



Badr  
**TAJINI**

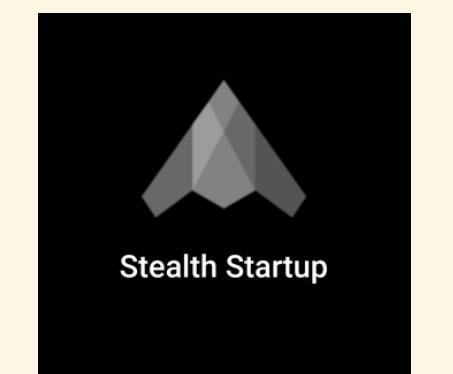
---

Lecturer-Researcher

-  
Co-founder & CTO

## About me

- Co-founder & CTO - Researcher in Robotics & AI
- Member of the Parietal team - GdR CNRS 2286 – Mathematical Imaging and its Applications (MIA)
- Research interests :
  - Agentic AI & Robotics (VLAs)
  - Computer Vision (VLMs),
  - Natural Language Processing (LLMs),
  - Brain NeuroImaging and Neurosciences.
- Last Job : ML Research & Systems, EDF
- Currently :
  - Lecturer-Researcher at ESIEE engineering school
  - Co-founder & CTO - SF Bay Area (Stealth Startup)



DevOps  
Data for  
SWE

# It's DevOps!



**Chapter 1, Introduction à DevOps et à la livraison de logiciels**

**Chapter 2, Comment gérer votre infrastructure en tant que code**

**Chapter 3, Comment déployer vos applications**

**Chapter 4, Comment versionner, construire et tester votre code**

**Chapter 5, Comment mettre en place l'intégration continue (CI) et la livraison continue (CD)**

**Chapter 6, Comment travailler avec plusieurs équipes et environnements**

**Chapter 7, Comment configurer un réseau**

# Chapter I :

# Introduction

# to DevOps and

# Software

# Delivery



# Chapter 1 : Introduction à DevOps et à la livraison de logiciels

"J'ai codé une application. Et maintenant ?"

# Chapter 1 :

## Métriques

### DORA et performance

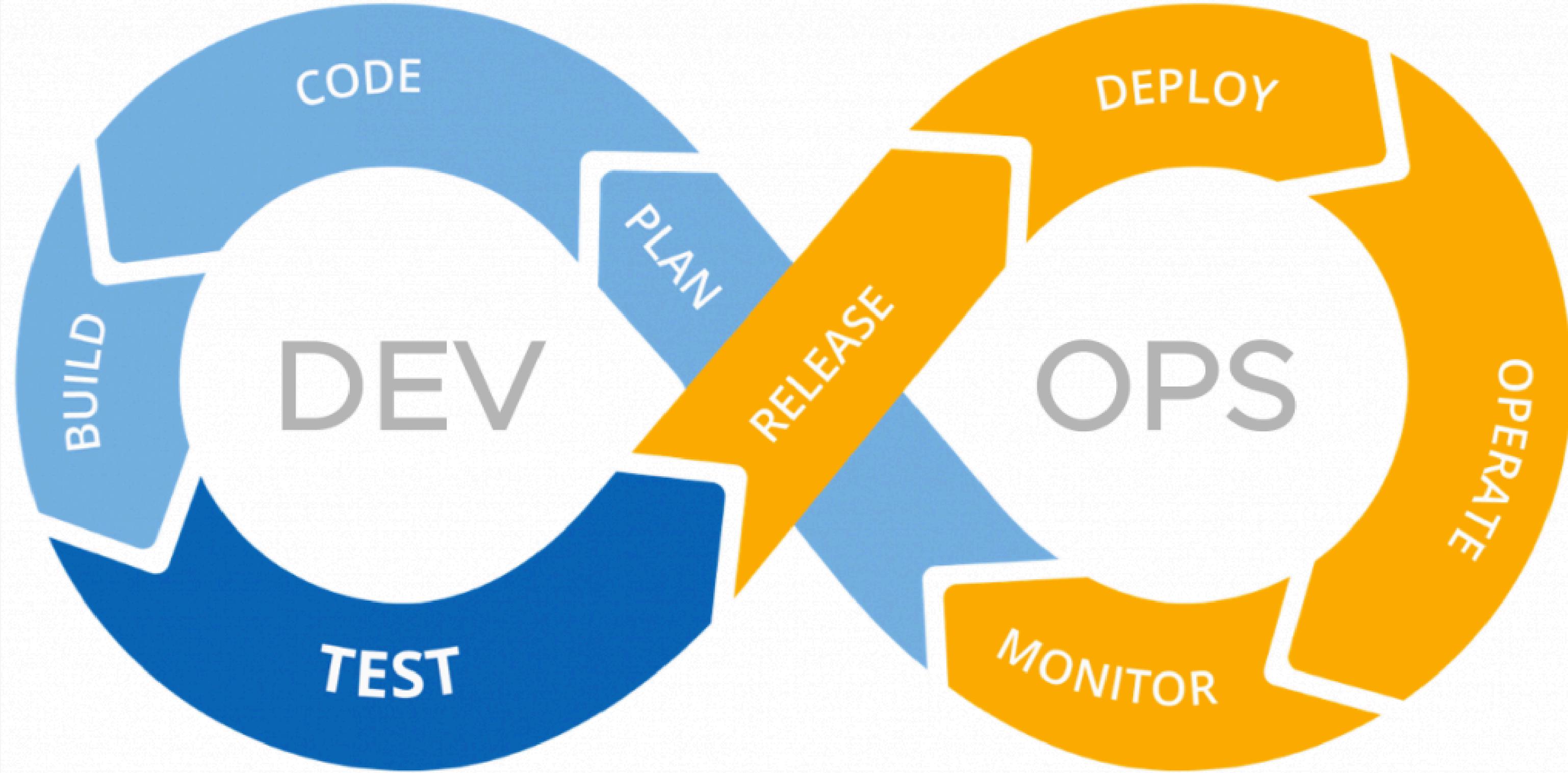
### DevOps

Metric	Description	Elite vs low performers <sup>a</sup>
Deployment frequency	How often you deploy to production	~10-200x more often
Lead time	How long it takes a change to go from committed to deployed	~10-200x faster
Change failure rate	How often deployments cause failures that need immediate remediation	~13x lower
Recovery time	How long it takes to recover from a failed deployment	~700-4000x faster

Observation “a” : le rapport indique que la fréquence de déploiement pour les entreprises d’élite est de « plusieurs déploiements par jour », ce que j'estime de manière prudente à 2-10 déploiements par jour, et que pour les entreprises peu performantes, elle se situe « entre une fois par semaine et une fois par mois ». Avec 5 jours de travail par semaine et 4 semaines par mois, cela donne 40 à 200 déploiements par mois pour l’élite, contre 1 à 4 déploiements par mois pour les moins performants, soit une différence de 10 à 200 fois.

Table 1-1. DORA metrics performance from the [2023 State of DevOps Report](#)

# Chapter 1 : Évolution et intégration de DevOps



# Chapter 1 : Définir DevOps et son évolution

Table 1-2. The shift in mindset with the cloud and DevOps

	Before	After
Teams	Devs toss code “over the wall” to Ops	Devs & Ops work together on cross-functional teams
Servers	Dedicated physical servers	Elastic virtual servers
Connectivity	Static IPs	Dynamic IPs, service discovery
Security	Physical, strong perimeter, high trust	Virtual, end-to-end, zero trust
Server provisioning	Manual	Infrastructure as Code (IaC) tools
Server configuration	Manual	Configuration management tools
Testing	Manual	Automated testing
Deployments	Manual	Automated
Change process	Change request tickets	Self-service
Deployment cadence	Weeks or months	Many times per day
Change cadence	Weeks or months	Minutes

# Chapter 1 : Évolution de DevOps : les stades (stages)

Stage 1

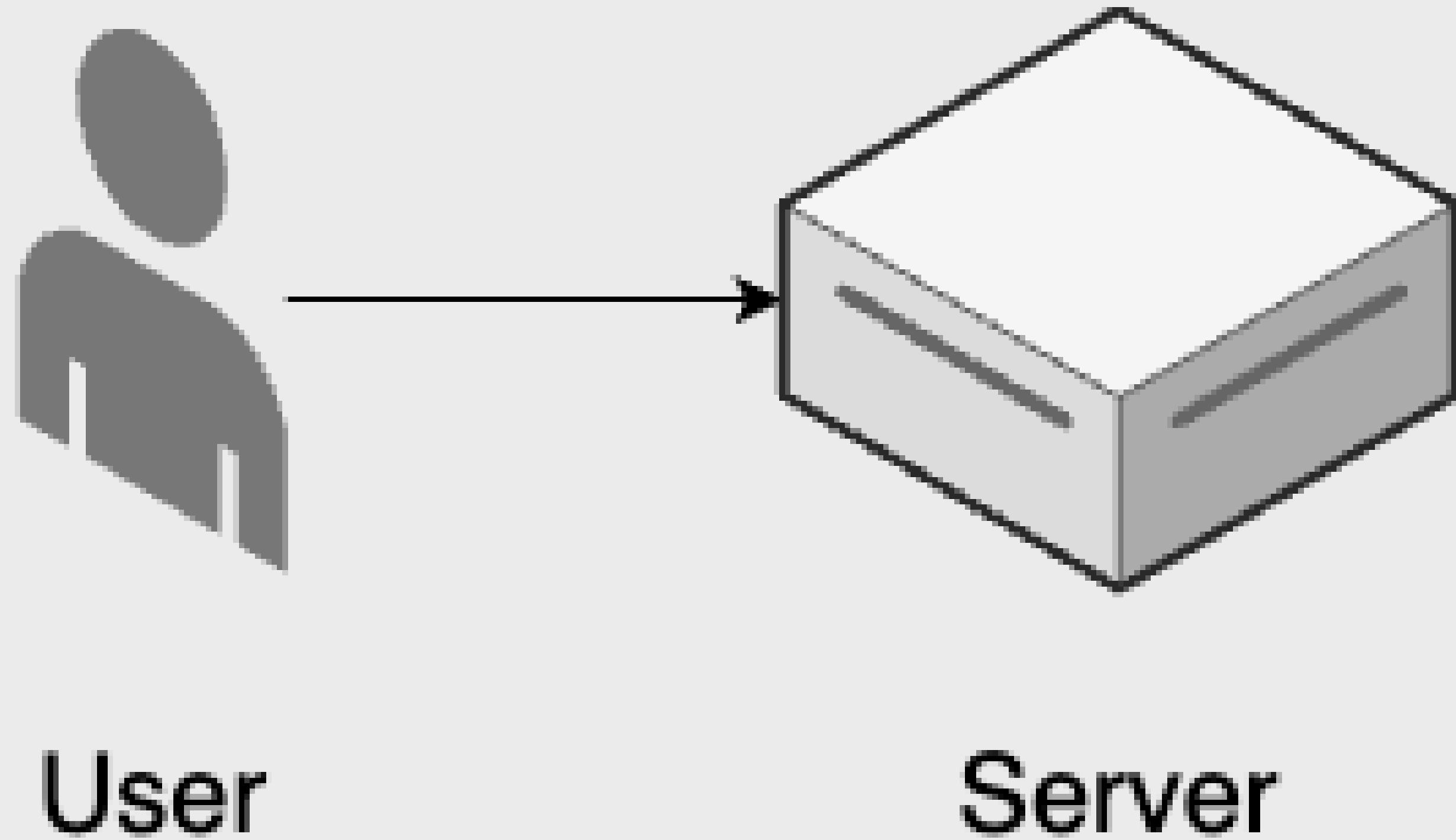


Figure 1.1 Step 1

# Chapter 1 : Évolution de DevOps : les stades (stages)

Stage 1

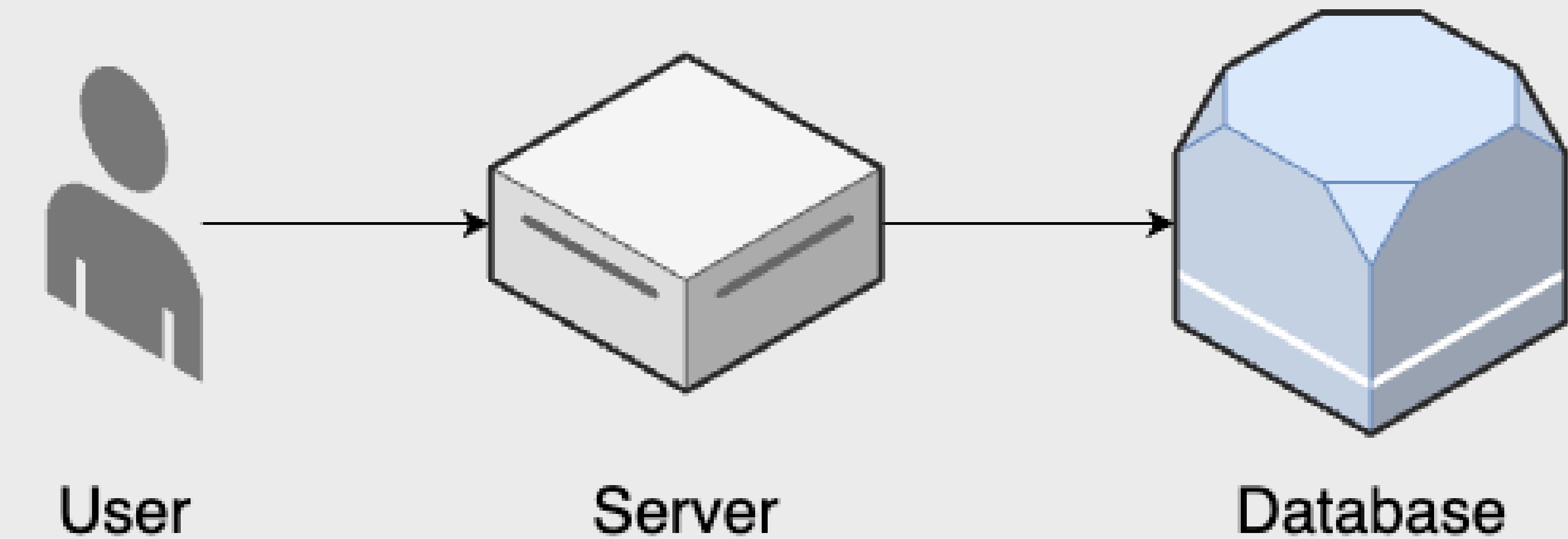
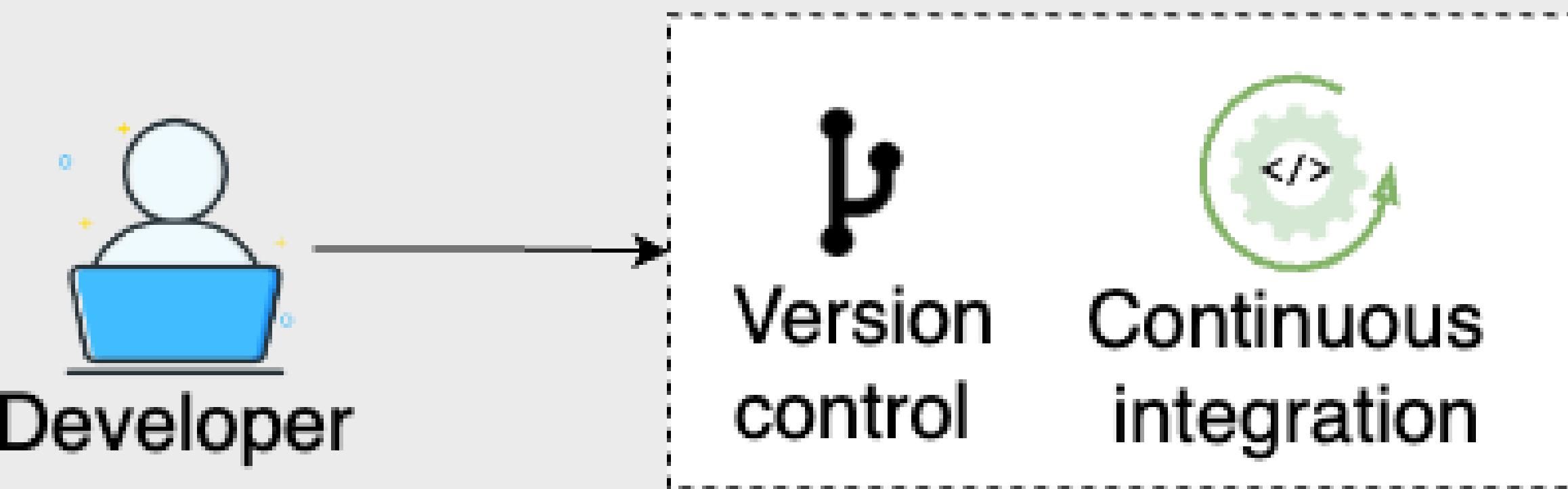


Figure 1.2 Step 2



# Chapter 1 :

## Évolution de DevOps : les stades (stages)

Stage 1

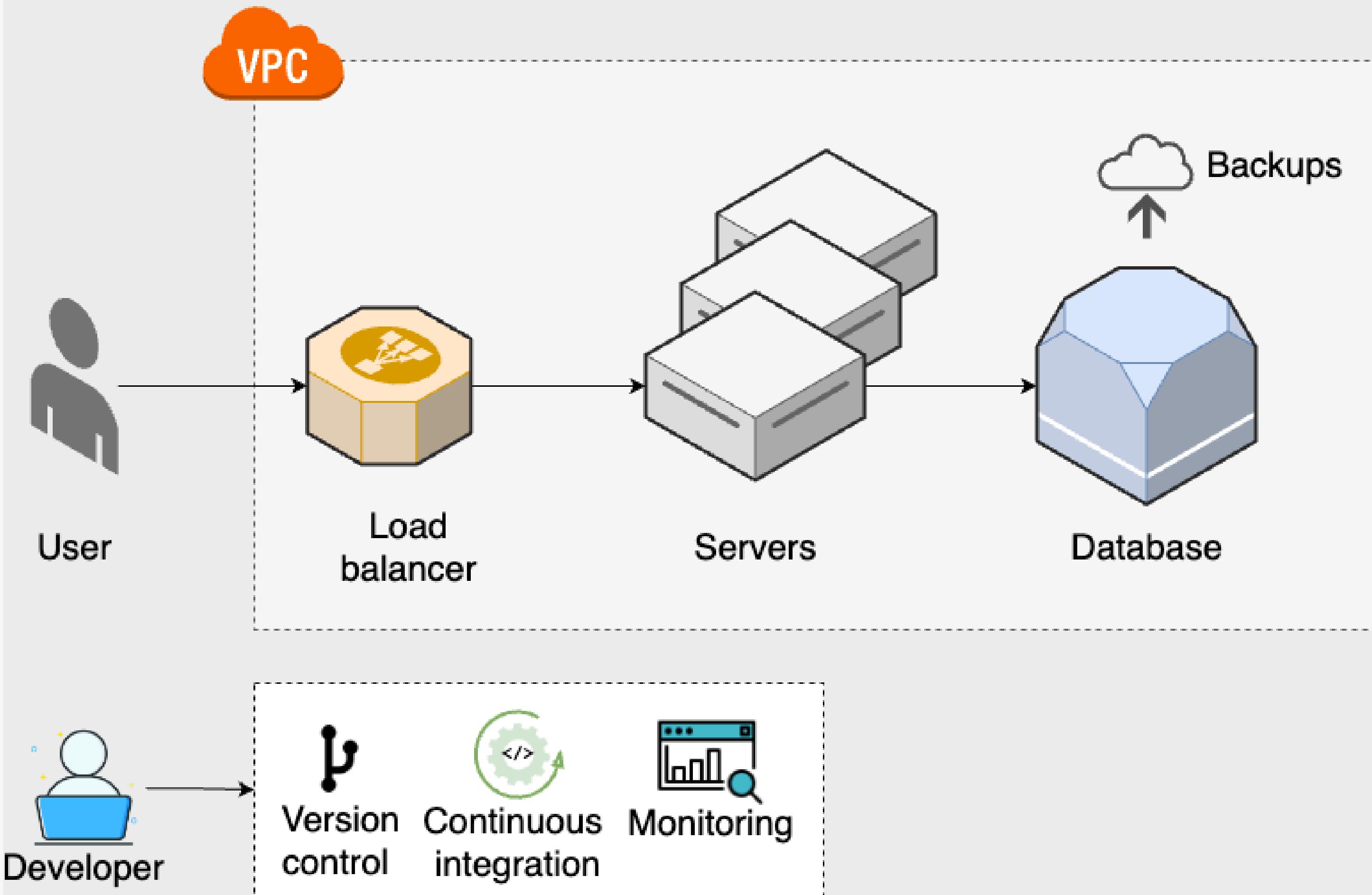


Figure 1.3 Step 3

# Chapter 1 : Évolution de DevOps : stades 2 et 3

Stage 2

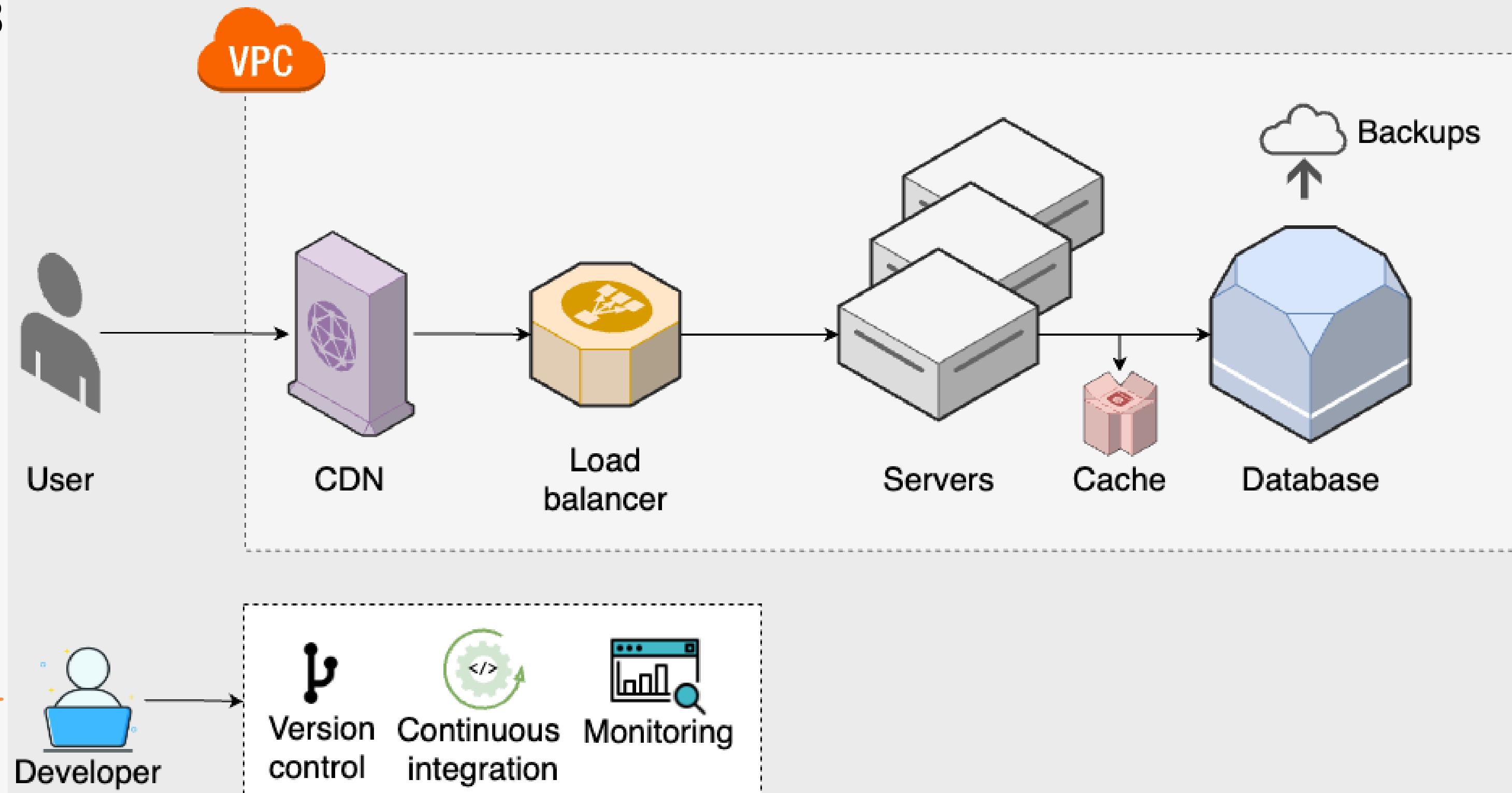


Figure 1.4 Step 4

# Chapter 1 :

## Évolution de DevOps : stades 2 et 3

Stage 2

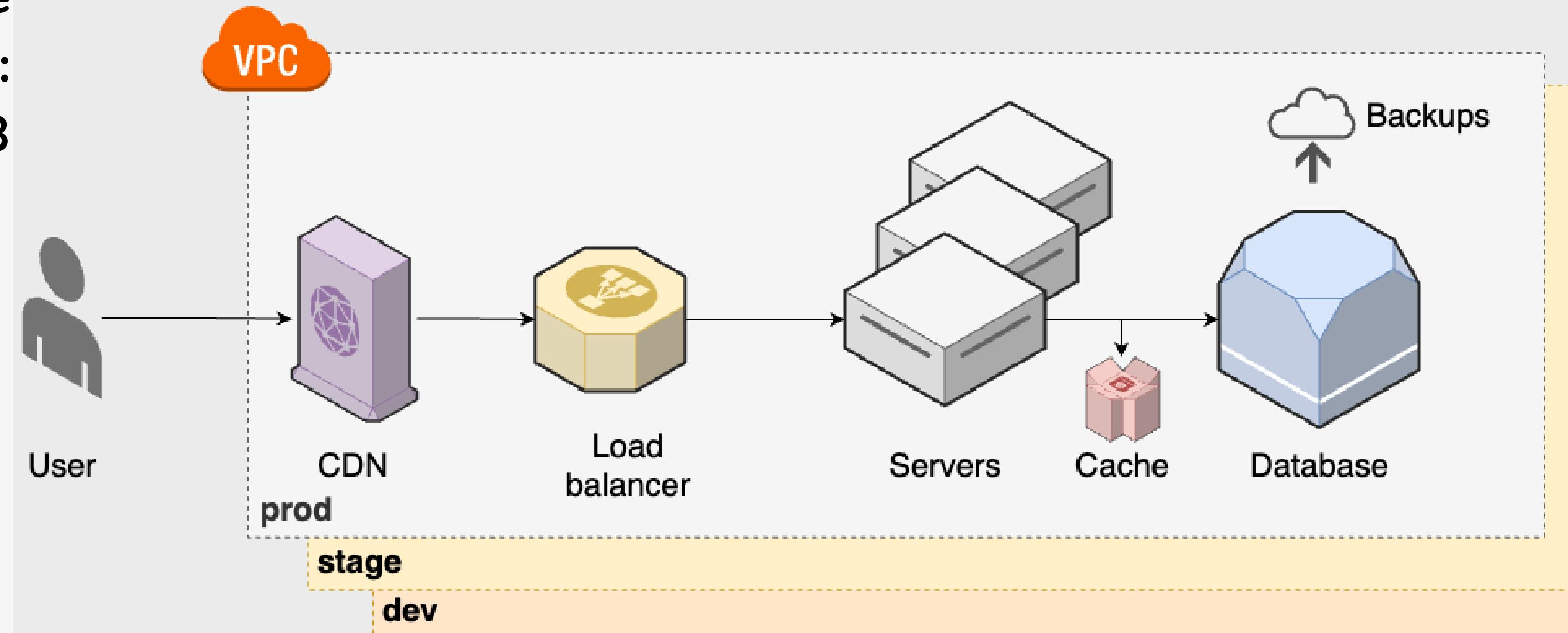
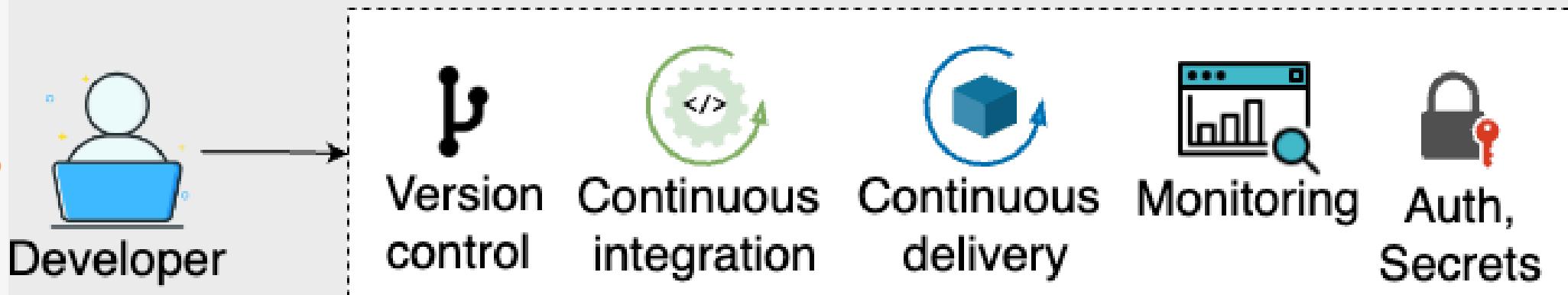


Figure 1.5 Step 5



# Chapter 1 :

## Évolution de DevOps : stades 2 et 3

Stage 2

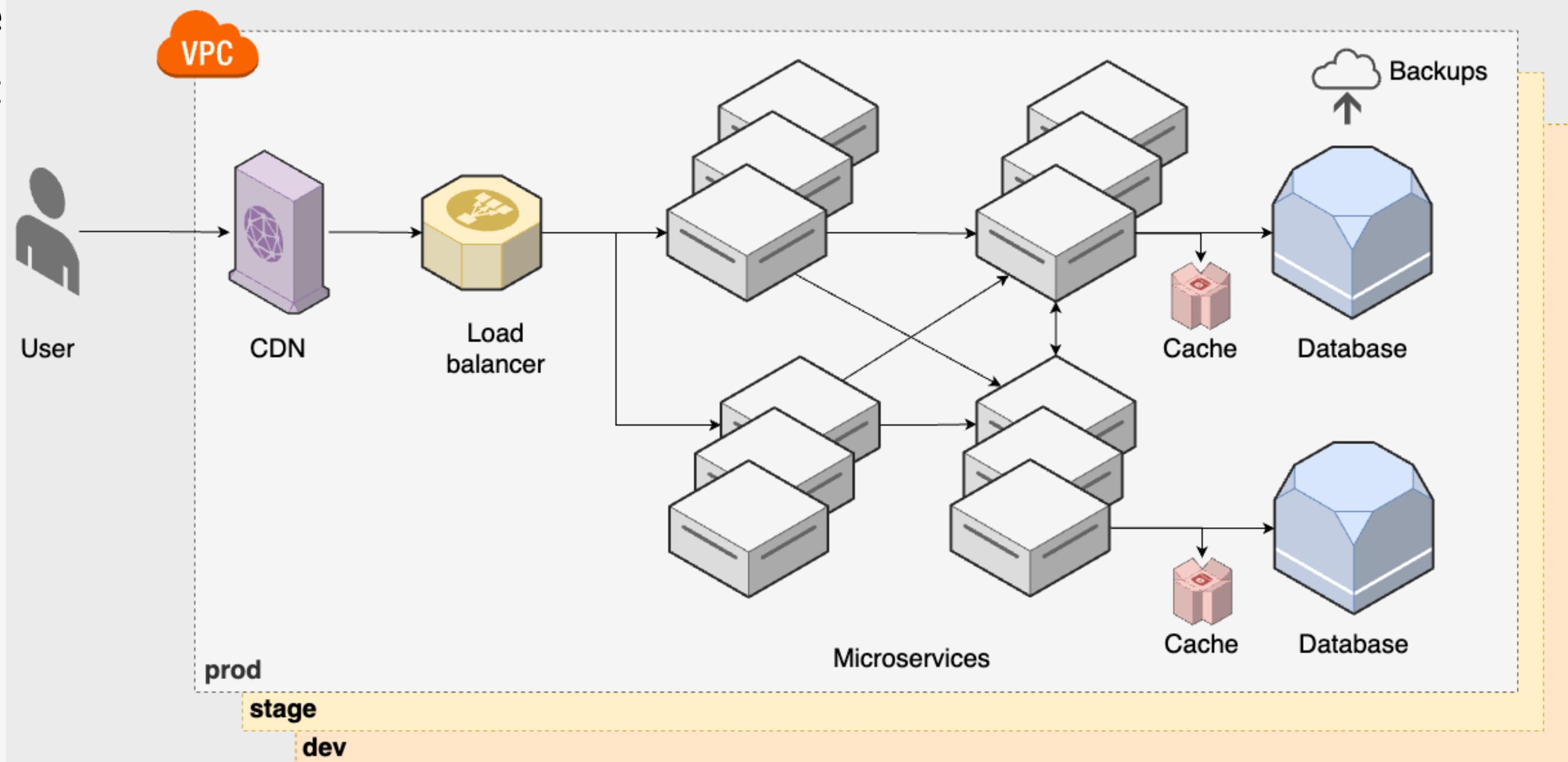


Figure 1.6 Step 6



# Chapter 1 : Évolution de DevOps : stades 2 et 3

Stage 3

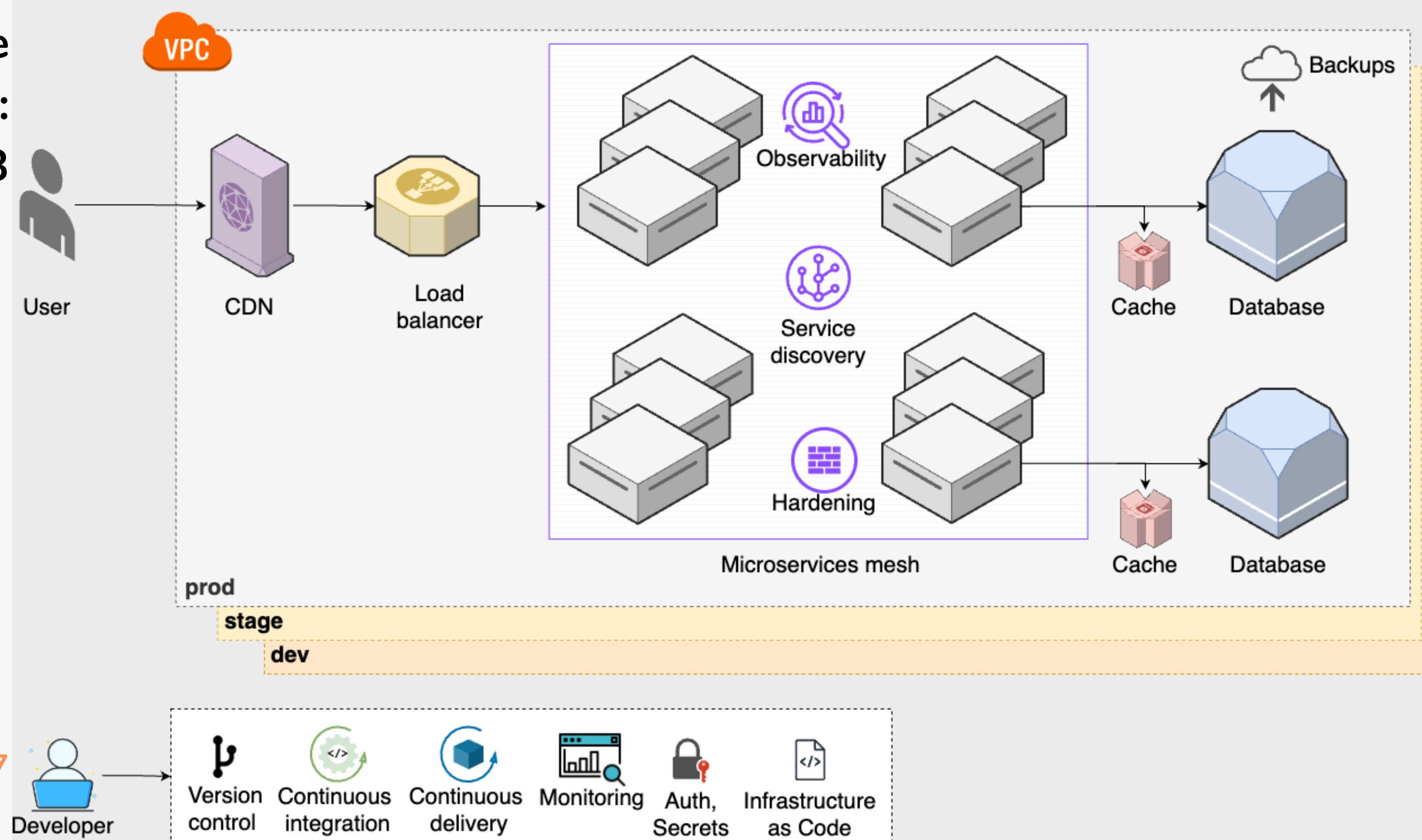
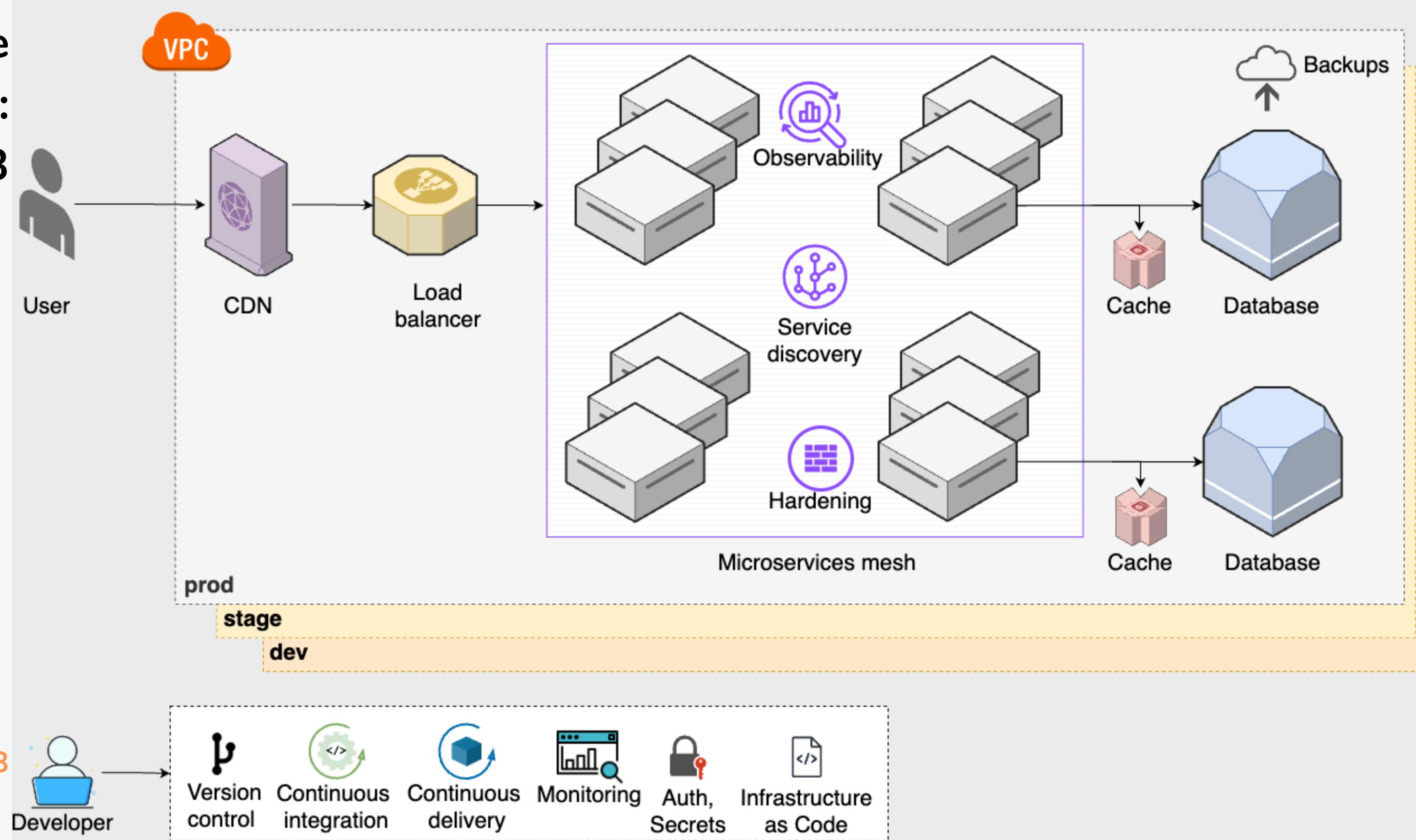


Figure 1.7 Step 7

# Chapter 1 : Évolution de DevOps : stades 2 et 3

Stage 3



# Chapter 1 :

## Évolution de DevOps : stades 2 et 3

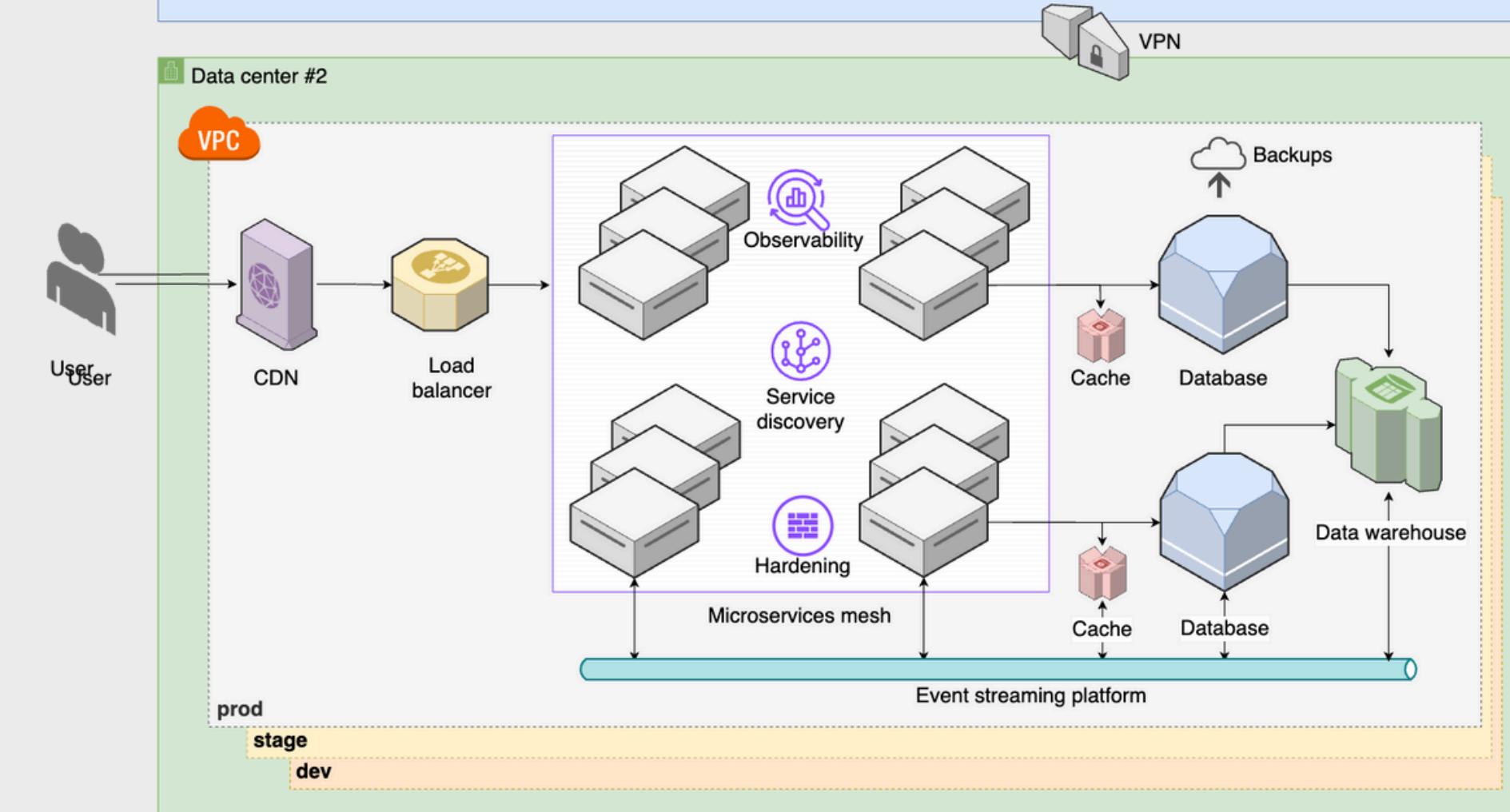
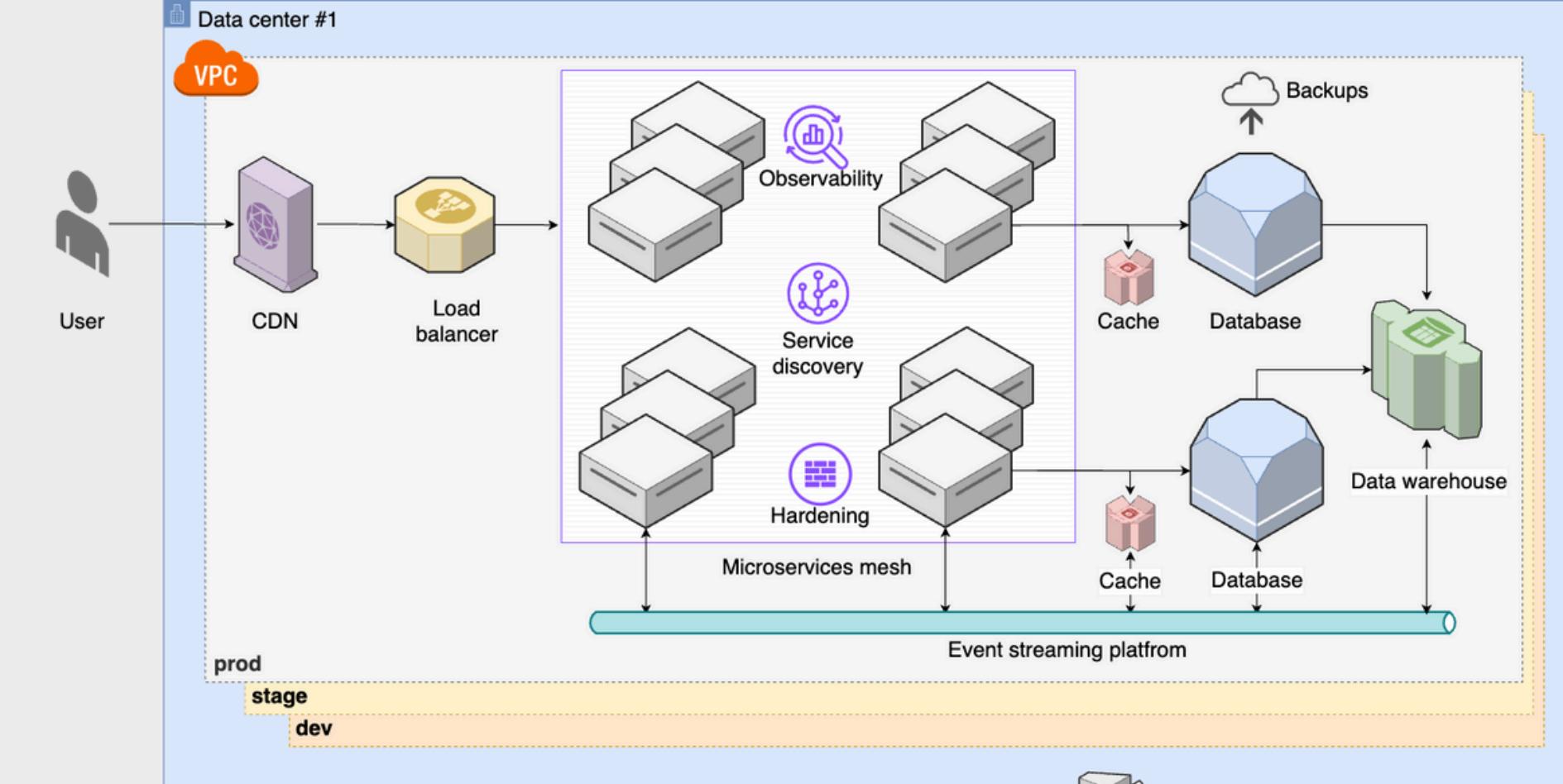


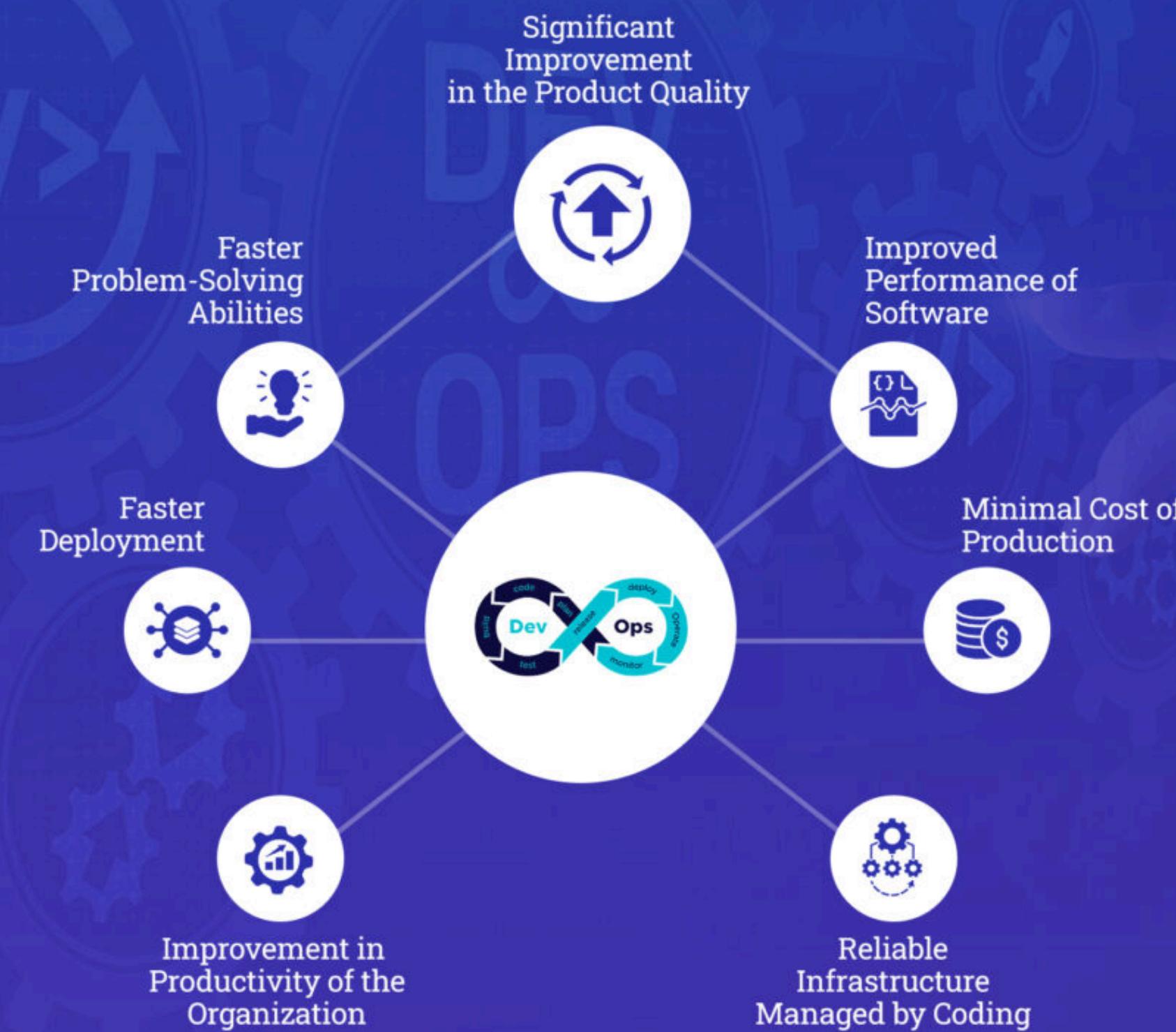
Figure 1-9. Step 9

# Chapter 1 :

## L'évolution de DevOps : Adaptabilité et adoption progressive



### Advantages of DevOps in Code Quality



## **Exemple : Exécution locale d'un exemple d'application**

## Introduction au déploiement d'applications

Déploiement d'applications : Environnements locaux et environnements de serveurs

Le déploiement local et ses limites

Avantages du déploiement de serveurs :

- 1. Sécurité
- 2. Disponibilité
- 3. Performance
- 4. Collaboration

Défis du déploiement sur site

# Chapter 1 : Introduction au déploiement d'applications

## Déploiement sur site ou via le cloud

### Transition vers le déploiement en cloud

Avantages du déploiement en cloud:

1. Pay-as-You-Go
2. Maintenance et expertise
3. Vitesse
4. Elasticité
5. Services gérés
6. Sécurité
7. Accès mondial
8. Scalabilité

# Chapter 1 : Déploiement sur site ou via le cloud

## Introduction au déploiement d'applications

### Déploiement sur site

Quand opter pour un déploiement sur site :

1. Infrastructure existante sur site
2. Modèles d'utilisation spécifiques
3. Maîtrise de la tarification
4. Conformité et exigences réglementaires

Démystifier l'enfermement des fournisseurs dans l'informatique dématérialisée :

Augmentation des coûts initiaux  
Inefficacités opérationnelles

## Déploiement sur site ou via le cloud

### Déploiement hybride

- 1. Migration partielle vers le cloud**
- 2. Outil adéquat pour le travail**

# Chapter 1 : Introduction au déploiement d'applications

## Modèles en cloud : PaaS vs. IaaS

Infrastructure en tant que service (IaaS)

Plateforme en tant que service (PaaS)

Avantages du PaaS:

- Évolutivité
- Gestion des domaines
  - Sécurité
  - Surveillance
- Déploiement automatisé

Limites du PaaS:

- Défis liés au débogage
- Contraintes de flexibilité
- Limites du service

## **Exemple : Déploiement d'une application à l'aide de Render (PaaS)**

# Chapter 1 : Introduction au déploiement d'applications

## Modèles en cloud : PaaS vs. IaaS

Infrastructure en tant que service (IaaS) :

1. Les fournisseurs de serveurs privés virtuels (VPS)
2. Fournisseurs de réseaux de diffusion de contenu (CDN) :
3. Fournisseurs de cloud à usage général :

Solutions spécialisées ou généralisées :

Pourquoi AWS pour l'IaaS dans l'apprentissage de DevOps ?

