Place avec gitlab server "scalable" sur AWS

### 9.3. Alternatives d'hébergement de site statique

- Le gitlab-runner pourrait être le serveur d'hébergement.
- AWS S3, ACM, Route 53, Cloudfront tout intégré AWS
- AWS EC2 Natif Ubuntu/Centos Apache HTTPD avec cloudfare/Let's Encrypt
- AWS EC2 Natif Ubuntu/Centos Nginx

## 10. Améliorations à proposer

- Choix du service SCM.
- Choix et emplacement du serveur d'intégration.
- Choix et emplacement des noeuds d'exécution.
- Choix du registry Docker
- Fabrication automatique des images "container" sur un registry (docker hub, gitlab hub ou encore AWS).
- Fabrication d'une seule image intégrée (gitbook et aw s-cl) en couches ou unique.
- Mieux intégrer npm et calibre, voire pandoc dans le stage "build".
- Découpler la phase "deploy" et ses dépendances pour chaque build.
- Ajouter un stage post-test sur les artefacts.
- Convertir en livre de jeu Ansible.

- Partir d'un contenu jekyll pour la sortie HTML et markdown sans frontmatter pour traitement ultérieur en PDF, MOBI, EPUB, Kindle, notamment avec Gitbook ou MkDocs-material.
- Test du site Web (OWASP Top 10)

# Serveur de génération sur AWS

- 1. Gitlab-ci sur AWS EC2
  - 1.1. AWS Marketplace
  - 1.2. Installation manuelle
  - 1.3. HA (Highly Available) configuration for GitLab on AWS
- 2. Alternative : Jenkins sur AWS EC2
- 3. Architecture Gitlab sur AWS
- 4. Installation et configuration de Gitlab Runner
- 5. Scénario de vie / Orchestration

Scénarios de montée en charge

## 1. Gitlab-ci sur AWS EC2

### 1.1. AWS Marketplace

Installation de [GitLab Community Edition AMI](#) sur l'AWS Marketplace :

The screenshot shows the AWS Marketplace interface for the GitLab Community Edition. The top navigation bar includes 'Categories', 'Delivery Methods', 'Solutions', 'Migration Mapping Assistant', 'Your Saved List', 'Partners', 'Sell in AWS Marketplace', and 'Amazon Web Services Home'. The search bar is on the right. Below the navigation is a product card for 'GitLab Community Edition' with a logo, a yellow 'Continue to Subscribe' button, and tabs for 'Overview', 'Pricing' (which is active), 'Usage', 'Support', and 'Reviews'. The main content area has a teal header 'Pricing Information' with the sub-section 'Estimating your costs'. It asks to choose a region (EU Frankfurt) and fulfillment option (64-bit x86 AMI). It shows 'Software Pricing Details' for 'GitLab Community Edition' at \$0/hr running on t2.medium, and 'Infrastructure Pricing Details' for an estimated cost of \$0.054 EC2/hr. A note about the 'Free Tier' is present. To the right, a table lists the current software and infrastructure pricing for services hosted in EU (Frankfurt) for various instance types:

GitLab Community Edition				
	Instance Type	Hourly Price	On-Demand Price	On-Demand Cost
<input type="radio"/>	t2.nano	\$0	\$0.007	\$0.007
<input type="radio"/>	t2.micro	\$0	\$0.013	\$0.013
<input type="radio"/>	t2.small	\$0	\$0.027	\$0.027
<input checked="" type="radio"/>	t2.medium	\$0	\$0.054	\$0.054
<small>★Vendor Recommended</small>				
<input type="radio"/>	t2.large	\$0	\$0.107	\$0.107
<input type="radio"/>	t3.nano	\$0	\$0.006	\$0.006
<input type="radio"/>	t3.micro	\$0	\$0.012	\$0.012

Figure 20 : GitLab Community Edition AMI sur l'AWS Marketplace

L'image est gratuite et une instance t2.medium (2 vCPUs et 4G RAM) est recommandé selon un modèle de coût pour une seule instance :

- 0,054 \$ /heure
- 1,3 \$ /jour
- 40,18 \$ /mois

## 1.2. Installation manuelle

- <https://about.gitlab.com/install/#ubuntu>

## 1.3. HA (Highly Available) configuration for GitLab on AWS

[Installing GitLab on Amazon Web Services \(AWS\)](#)