- Tier gratuit
- Des services complets
- Réputation et leadership sur le marché

## **Exemple : Déploiement d'applications sur AWS :** **Processus étape par étape (IaaS)**

## Exemple : Déploiement d'applications sur AWS : Processus étape par étape (IaaS)

Introduction au déploiement d'AWS

Étape 1 : Crédit d'un compte AWS

Étape 2 : Crédit d'un utilisateur IAM

Étape 3 : Se connecter en tant qu'utilisateur IAM

Étape 4 : Déploiement d'une instance EC2

Étape 5 : Configuration des données de l'utilisateur

Vérification du déploiement

## Exemple : Déploiement d'applications sur AWS : Processus étape par étape (IaaS)

Limites de l'approche simplifiée

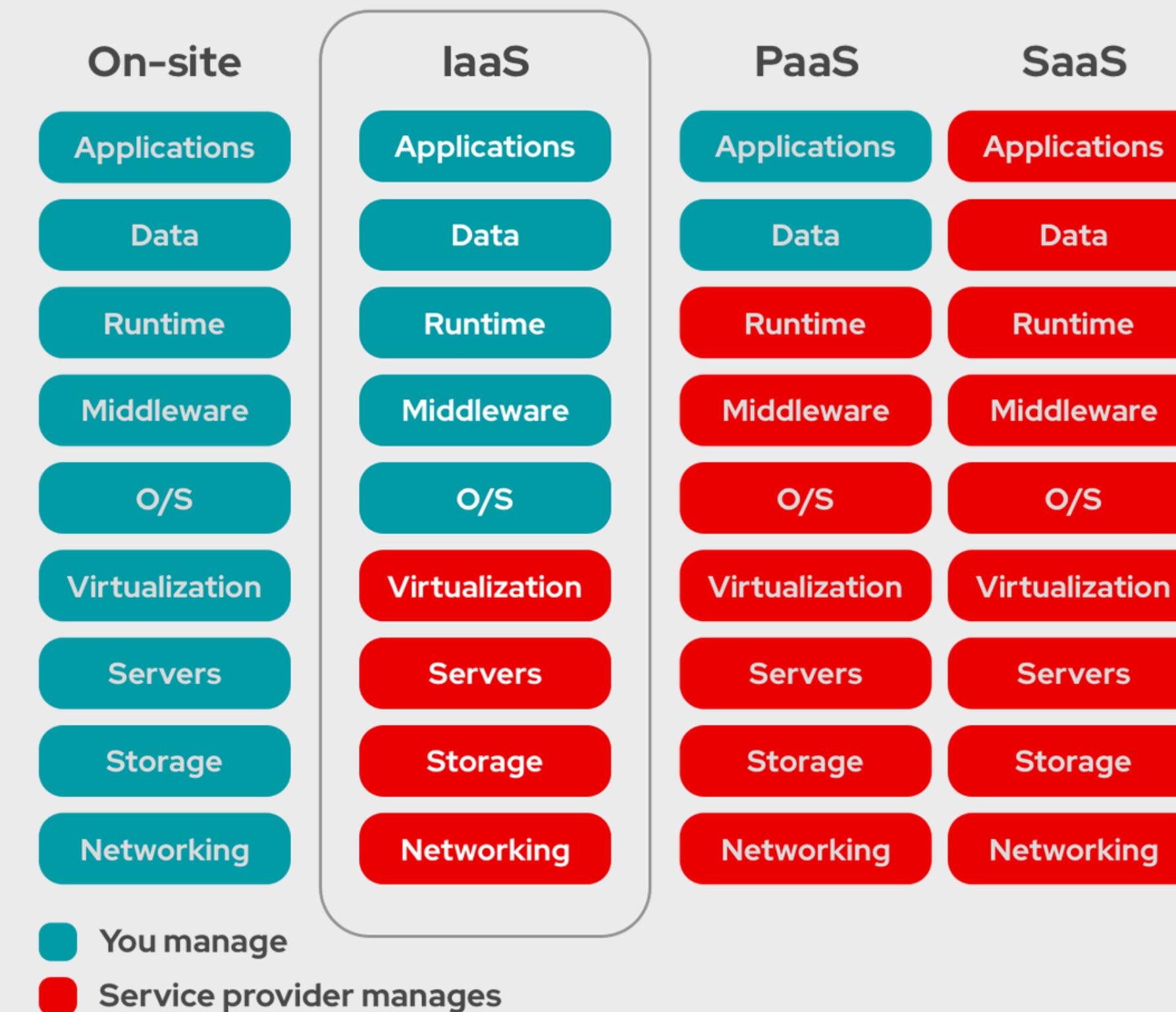
Recommandations pour la production

# Chapter 1 : Introduction au déploiement d'applications

## IaaS vs. PaaS : Choix stratégiques de déploiement

Infrastructure en tant que service (IaaS)

Plateforme en tant que service (PaaS)



# Chapter II :

## Manage Your Infrastructure as Code



# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC) : Améliorer les pratiques DevOps

Table 2-1. A key insight  
of DevOps is that you  
can manage almost  
everything as code

Auteur : Badr TAJINI

Task	How to manage as code	Example
Provision servers	Provisioning tools	Use OpenTofu to deploy a server
Configure servers	Server templating tools	Use Packer to create an image of a server
Configure apps	Configuration files and services	Read configuration from a JSON file
Configure networking	Software-defined networking	Use Kubernetes networking
Build apps	Build systems	Build your app with NPM
Test apps	Automated tests	Write automated tests using Jest
Deploy apps	Automated deployment	Do a rolling deployment with Kubernetes
Scale apps	Auto scaling	Set up auto scaling policies in AWS
Recover from outages	Auto healing	Set up liveness probes in Kubernetes
Manage databases	Schema migrations	Use Knex.js to update your database schema
Test for compliance	Policy as code	Check compliance using Open Policy Agent

# Chapitre 2 :

## Gestion de l'infrastructure en tant que code (IaC) :

### Améliorer les pratiques DevOps

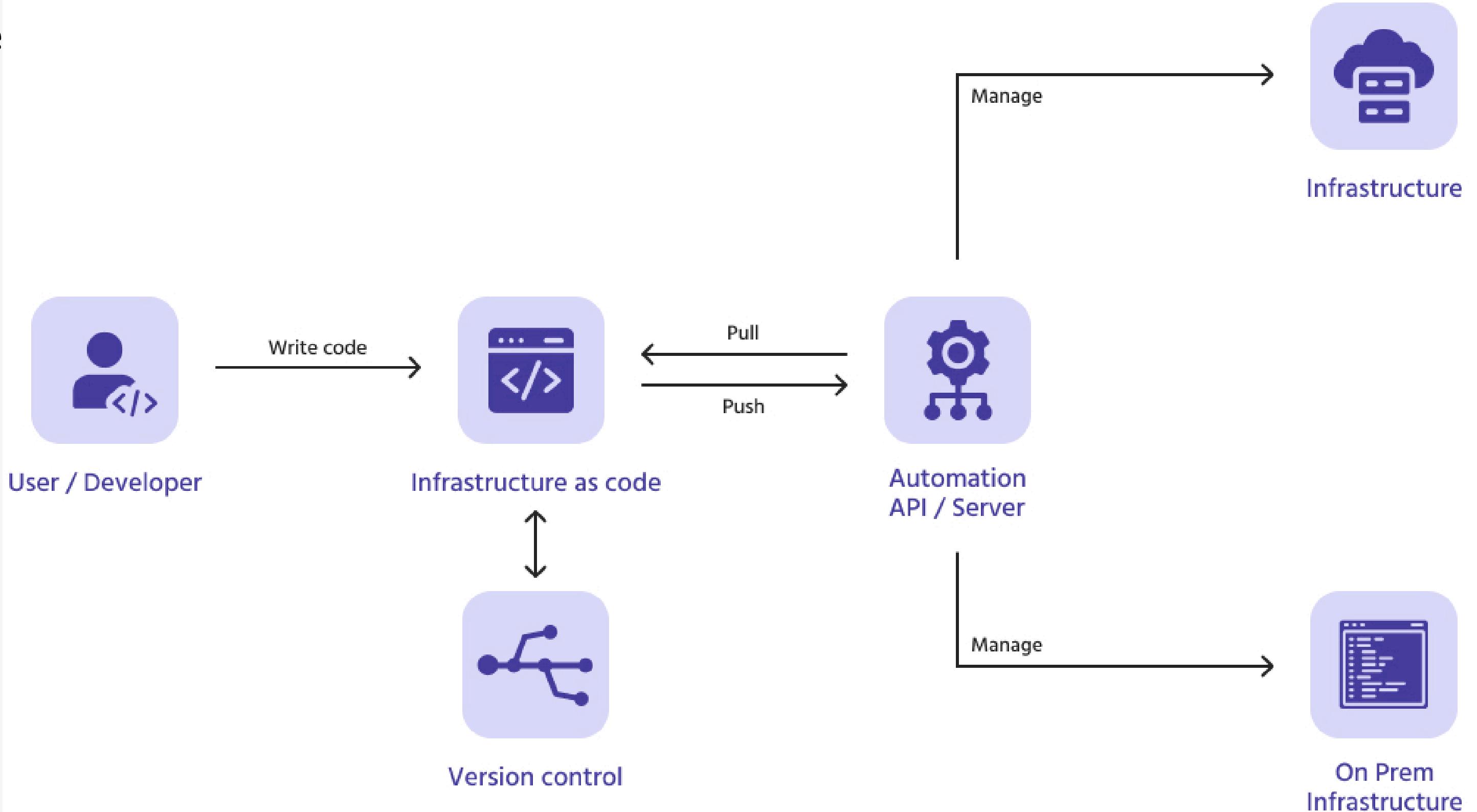
#### Avantages de l'infrastructure en tant que code

- Rapidité et sécurité
- Documentation
- Contrôle de version
- La validation
- Les développeurs peuvent déployer l'infrastructure de manière indépendante
  - Les composants modulaires
  - Le bonheur

# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC)

## Introduction à l'Infrastructure as Code (IaC)

Practice :  
Lab 2



## **Exemple : Déploiement d'une instance EC2 à l'aide d'un script Bash**

### **Limitations des scripts ad hoc**

- Les scripts sont souvent exécutés avec des privilèges élevés
  - Configuration des ports
  - Contraintes liées aux données de l'utilisateur
  - Supervision des processus
  - Problèmes d'évolutivité

### **Avantages de l'infrastructure en tant que code**

- Rapidité et sécurité
- Documentation
- Contrôle de version
- La validation
- Les développeurs peuvent déployer l'infrastructure de manière indépendante
- Les composants modulaires
- Le bonheur

# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC)

## Les scripts ad hoc comme outils d'IaC

Création de scripts  
Exécution

## Évaluation des scripts ad hoc par rapport aux critères de l'IaC

Opérations de CRUD  
Échelle  
Stratégies de déploiement  
Idempotence et traitement des erreurs  
Consistance  
Verbosité

## Limitations des scripts ad hoc

Support CRUD limité  
Problèmes d'évolutivité  
Absence de stratégies de déploiement  
Risque accru d'erreurs  
Les bases de code incohérentes  
Verbosité élevée

## Chapitre 2 : Transition vers les outils de gestion de la configuration

### Gestion de l'infrastructure en tant que code (IaC)

Category	Chef	Puppet	Ansible
Availability (in case of failure)	Backup Server	Alternative Master	Secondary instance
Configuration Language	Ruby DSL	Ruby, Puppet DSL, Embedded Ruby (ERB), DSL	Python, YAML
Architecture	Master-Agent	Master-Agent	Only Master (Agentless)
Installation Process	Time-intensive and complex due to Chef Workstation	Time-intensive due to master-agent certificate signing	Easy

# Chapitre 2 : Transition vers les outils de gestion de la configuration

## Gestion de l'infrastructure en tant que code (IaC)

Practice :  
Lab 2

**Exemple : Déploiement d'une instance EC2 à l'aide d'Ansible**

**Exemple : Configurer des serveurs avec Ansible**

## **Exemple : Configurer des serveurs avec Ansible**

# Chapitre 2 :

## Gestion de l'infrastructure en tant que code (IaC)

Évaluer les outils de gestion de la configuration en fonction des critères de l'IaC

Opérations de CRUD

Suppression

Échelle

Stratégies de déploiement

Idempotence et traitement des erreurs

Consistance

Verbosité

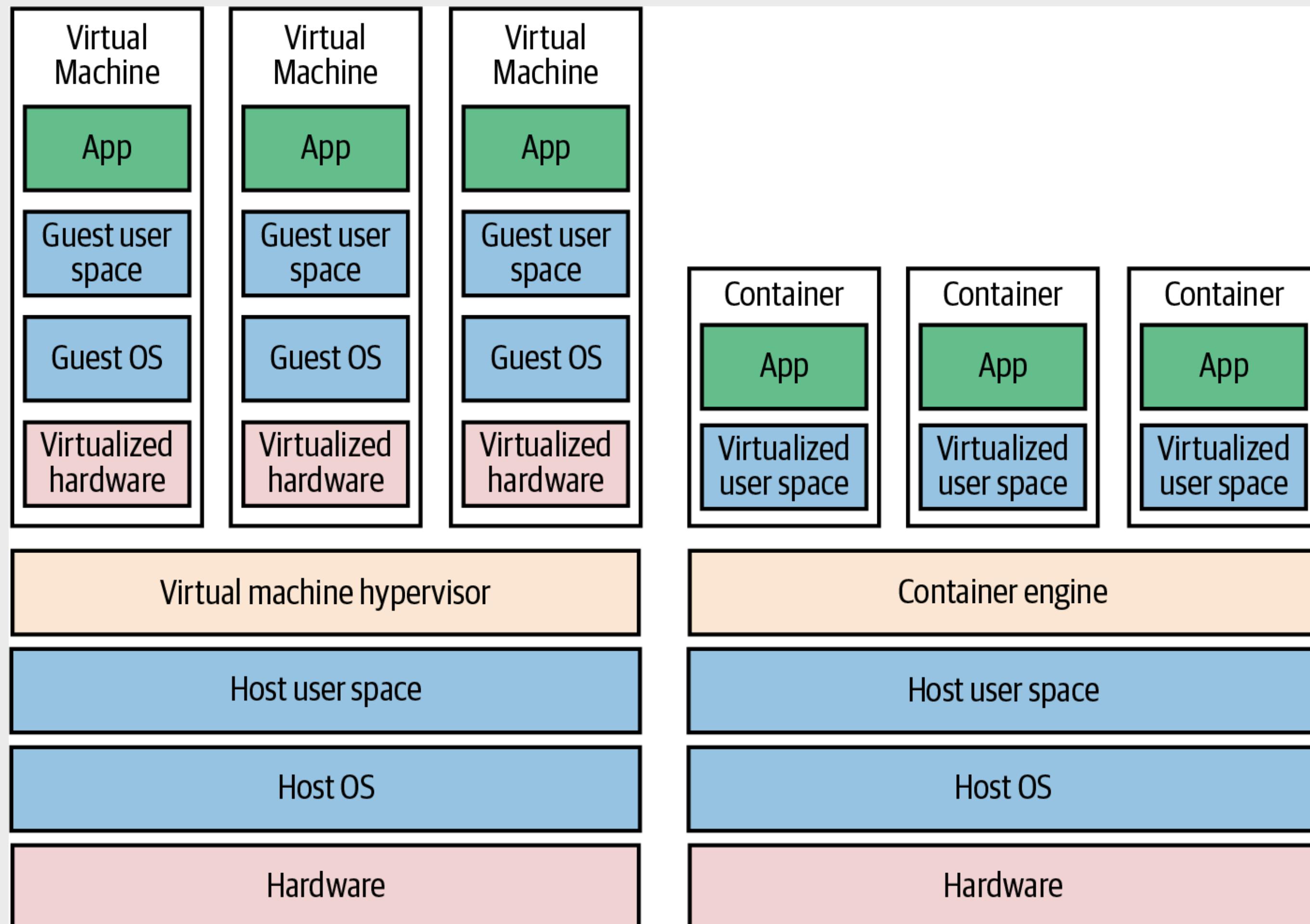
Avantages et inconvénients des outils de gestion de la configuration

Coûts d'installation

Paradigme d'infrastructure mutable

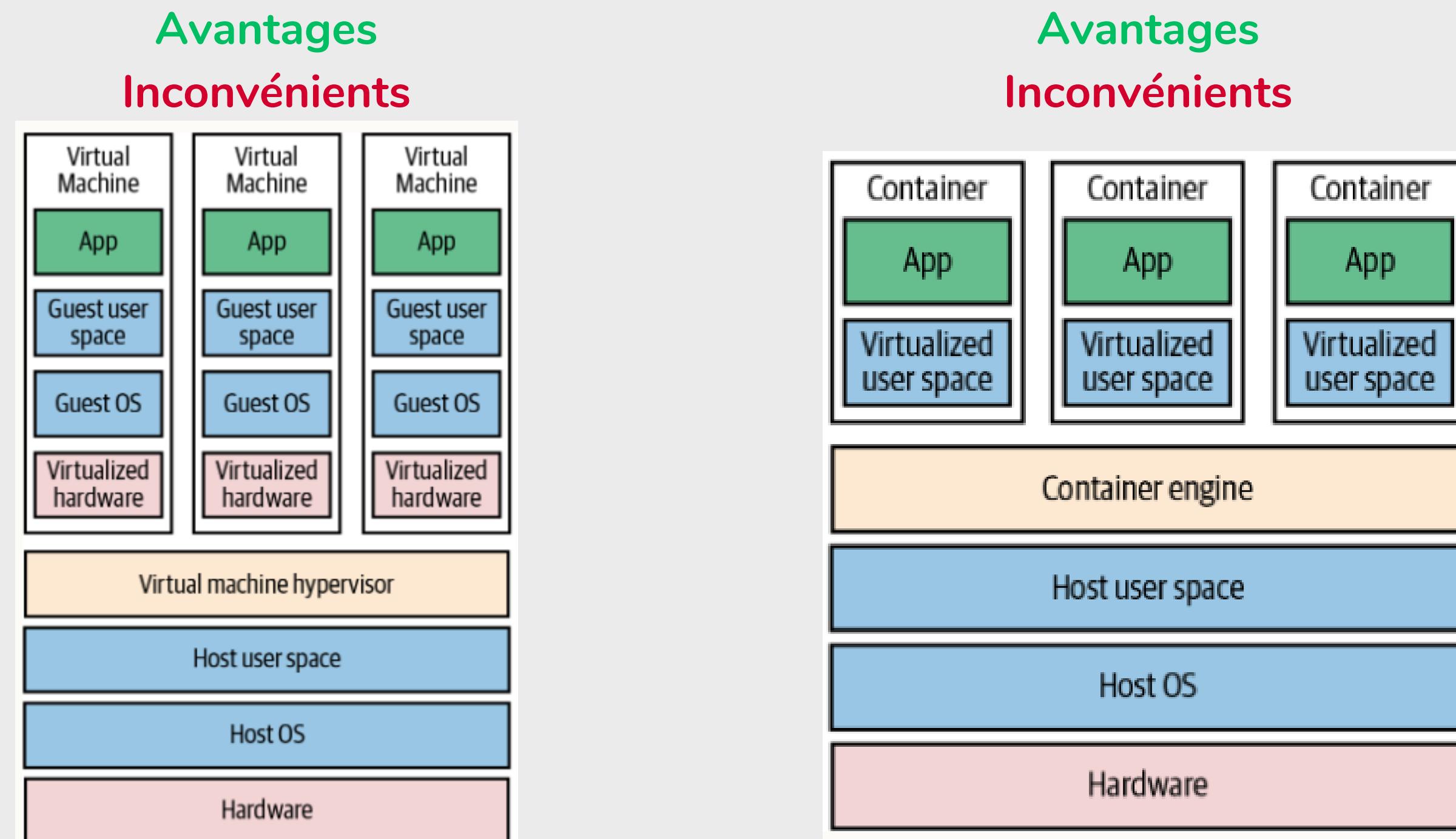
# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC)

## Transition vers un paradigme d'infrastructure immutable Introduction aux outils de modélisation de serveur



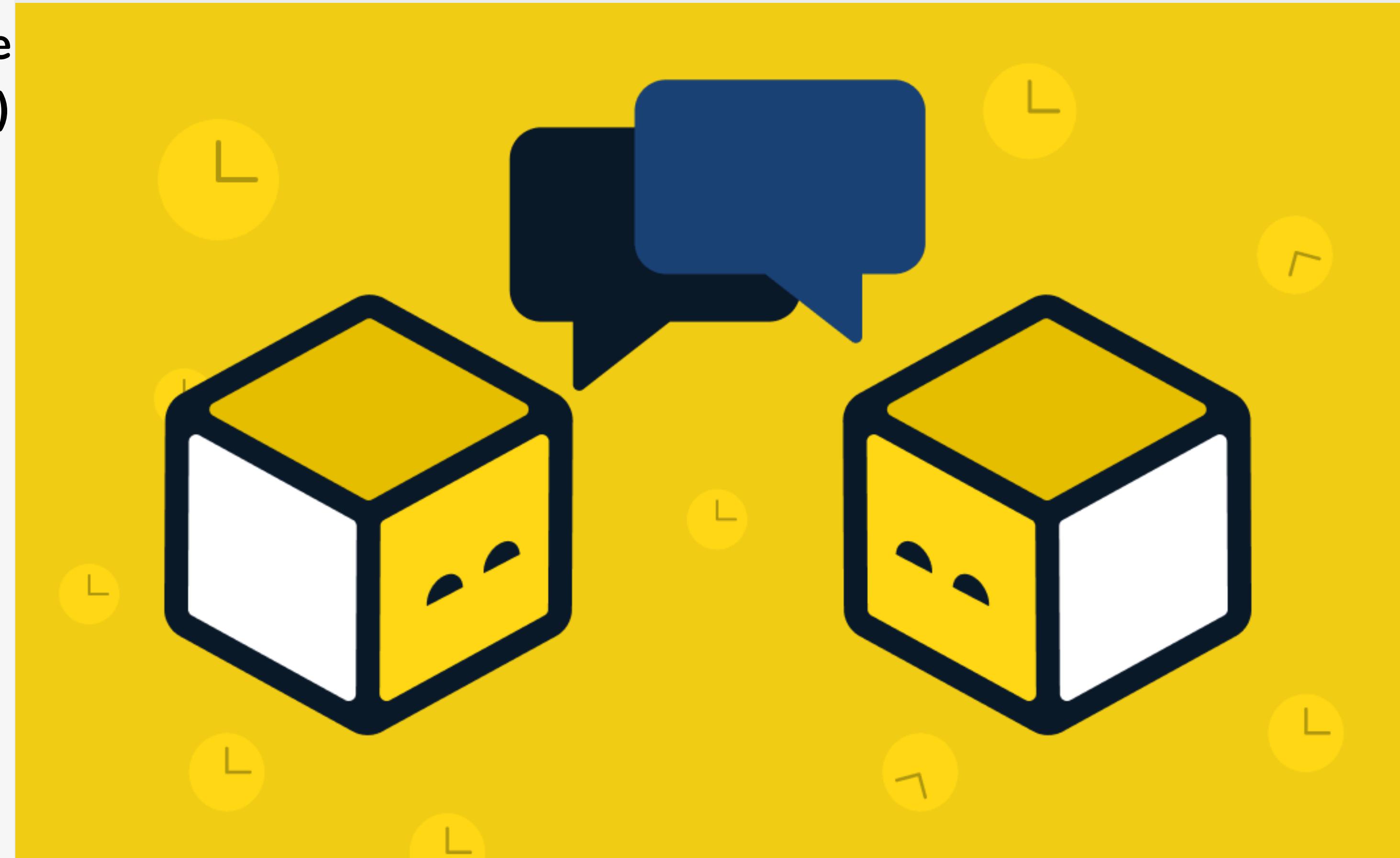
## Catégories d'outils de création de modèles de serveur

- Machines virtuelles (VM)
- Isolation et cohérence
- Outils
- Conteneurs
- Isolation et efficacité
- Outils



## Exemple : Création d'une image de VM à l'aide de Packer

## Définition de la syntaxe et des ressources d'OpenTofu



## Exemple : Déployer une instance EC2 avec OpenTofu

### Avantages de l'utilisation d'OpenTofu pour le provisionnement

- L'indépendance
- Évolutivité
- Cohérence
- Simplicité et concision
- Gestion des états

## **Exemple : Déployer une instance EC2 avec OpenTofu (SUITE)**

### **Avantages de l'utilisation d'OpenTofu pour le provisionnement**

- **L'indépendance**
- **Évolutivité**
- **Cohérence**
- **Simplicité et concision**
- **Gestion des états**

**Exemple : Déployer une instance EC2 à l'aide d'un module  
OpenTofu**

## **Exemple : Déployer une instance EC2 à l'aide d'un module OpenTofu**

**Avantages des modules réutilisables avec OpenTofu**

- **Modules éliminent la duplication du code**
- **Cohérence**
- **Efficacité**
- **Structures organisées en modules**

**Exemple : Déployer une instance EC2 en utilisant un module  
OpenTofu de GitHub**

## Exemple : Déployer une instance EC2 en utilisant un module OpenTofu de GitHub

### Avantages et bonnes pratiques

- Efficacité et cohérence
- Évolutivité et maintenabilité
- L'intégration avec d'autres outils IaC

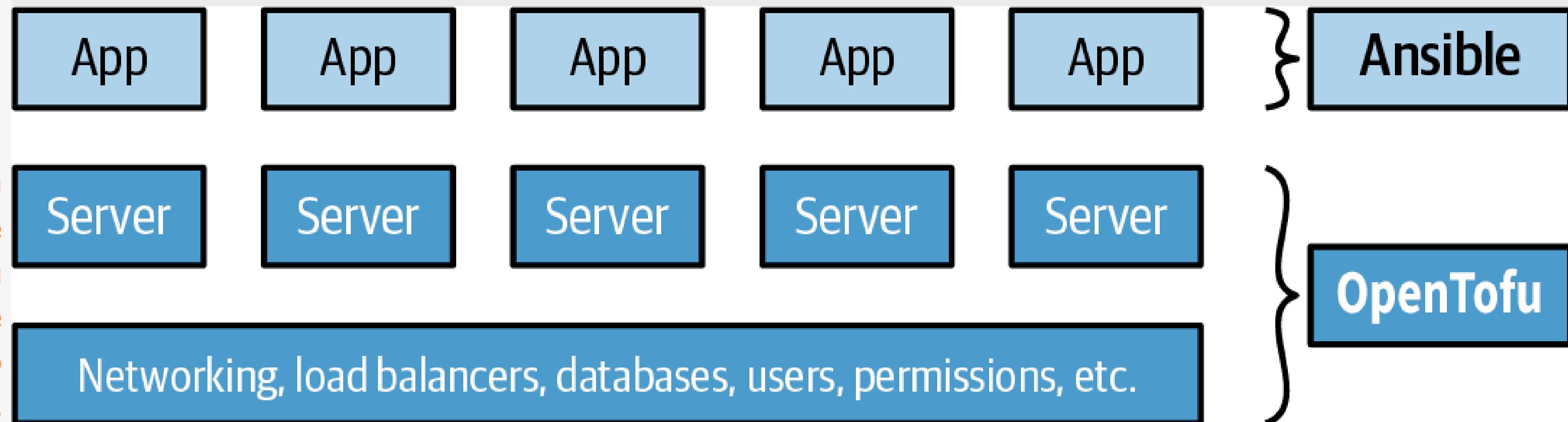
## **Principaux enseignements**

**Utilisation conjointe de plusieurs outils d'IaC**

# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC)

Intégrer plusieurs outils et adopter des pratiques IaC

Provisionnement et gestion de la configuration (OpenTofu et Ansible)

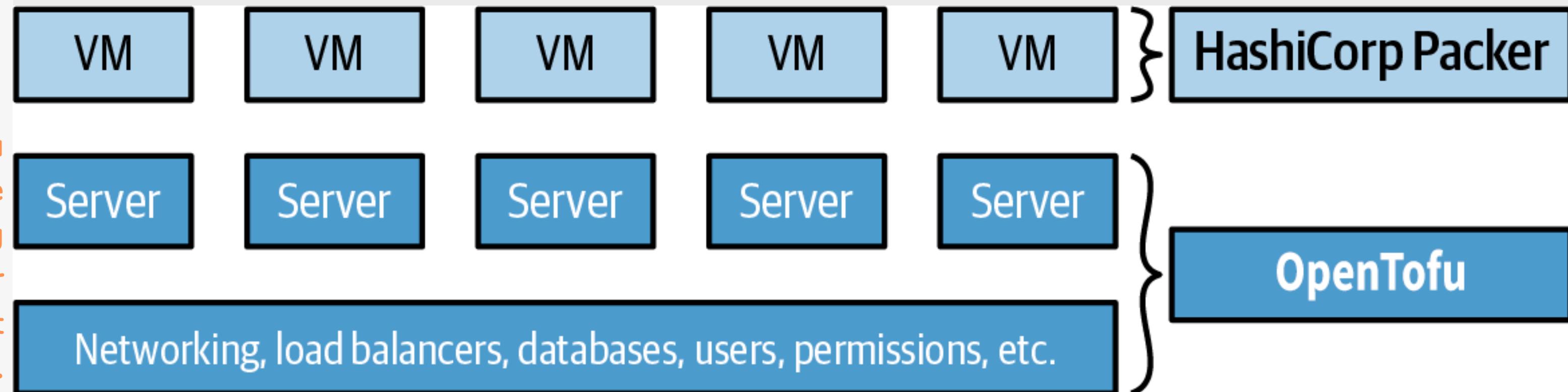


# Chapitre 2 :

## Gestion de l'infrastructure en tant que code (IaC)

### Intégrer plusieurs outils et adopter des pratiques IaC

#### Provisionnement et modélisation des serveurs (OpenTofu et Packer)

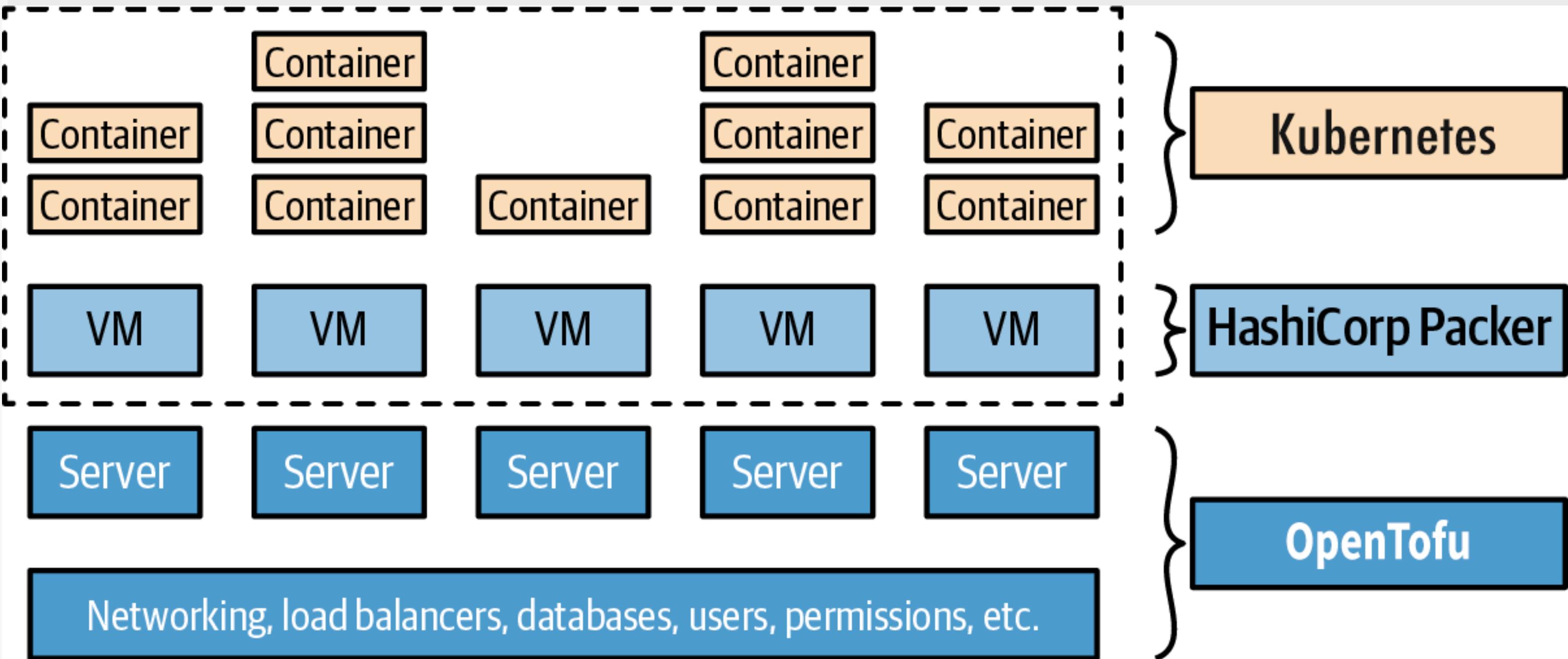


# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC)

## Intégrer plusieurs outils et adopter des pratiques IaC

### Provisionnement et modélisation des serveurs en plus de l'orchestration (OpenTofu, Packer, Docker et Kubernetes)

Figure 2-5. OpenTofu deploys the infrastructure, including servers; Packer creates the VMs that run on those servers; and Kubernetes manages those VMs as a cluster for running Docker containers..



# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC)

## Adopter l'infrastructure en tant que code (IaC)

1. Transformation de la culture et des processus
2. Conseils stratégiques pour l'adoption

# Chapitre 2 : Gestion de l'infrastructure en tant que code (IaC)

Infrastructure as Code (IaC) : Concepts clés et enseignements à tirer

1. Les scripts ad hoc
2. Les outils de gestion de la configuration (par exemple, Ansible)
3. Les outils de modélisation des serveurs (par exemple, Packer)
4. Les outils de provisionnement (par exemple, OpenTofu/Terraform)
5. La combinaison d'outils IaC
6. L'adoption de l'IaC

Le choix de l'outil d'IaC approprié dépend de la tâche spécifique :

- Provisionnement
- Configuration du serveur
- Scénarios hybrides

Différencier les outils de l'IaC au sein des catégories

Approfondissement des connaissances sur OpenTofu et Terraform

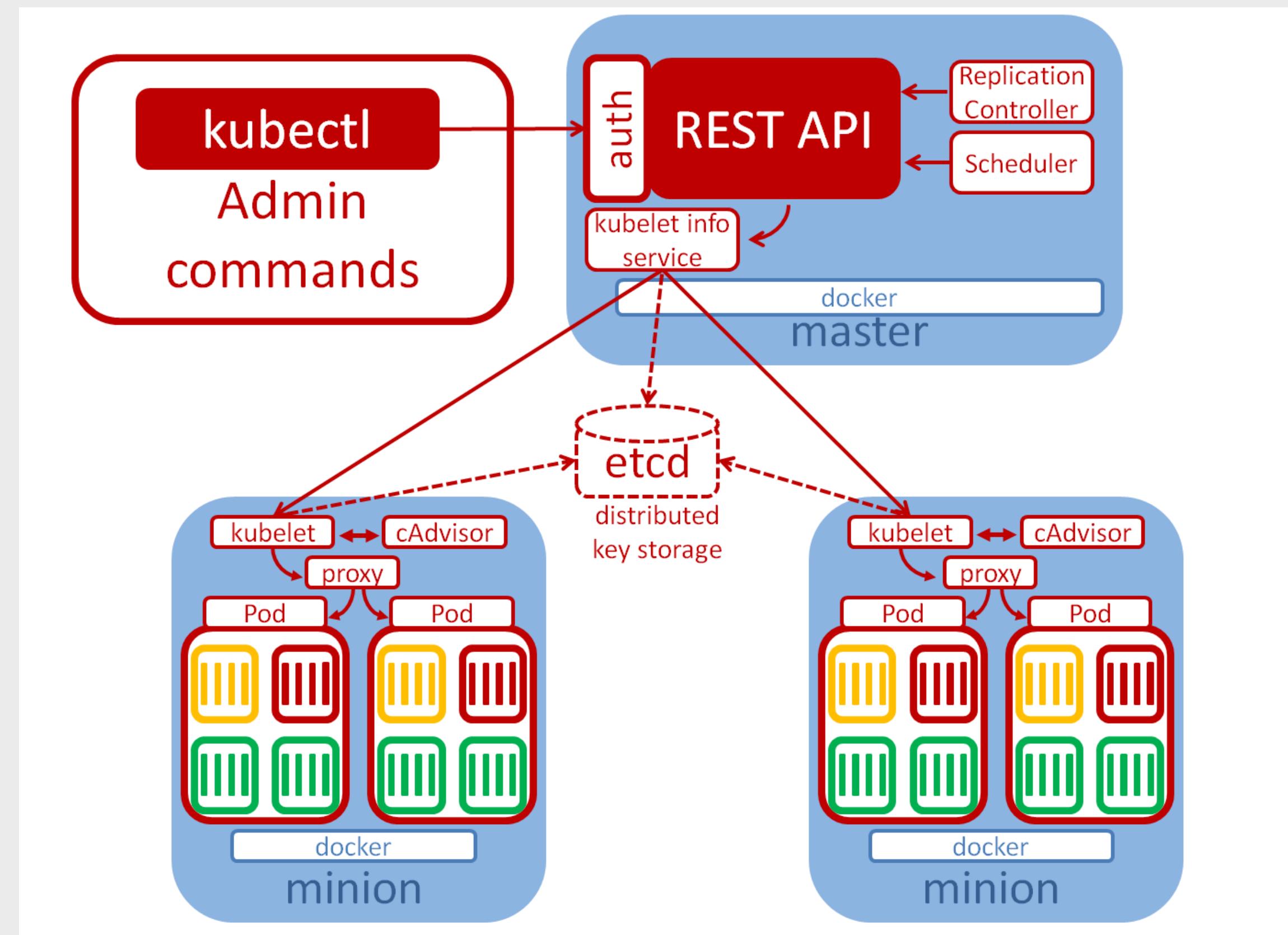


# Chapter III :

# How to deploy your Apps

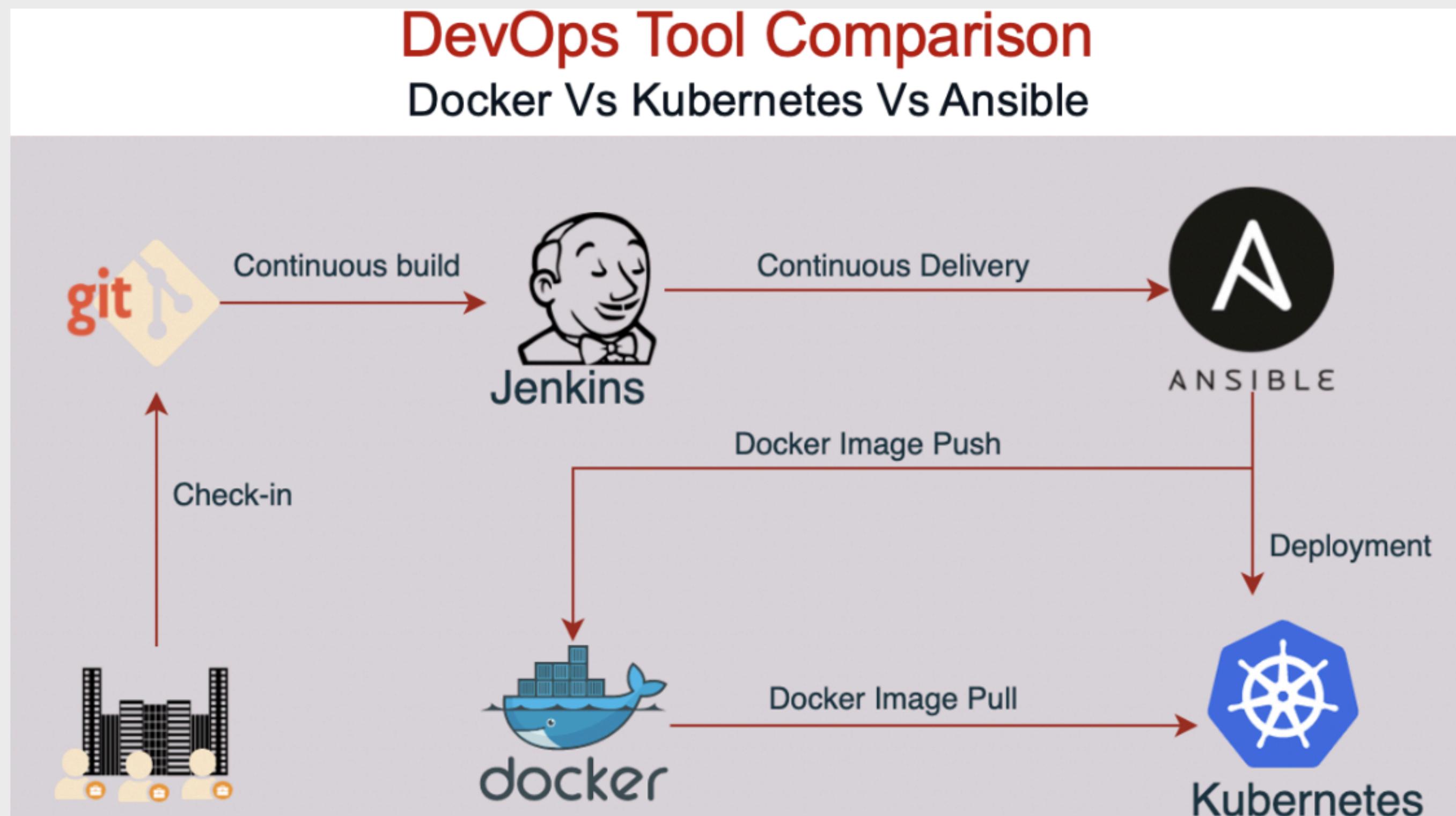
# Chapitre 3 : Comment déployer vos applications

- Les défis des applications à serveur unique
- Introduction à l'orchestration
- Catégories d'orchestration



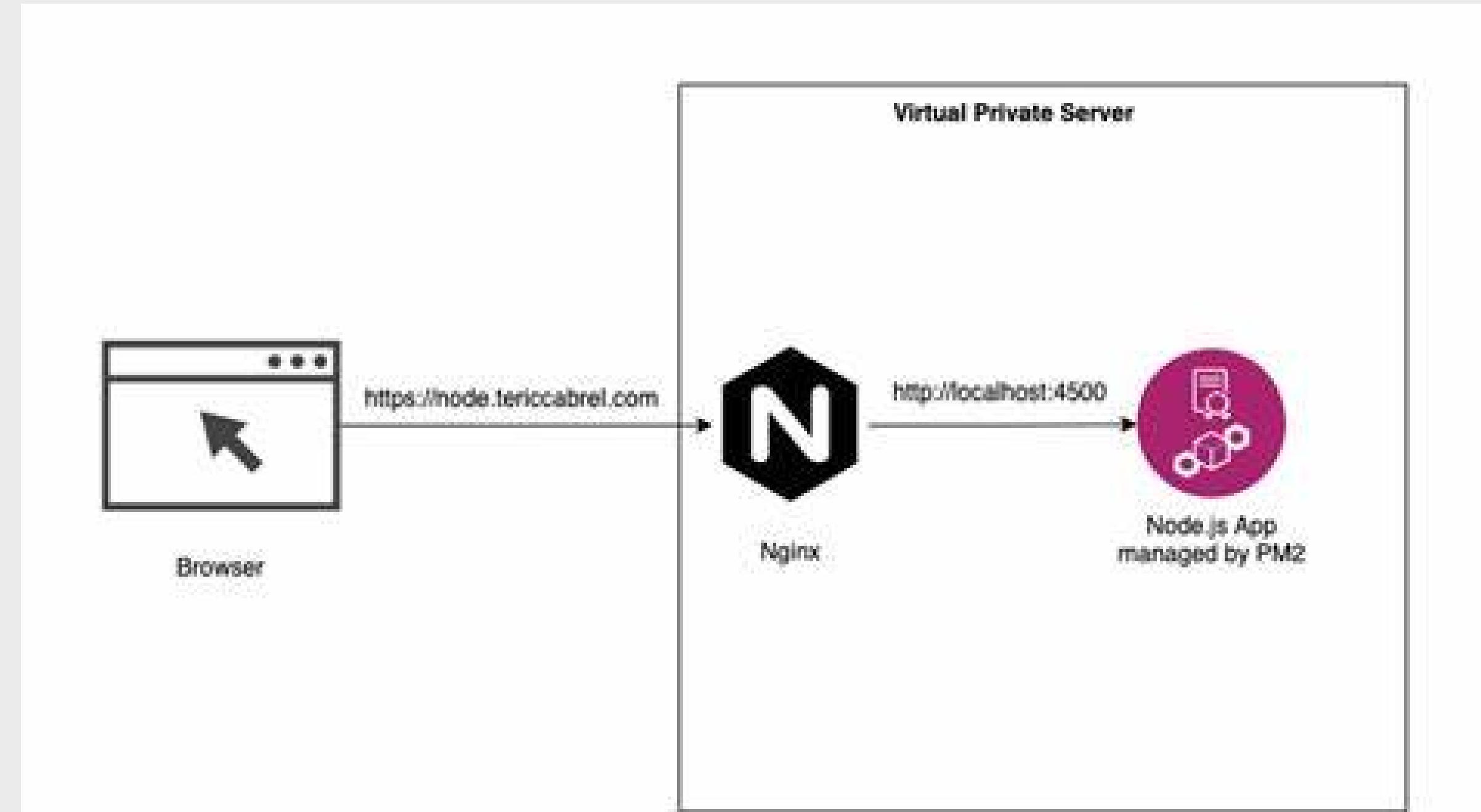
# Chapitre 3 : Comment déployer vos applications

- Introduction à l'orchestration de serveurs
- Déployer plusieurs serveurs avec Ansible
- Améliorer l'orchestration avec l'inventaire et les variables



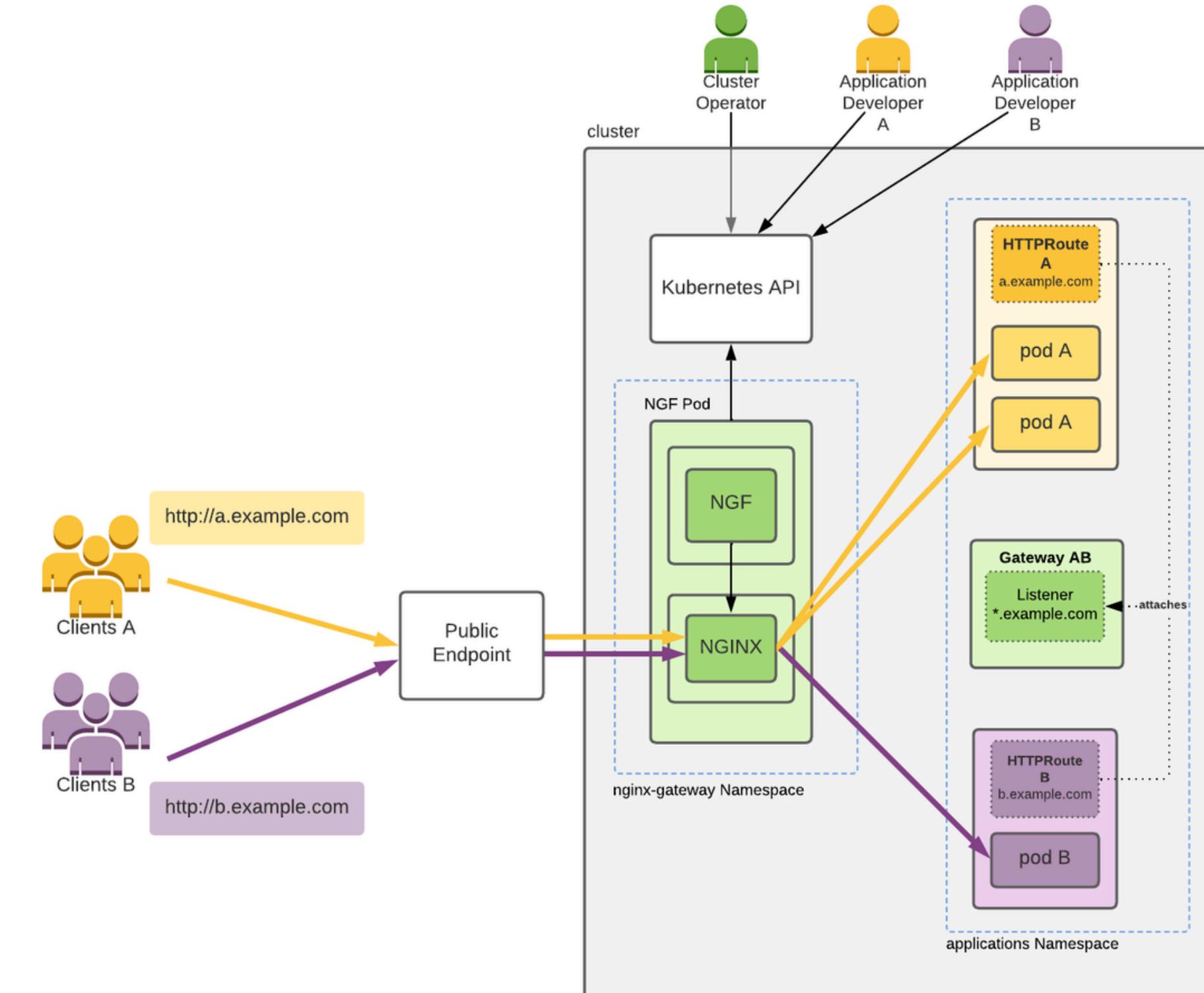
# Chapitre 3 : • Comment déployer vos applications

## Configuration des serveurs pour l'exécution d'une application Node.js PM2 et Auto-Healing pour les applications Node.js Conception de rôles modulaires et déploiement d'applications



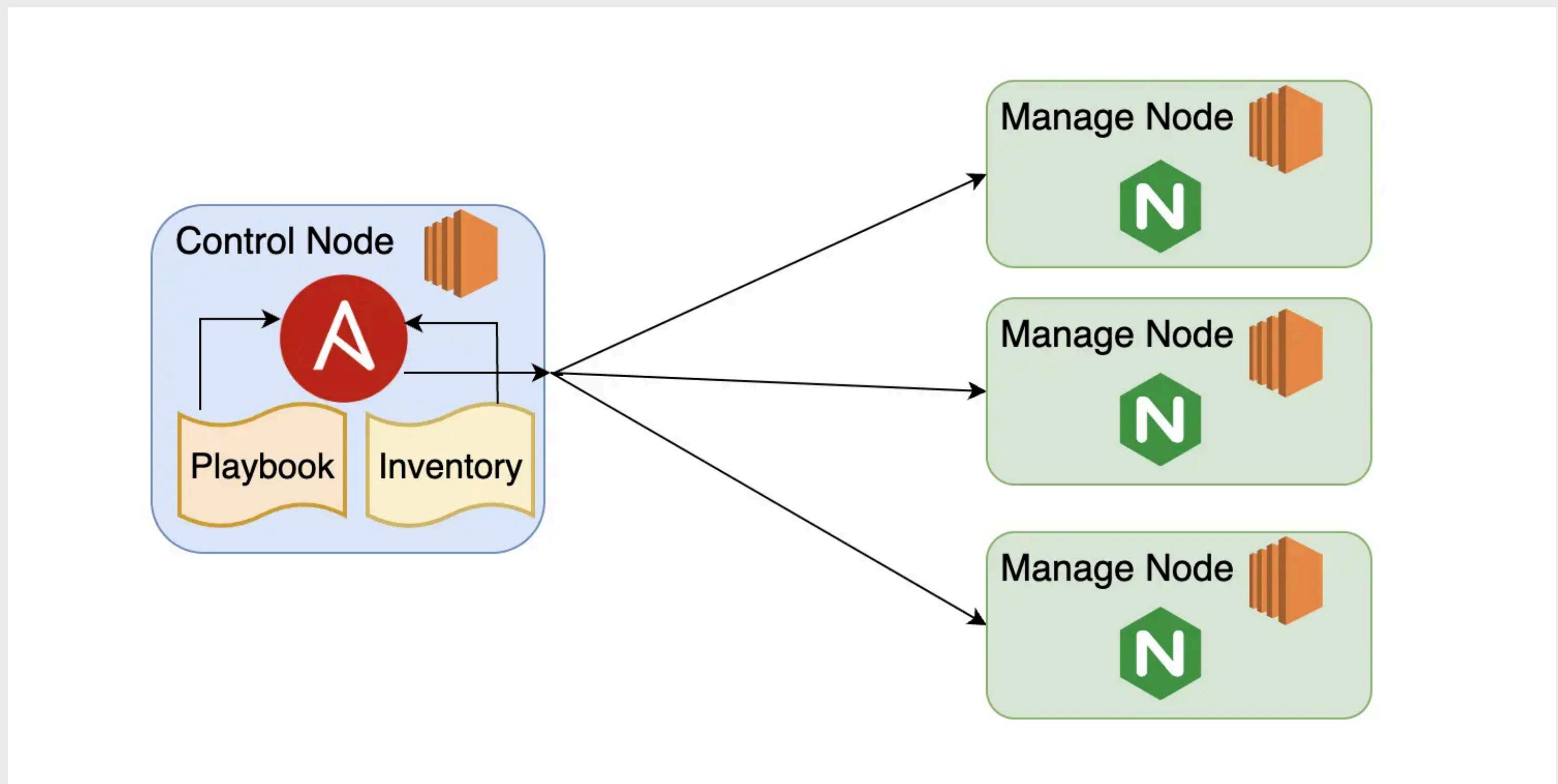
# Chapitre 3 : Comment déployer vos applications

- Finaliser le déploiement de l'application Node.js
- Présentation des équilibriseurs de charge
- Configurer Nginx avec Ansible



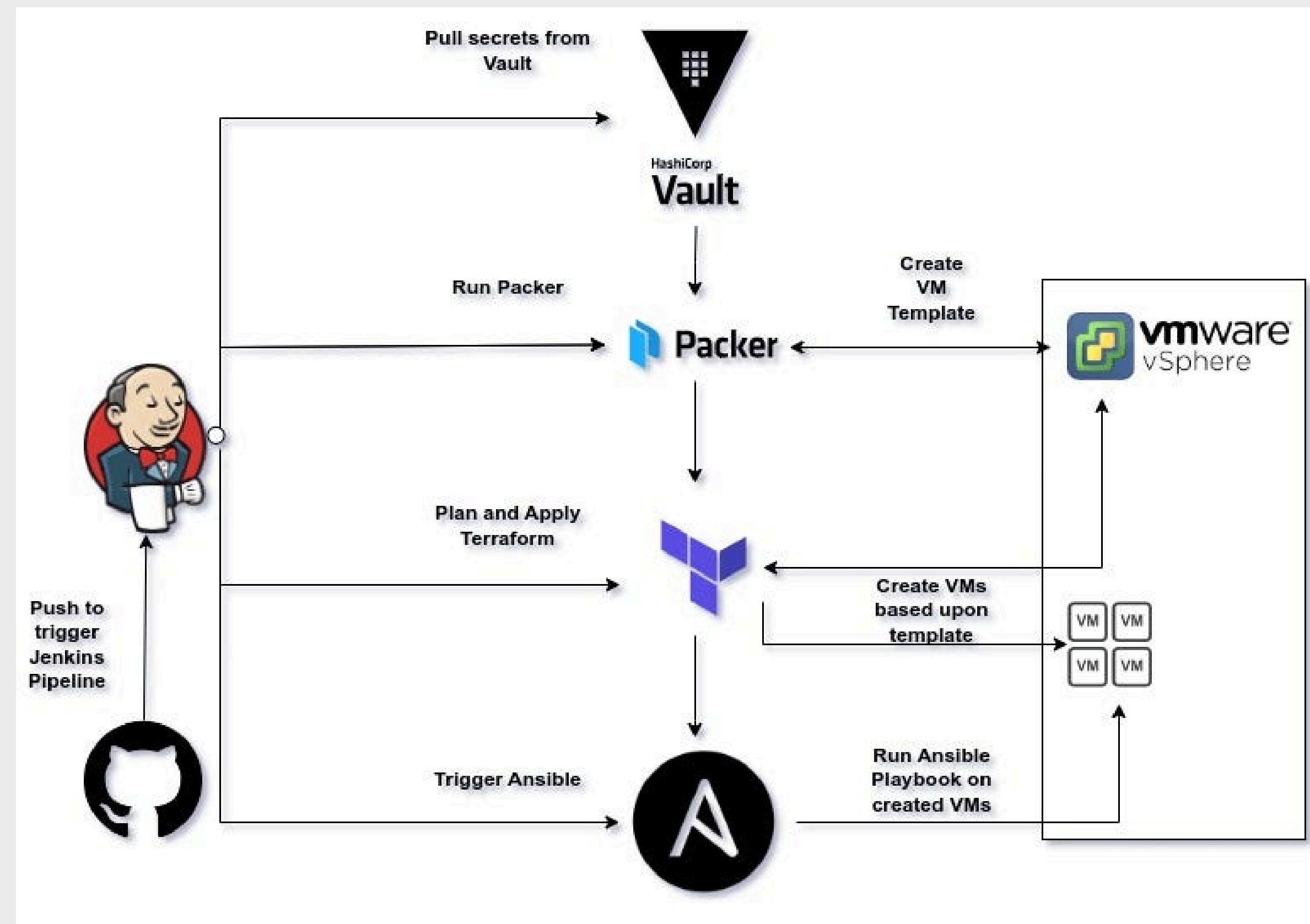
# Chapitre 3 : Comment déployer vos applications

- Configurer Nginx comme équilibrEUR de charge
- Les mises à jour en continu avec Ansible
- Résumé et concepts clés



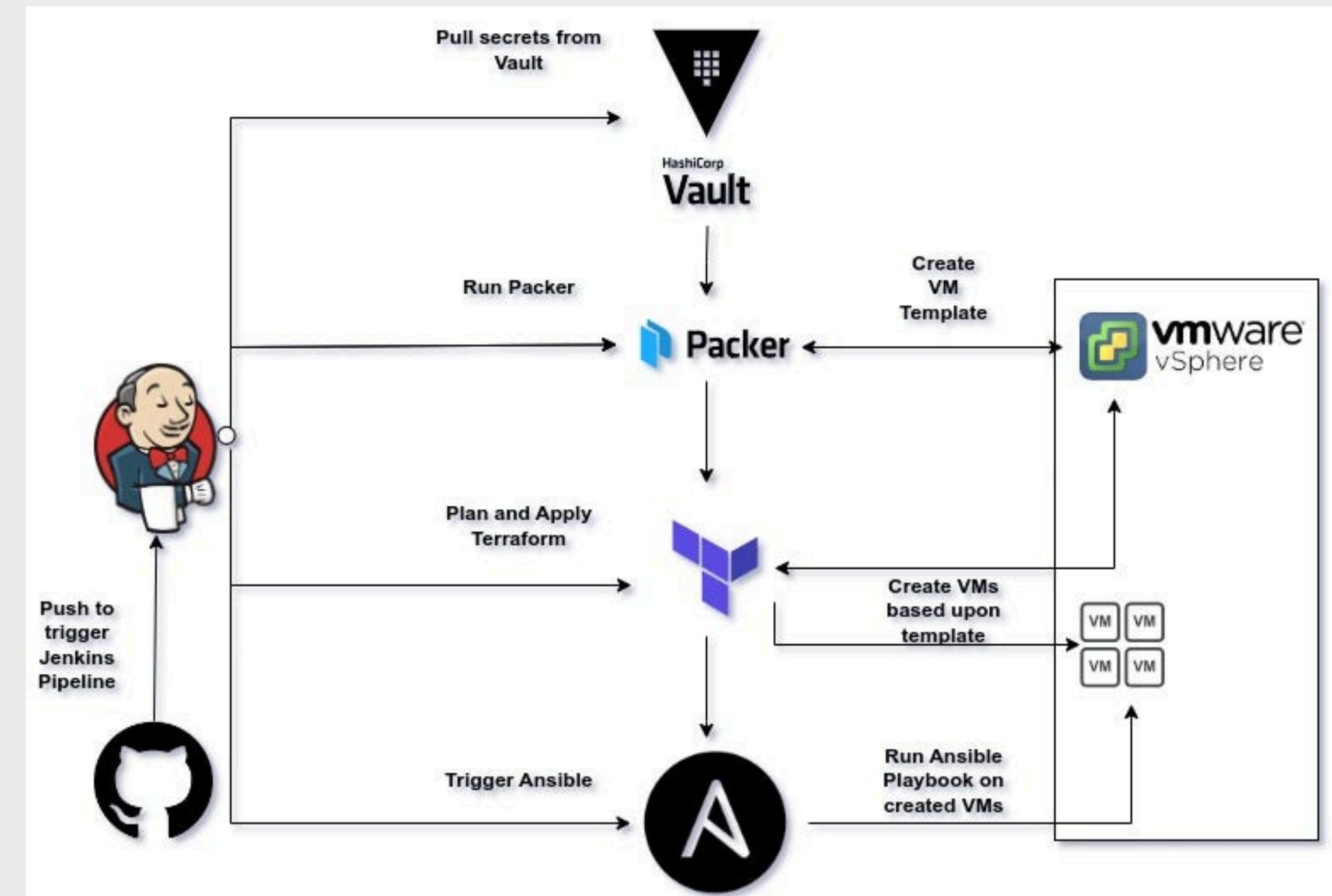
# Chapitre 3 : Comment déployer vos applications

- Les mises à jour en continu avec Ansible
- Introduction à l'orchestration de VM
- Construire des images de VM avec Packer



# Chapitre 3 : Comment déployer vos applications

- Construire et déployer une image de VM en utilisant Packer
- Déploiement d'images VM avec des groupes de mise à l'échelle automatique (ASG)
- Configuration d'un déploiement ASG



# Chapitre 3 :

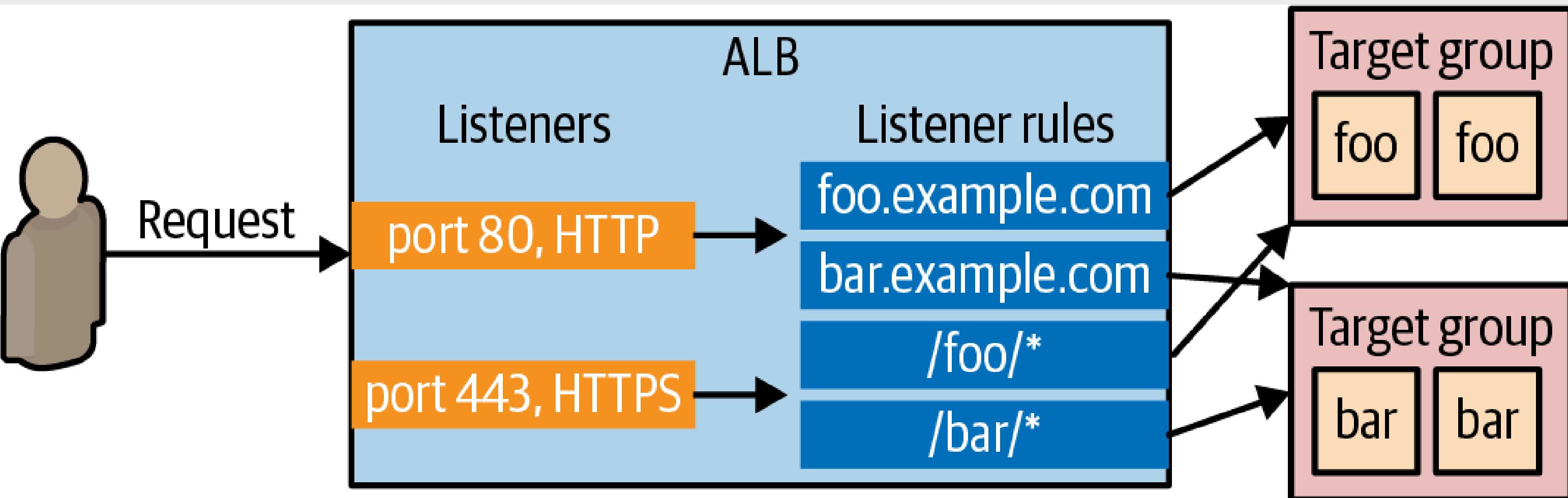
- Comment déployer vos applications

## Déploiement d'applications avec PM2

### Gestion des équilibriseurs de charge

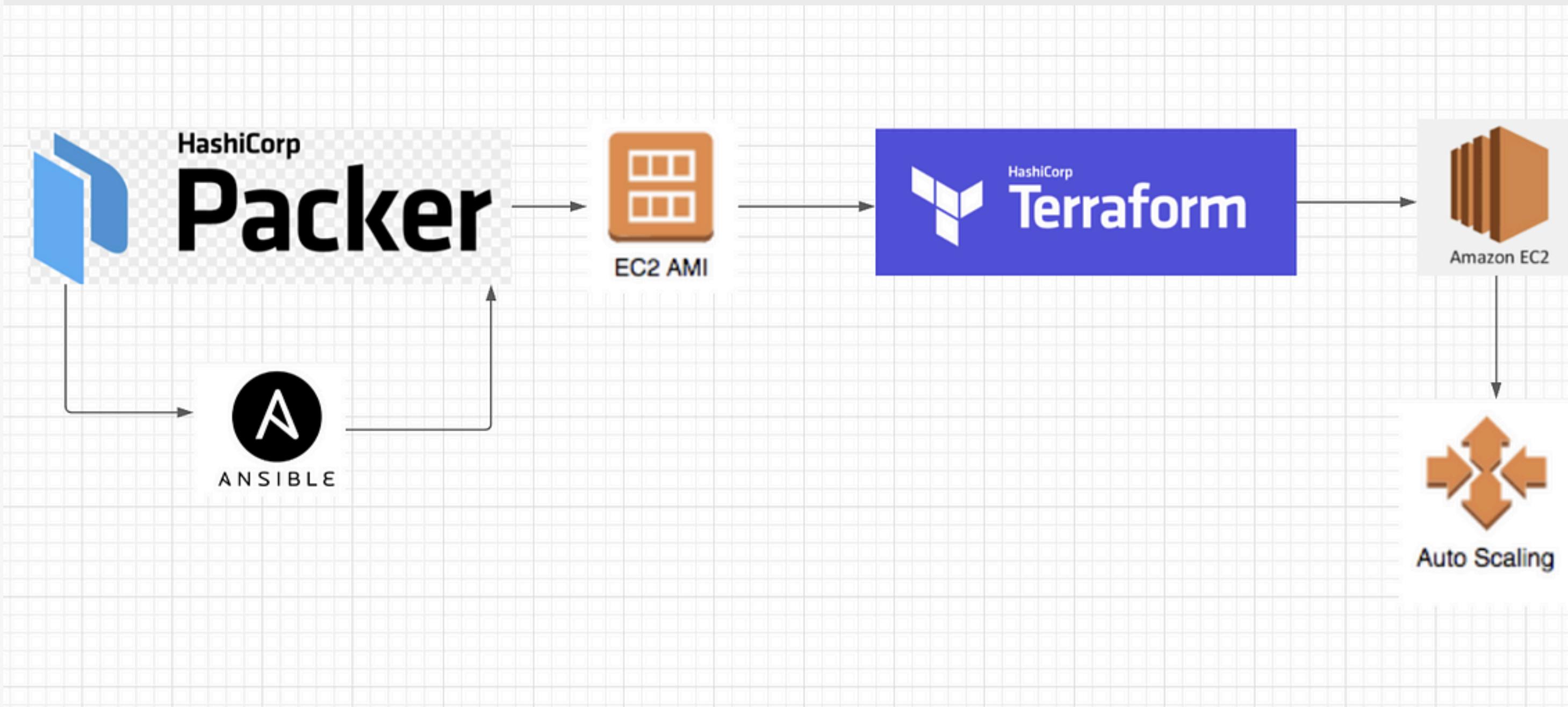
- Configurer AWS Application Load Balancer avec OpenTofu

Figure 3-1. An ALB consists of listeners, listener rules, and target groups.



# Chapitre 3 : Comment déployer vos applications

- Intégration des groupes de mise à l'échelle automatique avec les équilibriseurs de charge
- Mises à jour en continu avec l'actualisation de l'instance
- Reconstruction de “AMI” et déploiement final



# Chapitre 3 : Comment déployer vos applications

- Mise à jour des groupes de mise à l'échelle automatique (ASG) avec de nouvelles versions d'AMI
- Surveillance du déploiement
- Validation du déploiement sans temps d'arrêt

EC2 > Auto Scaling groups

### Auto Scaling groups (1/1) Info

C Launch configurations Launch templates Actions Create Auto Scaling group

Search your Auto Scaling groups

Name	Launch template/configuration	Instances	Status
<a href="#">sample-app-tofu20240414163932852100000003</a>	<a href="#">sample-app-tofu20240414163932852100000003</a>	6	

Auto Scaling group: sample-app-tofu20240414163932852100000003

Details Activity Automatic scaling Instance management Monitoring Instance refresh

#### Active instance refresh Info

C Actions

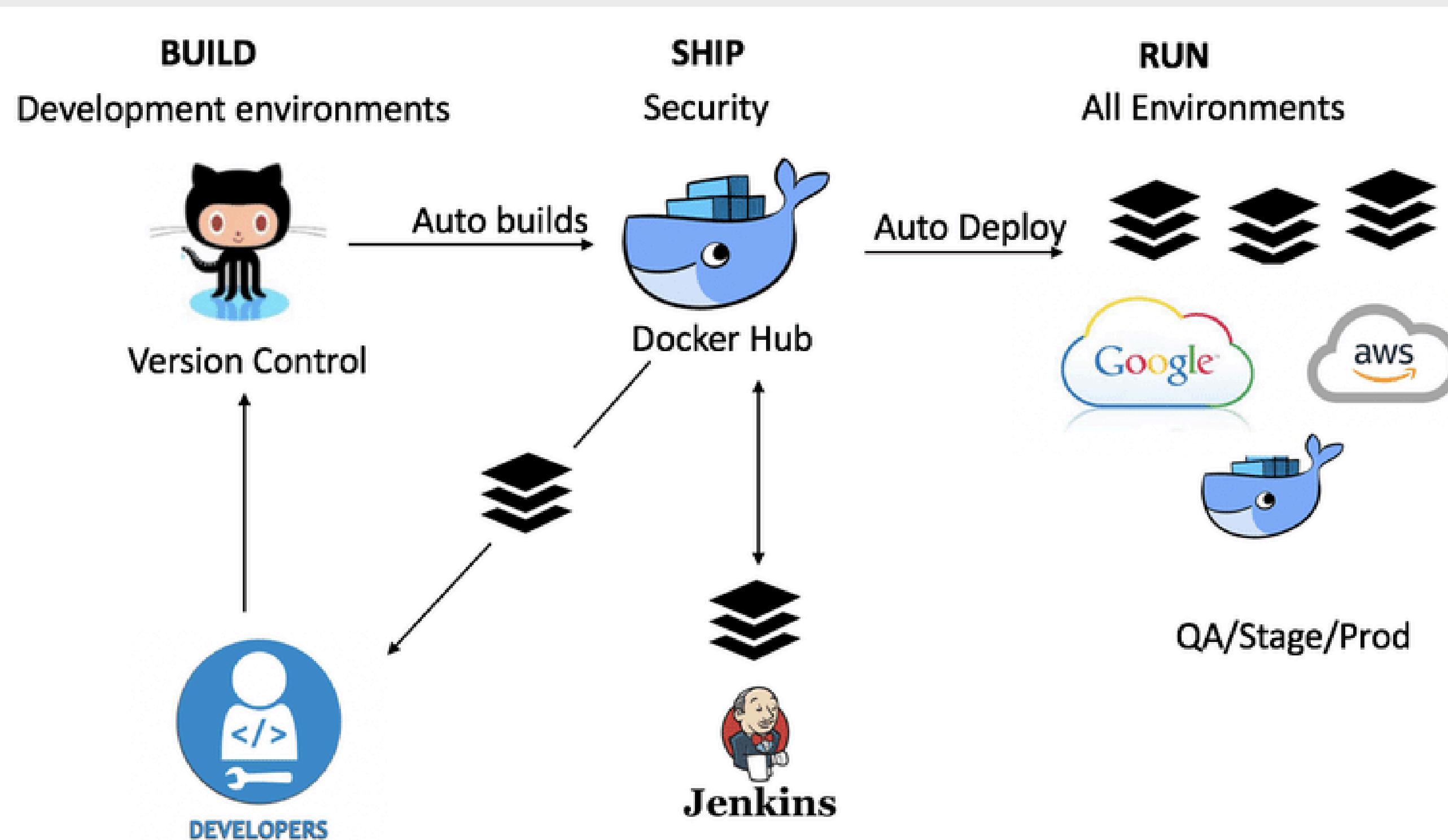
Instance refresh ID 796629e1-b841-4e89-927d-dd592a3c6572	Minimum healthy percentage 100%	Skip matching Disabled
Status <div style="width: 25%; background-color: #0070C0;"></div> 3 instances left to update	Instance warmup 300 seconds	Scale-in protected instances Ignore
Start time 2024 April 14, 12:45:14 PM -04:00	Checkpoints -	Standby instances Ignore

Figure 3-2. Using the EC2 console to see an ASG instance refresh in progress

# Chapitre 3 :

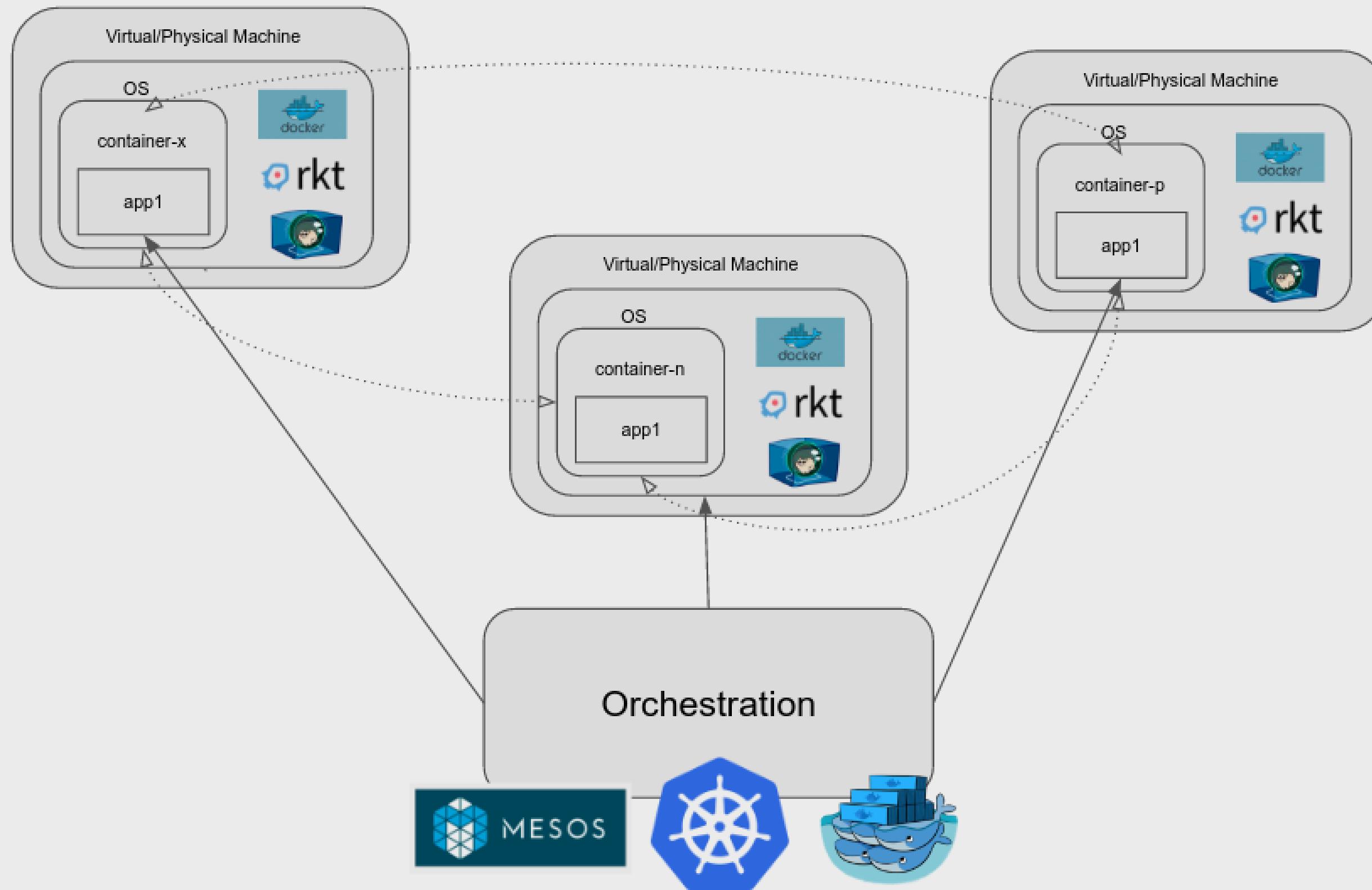
## Comment déployer vos applications

- Introduction à l'orchestration de conteneurs
- Vue d'ensemble de Docker
- Exploration des conteneurs Docker



# Chapitre 3 : Comment déployer vos applications

- Gestion des conteneurs avec Docker
- Construction d'images Docker
- Mise en réseau dans les conteneurs Docker



## Chapitre 3 : Comment déployer vos applications

Auteur : [Badr TAJINI](#)

71

- Exécution locale d'applications Dockerisées
- Introduction à Kubernetes
- Déploiement d'applications avec Kubernetes

Figure 3-3. The Kubernetes architecture consists of a control plane and worker nodes

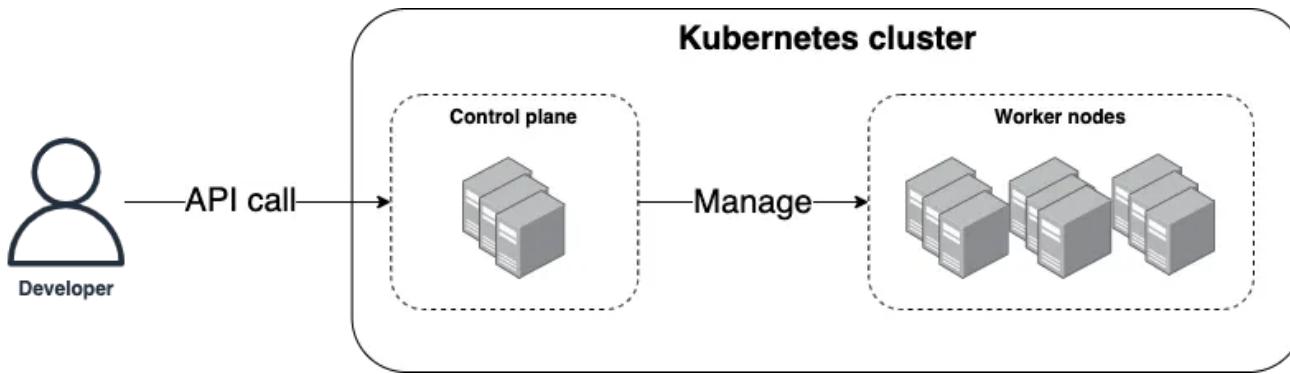
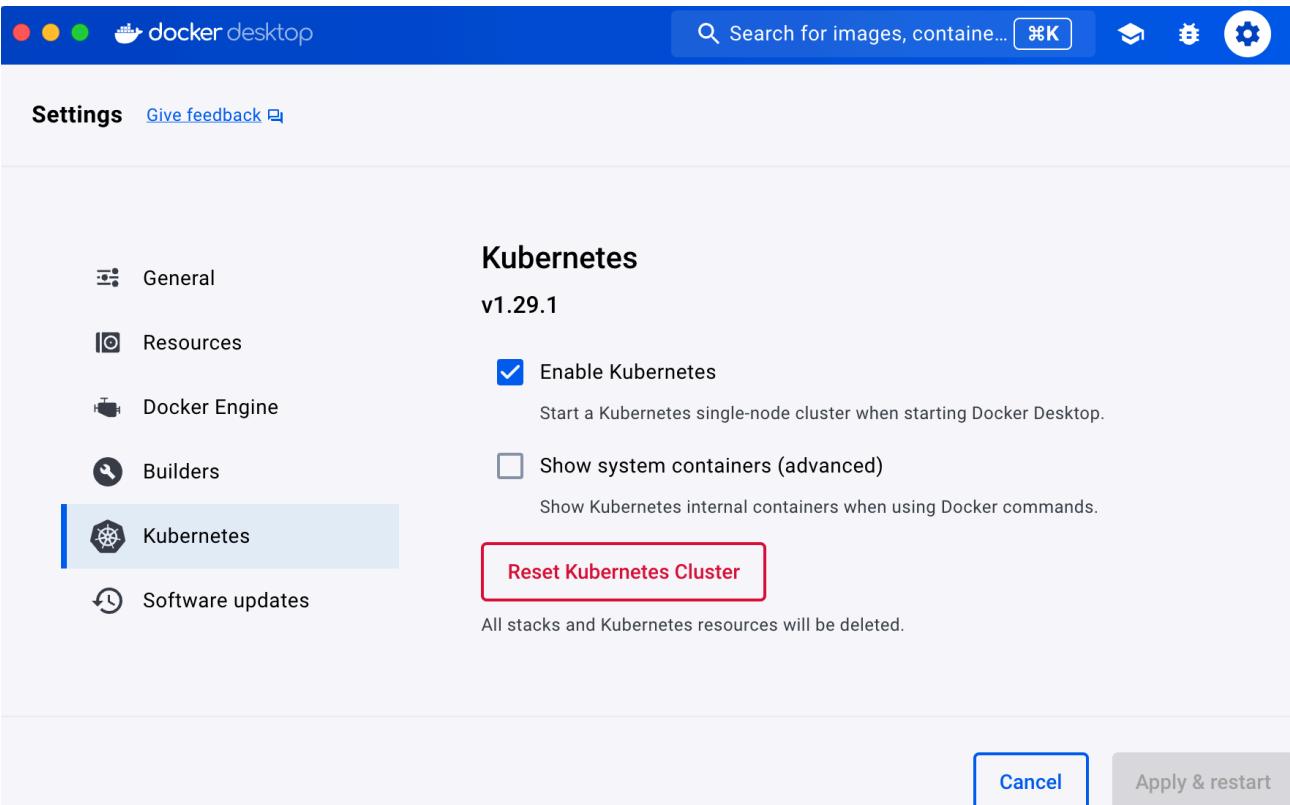
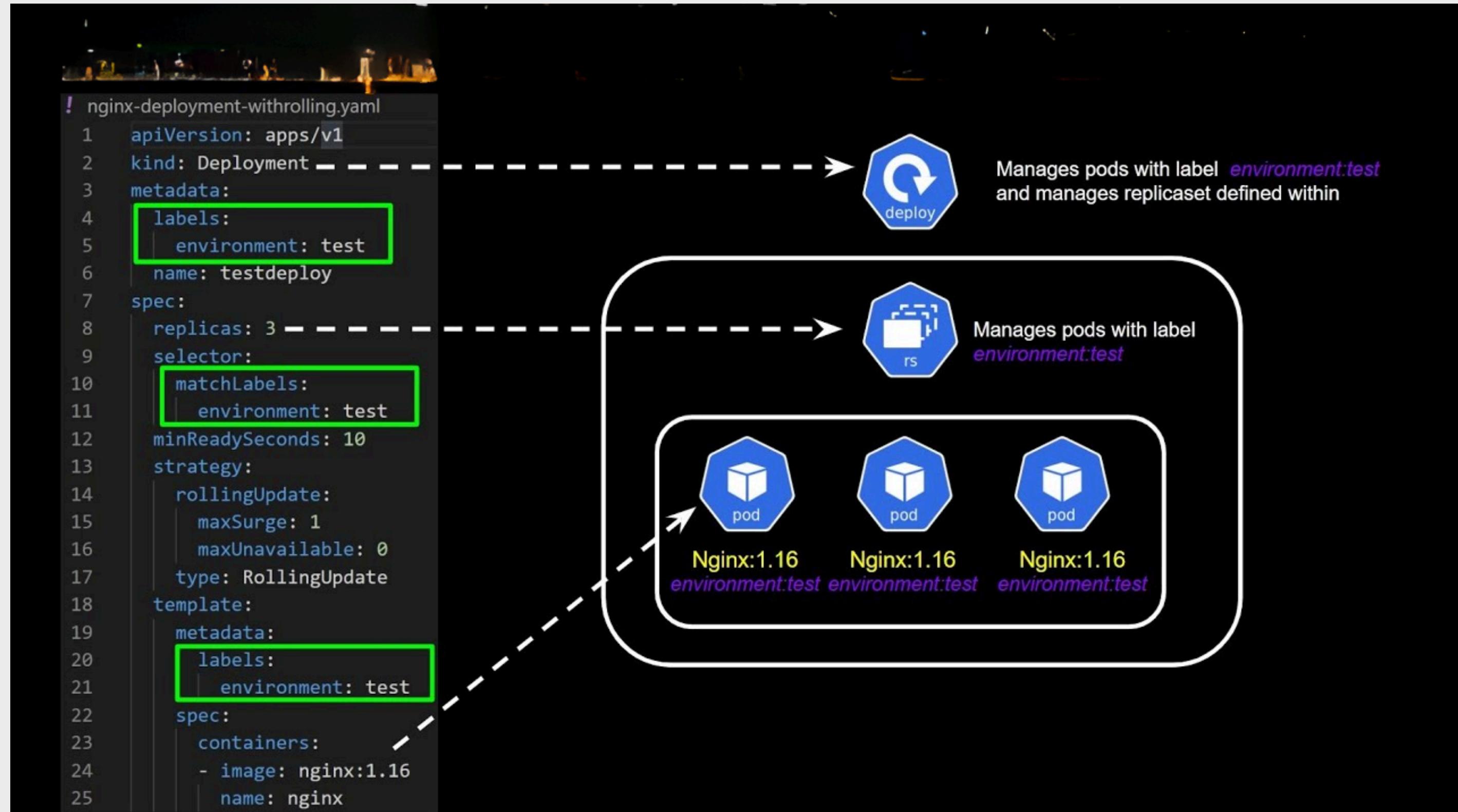


Figure 3-4. Enable Kubernetes on Docker Desktop.