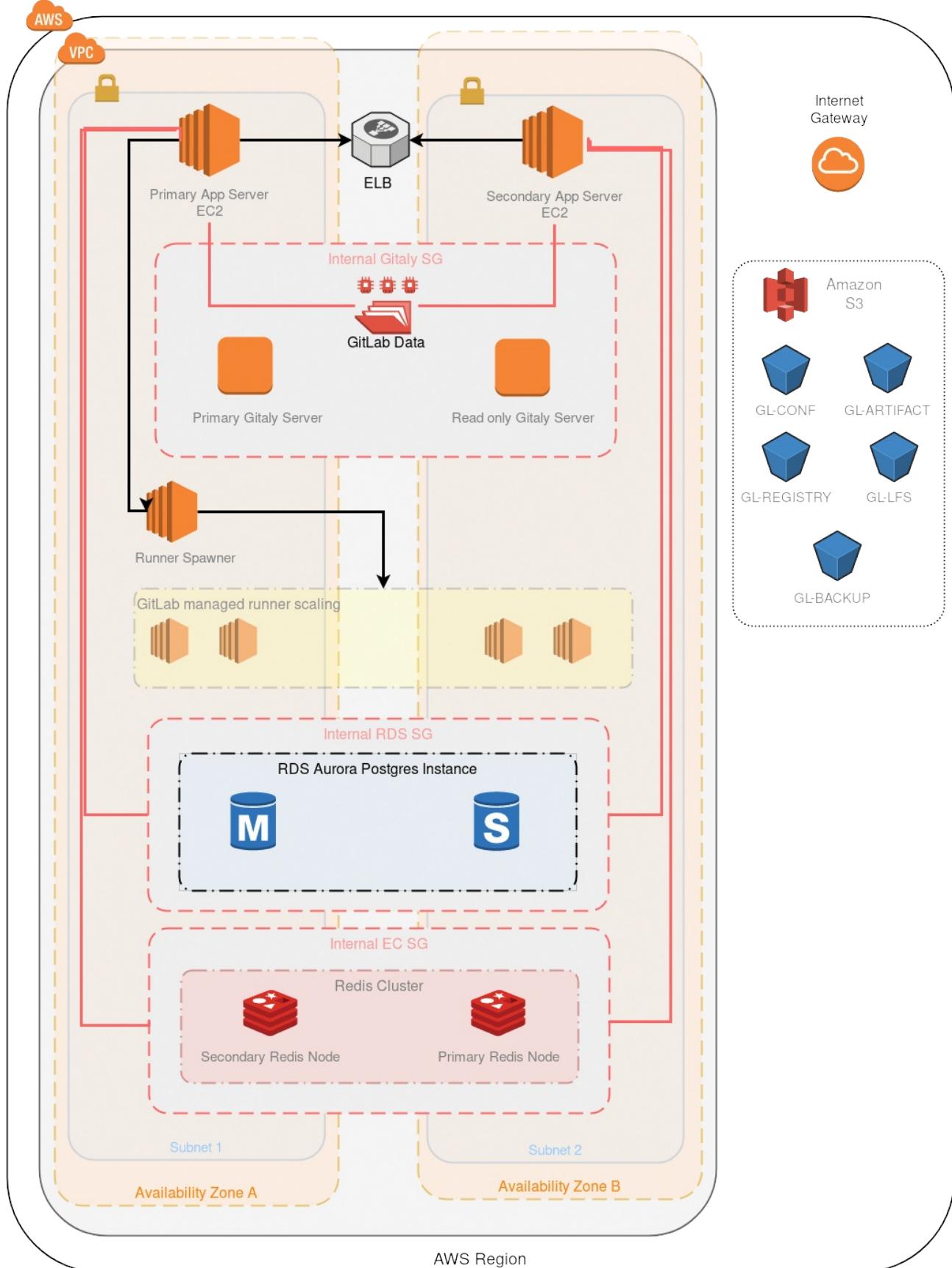


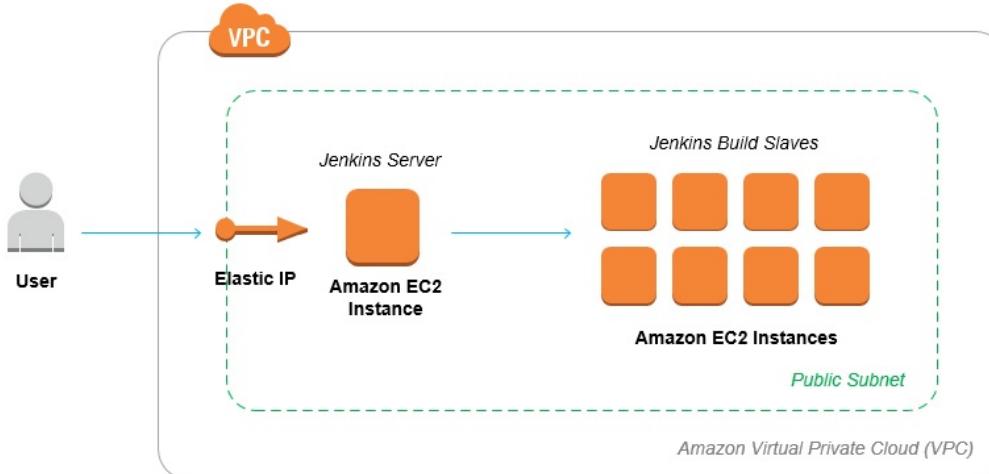
Figure 21 : AWS Architecture for Gitlab

## 2. Alternative : Jenkins sur AWS EC2

Configuration d'un serveur de génération Jenkins

## Set Up a Jenkins Build Server

Quickly create a build server for Continuous Integration (CI) on AWS



AWS Reference Architectures



Figure 22 : Configuration d'un serveur de génération Jenkins

### 3. Architecture Gitlab sur AWS

Autoscaling GitLab Runner on AWS

Autoscale GitLab CI runners and save 90% on EC2 costs

### 4. Installation et configuration de Gitlab Runner

...

### 5. Scénario de vie / Orchestration

...

## Exercices

- Projet 1. Créer un Bucket de type website et y pousser une page Web personnalisée
  - 1.a. En AWS CLI
  - 1.b. En Python 3
  - 1.c. Avec Ansible
- Projet 2. Reproduire le lab d'automation Wordpress
  - 2.a. En AWS CLI
  - 2.b. En Python 3
  - 2.c. Avec Ansible
- Projet 3. Déployer un site Web Statique en HTTPS CDN IPV6 sur un domaine
  - 3.a. Avec Ansible
  - 3.b. Avec Cloudformation
  - 3.c. Avec Terraform
- Projet 4. Configurer son projet Gitlab-gitbook
  - 4.a. Mise en place
  - 4.b. Améliorations à proposer
- Projet 5. Créer et publier un présentation en HTML 5

## Projet 1. Créer un Bucket de type website et y pousser une page Web personnalisée

### 1.a. En AWS CLI

Publier un repo `user/s3-website-cli`.

### 1.b. En Python 3

Publier un repo `user/s3-website-python3`.

### 1.c. Avec Ansible

Publier un repo `user/s3-website-ansible`.

## Projet 2. Reproduire le lab d'automation Wordpress

A partir d'une AMI Ubuntu par exemple.

### 2.a. En AWS CLI

publier un repo `user/ec2-wordpress-cli`.

Voir [Exécution de commandes sur votre instance Linux lors du lancement](#)