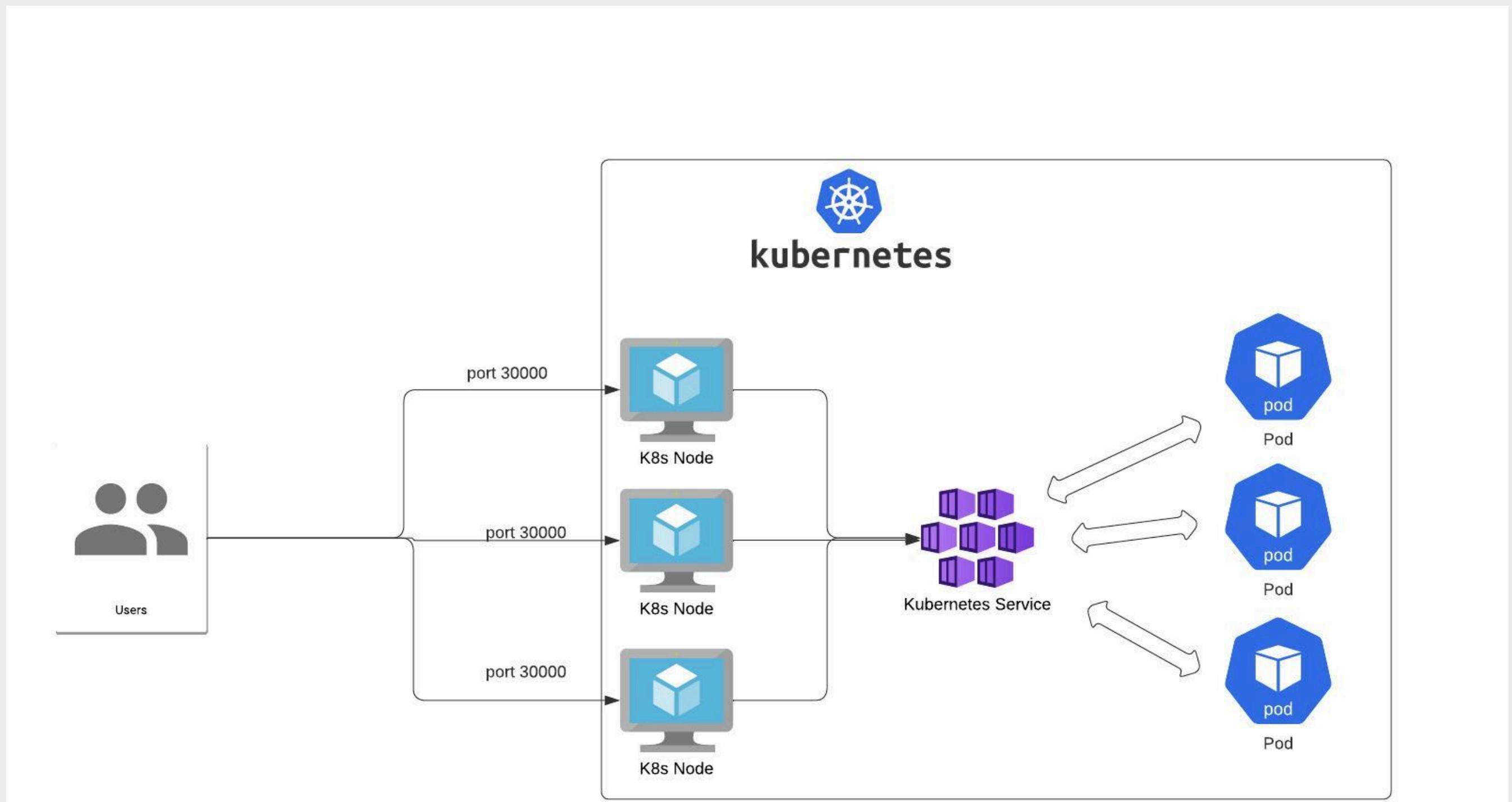
# Chapitre 3 : Comment déployer vos applications

- Configuration du déploiement de Kubernetes avec YAML
- Déploiement et gestion des ressources
- Exploitation des métadonnées et des contrôles d'état de Kubernetes



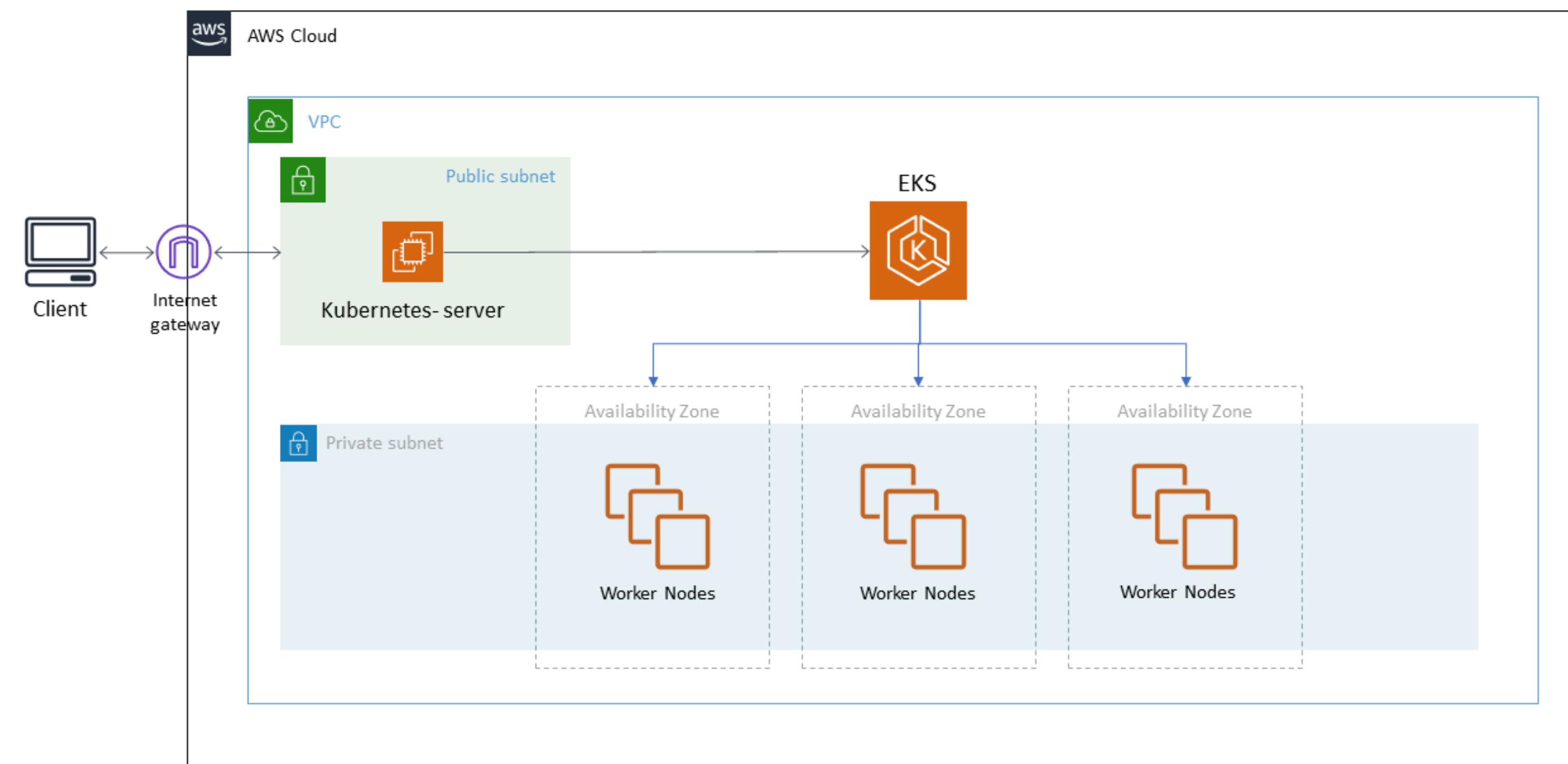
# Chapitre 3 : Comment déployer vos applications

- Déploiement d'un équilibrEUR de charge avec Kubernetes
- Mise à jour en continu avec Kubernetes
- Mise à jour des applications avec Kubernetes



# Chapitre 3 : Comment déployer vos applications

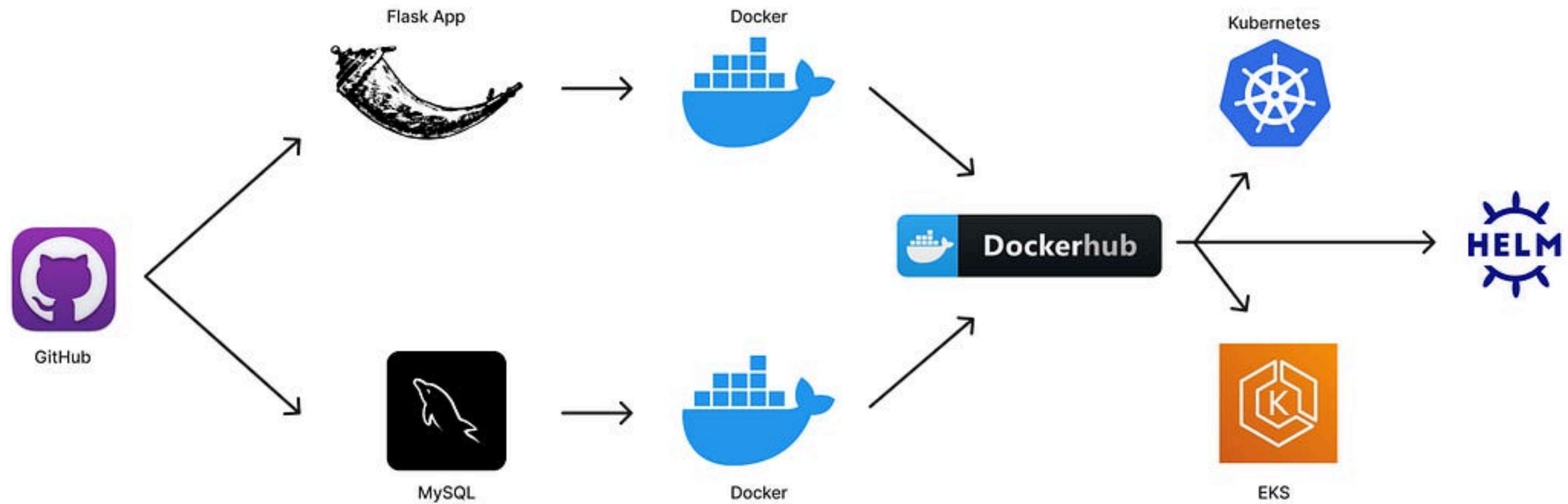
- Implémentation de mises à jour en continu avec Kubernetes
- Déploiement de clusters Kubernetes sur AWS avec EKS
- Caractéristiques principales de l'EKS et considérations



# Chapitre 3 :

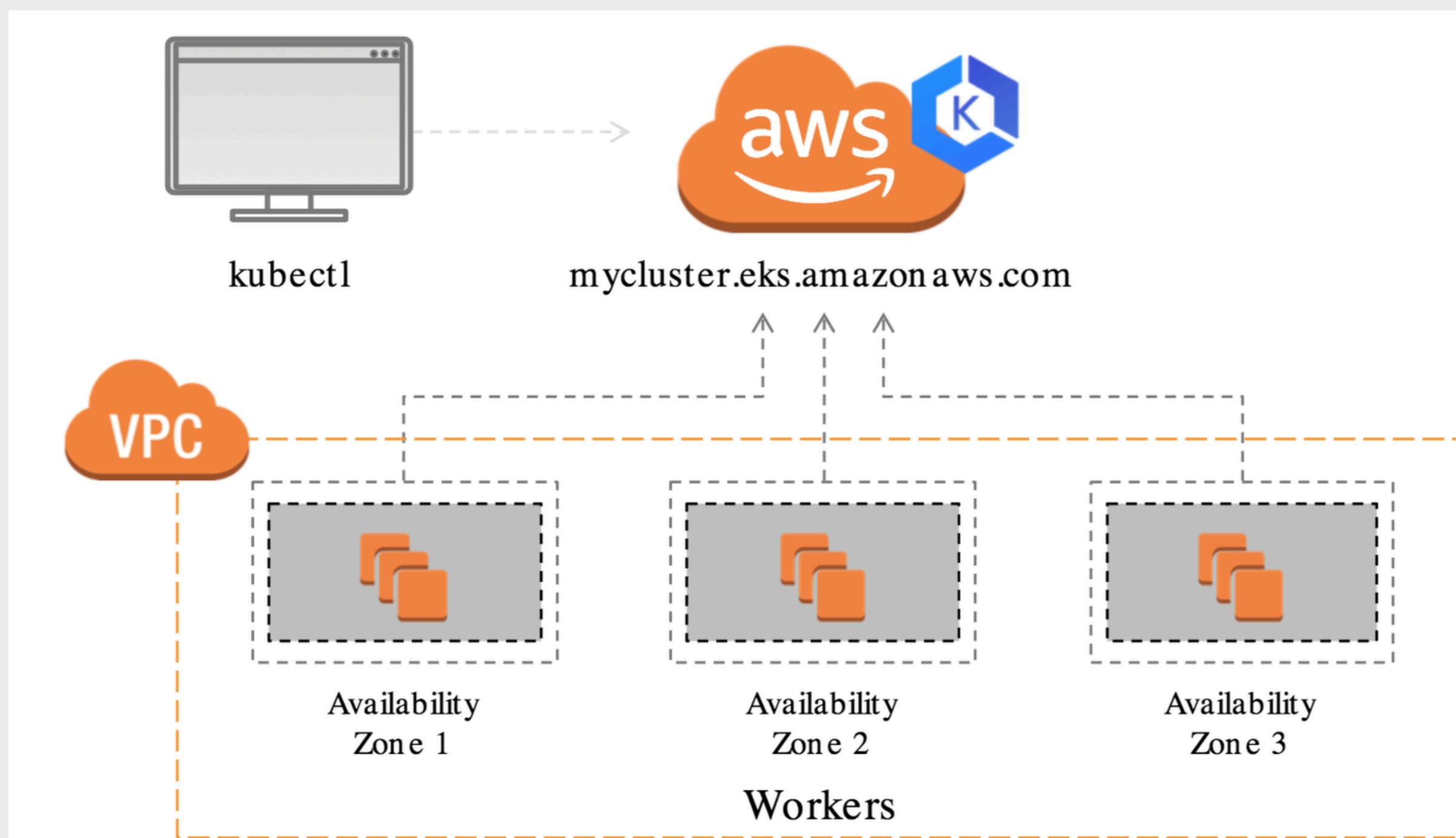
## Comment déployer vos applications

- Configuration du cluster EKS et authentification
- Création d'un dépôt ECR pour les images Docker
- Construction d'images Docker multi-architectures



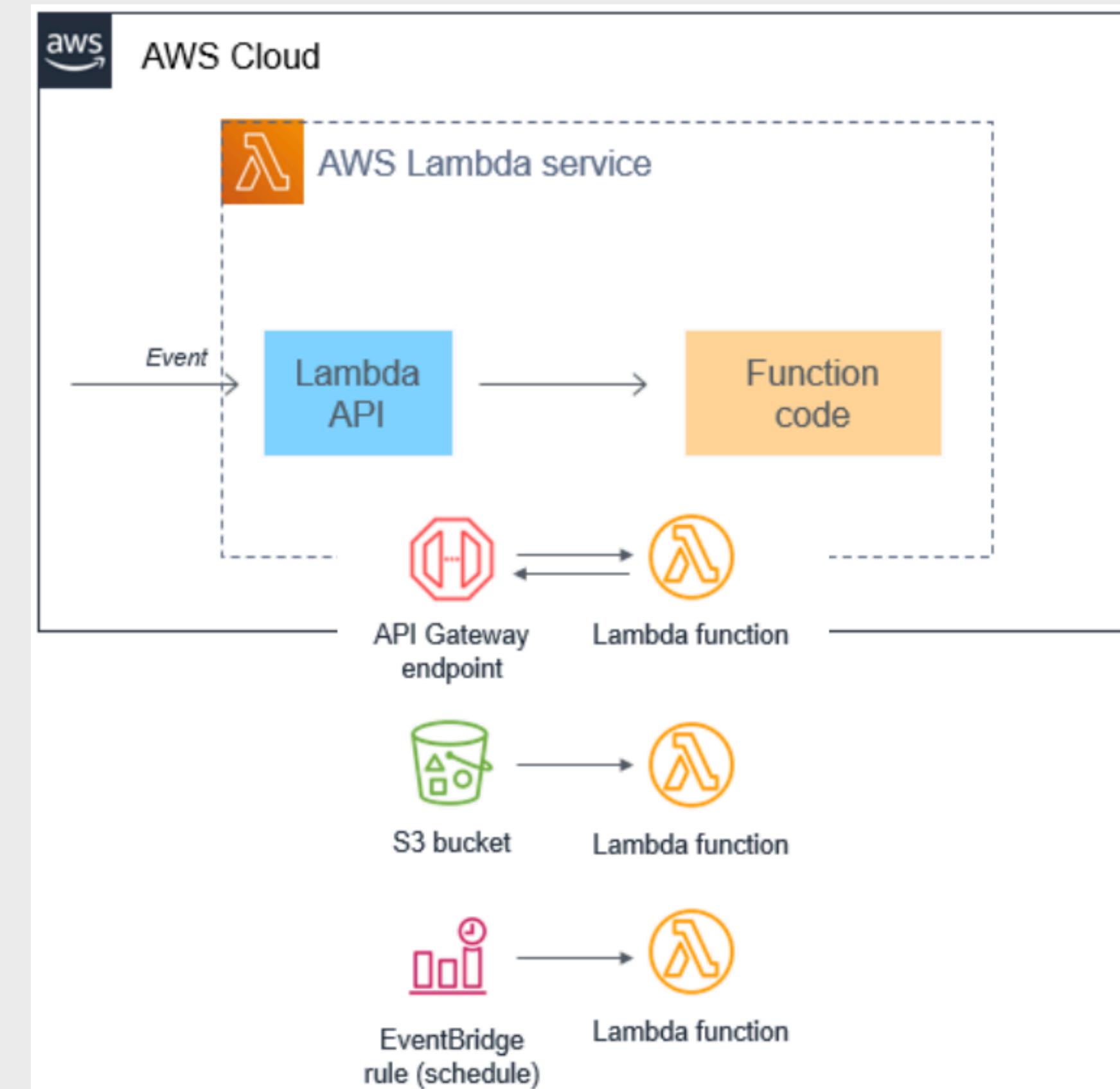
# Chapitre 3 : Comment déployer vos applications

- Déploiement d'une application Dockerisée dans EKS
- Introduction à l'orchestration sans serveur
- Considérations pratiques pour les déploiements sans serveur



# Chapitre 3 : Comment déployer vos applications

- **Avantages et limites de l'architecture sans serveur**
- **Déploiement d'une fonction Serverless avec AWS Lambda**
- **Utilisation pratique de Serverless avec AWS Lambda**

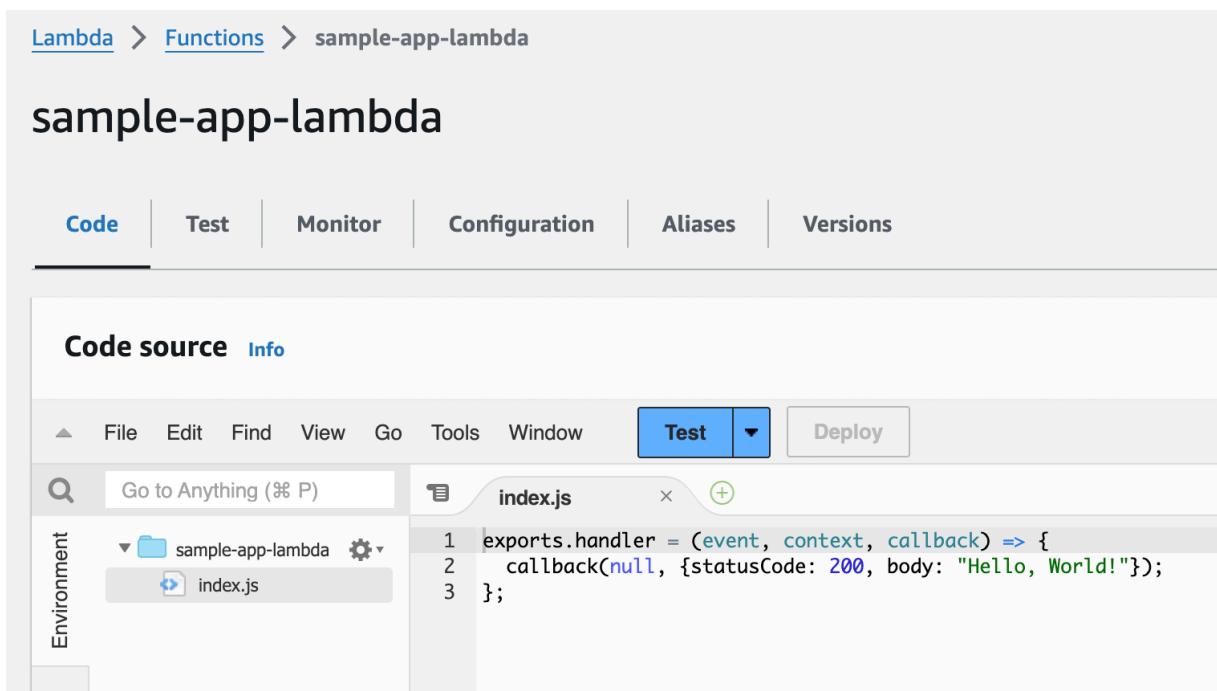


**Chapitre 3 :**  
Comment  
déployer vos  
applications

Auteur : Badr TAJINI

78

- Mise en œuvre d'une fonction AWS Lambda de base
- Test des fonctions Lambda et déclenchement via API Gateway
- Configuration de API Gateway pour l'intégration Lambda



Lambda > Functions > sample-app-lambda

## sample-app-lambda

Code Test Monitor Configuration Aliases Versions

**Code source** Info

File Edit Find View Go Tools Window Test Deploy

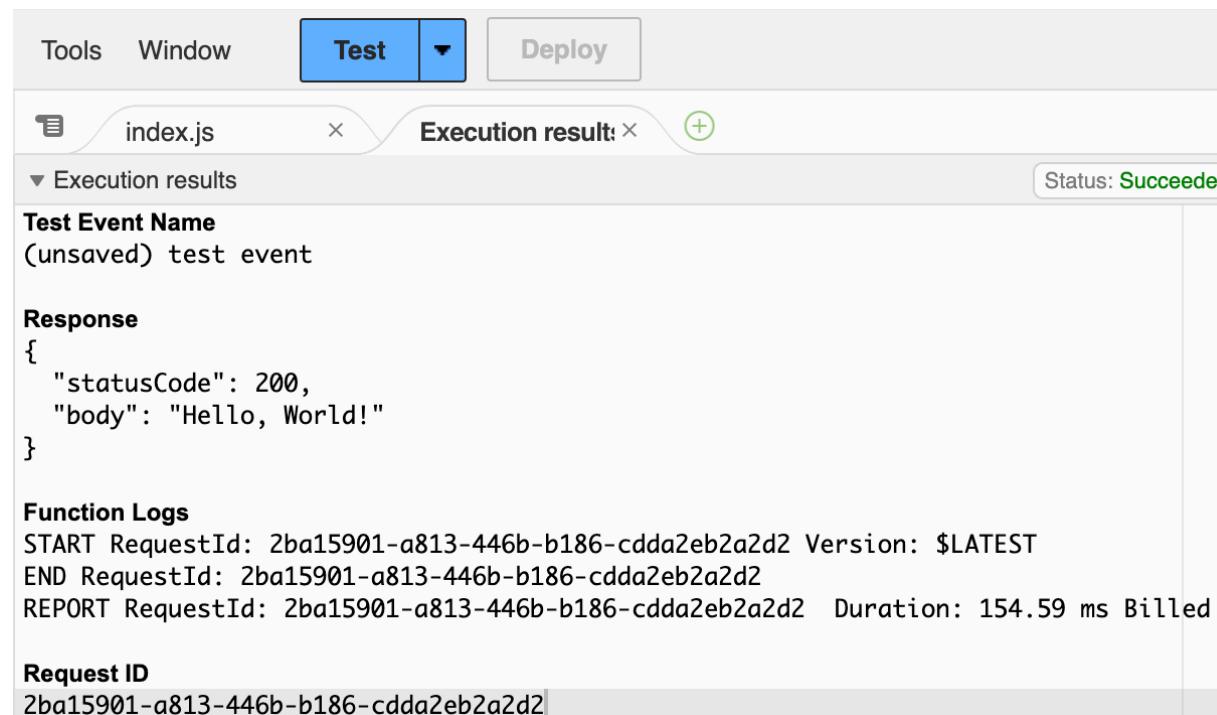
Go to Anything (% P)

Environment

index.js

```
1 exports.handler = (event, context, callback) => {
2   callback(null, {statusCode: 200, body: "Hello, World!"});
3 }
```

Figure 3-5. The Lambda console shows your newly created function



Tools Window Test Deploy

index.js Execution result: Succeeded

Execution results Status: Succeeded

Test Event Name (unsaved) test event

Response

```
{ "statusCode": 200, "body": "Hello, World!" }
```

Function Logs

```
START RequestId: 2ba15901-a813-446b-b186-cdda2eb2a2d2 Version: $LATEST
END RequestId: 2ba15901-a813-446b-b186-cdda2eb2a2d2
REPORT RequestId: 2ba15901-a813-446b-b186-cdda2eb2a2d2 Duration: 154.59 ms Billed Duration: 154.59 ms冷启动: yes
```

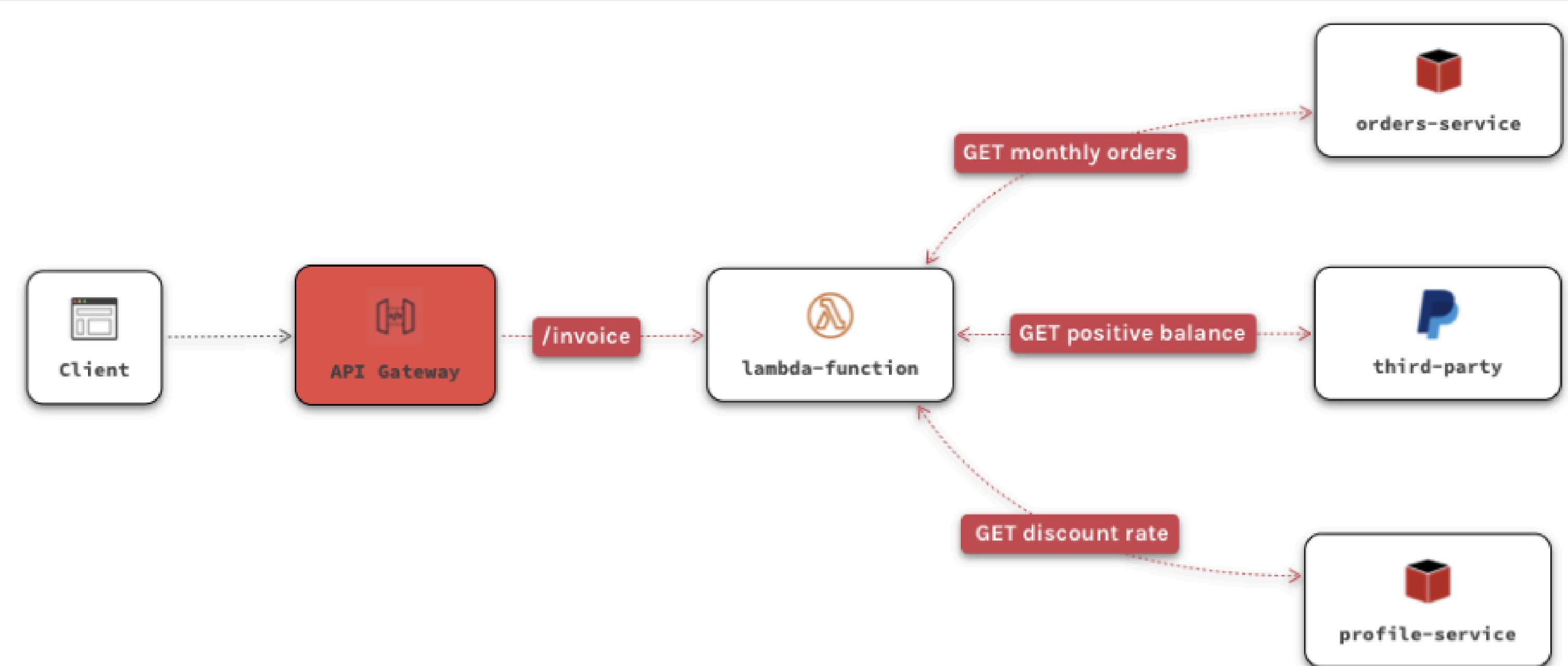
Request ID

```
2ba15901-a813-446b-b186-cdda2eb2a2d2
```

Figure 3-6. The output from manually triggering the Lambda function with a test event

# Chapitre 3 : Comment déployer vos applications

- Déploiement et mise à jour des fonctions AWS Lambda avec API Gateway
- Introduction aux modèles d'orchestration



# Chapitre 3 : Comment déployer vos applications

- Comparaison des modèles d'orchestration
- Avantages et limites de l'orchestration sans serveur

Problem	Server Orchestration	VM Orchestration	Container Orchestration	Serverless Orchestration
Deployment	Supported Example: <a href="#">Ansible</a> deploys apps to servers.	Supported Example: Auto Scaling Group (ASG) deploys from a launch template.	Strong Support Example: <a href="#">Kubernetes</a> deploys containers from deployment specifications.	Strong Support Example: AWS Lambda deploys from a deployment package.
Update Strategies	Supported Example: <a href="#">Rolling updates</a> with Ansible.	Supported Example: ASG supports rolling updates.	Strong Support Example: <a href="#">Kubernetes</a> supports rolling, canary, and blue-green deployments.	Strong Support Example: Lambda supports blue-green, canary, and traffic shifting.
Scheduling	Not Supported No scheduler, pay per server.	Supported Example: ASG scheduler, pay per VM.	Strong Support Example: <a href="#">Kubernetes</a> scheduler, pay per server.	Strong Support Example: Lambda scheduler, pay per execution time.
Rollback	Not Supported No rollback with mutable infrastructure.	Strong Support Rollback to previous immutable version.	Strong Support Rollback to previous immutable version.	Strong Support Rollback to previous immutable version.

# Chapitre 3 : Comment déployer vos applications

- Analyse comparative des modèles d'orchestration

Problem	Server Orchestration	VM Orchestration	Container Orchestration	Serverless Orchestration
Auto Scaling	Not Supported <a href="#">All scaling is manual.</a>	Supported <a href="#">Example: Auto Scaling Group (ASG) auto scaling.</a>	Supported <a href="#">Example: Kubernetes auto scaling.</a>	Strong Support <a href="#">Automatically scaled with option to scale to zero.</a>
Auto Healing	Not Supported <a href="#">All healing is manual.</a>	Supported <a href="#">Example: ASG health checks.</a>	Supported <a href="#">Example: Kubernetes health probes.</a>	Strong Support <a href="#">Fully managed and automatic.</a>
Configuration	Strong Support <a href="#">Example: Ansible roles, templates, etc.</a>	Supported <a href="#">Example: OpenTofu exposes variables.</a>	Strong Support <a href="#">Example: Kubernetes ConfigMaps.</a>	Strong Support <a href="#">Example: Lambda integrated with SSM Parameter Store.</a>
Secrets Management	Supported <a href="#">Example: Ansible Vault.</a>	Manual <a href="#">Example: Read from a secret store during boot.</a>	Strong Support <a href="#">Example: Kubernetes Secrets.</a>	Strong Support <a href="#">Example: Lambda integrated with AWS Secrets Manager.</a>

# Chapitre 3 : Comment déployer vos applications

Problem	Server Orchestration	VM Orchestration	Container Orchestration	Serverless Orchestration
Load Balancing	Manual  Example:  <a href="#">Deploy Nginx</a>	Strong Support  Example: ALBs	Strong Support  Example:  <a href="#">Kubernetes Services</a>	Strong Support  Example: API Gateway
Service Communication	Manual  Example:  <a href="#">Ansible inventory IPs</a>	Manual  Example: Use ALBs with ASGs	Strong Support  Example:  <a href="#">Kubernetes Service Discovery</a>	Strong Support  Example:  <a href="#">Lambda Invoke API, event-driven</a>
Disk Management	Manual  <a href="#">Manually attach and manage drives</a>	Supported  Ephemeral disks only, not permanent	Strong Support  Example:  <a href="#">Kubernetes Volumes and Persistent Volumes</a>	Not Supported  <a href="#">Lambda file system is read-only</a>

- Répartition de la charge, communication et gestion des disques

# Chapitre 3 :

## Comment déployer vos applications

Dimension	Server Orchestration	VM Orchestration	Container Orchestration	Serverless Orchestration
Deployment	Weak	Moderate	Strong	Very Strong
Speed	5-60 minutes	5-30 minutes	1-5 minutes	1 minute
Maintenance	Weak Servers, OS, tools, apps	Moderate Servers, VMs, tools, apps	Weak Servers, containers, tools, apps	Very Strong Apps only
Ease of Learning	Strong Reasonably easy	Strong Reasonably easy	Weak Can be difficult (especially Kubernetes)	Very Strong Very easy
Dev/Prod Parity	Weak Rarely used in dev	Weak Limited options for dev environment	Very Strong Commonly used in dev (Docker)	Very Strong Commonly used in dev (serverless apps)

- Évaluation des dimensions de l'orchestration

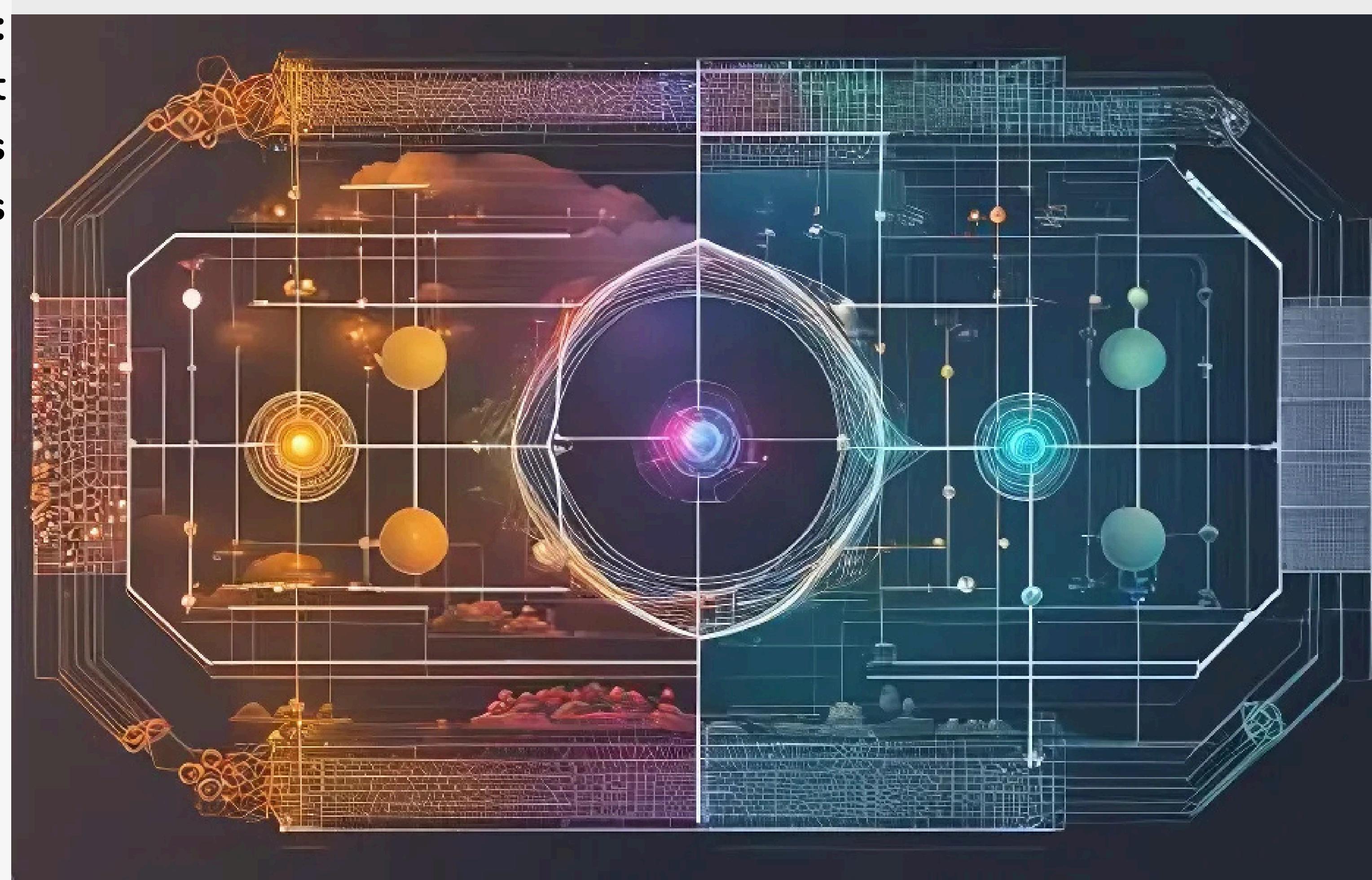
# Chapitre 3 :

## Comment déployer vos applications

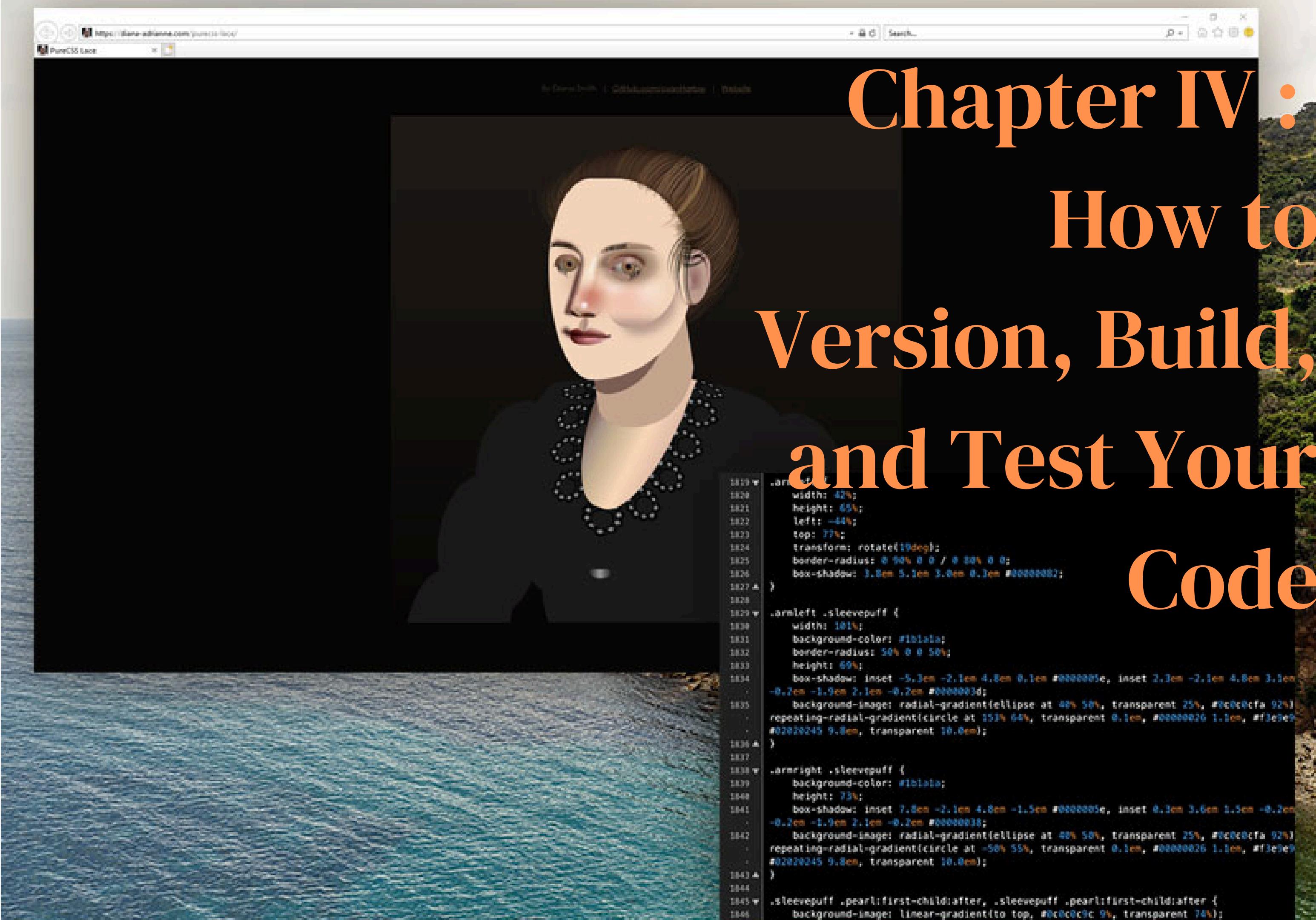
Dimension	Server Orchestration	VM Orchestration	Container Orchestration	Serverless Orchestration
Maturity	Strong Oldest approach, mostly open source	Moderate Second-oldest, mostly proprietary	Strong Newer, mostly open source	Weak Newest, mostly proprietary
Debugging	Strong Server access, but mutable	Very Strong Server access, immutable	Weak Server access, but complex abstractions	Weak No server access
Long-running Tasks	Very Strong Simple	Very Strong Simple	Very Strong Simple	Weak Complex
Performance Tuning	Very Strong Full hardware control	Strong Full control but possible noisy neighbors	Moderate Limited control, more noisy neighbors	Weak No control over hardware, cold starts

- Comparaison de la maturité, du débogage et des tâches de longue durée entre les modèles d'orchestration
- Optimisation des performances et rentabilité

# Chapitre 3 : Comment déployer vos applications



- Conclusions sur les modèles d'orchestration



# Chapter IV:

## How to

# Version, Build,

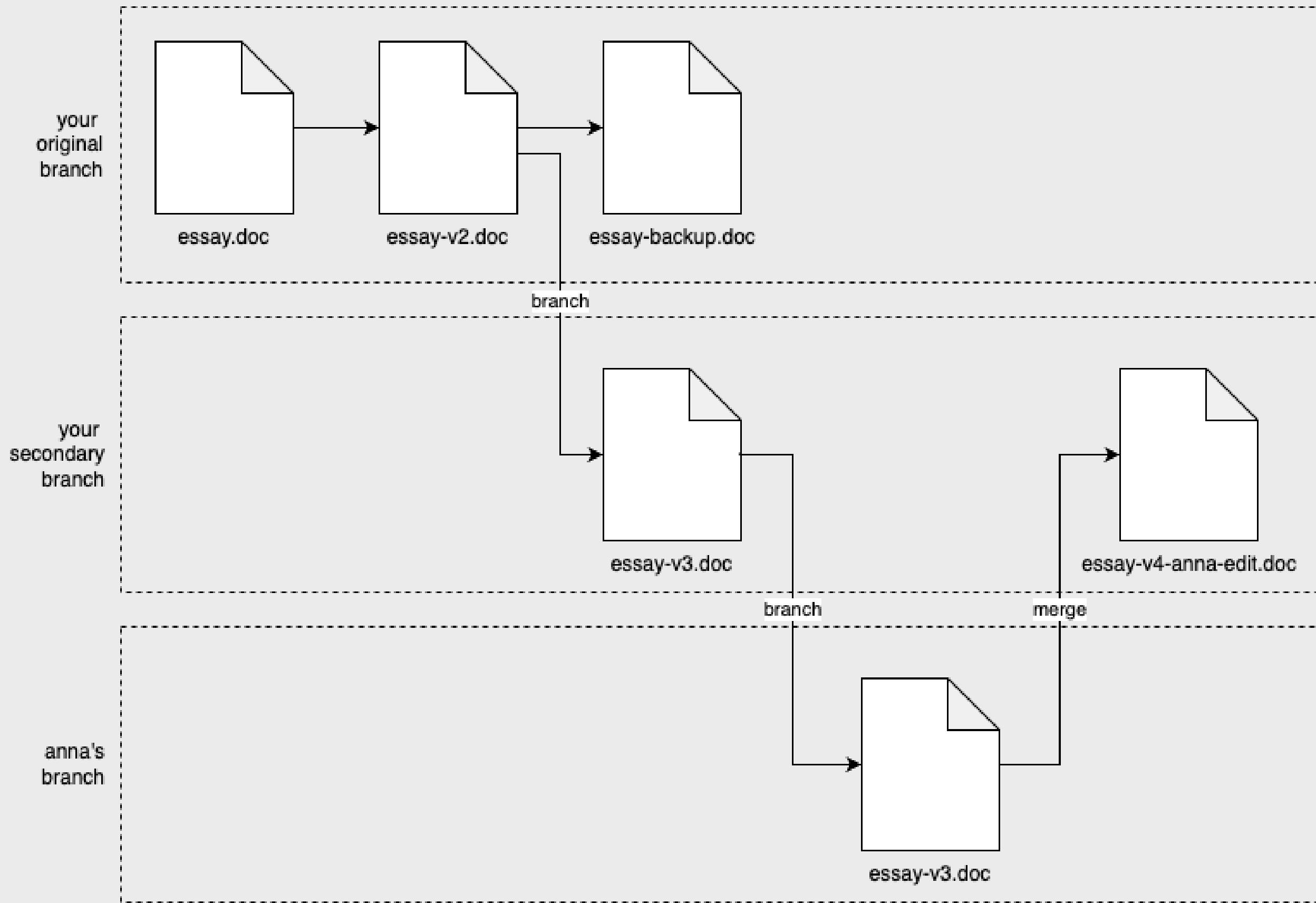
# and Test Your

# Code

# Chapitre 4 : Comment versionner, construire et tester votre code

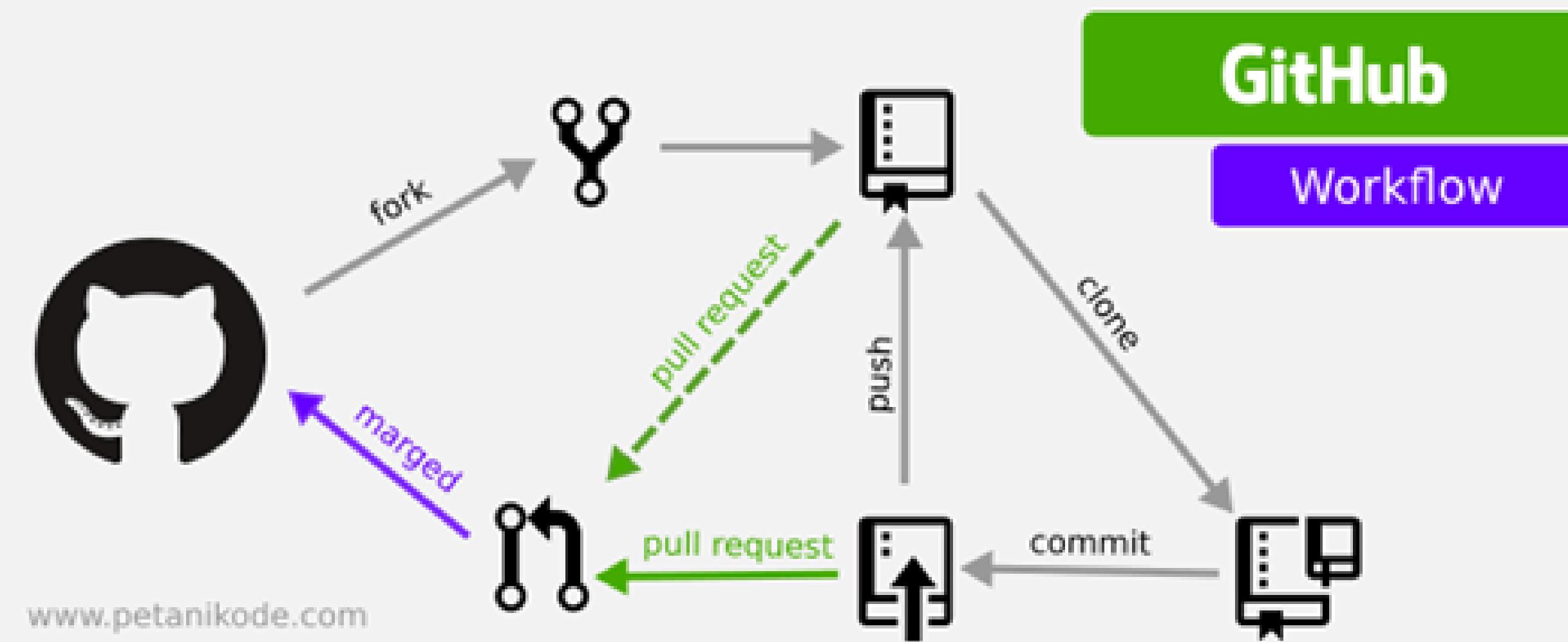
Figure 4-1. Visualizing  
your process with  
Word documents as  
version control

- Introduction au contrôle de version, aux systèmes de construction et aux tests automatisés
- Principes de base du contrôle de version
- L'importance du contrôle de version



# Chapitre 4 : Comment versionner, construire et tester votre code

- Comprendre Git en tant qu'outil de contrôle de version
- Retracer les modifications avec Git
- Résoudre les conflits et conserver l'historique



# Chapitre 4 :

## Comment versionner, construire et tester votre code

- Fonctionnalités avancées de Git et branchement
- Contrôle de version distribué avec GitHub
- Travaux pratiques avec Git et GitHub



# Chapitre 4 : Comment versionner, construire et tester votre code

- Initialisation d'un dépôt Git et gestion des fichiers avec `.gitignore`
- Staging et Committing de fichiers sur Git
- Création et liaison d'un dépôt GitHub

Figure 4-2. Create a new GitHub repo

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

Owner \*      Repository name \*

brikis98 / fundamentals-of-devops

fundamentals-of-devops is available.

Great repository names are short and memorable. Need inspiration? How about [solid-barnacle](#) ?

Description (optional)

---

 **Public**  
Anyone on the internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

---

[Create repository](#)

## Chapitre 4 : Comment versionner, construire et tester votre code

Auteur : Badr TAJINI

91

- Liaison d'un dépôt Git local à GitHub
- Création d'un README pour votre dépôt GitHub
- Synchronisation des modifications locales et distantes

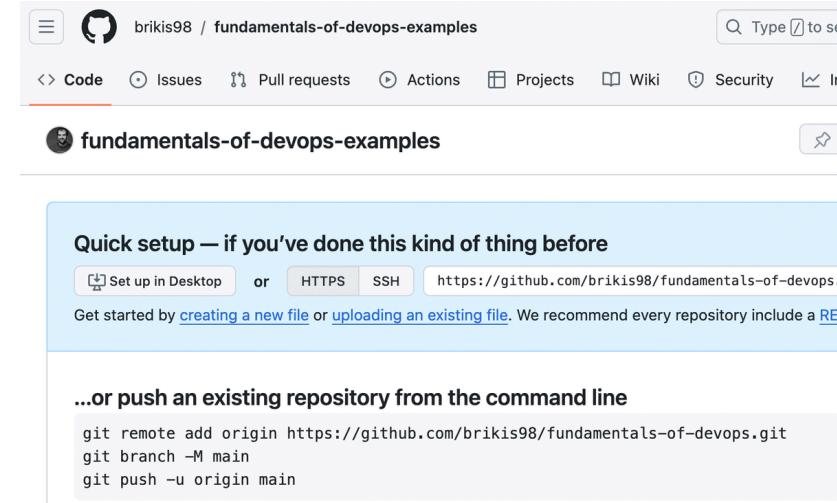


Figure 4-3. A newly created, empty GitHub repo

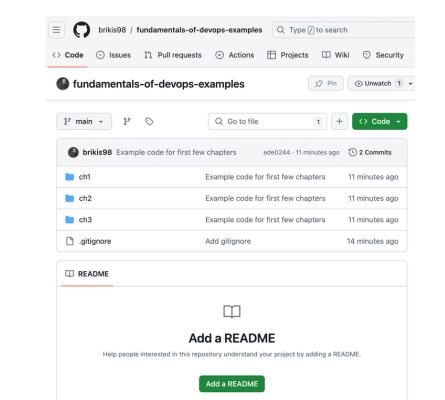


Figure 4-4. Your GitHub repo with code in it

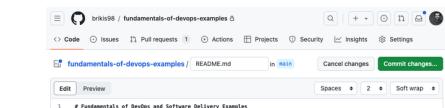


Figure 4-5. Filling in a README for the GitHub repo

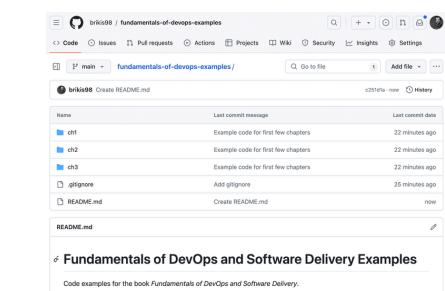


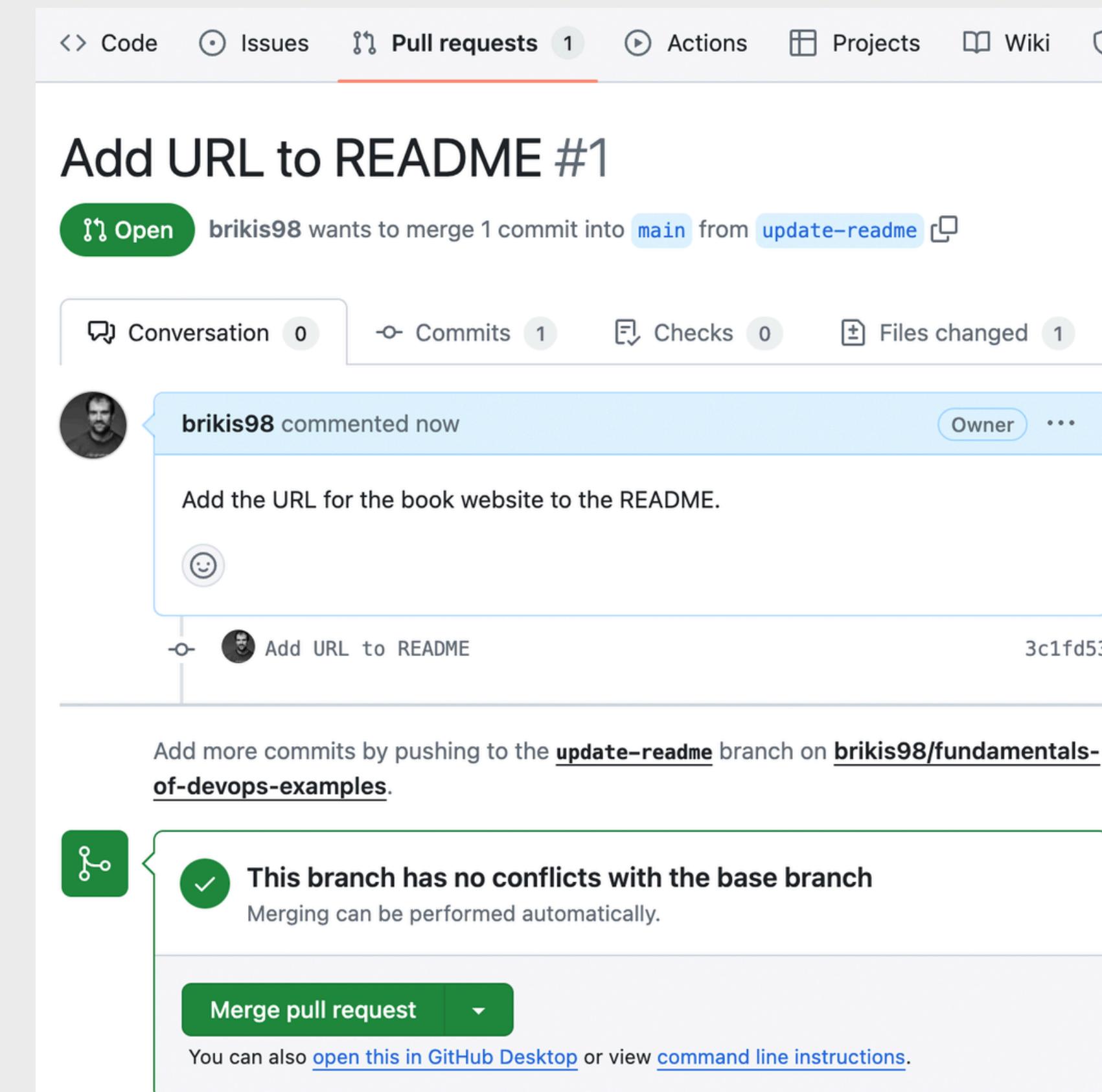
Figure 4-6. Your GitHub repo with a README

# Chapitre 4 :

## Comment versionner, construire et tester votre code

Figure 4-7. An open pull request in GitHub

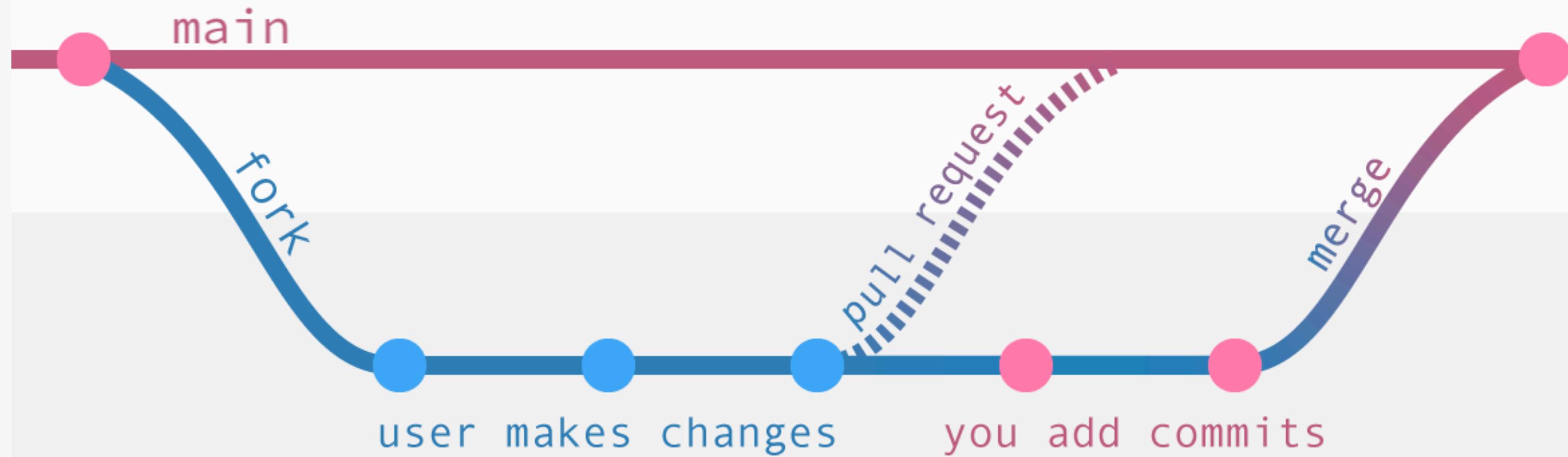
- Flux de travail Git de base et extraction de modifications
- Ouverture d'une Pull Request sur GitHub
- Review des Pull Request et bonnes pratiques



# Chapitre 4 :

## Comment versionner, construire et tester votre code

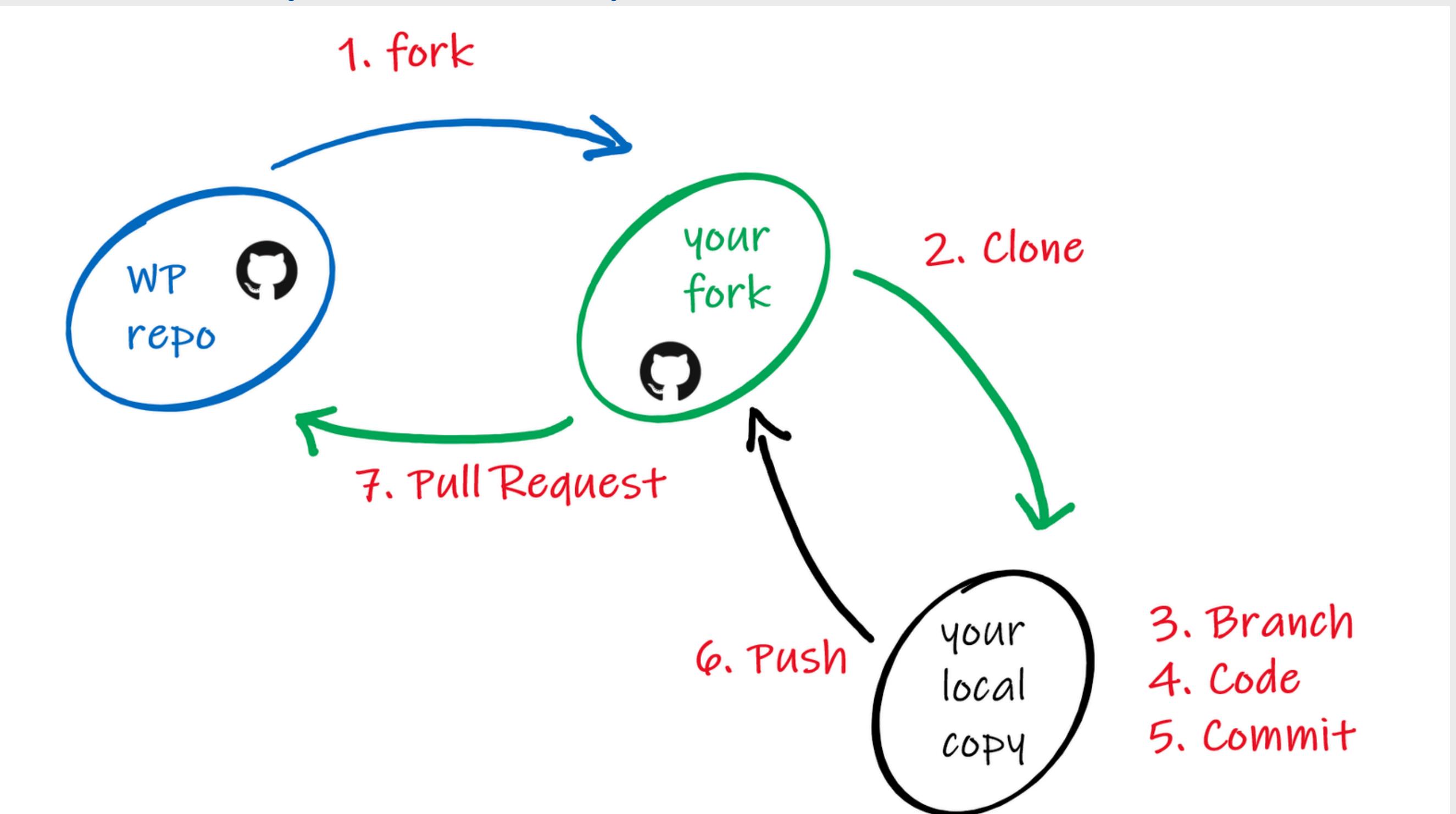
- Bonnes pratiques pour l'utilisation du contrôle de version
- Principes des Commits et des PR atomiques
- Utilisation des révisions de code pour le contrôle de la qualité



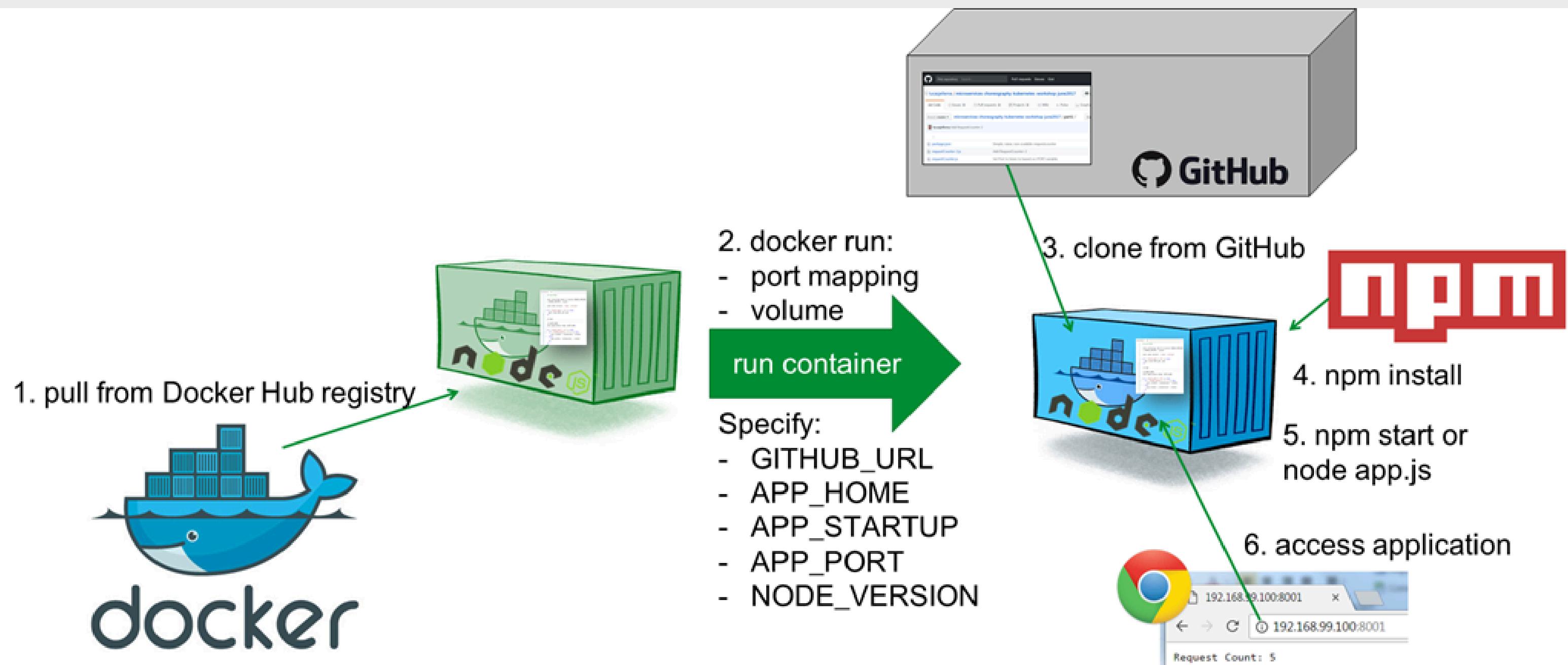
# Chapitre 4 : Comment versionner, construire et tester votre code

## • Garantir la qualité du code grâce aux révisions et à la protection

- Introduction aux systèmes de construction
- Mise en place d'un système de construction avec NPM

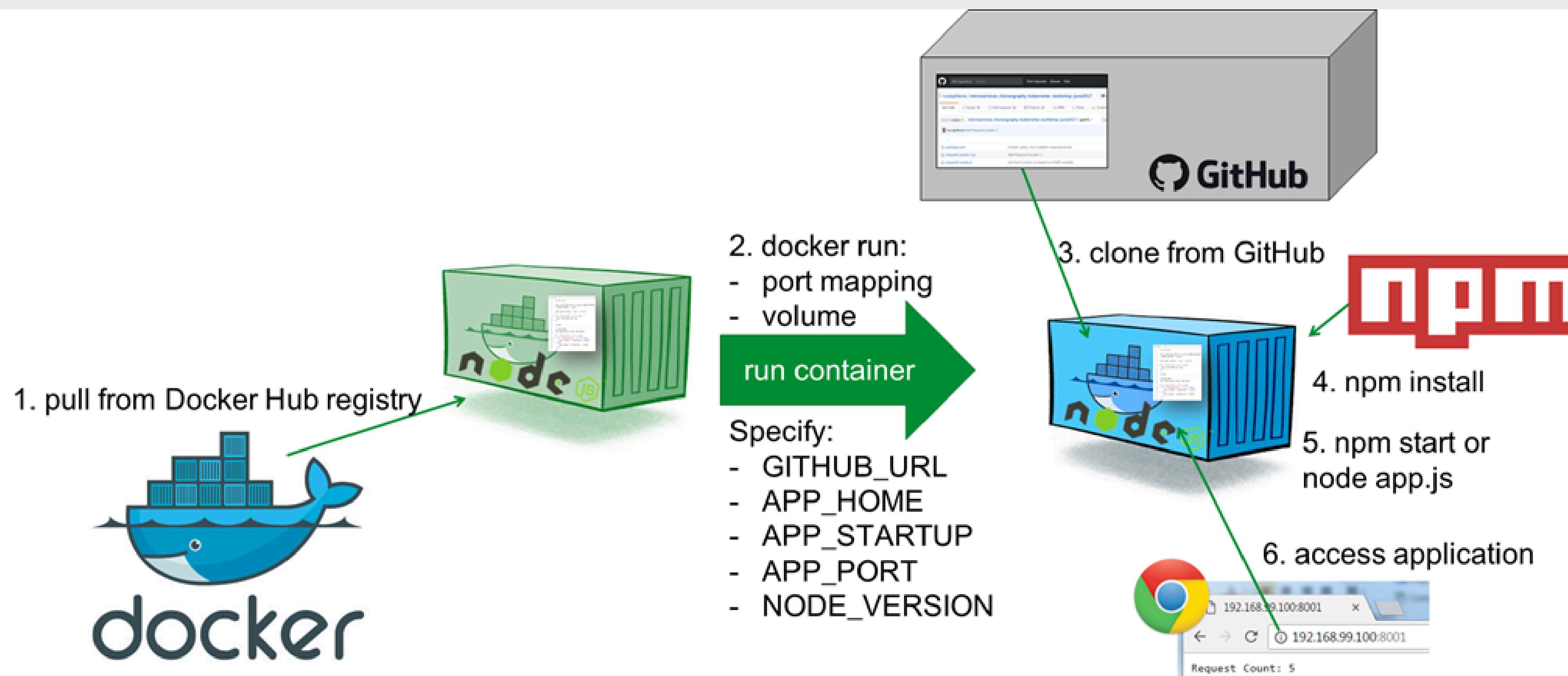


- Améliorer les processus de construction avec NPM et Docker
  - Gestion des dépendances dans les systèmes de compilation
  - Principaux conseils



## Gestion des dépendances dans les projets logiciels modernes

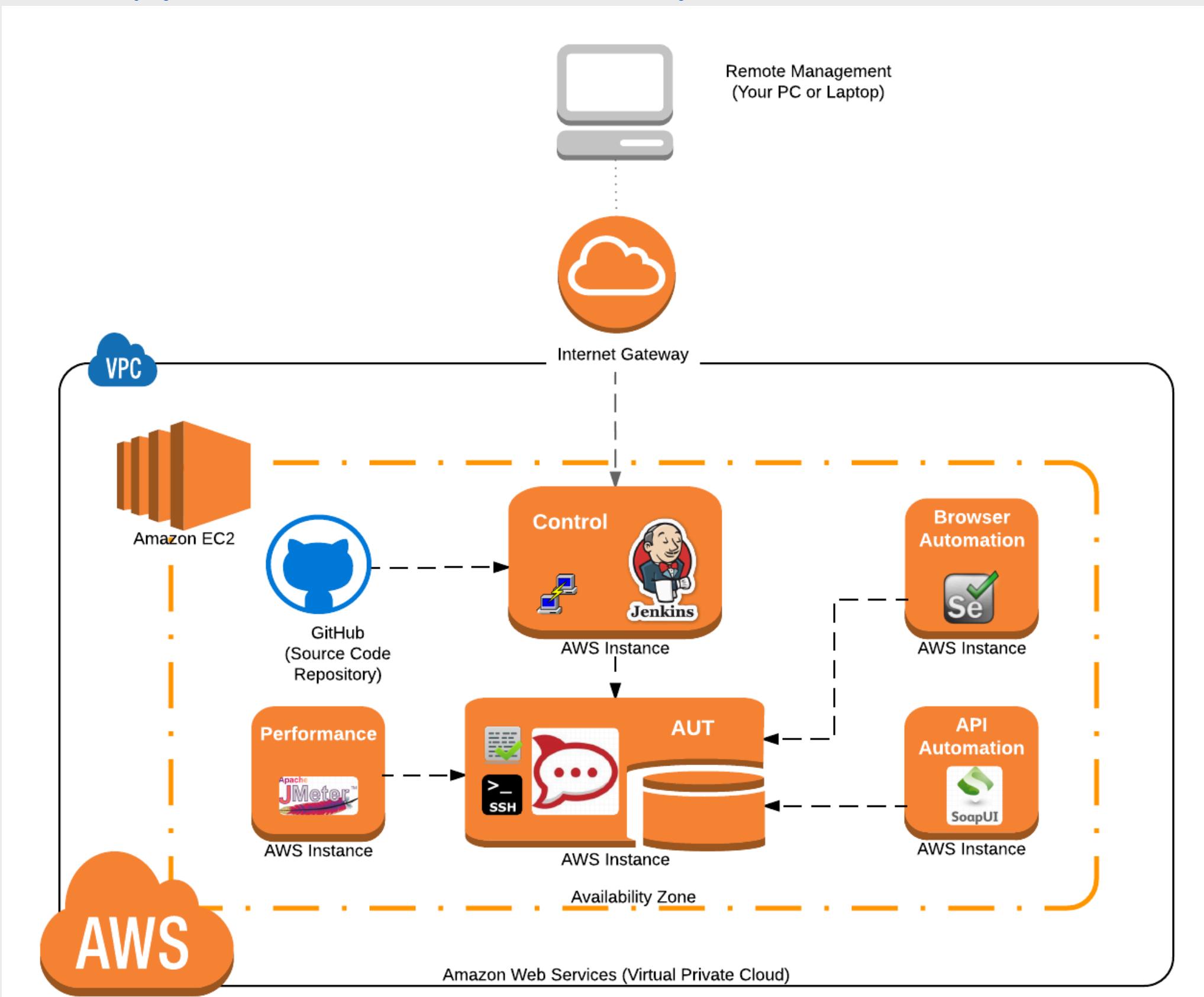
- L'ajout et la gestion des dépendances avec NPM
- Réécriture d'applications avec un framework



# Chapitre 4 :

## Comment versionner, construire et tester votre code

- Gestion des dépendances dans les constructions
- Compréhension du code hérité et des tests automatisés
- Types de tests statiques et automatisés



# Chapitre 4 : Comment versionner, construire et tester votre code

- Introduction aux tests automatisés
- Mise en place des tests pour les applications Node.js
- Écriture et exécution de tests automatisés

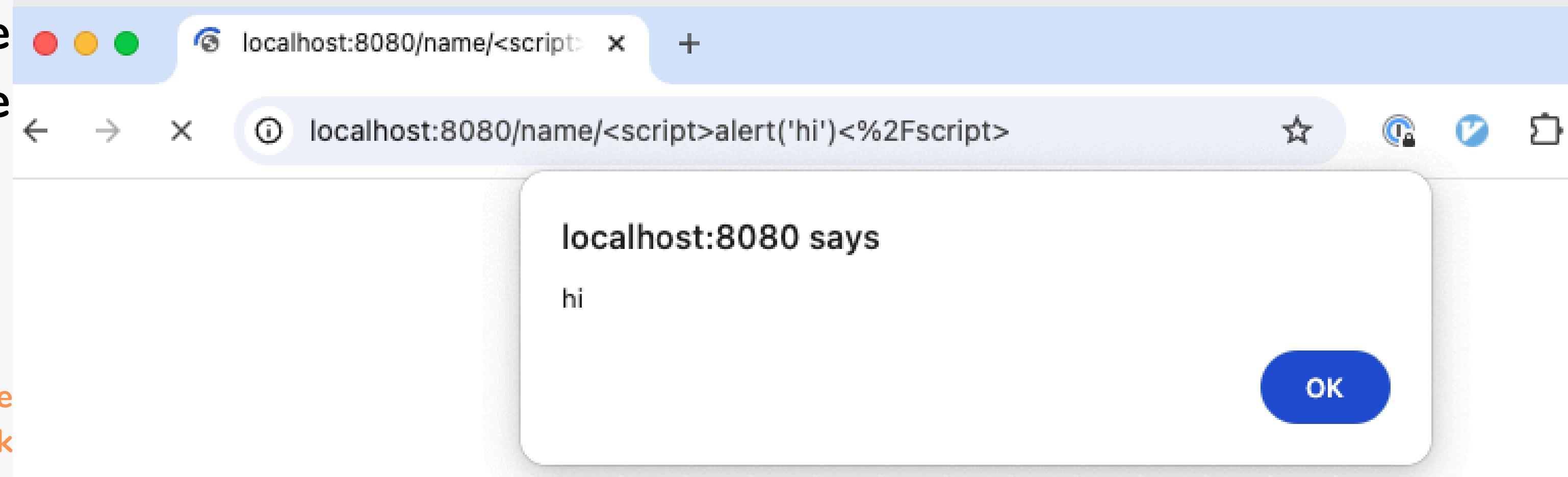


Figure 4-8. An example of an injection attack

# Chapitre 4 :

## Comment versionner, construire et tester votre code

- Tests avancés et automatisation des points finaux
- Identification et résolution des vulnérabilités de sécurité
- Mise en œuvre de l'assainissement pour prévenir les XSS

The screenshot shows a GitHub Actions comment from the bot (@github-actions) on a commit 9bad66a, posted 19 minutes ago. The comment displays the deployment status as "success" and provides Observatory results for the URL blog-mfch8vn8m.vercel.app, stating a score of 45 of 100. It also links to the full report at <https://observatory.mozilla.org/analyze/blog-mfch8vn8m.vercel.app>. A table titled "Highlights" lists five findings with their respective scores and descriptions.

Passed	Score	Description
●	-25	Content Security Policy (CSP) header not implemented
●	5	Preloaded via the HTTP Strict Transport Security (HSTS) preloading process
●	-5	X-Content-Type-Options header not implemented
●	-20	X-Frame-Options (XFO) header not implemented
●	-10	X-XSS-Protection header not implemented

# Chapitre 4 : Comment versionner, construire et tester votre code

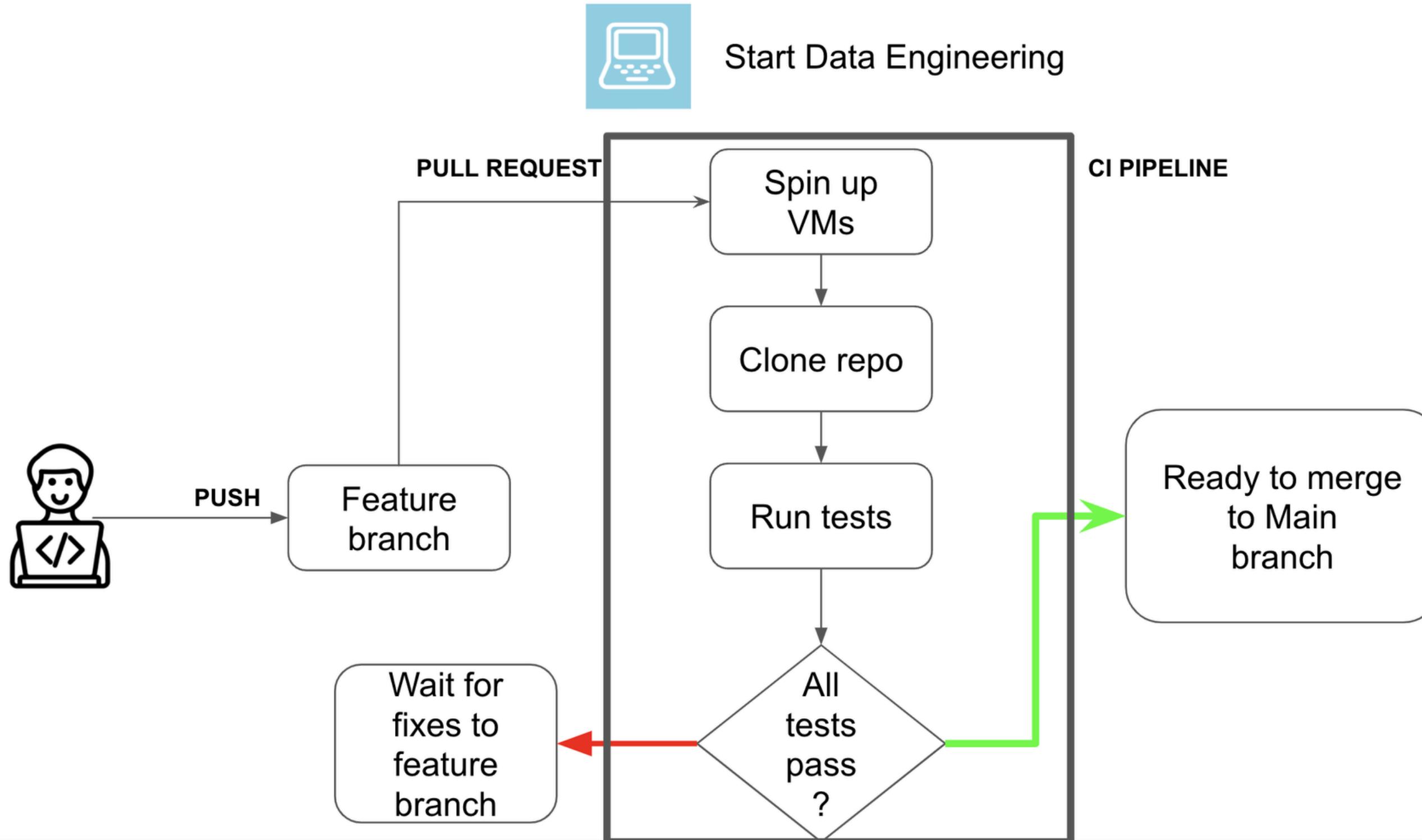
- Traitement des tests qui échouent avec HTML Templating
- Les boucles de rétroaction automatisées
- Extension des tests à l'infrastructure

Test Suite Reporter				
Started: 2020-03-30 09:21:51				
Suites (2)		Tests (7)		
<a href="#">/jest-html-reporter/test/demo.spec.ts</a>				7.077s
Demo	should demonstrate a passing test		passed	0.001s
Demo	should demonstrate a pending test		pending	5s
Demo	should demonstrate a failing test		failed	5.011s
<pre>Error: Expected value to equal:   "that" Received:   "this"  at Object.it (test/demo.spec.js:6:16) at new Promise () at at process._tickCallback (internal/process/next_tick.js:188:7)</pre>				
<a href="#">/path/to/another-spec.ts</a>				1.812s
index	should demonstrate another test spec		passed	0.015s
<a href="#">/path/to/yet.another.spec.ts</a>				1.575s
Another > Spec	should demonstrate yet another test		passed	0.005s
Another > Spec > Test	should demonstrate a nested test		passed	0.003s
Another > Spec > Test	should demonstrate yet another nested test		passed	0s

# Chapitre 4 :

## Comment versionner, construire et tester votre code

- Automatisation des tests d'infrastructure
- Recommandations en matière de tests

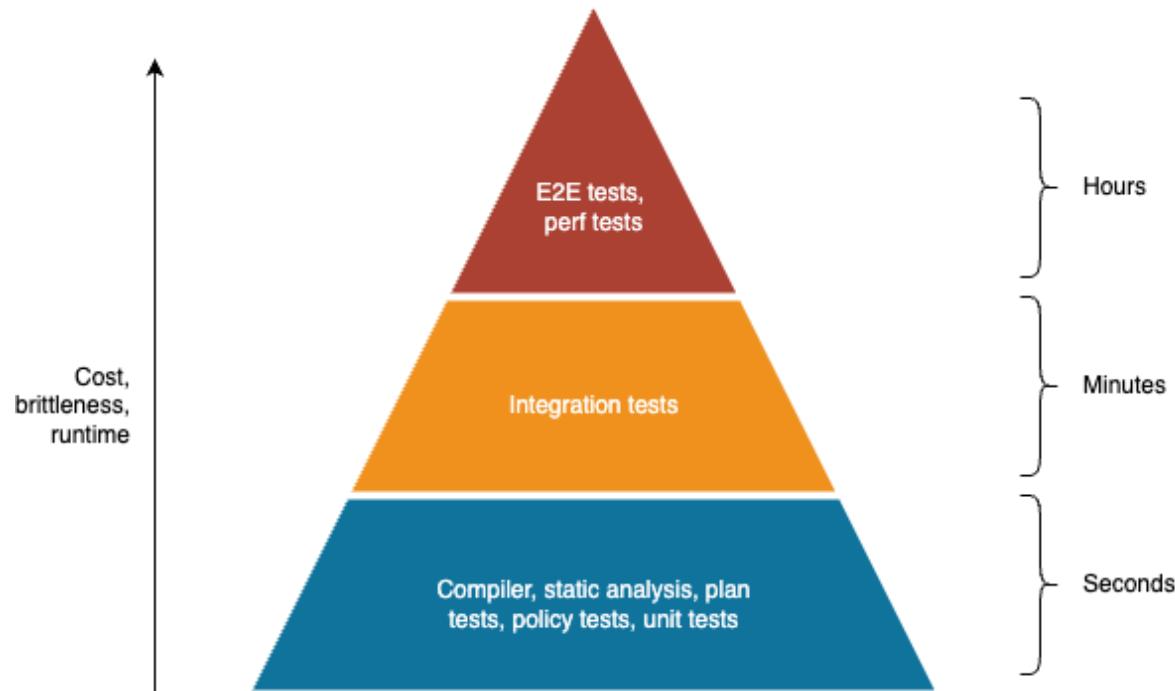


- Vue d'ensemble des types de tests et de leurs caractéristiques
- Cadre de la pyramide des tests
- Directives de mise en œuvre
- Avantages stratégiques

Table 4-1. A comparison of different types of automated tests

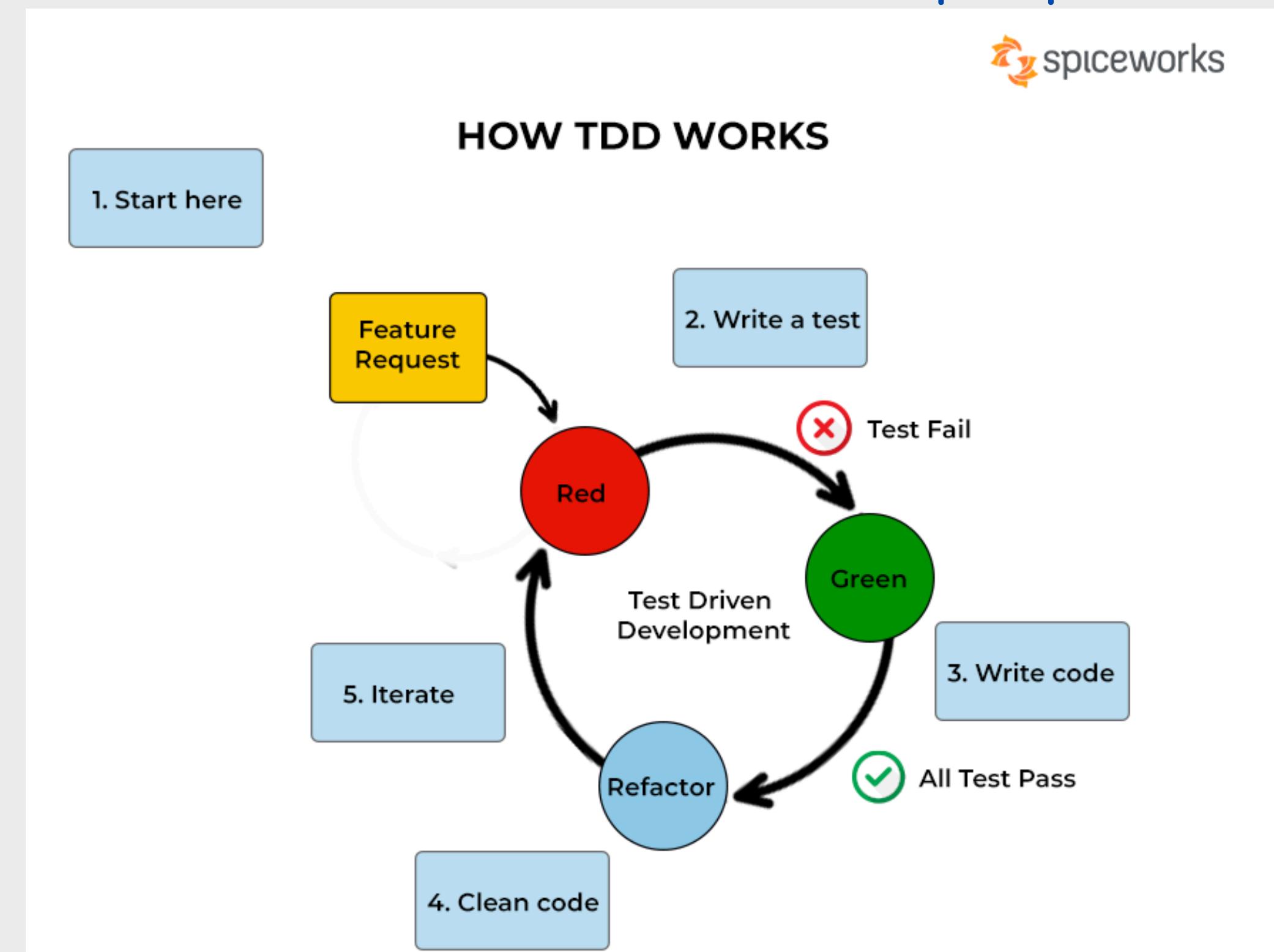
	Compiler	Static analysis	Policy tests	Plan tests	Unit tests	Integration tests	E2E tests	Perf tests
Syntax errors	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Type errors	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Compliance errors	No	Yes	Yes	Yes	No	No	No	No
Unit functionality	No	No	No	Partial	Yes	Yes	No	No
Interaction of units	No	No	No	No	No	Yes	Yes	No
System functionality	No	No	No	No	No	No	Yes	Yes
Functionality under load	No	No	No	No	No	No	No	Yes
Speed	Very fast	Very fast	Very fast	Fast	Fast	Slow	Very slow	Very slow
Stability	Very stable	Very stable	Very stable	Stable	Stable	Stable	Flaky	Very flaky
Ease of use	Easy	Easy	Easy	Easy	Moderate	Hard	Very hard	Very hard

Figure 4-9. The test pyramid



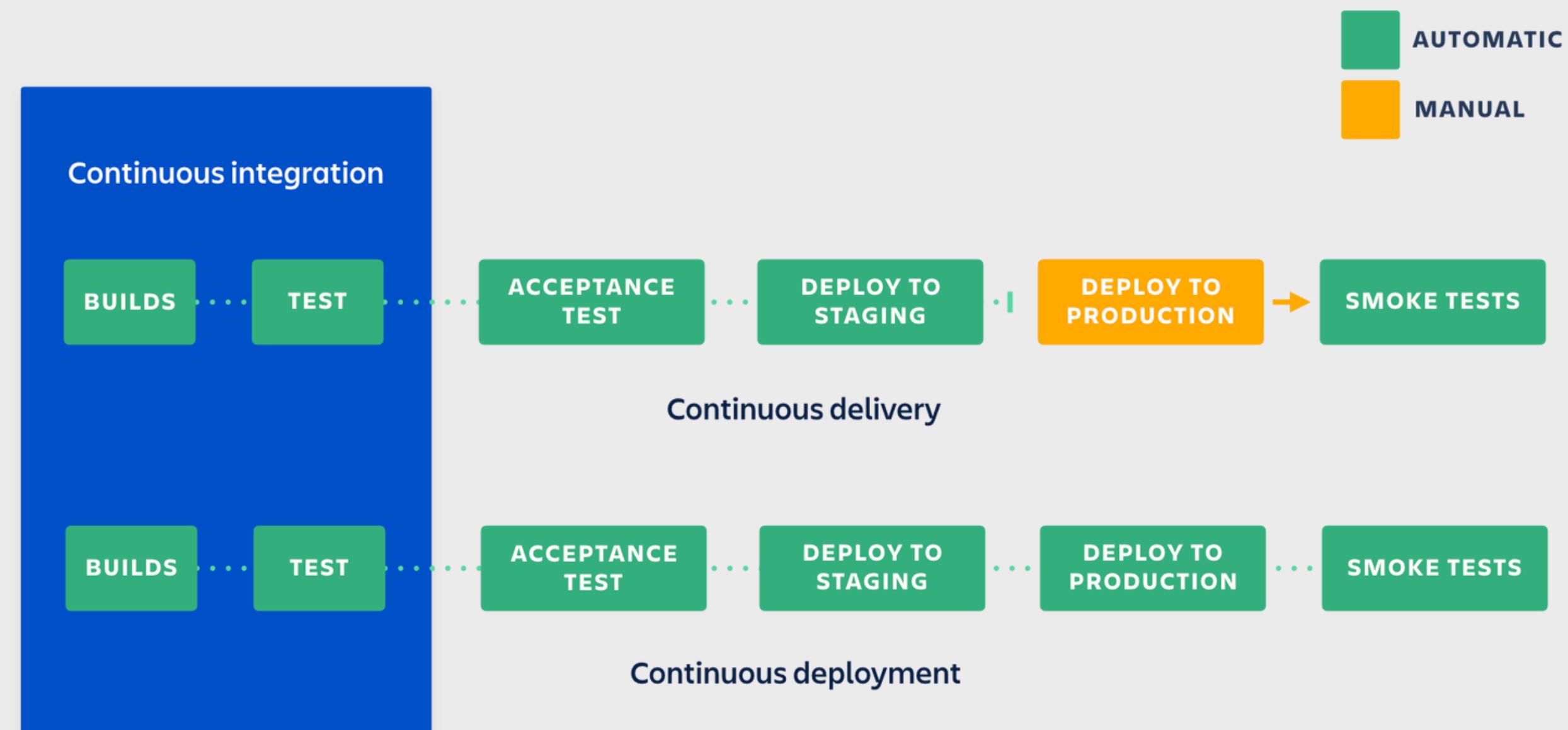
# Chapitre 4 : Comment versionner, construire et tester votre code

- Évaluation des investissements dans les tests automatisés
- Introduction au développement piloté par les tests (TDD)
- Avantages du TDD
- Applications pratiques du TDD
- Recommandations finales et bonnes pratiques



# Chapitre 4 : Comment versionner, construire et tester votre code

- Recommandations clés pour un développement efficace des logiciels
- Faire face aux défis de la collaboration
- Transition vers l'intégration et la livraison continues
- Implication de l'intégrité des engagements



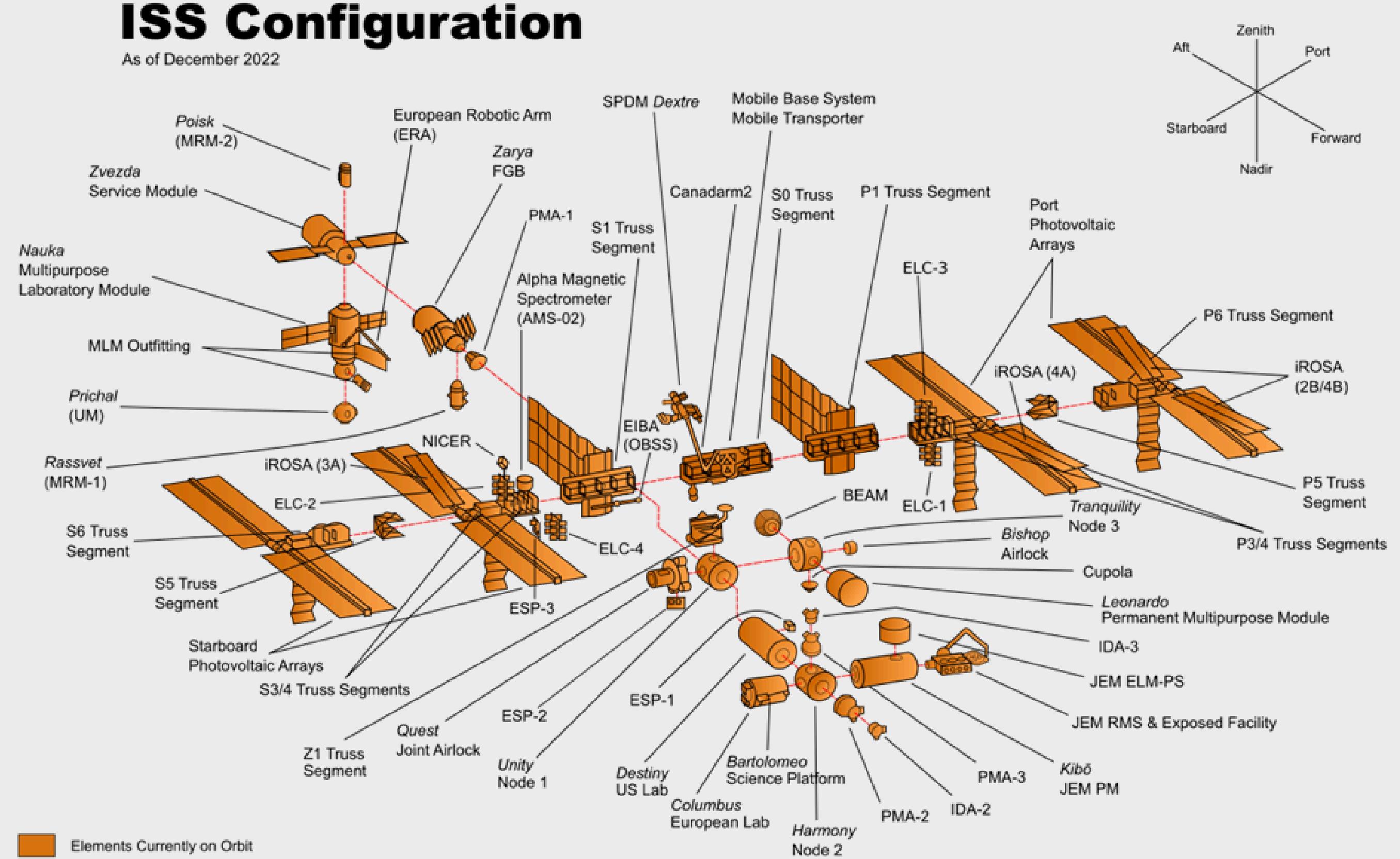


# Chapter V:

# How to Set Up Continuous Integration (CI) and Continuous Delivery (CD)

# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

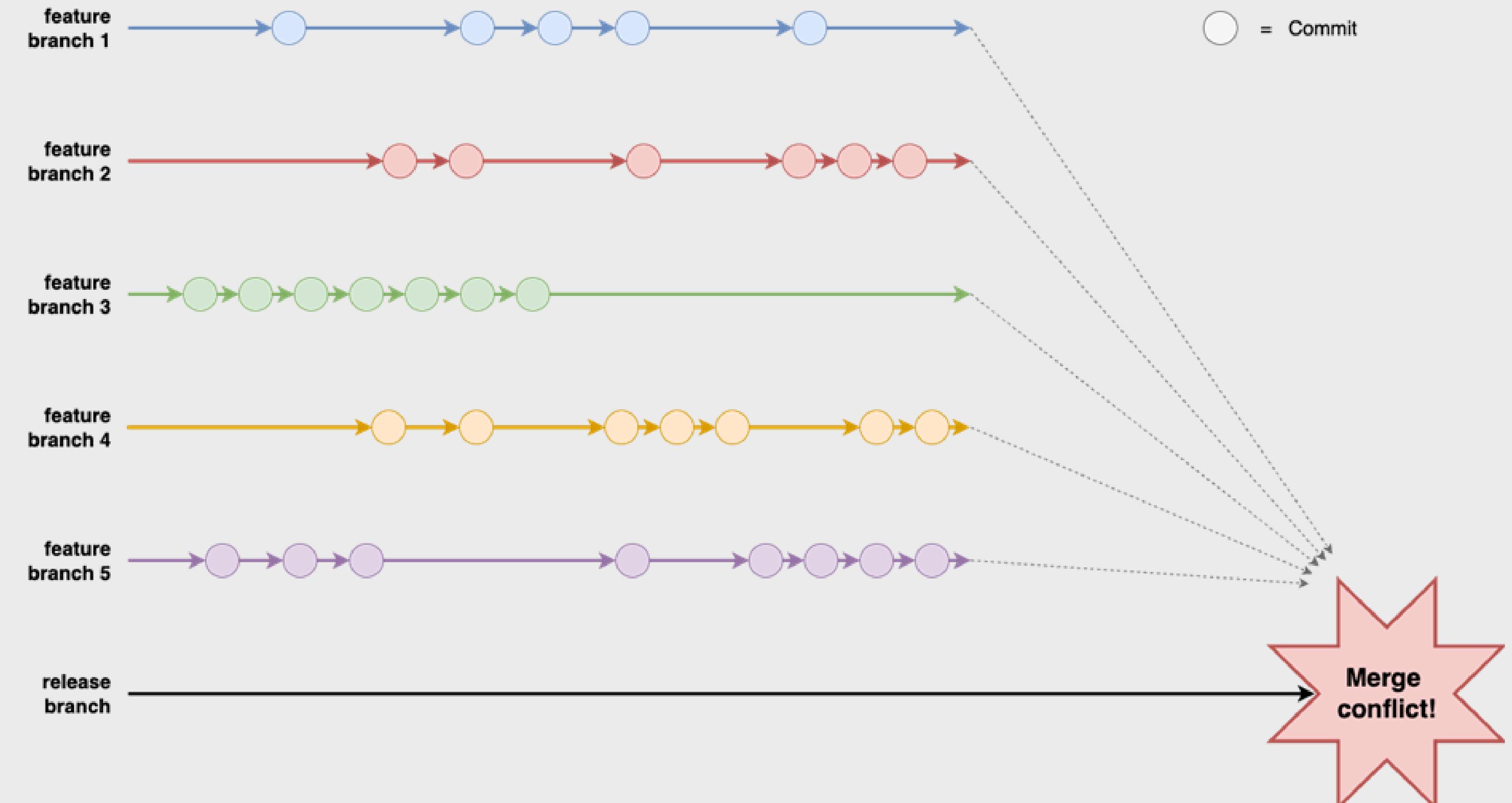
Figure 5-1. The International Space Station. Image from Wikipedia.