Script "init" disponible sur <https://raw.githubusercontent.com/goffinet/aws-112018/master/scripts/script8.sh>.

### 2.b. En Python 3

### 2.c. Avec Ansible

## Projet 3. Déployer un site Web Statique en HTTPS CDN IPV6 sur un domaine

### 3.a. Avec Ansible

publier un repo `user/s3-https-cdn-ansible` .

### 3.b. Avec Cloudformation

publier un repo `user/s3-https-cdn-cf` .

### 3.c. Avec Terraform

publier un repo `s3-https-cdn-tf` .

## Projet 4. Configurer son projet Gitlab-gitbook

### 4.a. Mise en place

publier un repo `user/gitlab-gitbook`

- Fork du projet
- clé SSH
- image conteneur docker
- fichier `gitlab-ci.yml`
- Bucket S3
- Variables

### 4.b. Améliorations à proposer

- Choix du service SCM.
- Choix et emplacement du serveur d'intégration.
- Choix et emplacement des noeuds d'exécution.
- Choix du registry Docker
- Fabrication automatique des images "container" sur un registry (docker hub, gitlab hub ou encore AWS).
- Fabrication d'une seule image intégrée (gitbook et aws-cli) en couches ou unique.
- Mieux intégrer npm et calibre, voire pandoc dans le stage "build".
- Découpler la phase "deploy" et ses dépendances pour chaque build.
- Ajouter un stage post-test sur les artefacts.
- Convertir en livre de jeu Ansible.
- Partir d'un contenu jekyll pour la sortie HTML et markdown sans frontmatter pour traitement ultérieur en PDF, MOBI, EPUB, Kindle, notamment avec Gitbook ou MkDocs-material.
- Test du site Web (OWASP Top 10)

## Projet 5. Créer et publier un présentation en HTML 5

publier un repo `user/devops-aws-ppt` .

Exemples :

- [LandSlide - Generate HTML5 slideshows from markdown, ReST, or textile](#)
- [Example Presentations in reveal.js](#)
- [Gitpitch](#)