- Introduction à CI/CD
- Analogie : La construction de la Station spatiale internationale (ISS)
- Intégration continue (CI)

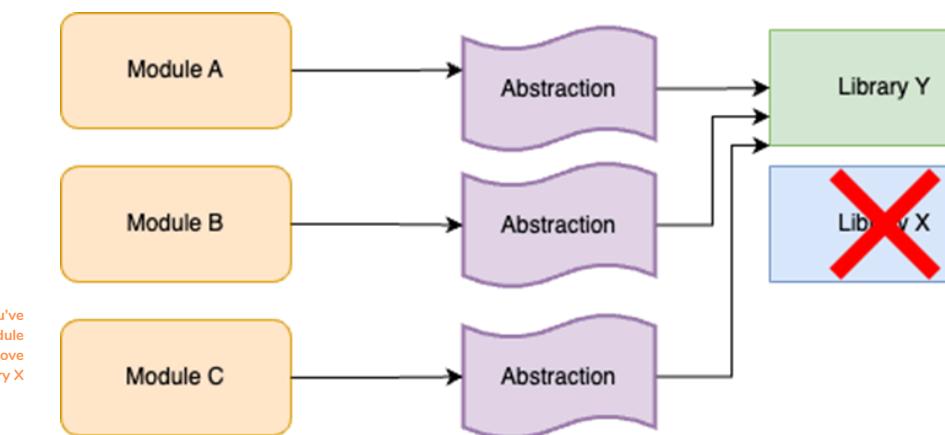
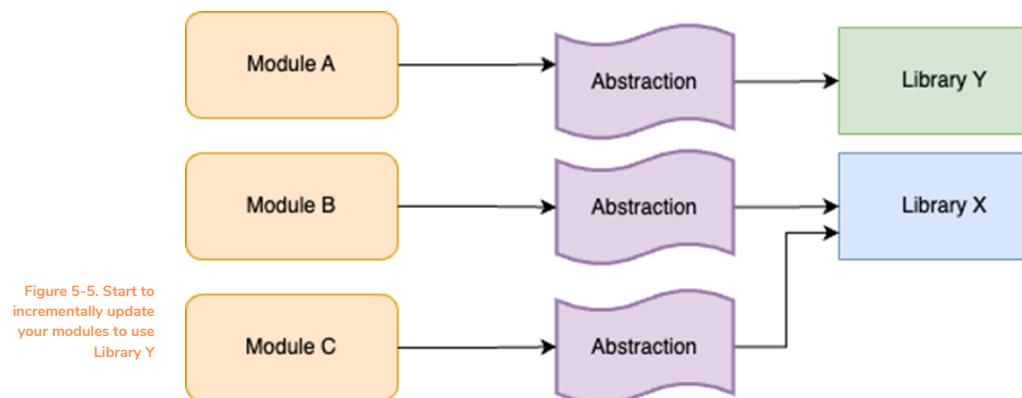
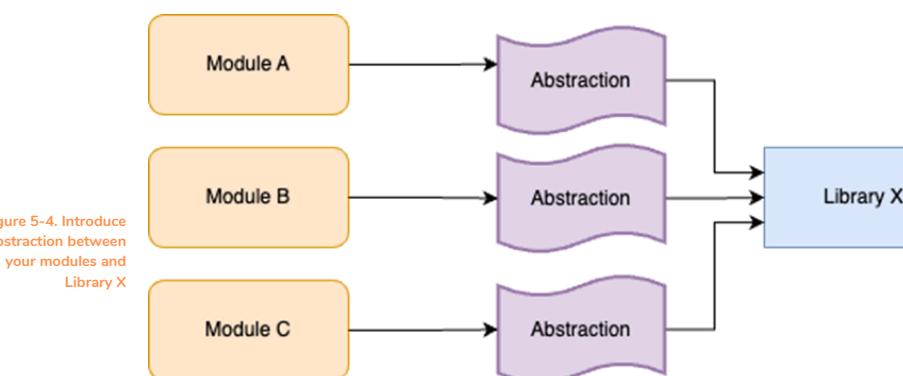
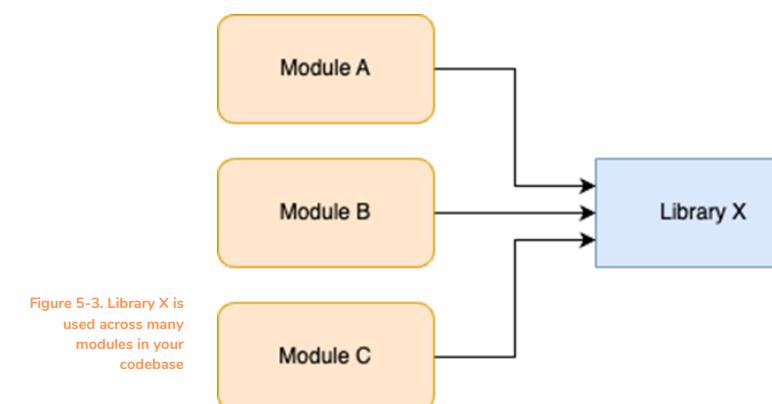
# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

Figure 5-2. The huge merge conflicts that you get as a result of late integration



- Gestion des conflits de fusion dans le cadre d'un développement basé sur un tronc d'arbre
- Prévention des ruptures de builds avec des builds auto-testées
- Gestion des modifications à grande échelle

- Branch by Abstraction (Branche par abstraction)
- Bascules de fonctionnalités
- Exemple : Utiliser les actions GitHub pour la CI



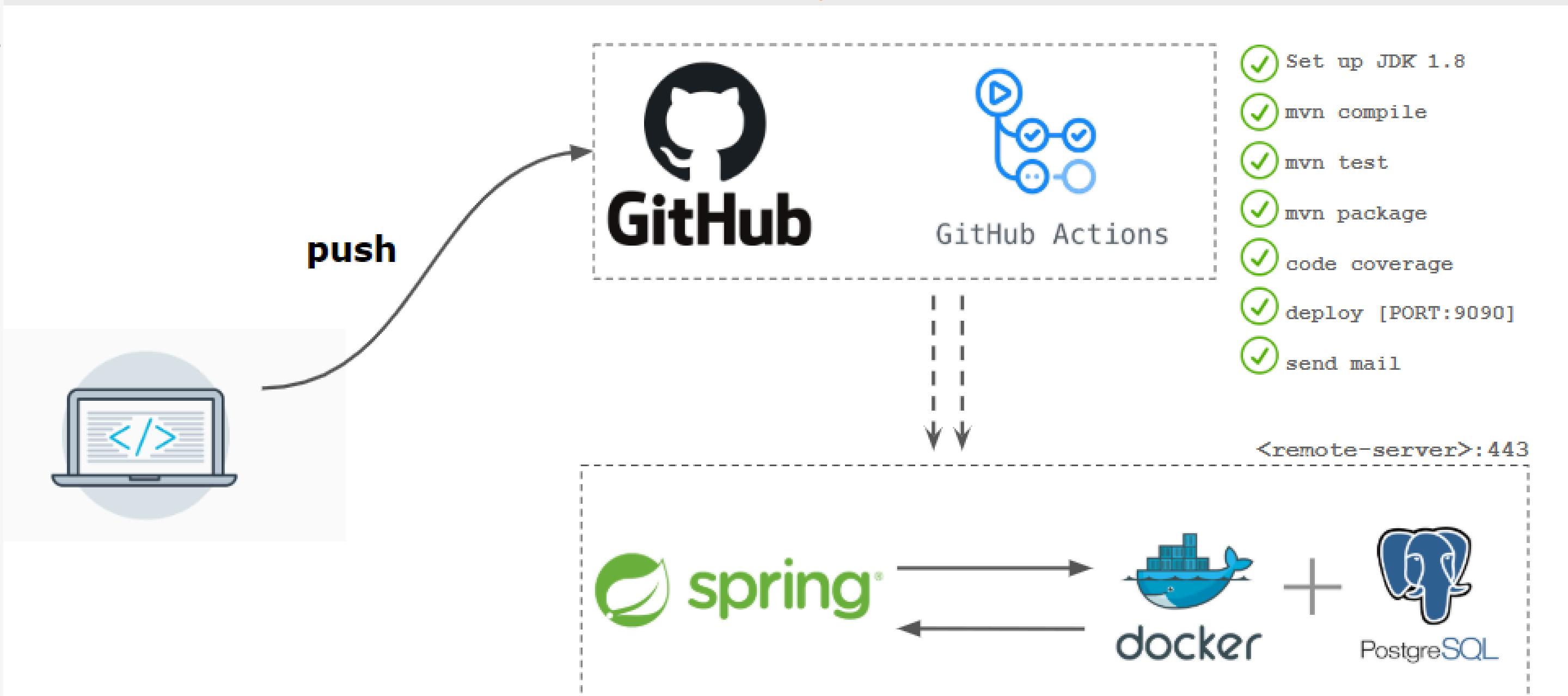
# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

- Mise en place d'un workflow d'actions GitHub

- Exécution du flux de travail

- L'automatisation de la CI en action

- Concepts clés



- Tests automatisés dans les Pull Requests

- Mise à jour et re-test

- Finalisation de la « Pull Request » (demande d'extraction)

- Meilleures pratiques et exercices

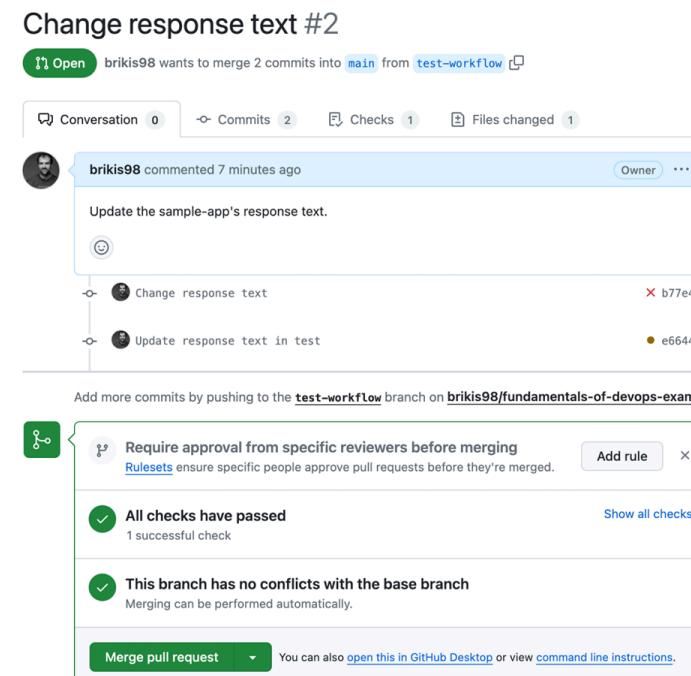
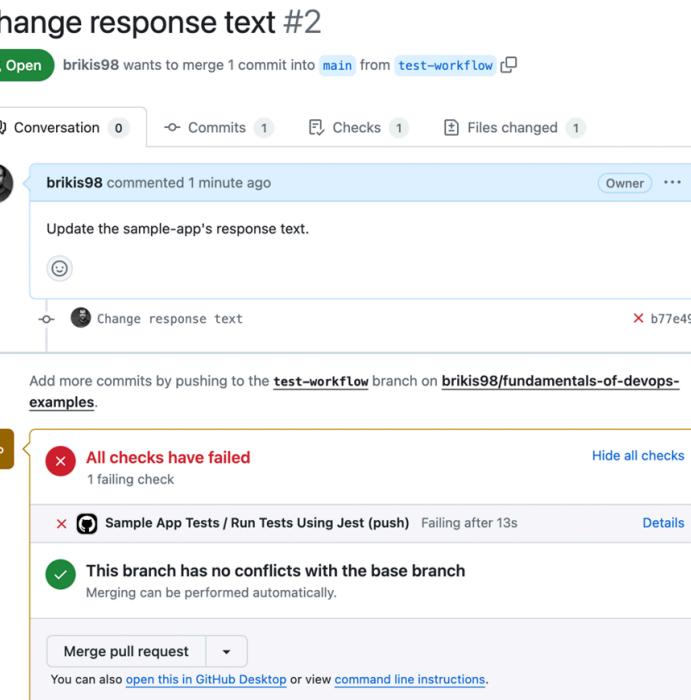
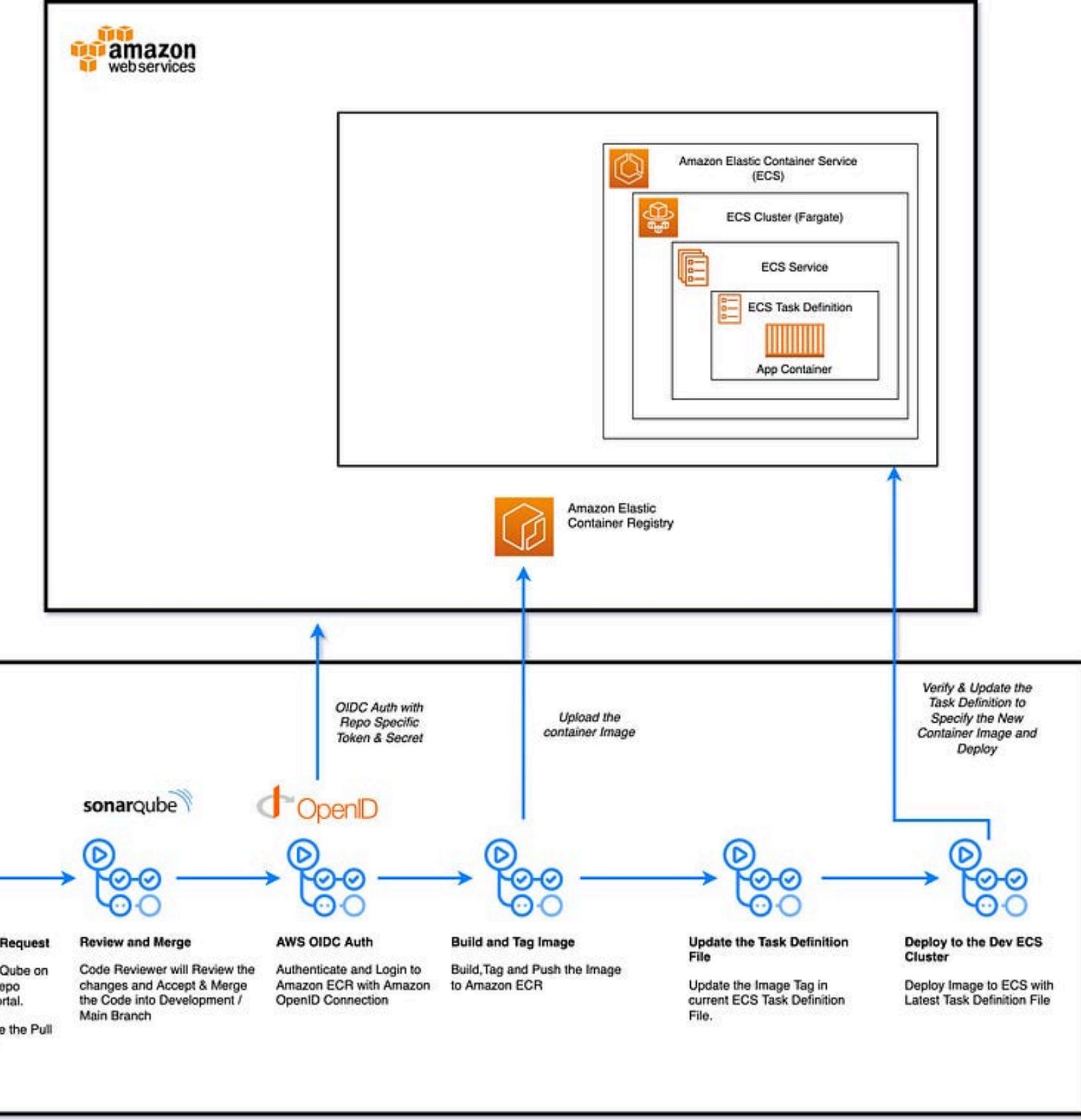


Figure 5-8. Looking into the cause of the test failure

```
Run Tests Using Jest
failed 2 minutes ago in 11s
Search logs

Run tests
15 FAIL ./app.test.js
16   Test the app
17     x Get / should return Hello, World! (23 ms)
18
19   ● Test the app > Get / should return Hello, World!
20
21     expect(received).toBe(expected) // Object.is equality
22
23     Expected: "Hello, World!"
24     Received: "Ready for Production!"
25
26       6 |   const response = await request(app).get('/');
27       7 |   expect(response.statusCode).toBe(200);
28     >  8 |   expect(response.text).toBe('Hello, World!');
29
30     9 |
31   10 | });
32     11 |
33
34   at Object.toBe (app.test.js:8:27)
```

# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?



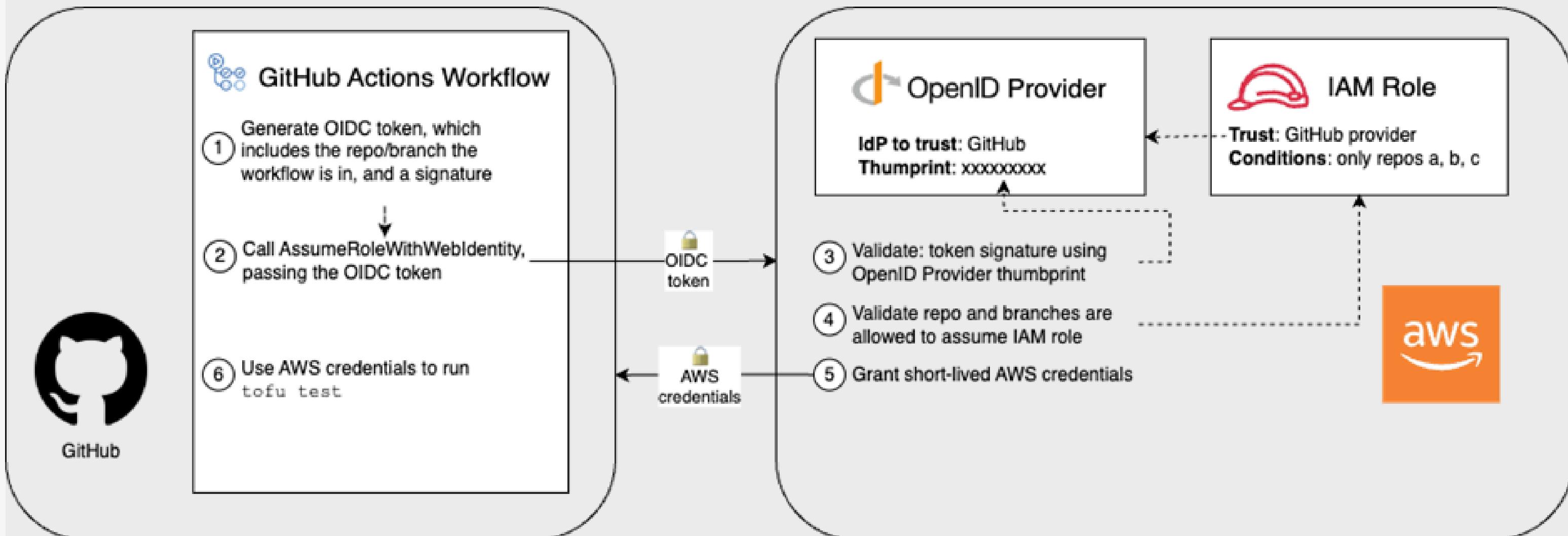
- Introduction à l'authentification CI/CD
- Utiliser des méthodes d'authentification sécurisées
- Informations d'identification de l'utilisateur de la machine
- Informations d'identification fournies automatiquement
- Exemple de flux de travail avec OIDC
- Application pratique

# Chapitre 5 :

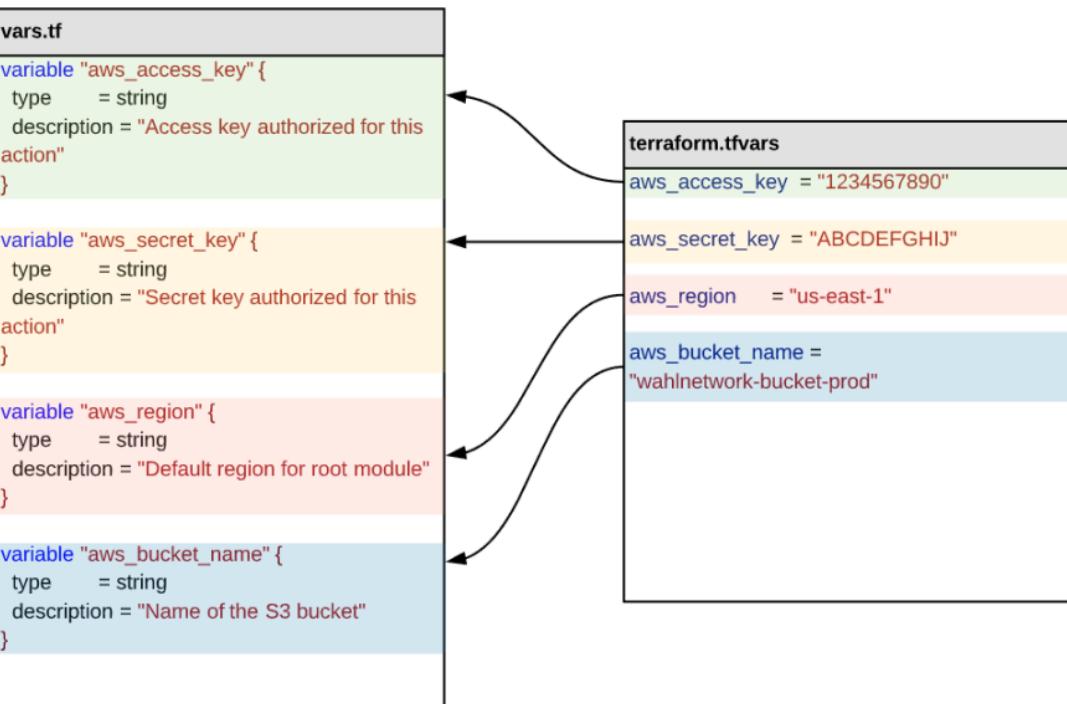
## Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

Figure 5-10. With OIDC, you configure AWS to trust an IdP such as GitHub, which allows that IdP to exchange an OIDC token for short-lived AWS credentials

- Configuration du fournisseur OIDC
- Configuration des rôles IAM
  - Exécution de tests automatisés de l'infrastructure
  - Avantages de cette configuration



- Définir et utiliser des variables d'entrée dans les modules Terraform
- Création d'un flux d'actions GitHub pour les tests d'infrastructure
- Avantages des variables dynamiques et des tests automatisés



### Add infrastructure tests #5

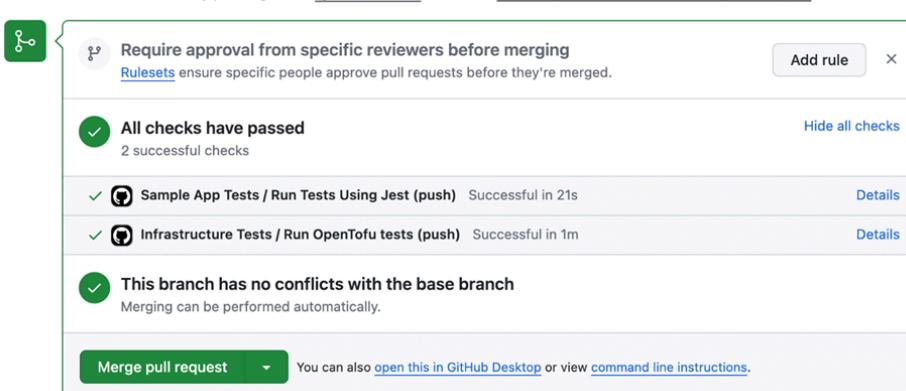
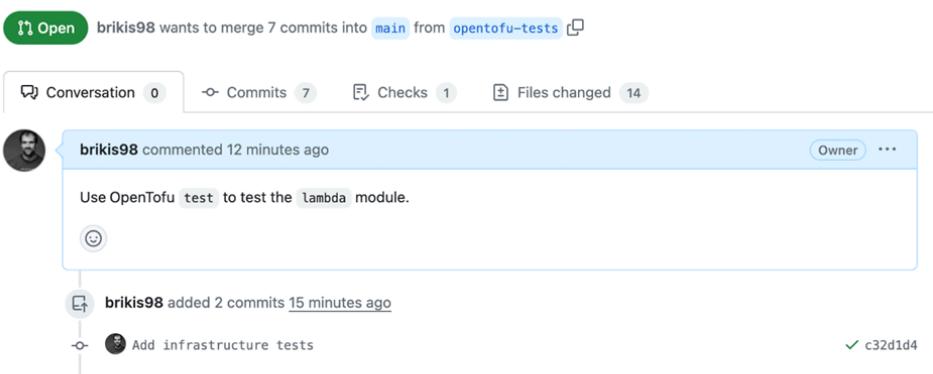


Figure 5-11. A PR showing the sample app and OpenTofu tests running

• Livraison continue (CD) : Vue d'ensemble

- Stratégies de déploiement
- Principaux enseignements

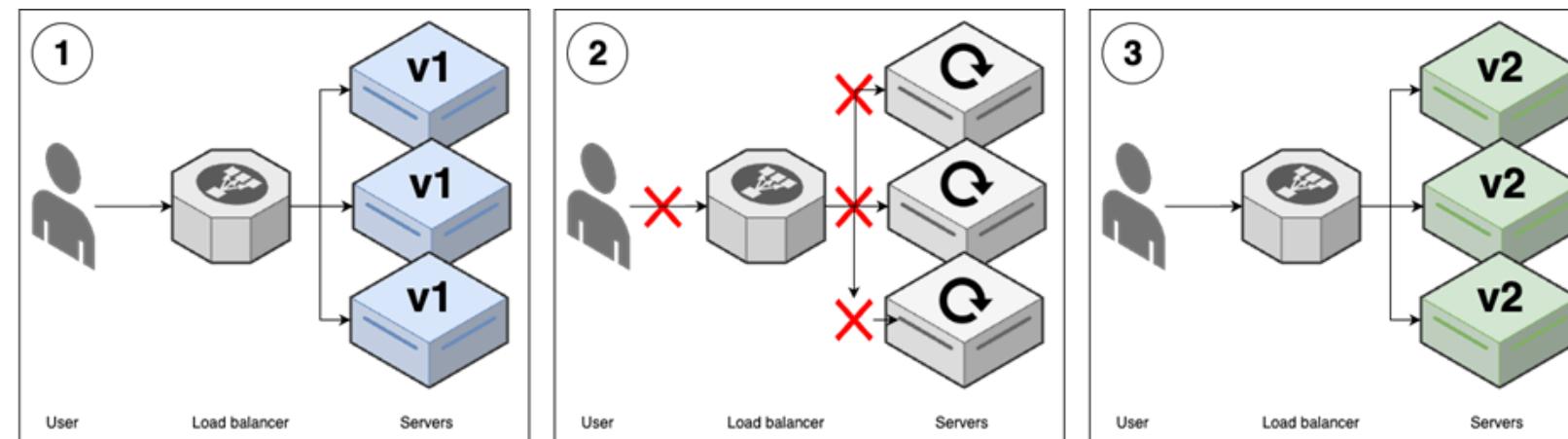


Figure 5-12. Downtime deployment

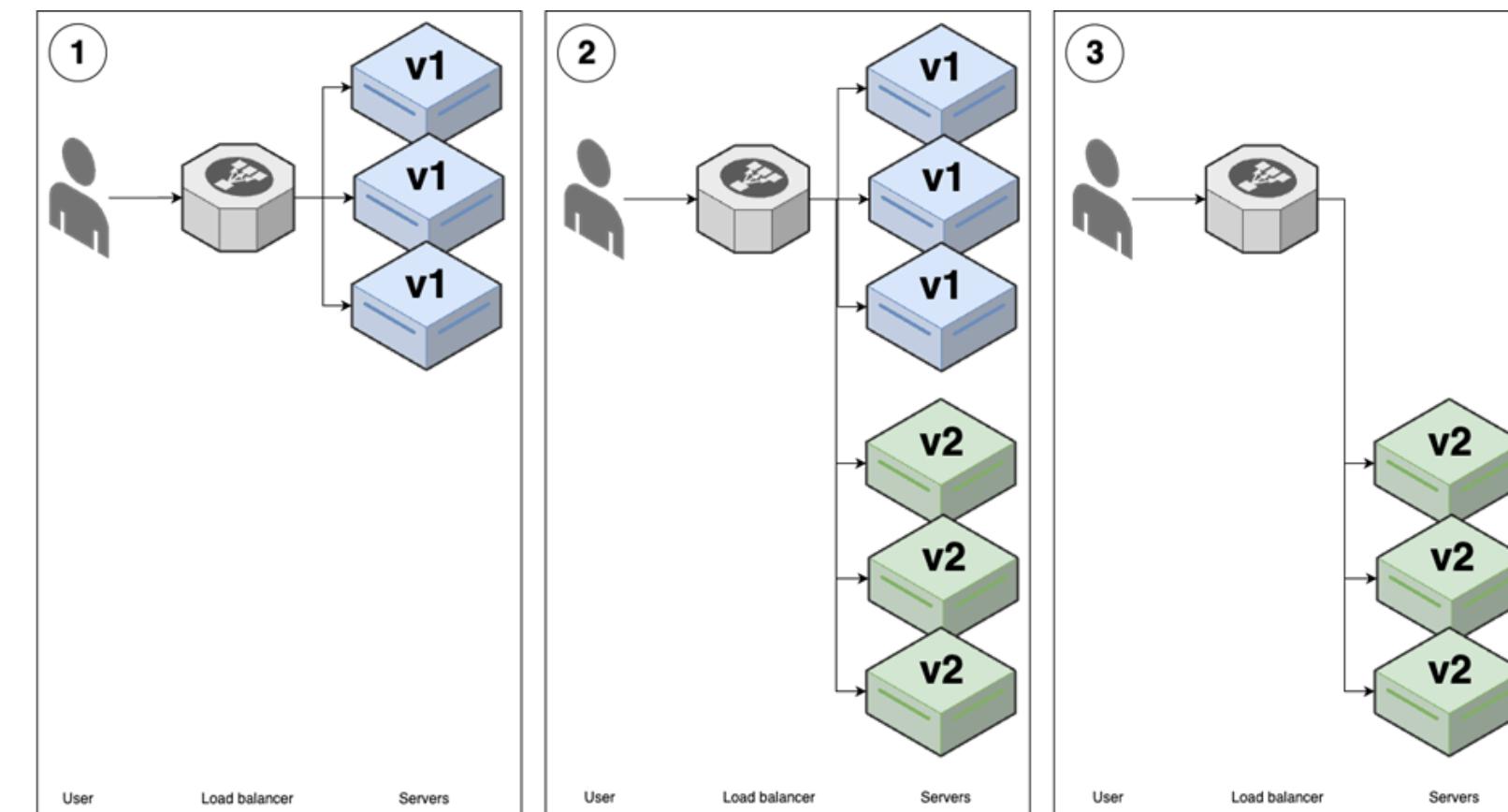


Figure 5-13. Rolling deployment without replacement

- Déploiement progressif avec remplacement

- Déploiement bleu-vert

- Comparaison des stratégies de déploiement

- Recommandations

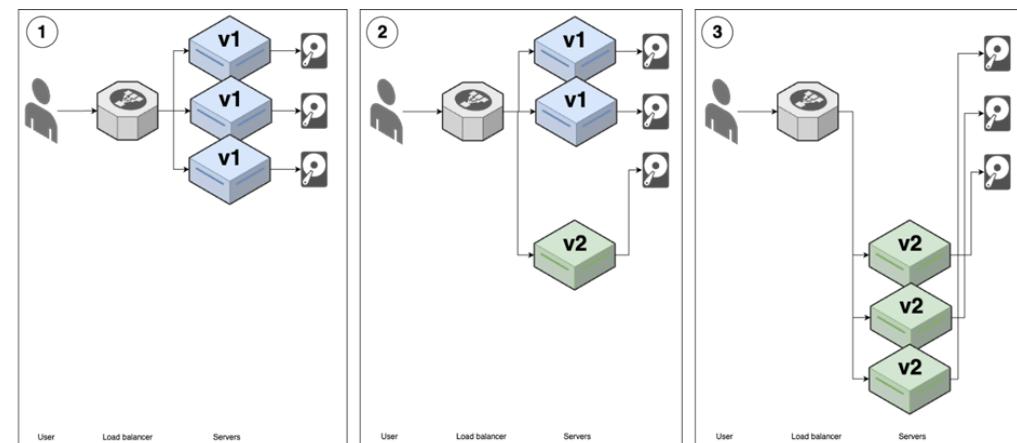


Figure 5-14. Rolling deployment with replacement

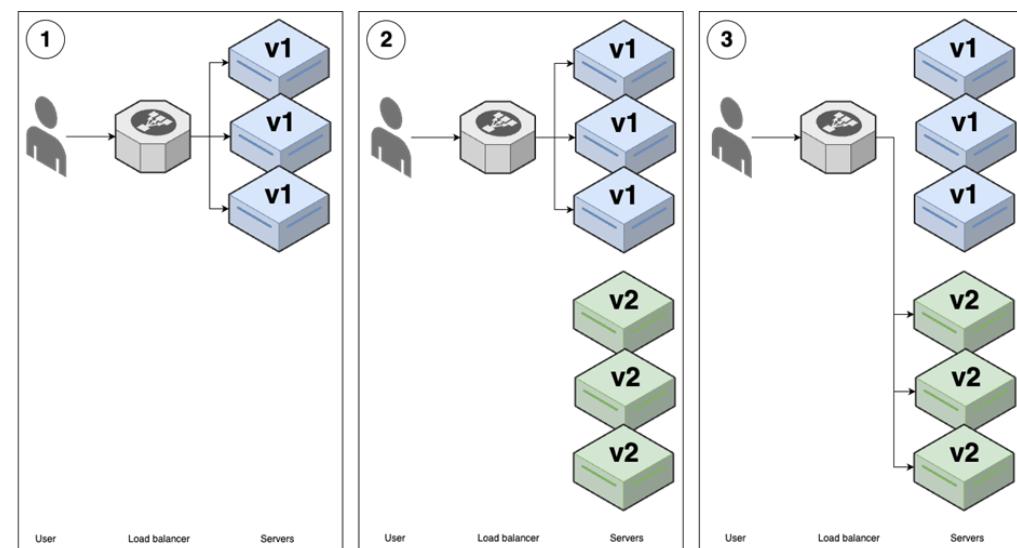


Figure 5-15. Blue-green deployment

Strategy	User experience	Stateless apps	Stateful apps	Support
Downtime deployment	Downtime	Supported	Supported	Widely supported
Rolling without replacement	Alternates between versions	Supported	Not supported	Widely supported
Rolling with replacement	Alternates between versions	Supported	Supported	Widely supported
Blue-green	Instantaneous switchover	Supported	Not supported	Limited support

Table 5-1. A comparison of primary deployment strategies

- Déploiement du canari
- Déploiement de la bascule de fonction
- Déploiement de la promotion
- Comparaison des stratégies d'extension

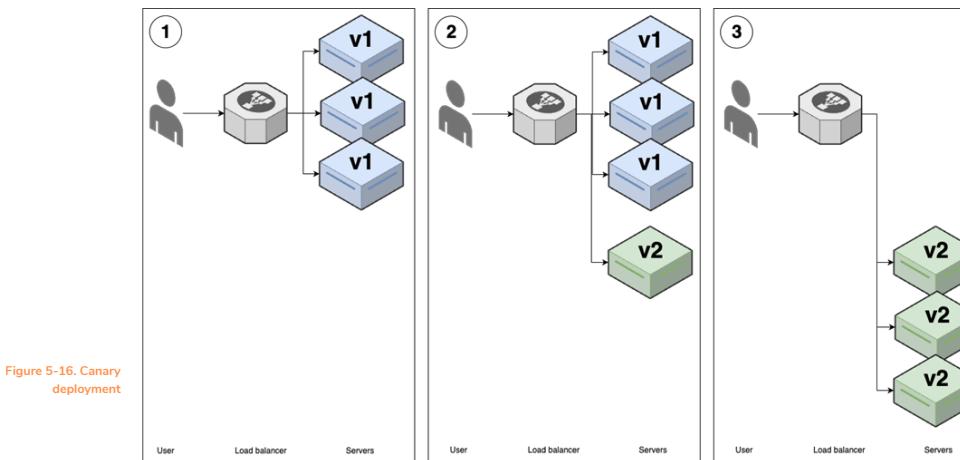


Figure 5-16. Canary deployment

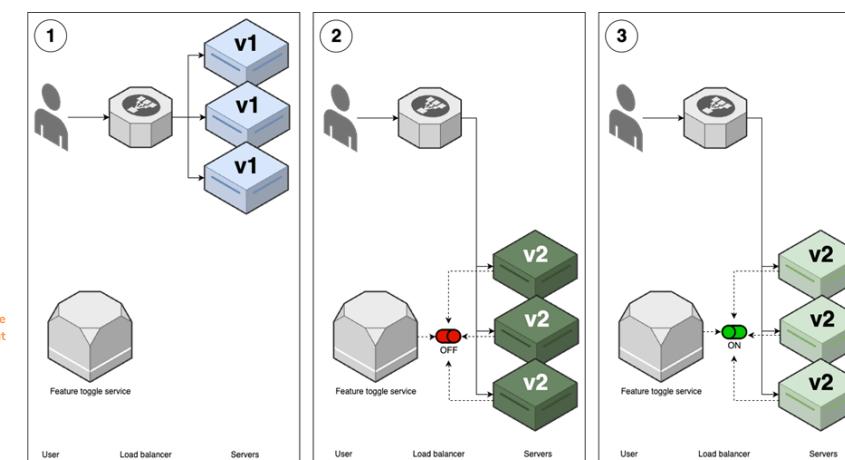


Figure 5-17. Feature toggle deployment

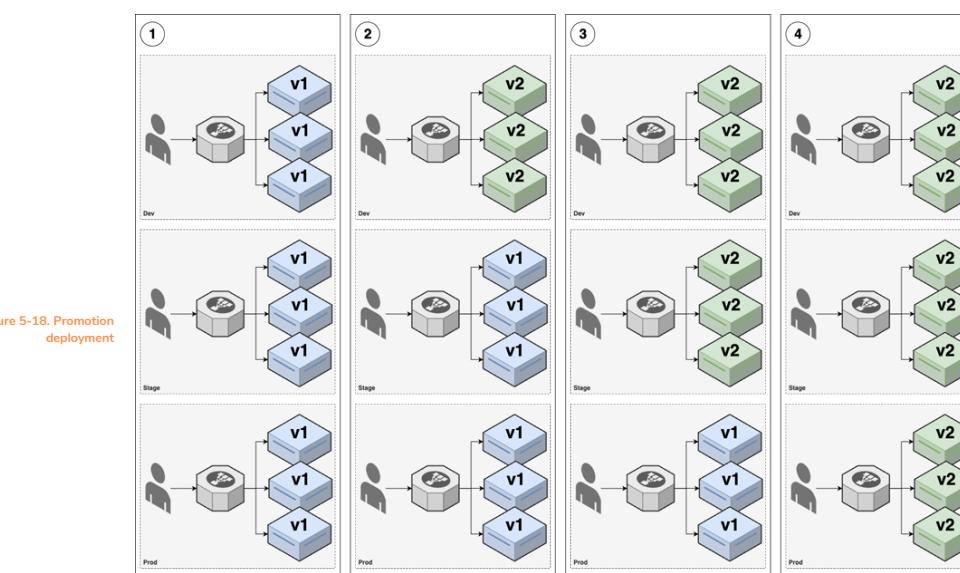
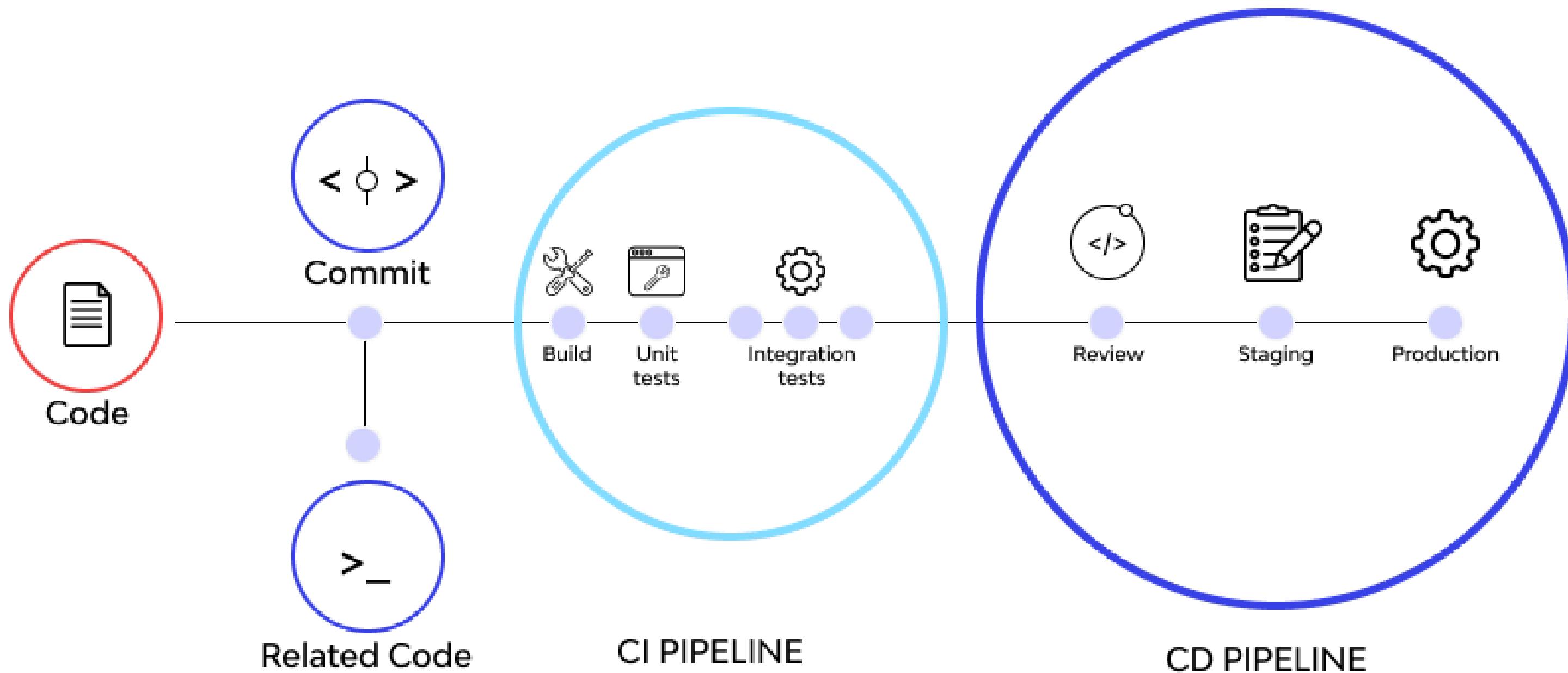


Figure 5-18. Promotion deployment

# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

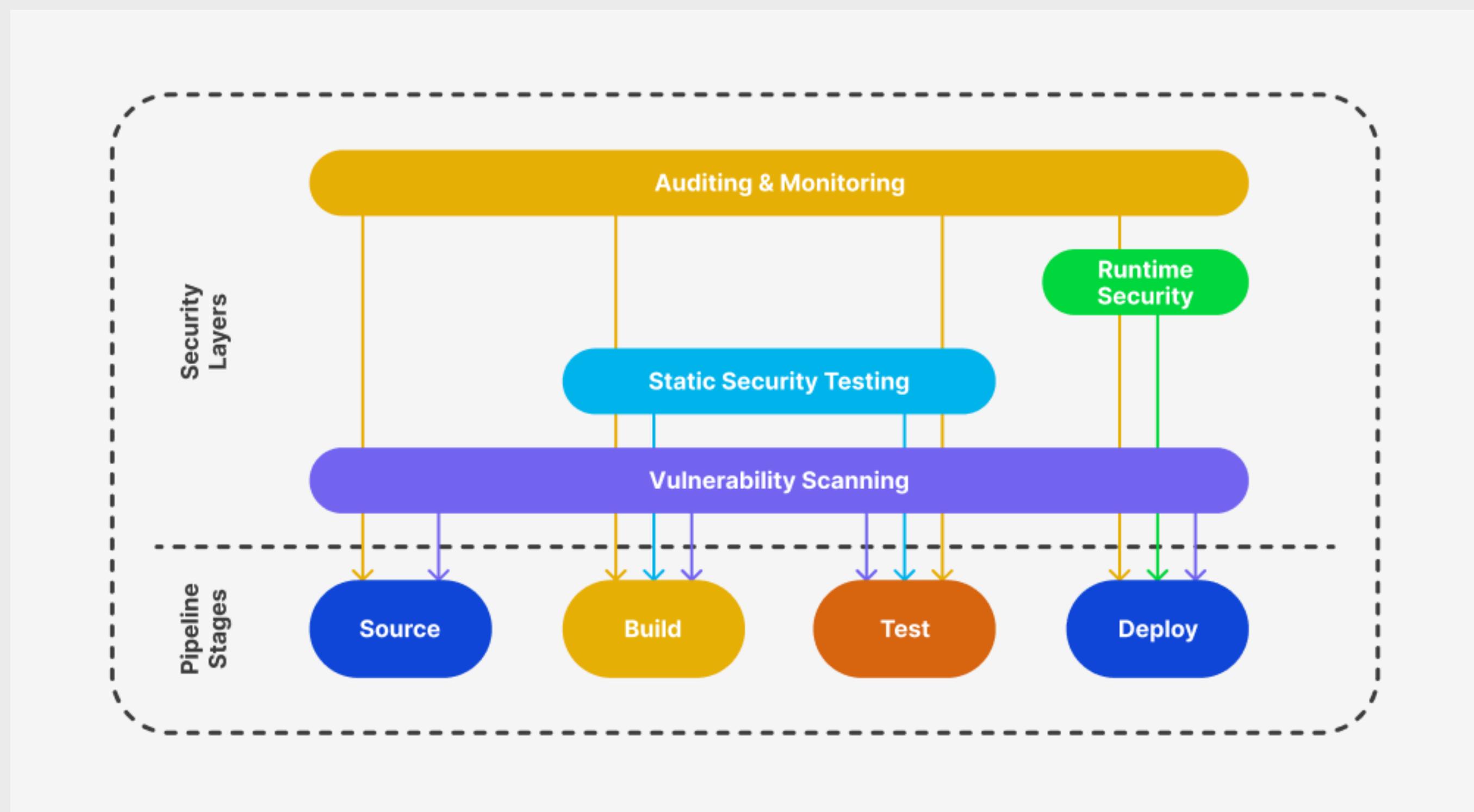
- Résoudre les problèmes sans nouveaux déploiements
- Pipelines de déploiement
- Considérations relatives aux pipelines de déploiement

## CI/CD pipeline



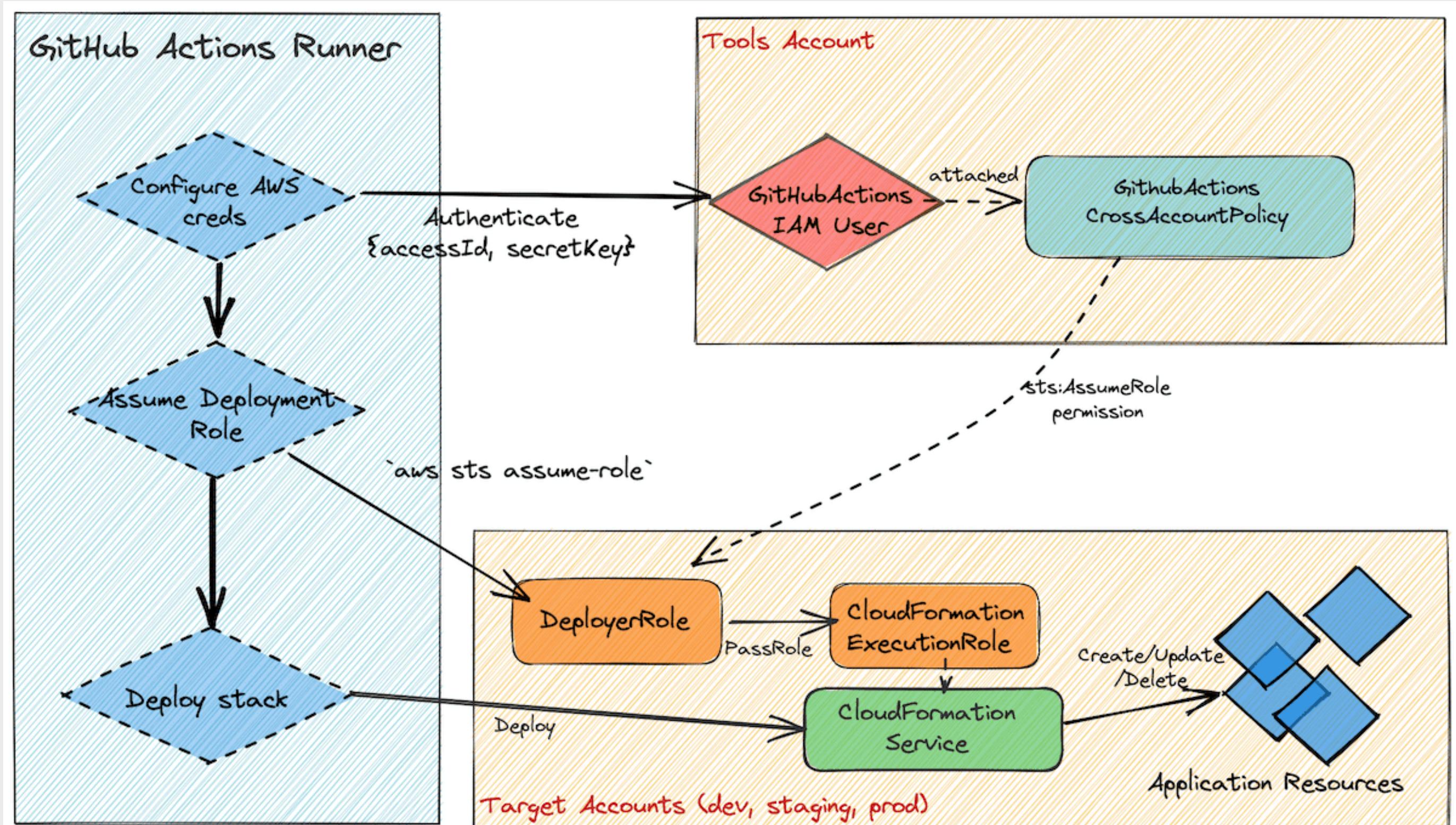
# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

- Sécurisation des pipelines de déploiement
- Défis liés au stockage d'états locaux
- Avantages de l'utilisation d'Amazon S3 comme backend distant
- Étapes de mise en œuvre d'un backend distant



# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

- Configuration de l'état à distance avec OpenTofu
- Configuration de Remote State avec S3 et DynamoDB
- Etapes de l'intégration de Remote State
- Exemple : Ajout de rôles IAM pour les déploiements d'infrastructure
- Avantages de ces configurations

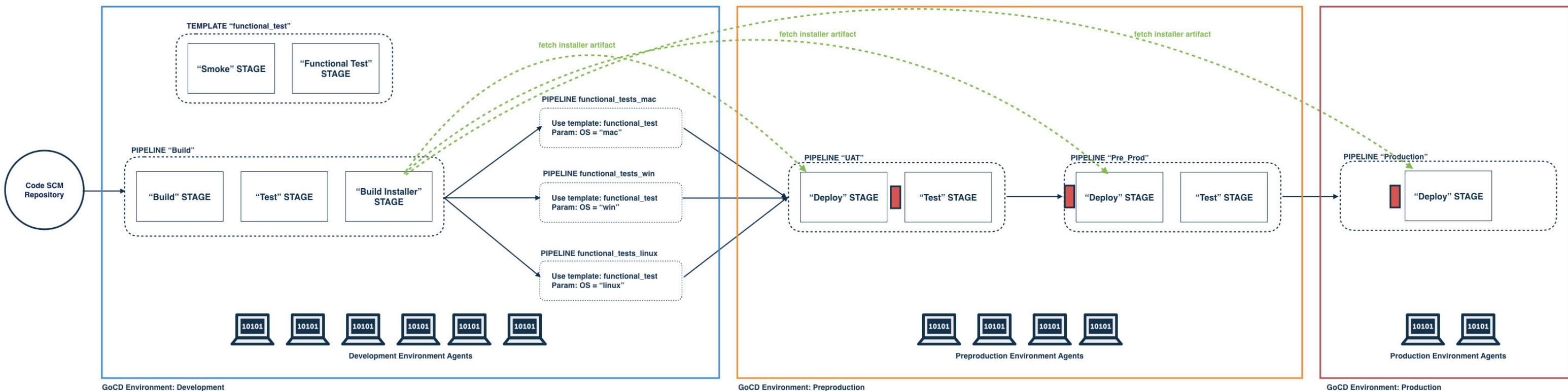


**Chapitre 5 :**  
**Comment**  
**mettre en place**  
**l'intégration**  
**continue (CI) et**  
**la livraison**  
**continue (CD) ?**

Auteur : Badr TAJINI

120

- Activation des rôles IAM pour les opérations de planification et d'application
- Définition d'un pipeline pour les déploiements d'infrastructure
- Principaux avantages de cette configuration



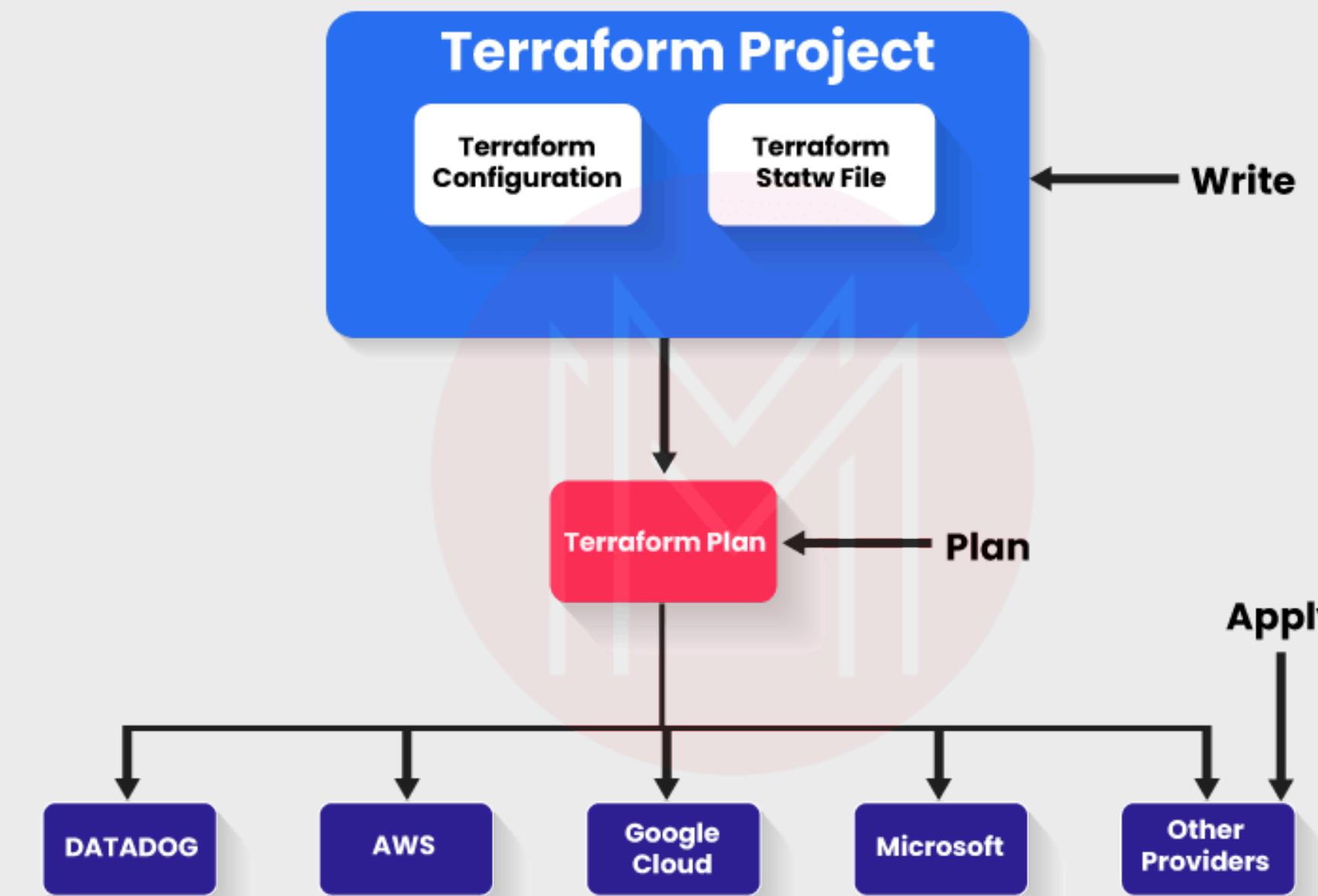
# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

- Workflow pour le plan d'infrastructure

- Workflow pour l'application de l'infrastructure

- Distinctions entre les flux de travail de planification et d'application

- Mise en œuvre pratique



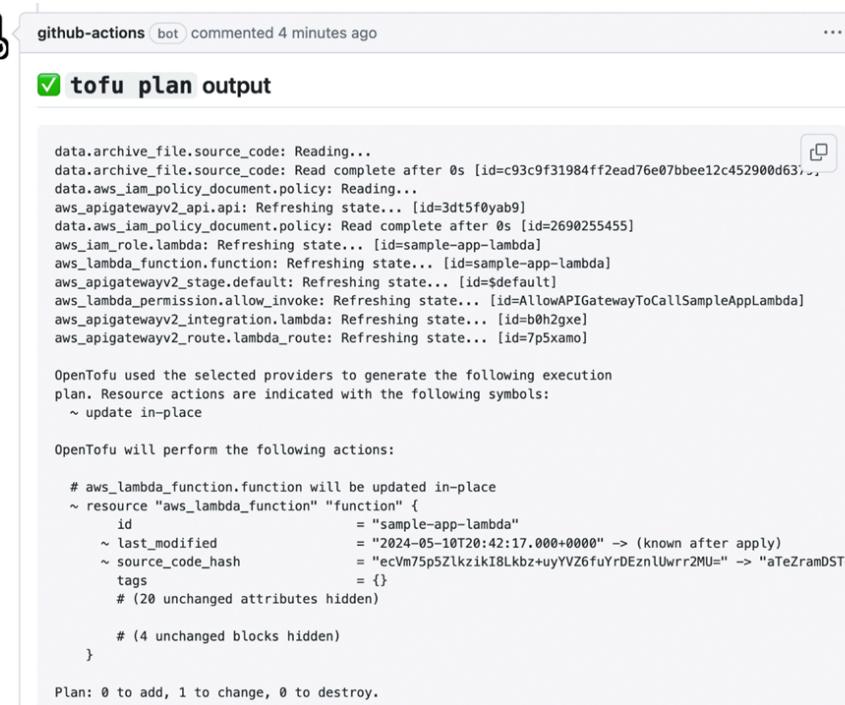
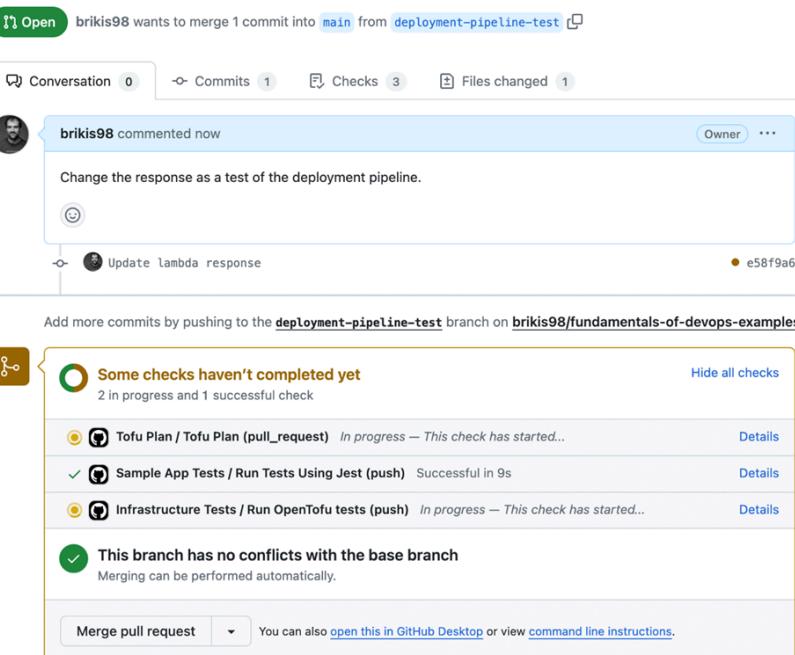
**Chapitre 5 :**  
Comment  
mettre en place  
l'intégration  
continue (CI) et  
la livraison  
continue (CD) ?

Auteur : Badr TAJINI

122

- Mise en œuvre et test des modifications du code
- Workflow du pipeline de déploiement en action
- Révision et application des modifications
- Principaux avantages du pipeline de déploiement

### Update lambda response #12



# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

Figure 5-21. The pipeline adds a comment with the apply output to merged PRs

Merged

Update lambda response #12  
brikis98 merged 2 commits into main from deployment-pipeline-test 1 minute ago

brikis98 merged commit 5b8f4c3 into main 1 minute ago  
View details Revert  
4 checks passed

github-actions bot commented now

**tofu apply output**

```
data.archive_file.source_code: Reading...
data.archive_file.source_code: Read complete after 0s [id=c93c9f31984ff2ead76e07bbe12c452900d63]...
data.aws_iam_policy_document.policy: Reading...
aws_apigatewayv2_api.api: Refreshing state... [id=3dt5f0yab9]
data.aws_iam_policy_document.policy: Read complete after 0s [id=2690255455]
aws_iam_role.lambda: Refreshing state... [id=sample-app-lambda]
aws_apigatewayv2_stage.default: Refreshing state... [id=$default]
aws_lambda_function.function: Refreshing state... [id=sample-app-lambda]
aws_lambda_permission.allow_invoke: Refreshing state... [id=AllowAPIGatewayToCallSampleAppLambda]
aws_apigatewayv2_integration.lambda: Refreshing state... [id=b0h2gxe]
aws_apigatewayv2_route.lambda_route: Refreshing state... [id=7p5xamo]
```

OpenTofu used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:  
~ update in-place

OpenTofu will perform the following actions:

```
# aws_lambda_function.function will be updated in-place
~ resource "aws_lambda_function" "function" {
    id                      = "sample-app-lambda"
    ~ last_modified          = "2024-05-10T20:42:17.000+0000" -> (known after apply)
    ~ source_code_hash        = "ecVm75p5ZlkzikI8Lkbz+uyYVZ6fuYrDEznLUwrr2MU=" -> "aTeZramDST"
    tags                     = {}
    # (20 unchanged attributes hidden)

    # (4 unchanged blocks hidden)
}
```

Plan: 0 to add, 1 to change, 0 to destroy.

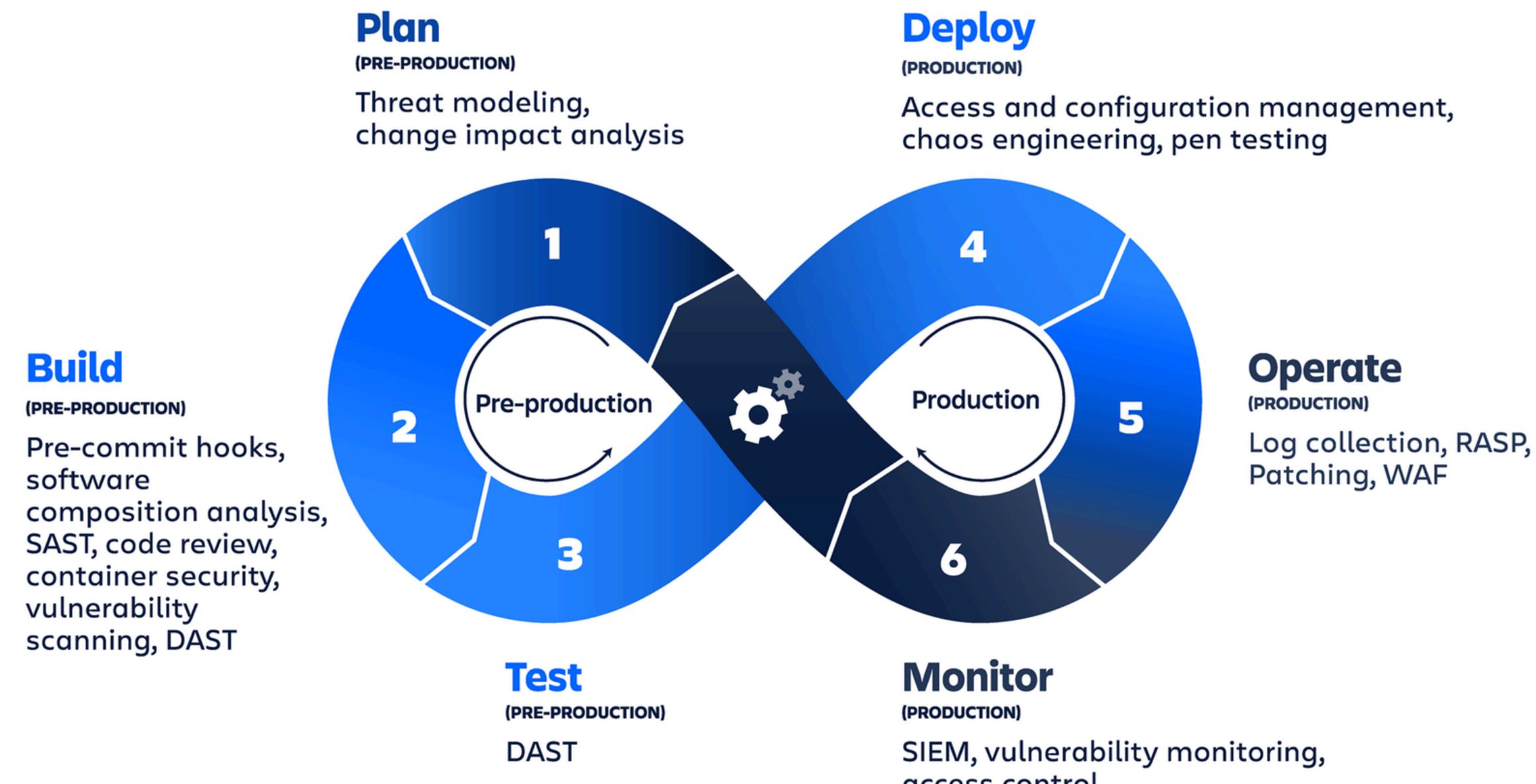
```
aws_lambda_function.function: Modifying... [id=sample-app-lambda]
aws_lambda_function.function: Modifications complete after 6s [id=sample-app-lambda]
```

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

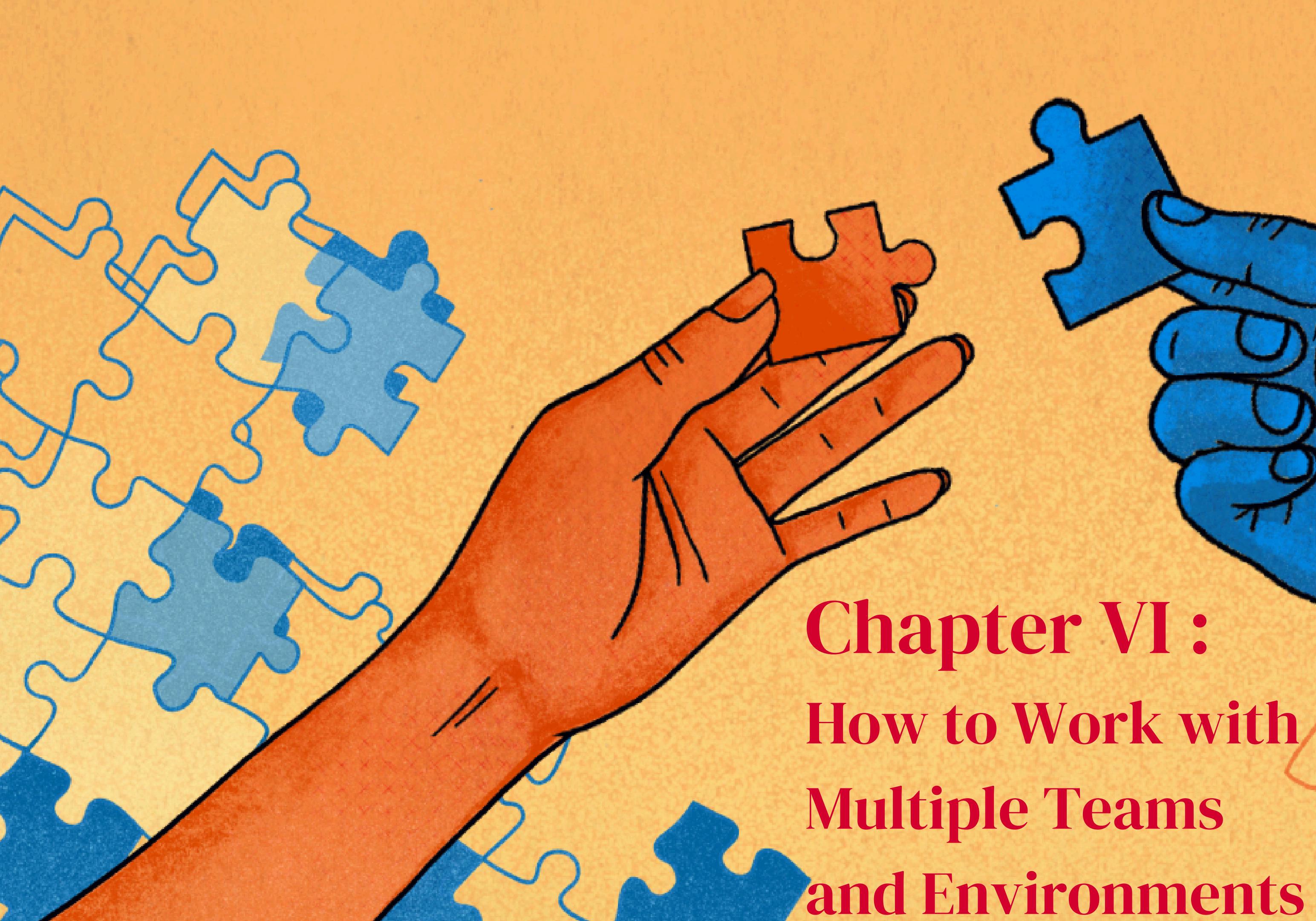
- Aperçu du pipeline de déploiement
- Recommandations pour des pipelines efficaces
- Considérations de sécurité avancée

# Chapitre 5 : Comment mettre en place l'intégration continue (CI) et la livraison continue (CD) ?

## DevSecOps



- Modulariser les pipelines de déploiement
- Construire des pipelines sécurisés et fiables
- Principaux enseignements pour l'adoption de CI/CD
- Relation entre agilité et sécurité



## Chapter VI :

# How to Work with Multiple Teams and Environments

# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

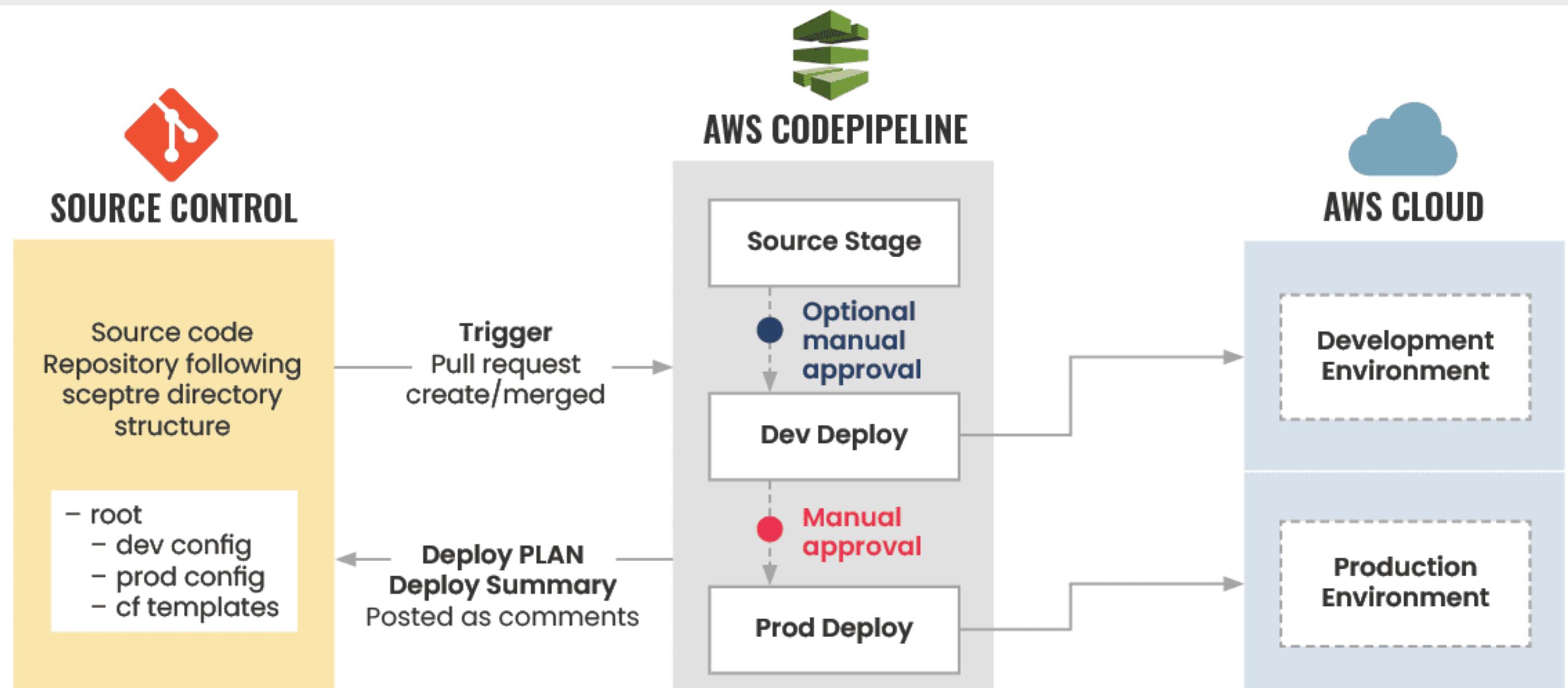
- Introduction à la mise à l'échelle de plusieurs équipes et environnements
- Avantages des environnements de déploiement multiples
- Isolement des tests dans les différents environnements
- Réduction de la latence
- Isolation des produits et des équipes
- Principaux enseignements

Table 6-1. Typical latency numbers of common computer operations<sup>a</sup>

Operation	Time in ns
Random read from CPU cache (L1)	1
Random read from main memory (DRAM)	100
Compress 1 kB with Snappy	2,000
Read 1 MB sequentially from DRAM	3,000
Random read from solid state disk (SSD)	16,000
Read 1 MB sequentially from SSD	49,000
TCP packet round trip within same datacenter	500,000
Random read from rotational disk	2,000,000
Read 1 MB sequentially from rotational disk	5,000,000
TCP packet round trip from California to New York	40,000,000
TCP packet round trip from California to Australia	183,000,000

# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

- Gestion de la latence dans les déploiements mondiaux
- Conformité réglementaire et environnements dédiés
- Augmentation de la résilience avec les environnements distribués
- Définition et mise en place d'environnements multiples
- Défis posés par les environnements multiples



## Chapitre 6 :

# Comment travailler avec des équipes et des environnements multiples ?

- Complexité accrue de la configuration des applications
- Causes des pannes : Changements de configuration et changements binaires
- Gestion efficace de la configuration
- Décomposition des déploiements pour l'évolutivité
- Configuration de plusieurs comptes AWS

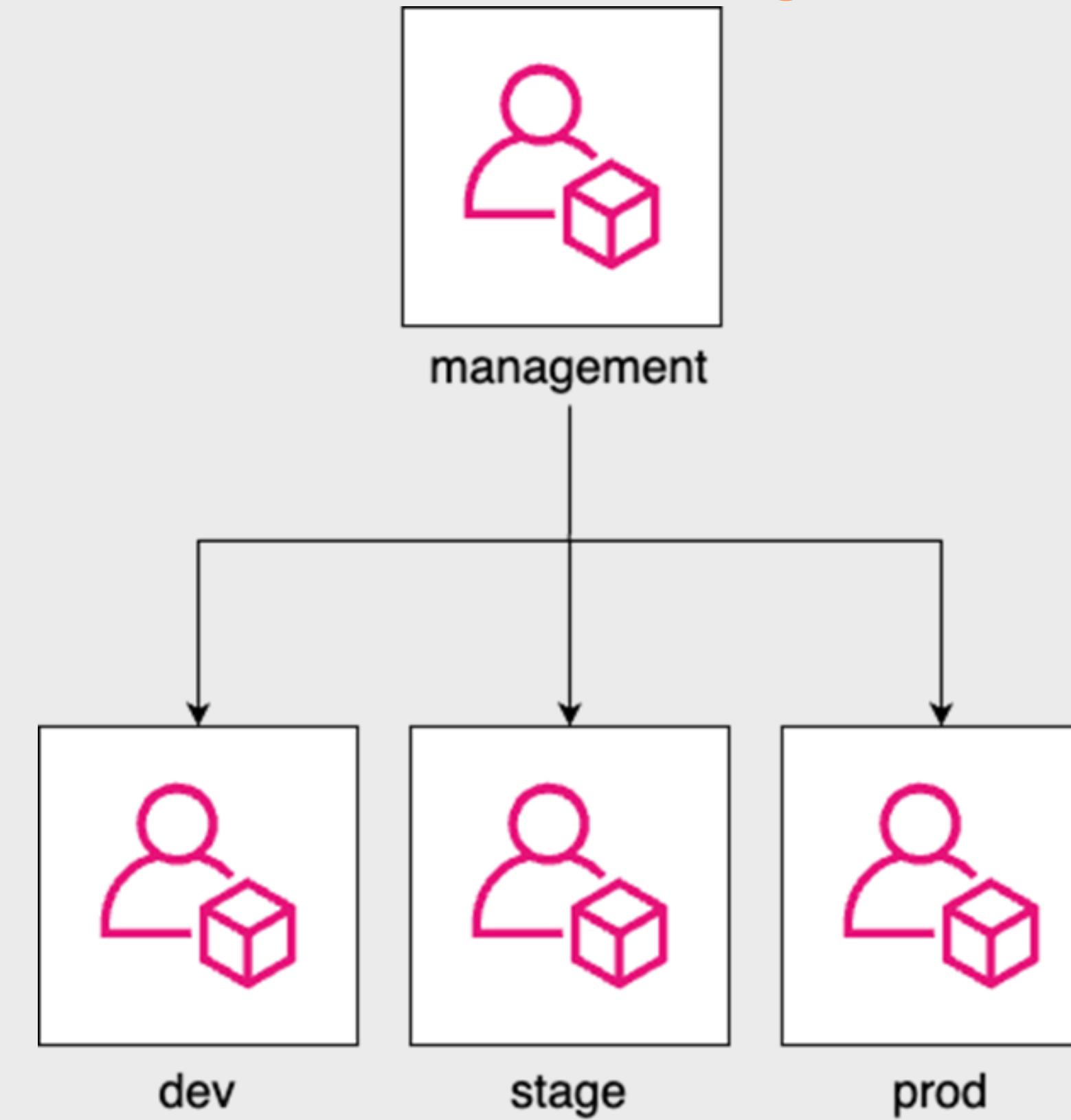
Table 6-2. Top causes of outages at Google, 2010-2017

Cause	Percent of outages
Binary push	37%
Configuration push	31%
User behavior change	9%
Processing pipeline	6%
Service provider change	5%
Performance decay	5%
Capacity management	5%
Hardware	2%

# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

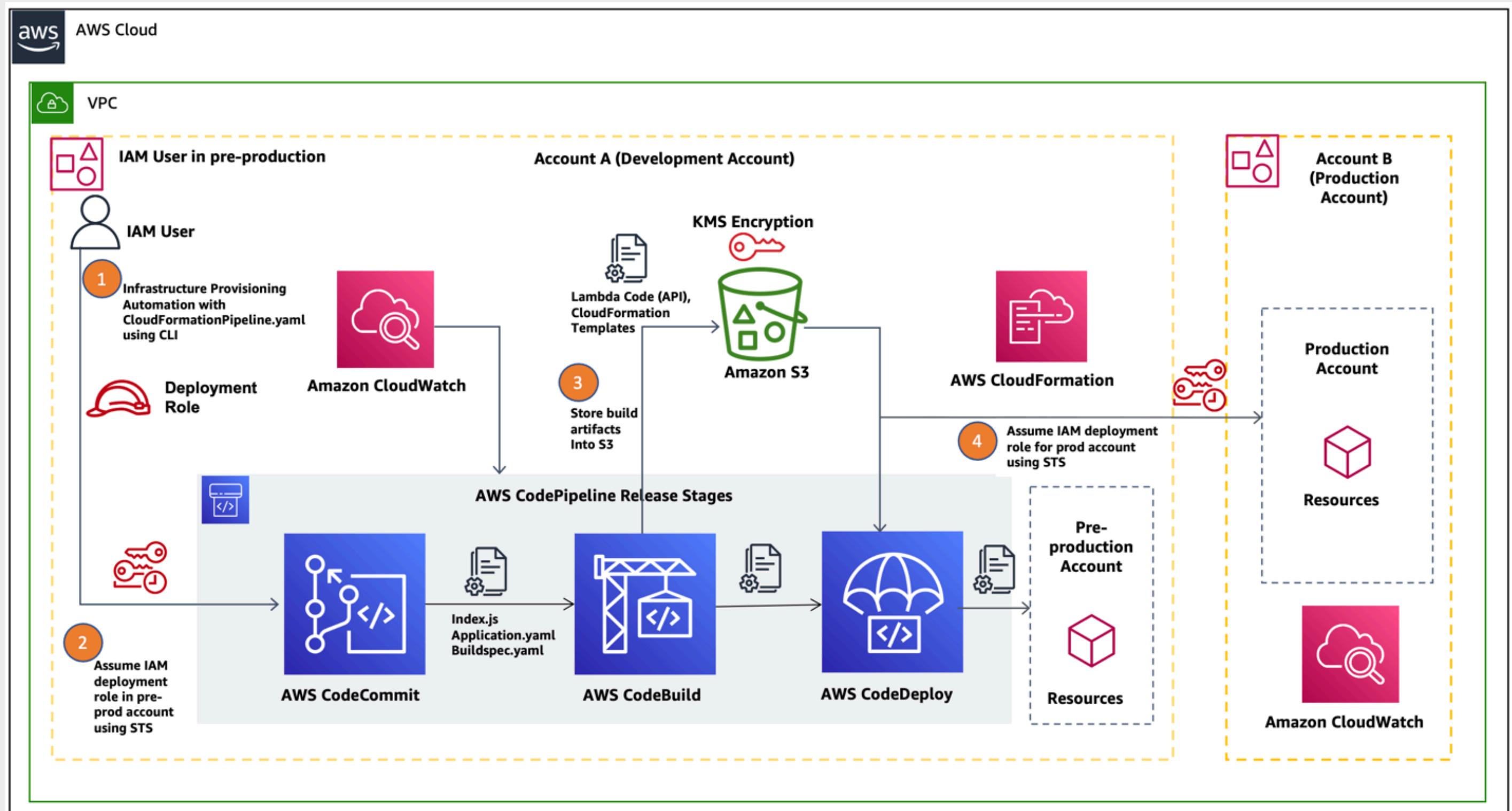
Figure 6-1. AWS multi-account structure

- L'importance des stratégies multi-comptes
- Organisations AWS pour la gestion des comptes
- Étapes de la création de comptes enfants
- Détails de la configuration



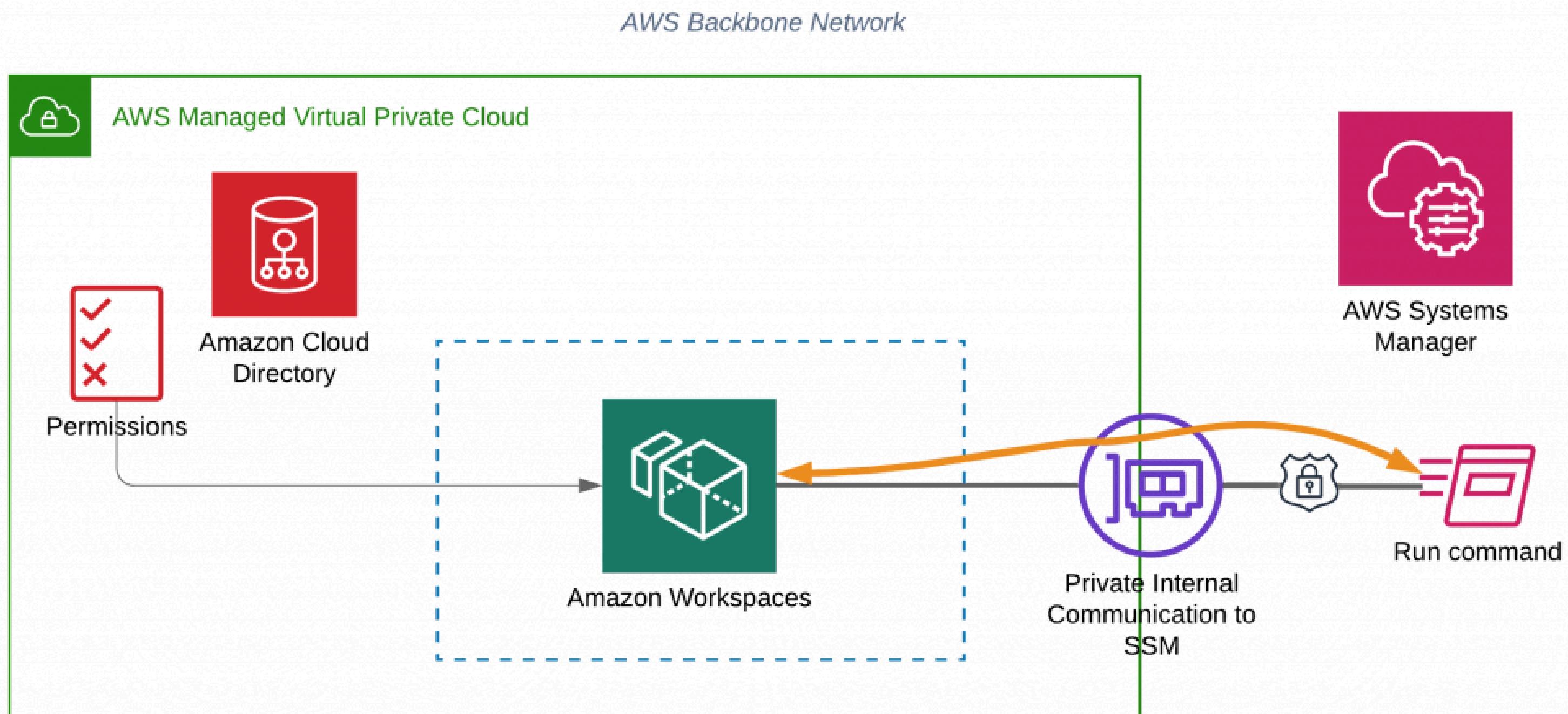
# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

- Utilisation de variables de sortie pour gérer les rôles
- Accès aux comptes enfants
- Déploiement d'une infrastructure dans plusieurs environnements
- Principaux enseignements



# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

- Utilisation des espaces de travail pour les déploiements isolés
- Différencier les configurations par environnement
- Récapitulation du flux de travail de déploiement
- Principaux enseignements



# Chapitre 6 :

## Comment travailler avec des équipes et des environnements multiples ?

- Finalisation et fermeture des comptes enfants AWS
- Décomposition de la base de code : Motivation et avantages
- Avantages de la décomposition de la base de code

Table 6-3. Bug density in software projects of various sizes.

Project size (lines of code)	Bug density (bugs per 1K lines of code)
< 2K	0 – 25
2K – 6K	0 – 40
16K – 64K	0.5 – 50
64K – 512K	2 – 70
> 512K	4 – 100

<sup>a</sup> *Code Complete: A Practical Handbook of Software Construction* by Steve McConnell (Microsoft Press).

# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

Figure 6-2. Part A depends on the source code of parts B and C

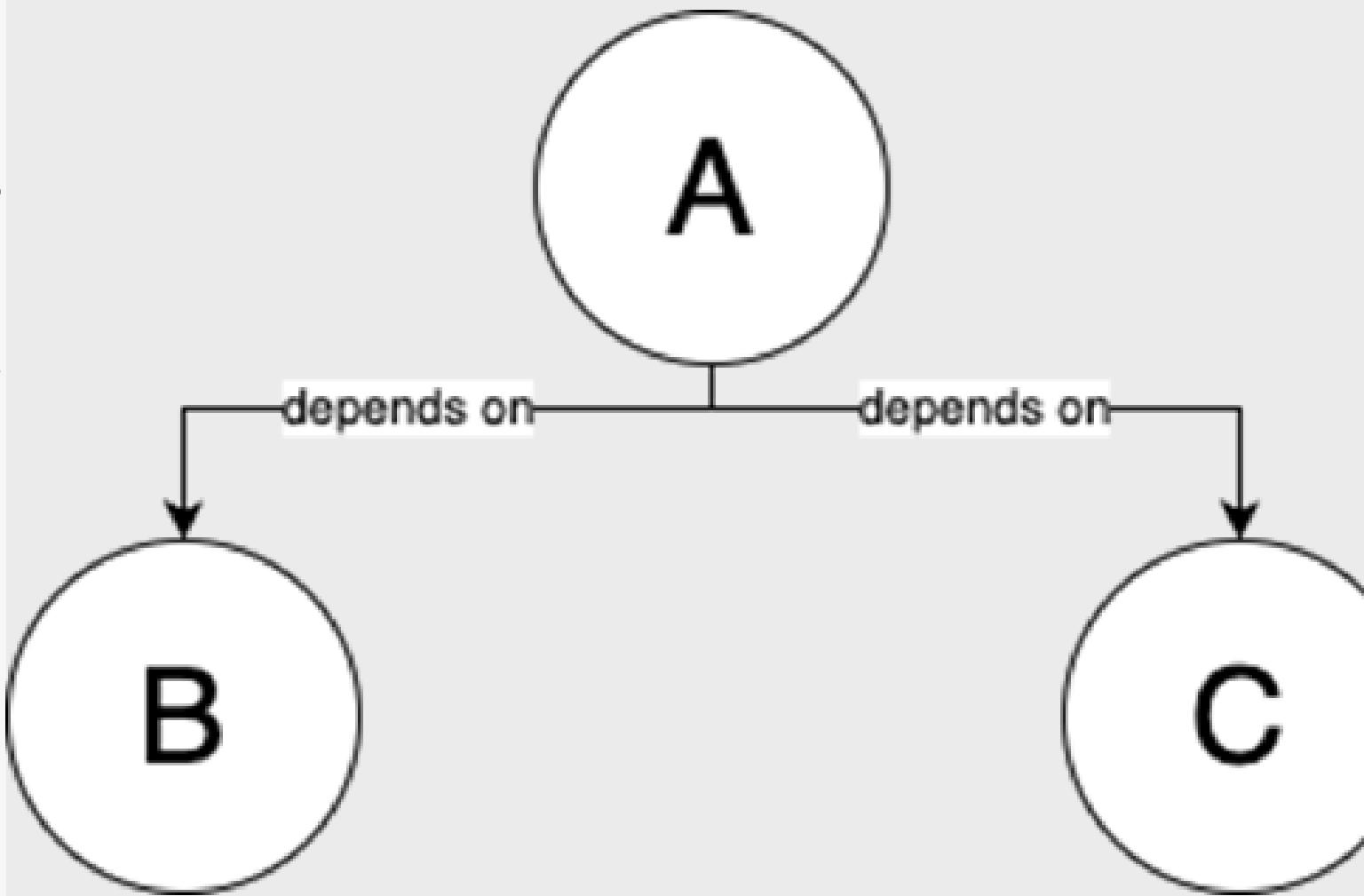
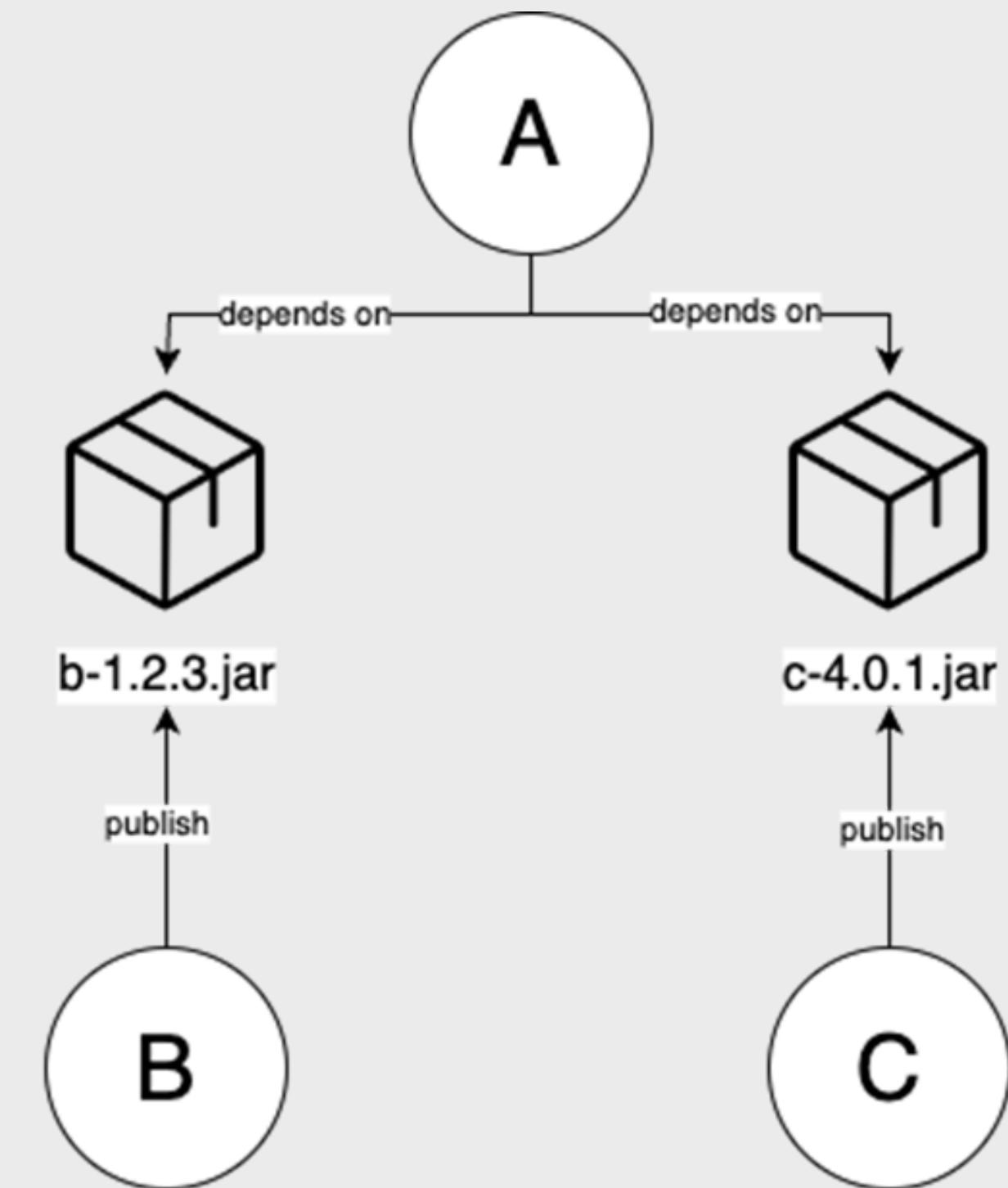


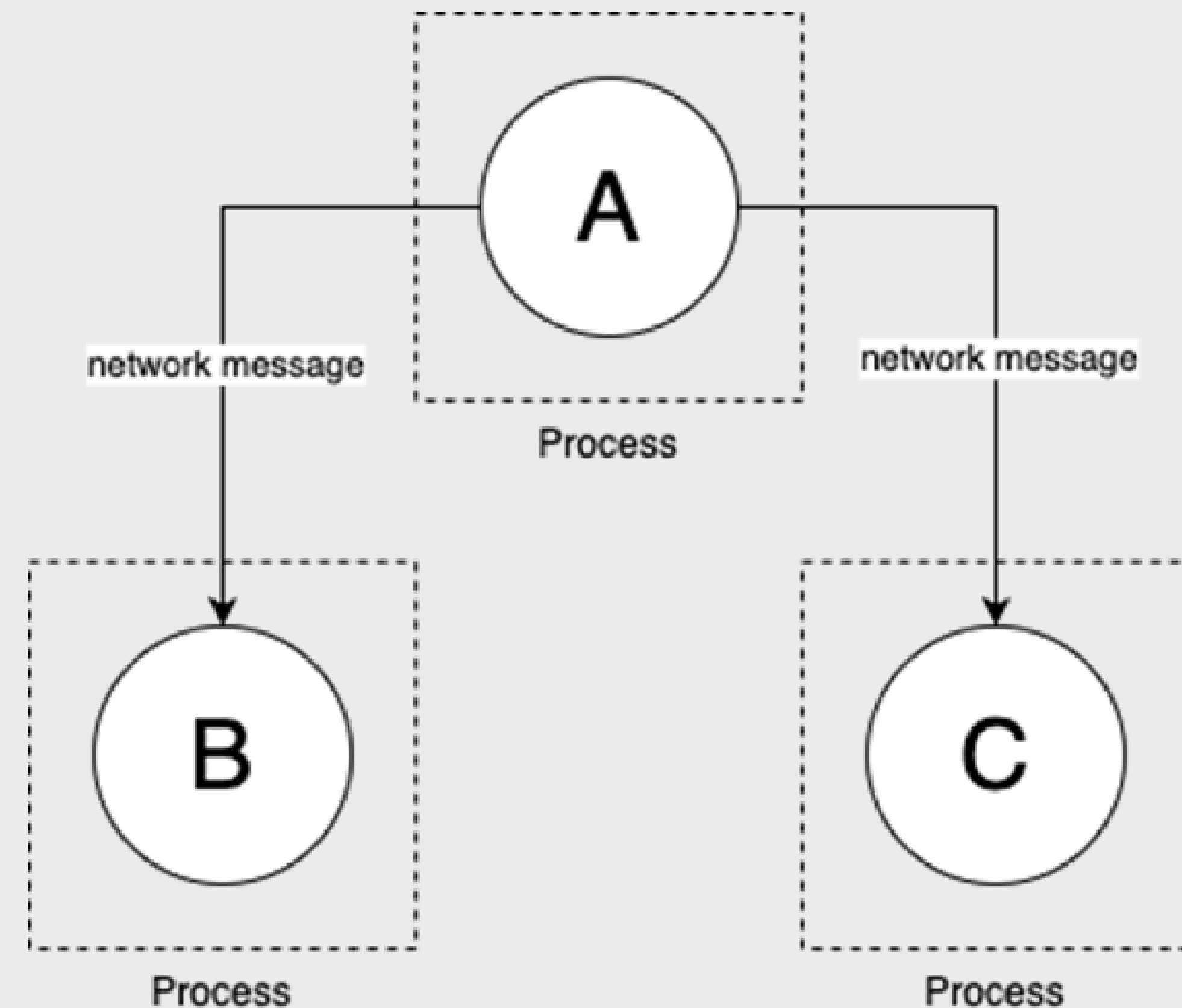
Figure 6-3. Part A depends on artifacts published by libraries B and C



# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

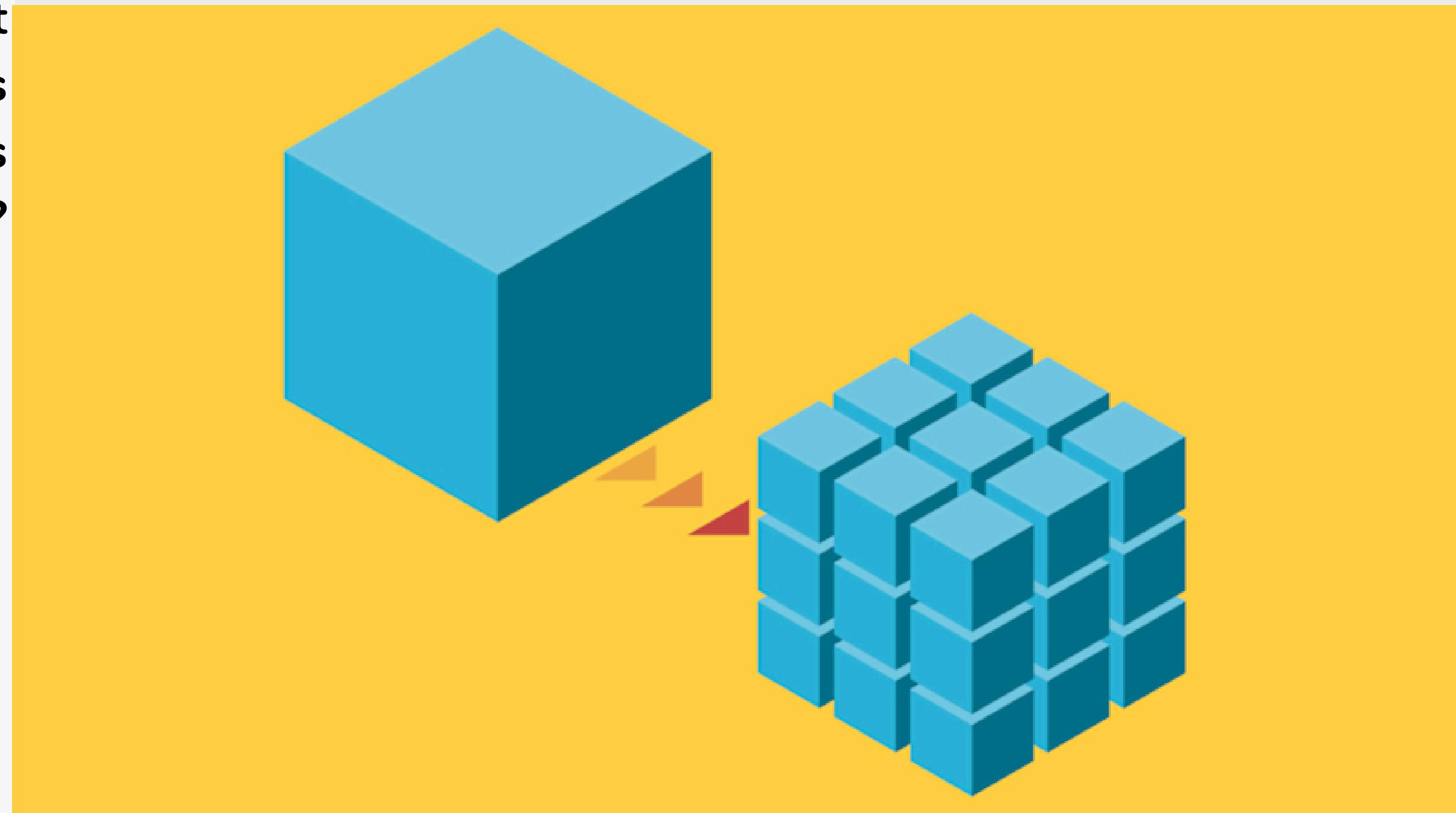
Figure 6-4. All the parts of the codebase run in separate processes and communicate via messages over the network

- Transition vers les architectures orientées services
- Défis liés à la rupture de la base de code



# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

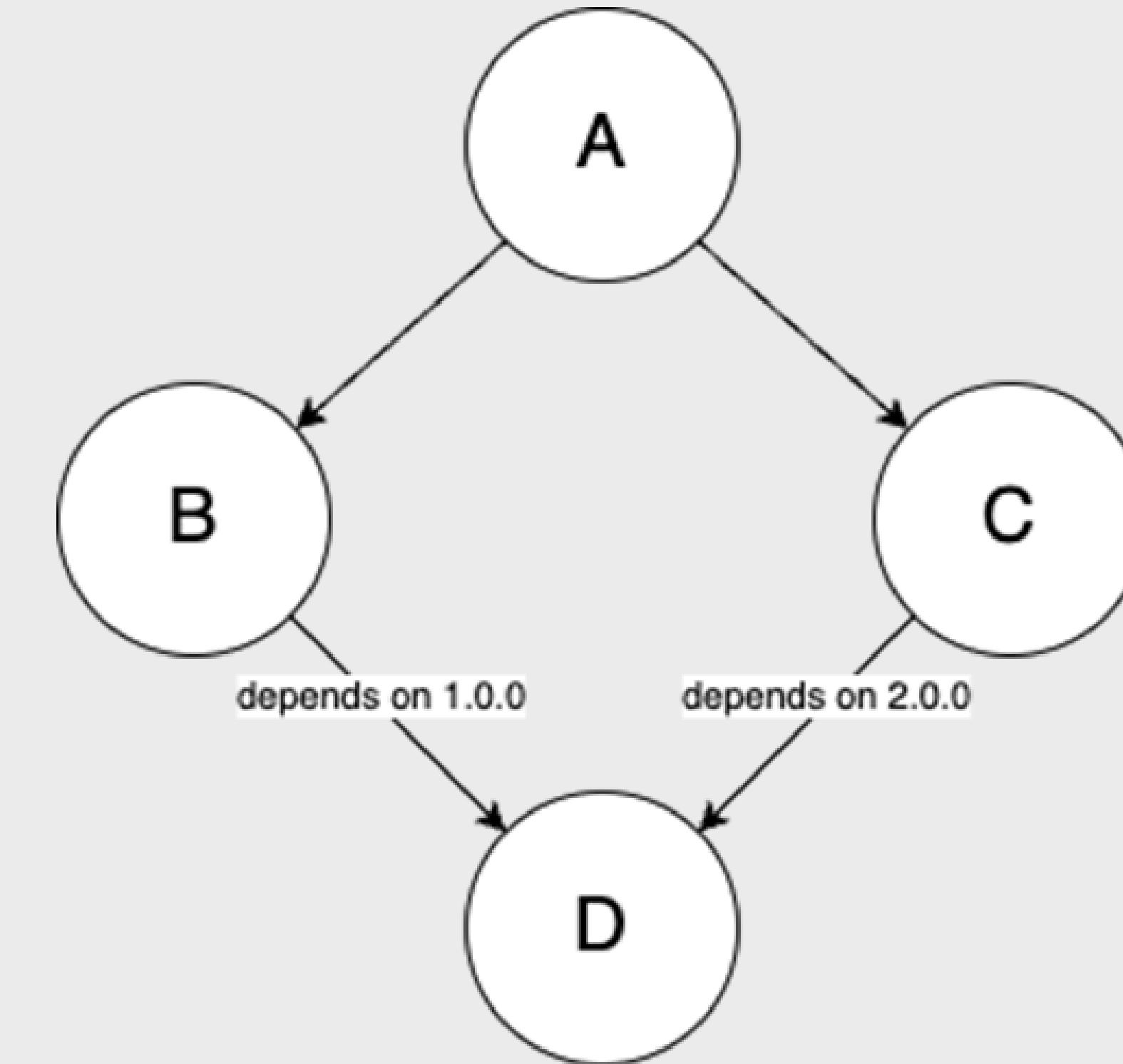
- Gestion de la compatibilité ascendante
- Défis posés par les modifications globales
- Décision concernant la division de la base de code



# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

Figure 6-5.  
Diamond  
dependencies

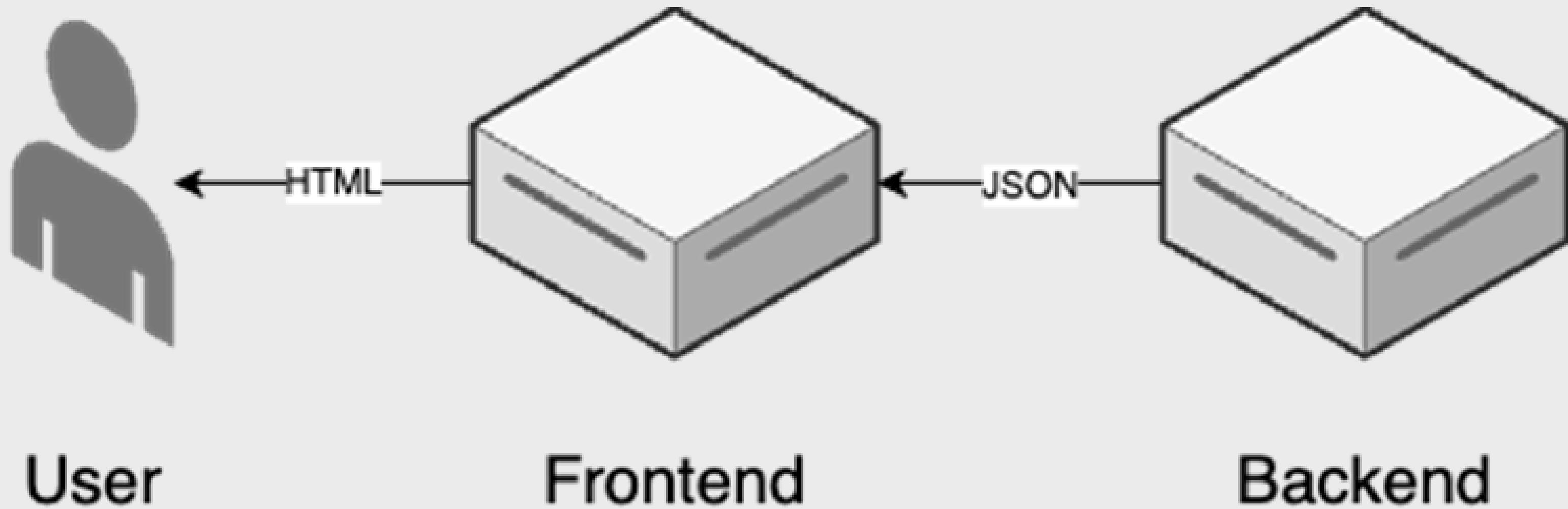
- Intégration tardive ou continue
- L'enfer des dépendances : Un défi majeur pour les bibliothèques
- Frais généraux opérationnels dans les services distribués
- Frais généraux liés à l'ordonnancement du déploiement
- Frais généraux de débogage dans les microservices



# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

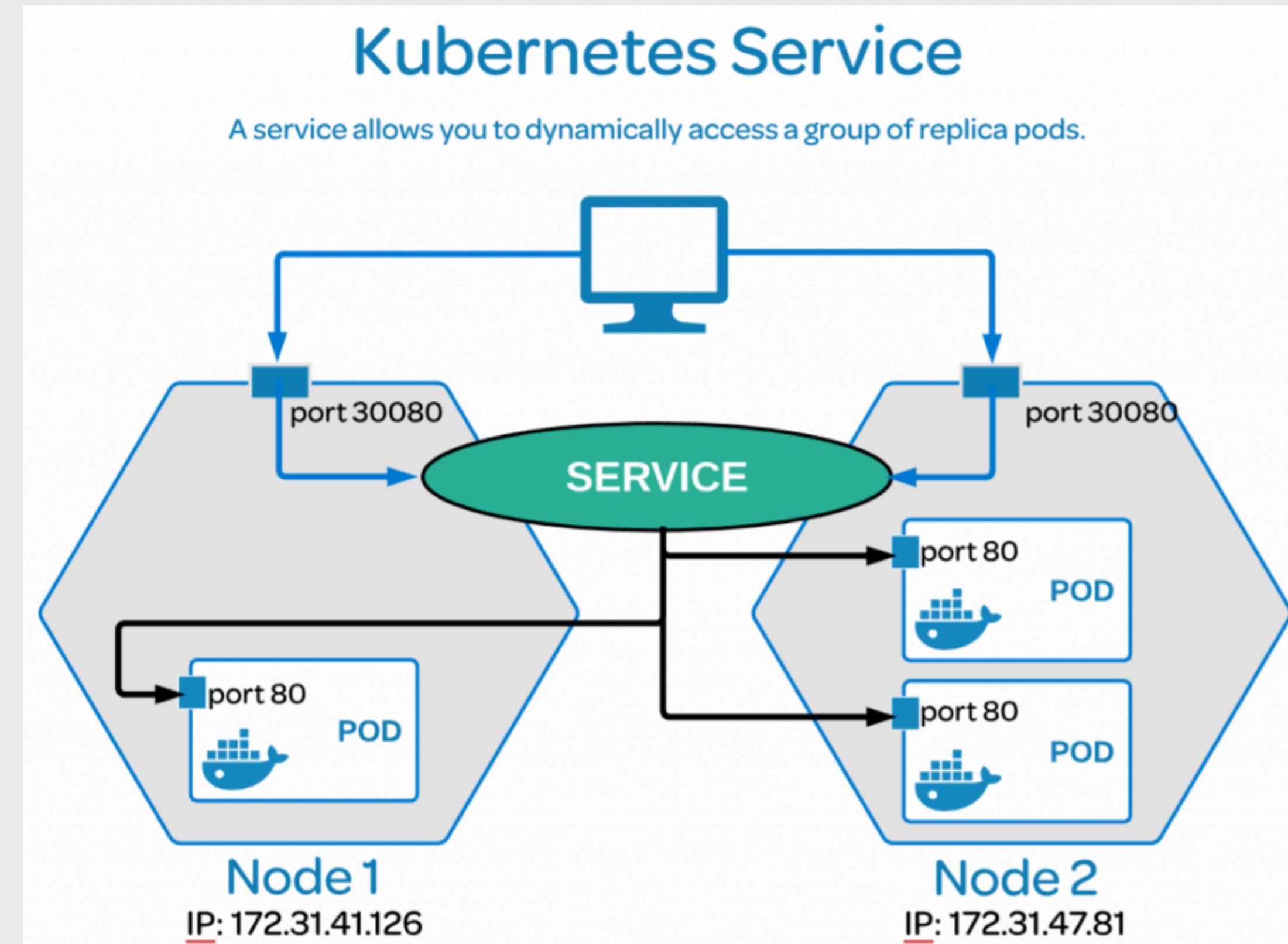
**Figure 6-6. A frontend microservice fetches data from a backend microservice and presents that data to the user**

- Introduction aux microservices
  - Surcharge de performance
  - Déploiement de microservices dans Kubernetes

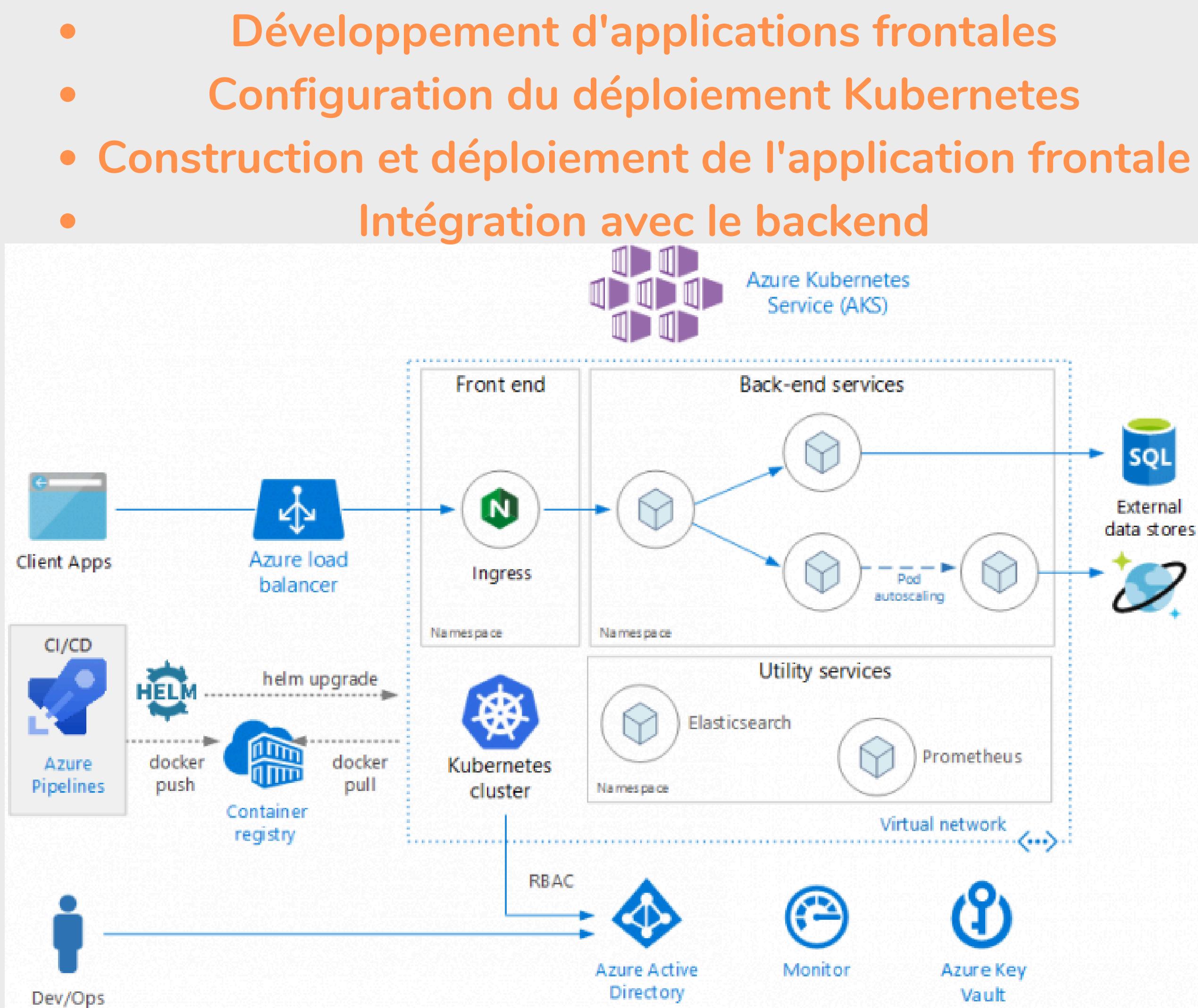


# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

- Installation et configuration du backend
- Création d'un service Kubernetes
- Construction et déploiement du backend
- Transition vers un microservice frontend



# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?



# Chapitre 6 : Comment travailler avec des équipes et des environnements multiples ?

- Exécuter des microservices dans Kubernetes
- Huit points clés sur les microservices et les déploiements
- Implications plus larges
- Perspectives d'avenir

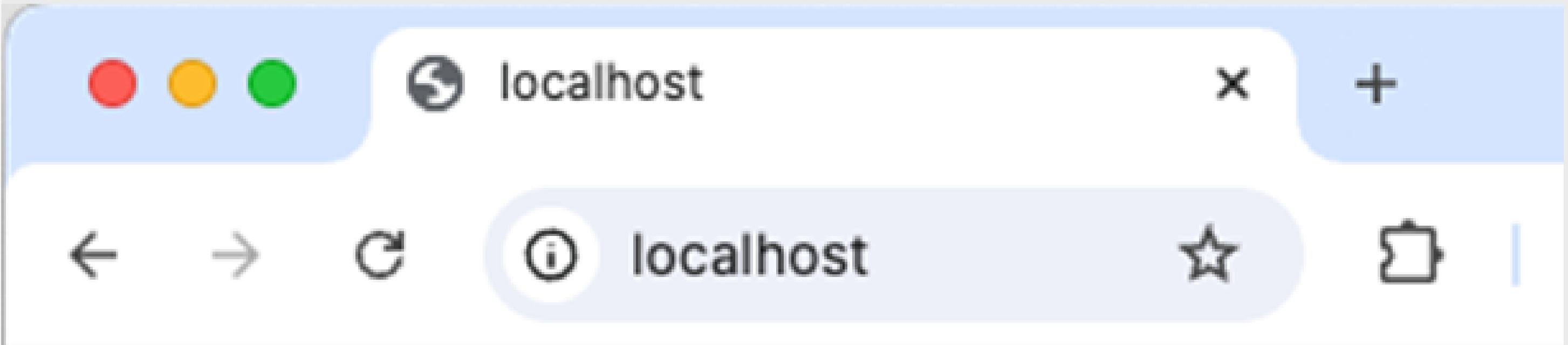
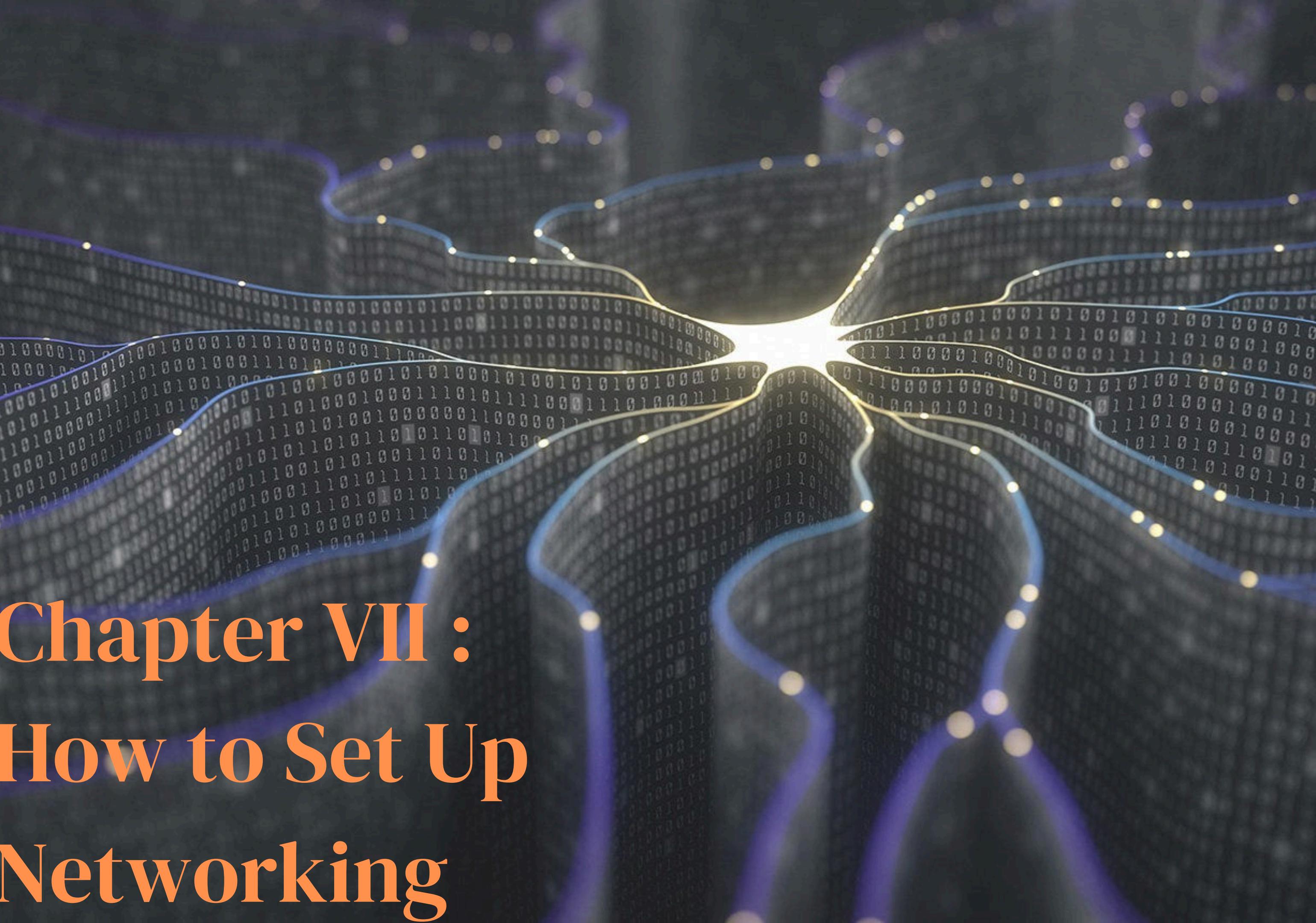


Figure 6-7. The HTML response from the frontend



# Chapter VII: How to Set Up Networking

Chapitre 7 :  
Comment  
mettre en place  
un réseau

Auteur : Badr TAJINI

142

- Aperçu de la Mise en Réseau
- Mise en Réseau Publique
- Système de Noms de Domaine (DNS)
- Pertinence Pratique

Concept	Description	Example
<b>Public networking</b>	Manage access to your apps over the public Internet with public IPs and domain names.	Deploy servers with public IPs in AWS and register a domain name for them in Route 53.
<b>Private networking</b>	Run your apps in a private network to protect them from public Internet access.	Create a Virtual Private Cloud (VPC) in AWS and deploy servers into it.
<b>Network access</b>	Learn to how to securely access private networks using SSH, RDP, and VPN.	Connect to a server in a VPC in AWS using a bastion host and SSH.
<b>Service communication</b>	Connect and secure communication between apps in a microservices architecture.	Use Istio as a service mesh for microservices running in Kubernetes.

Table 7-1. Concepts and examples you'll go through in this chapter

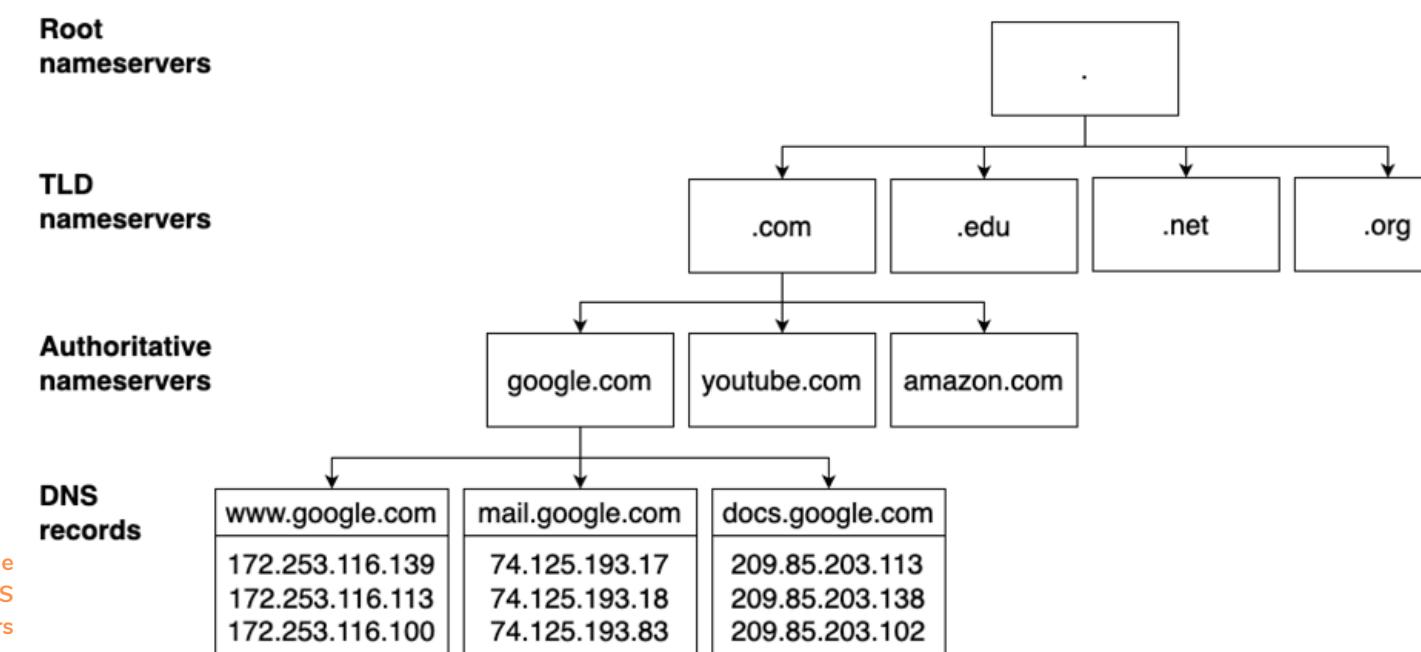


Figure 7-1. The hierarchy of DNS servers

**Chapitre 7 :**  
**Comment**  
**mettre en place**  
**un réseau**

Auteur : [Badr TAJINI](#)

143

- Etapes de l'enregistrement d'un domaine
- Configuration de la zone hébergée
- Déploiement des instances EC2

Search for domain

Check availability for a domain

X Search

Figure 7-2. Find a domain name that is available



Domain	Price/year	Actions
fundamentals-of-devops-example.com <span>Exact match</span>	14.00 USD	<a href="#">Select</a>

Search result

Domain	Price/year	Actions
fundamentals-of-devops-example.com <span>Exact match</span>	14.00 USD	<a href="#">Select</a>

Figure 7-2. Find a domain name that is available

fundamentals-of-devops-example.com [Info](#)

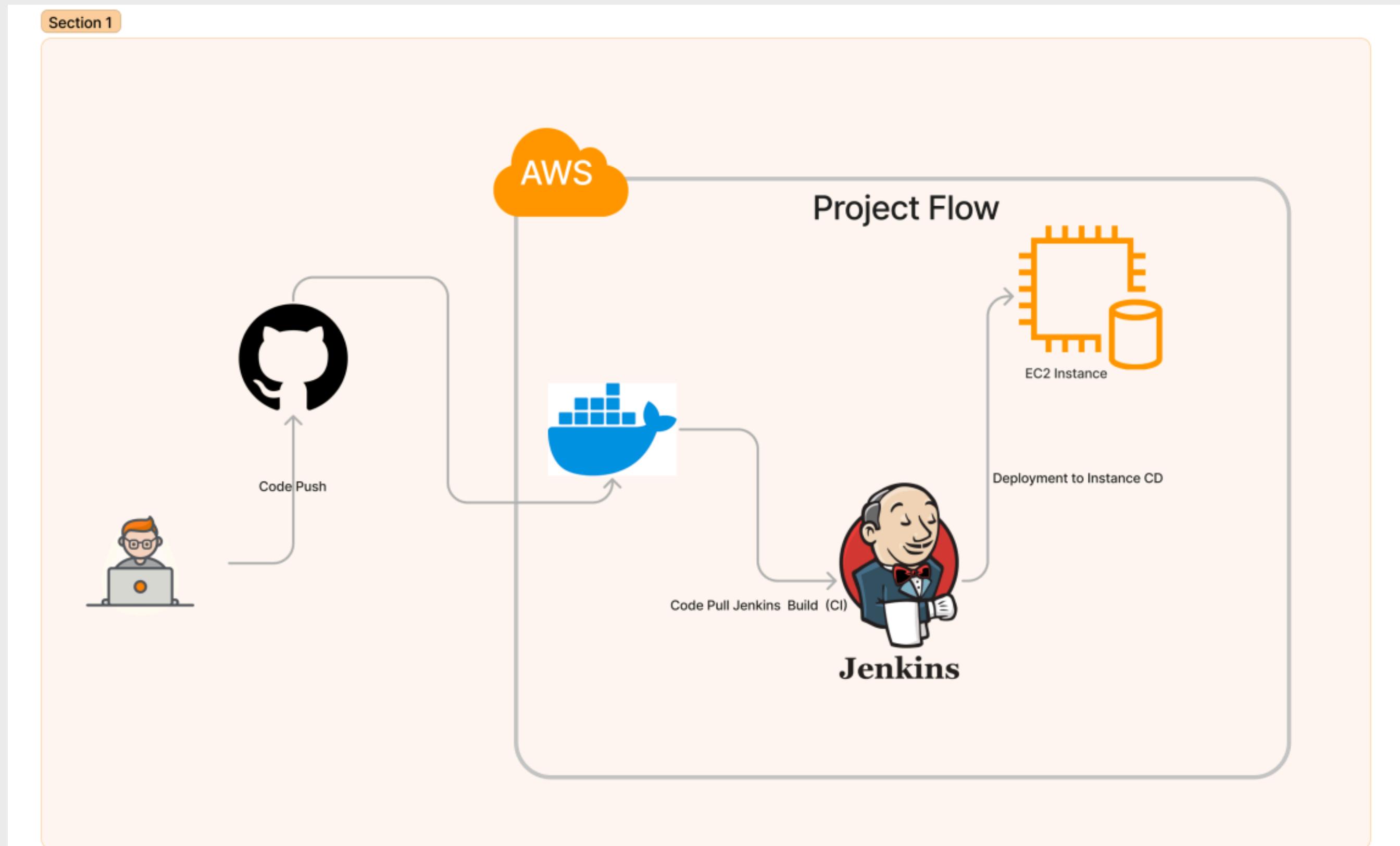
Details [Info](#) Actions ▾

Registration date July 24, 2024, 14:45 (UTC:+01:00)	Auto-renew On	Domain status code addPeriod ok	Name servers ns-1262.awsdns-29.org ns-153.awsdns-19.com ns-1542.awsdns-00.co.uk ns-682.awsdns-21.net
Expiration date July 24, 2025	Transfer lock Off	DNSSEC status Not configured	

Figure 7-3. Details for your registered domain name

# Chapitre 7 : Comment mettre en place un réseau

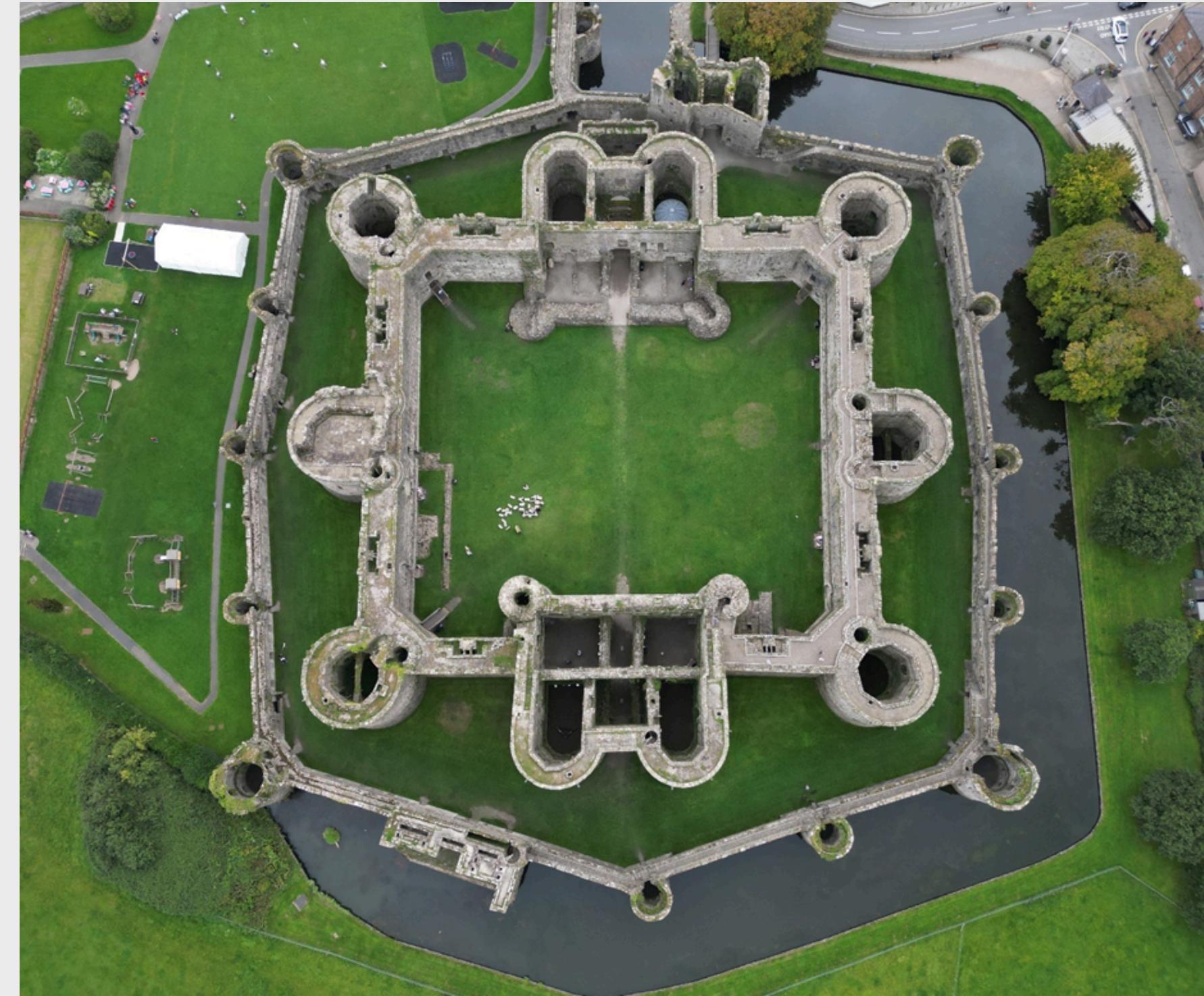
- Déploiement des instances EC2
- Configuration des enregistrements DNS
- Exercices pratiques



# Chapitre 7 : Comment mettre en place un réseau

- Passage d'un réseau public à un réseau privé
- Le concept de réseau privé
- Mise en œuvre pratique
- Comparaison : Réseaux publics et réseaux privés

Figure 7-5. Beaumaris  
Castle ([photo by  
Llywelyn2000](#))



- Structures de réseaux : De la simplicité à la complexité
- Réseaux privés : Caractéristiques et applications
- Avantages des réseaux privés

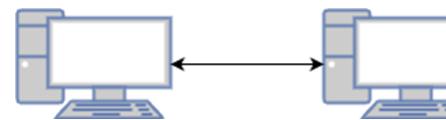


Figure 7-6. Connecting two computers

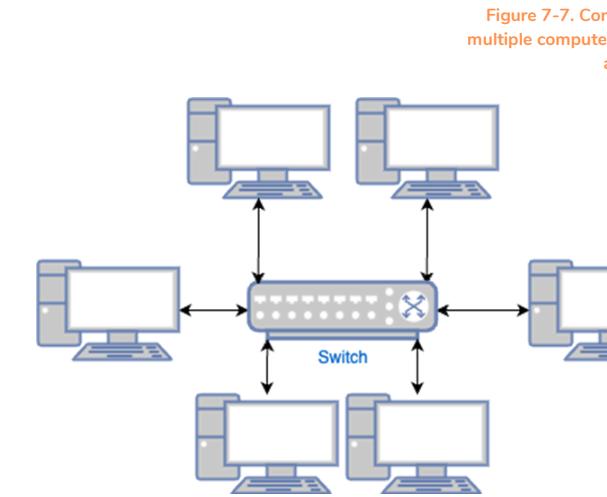


Figure 7-7. Connecting multiple computers using a switch

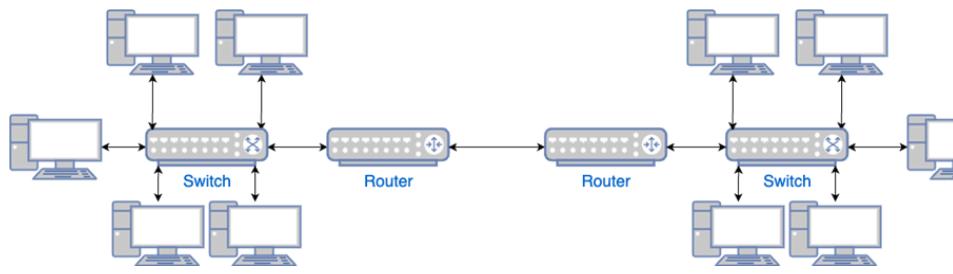


Figure 7-8. Connecting two networks using routers

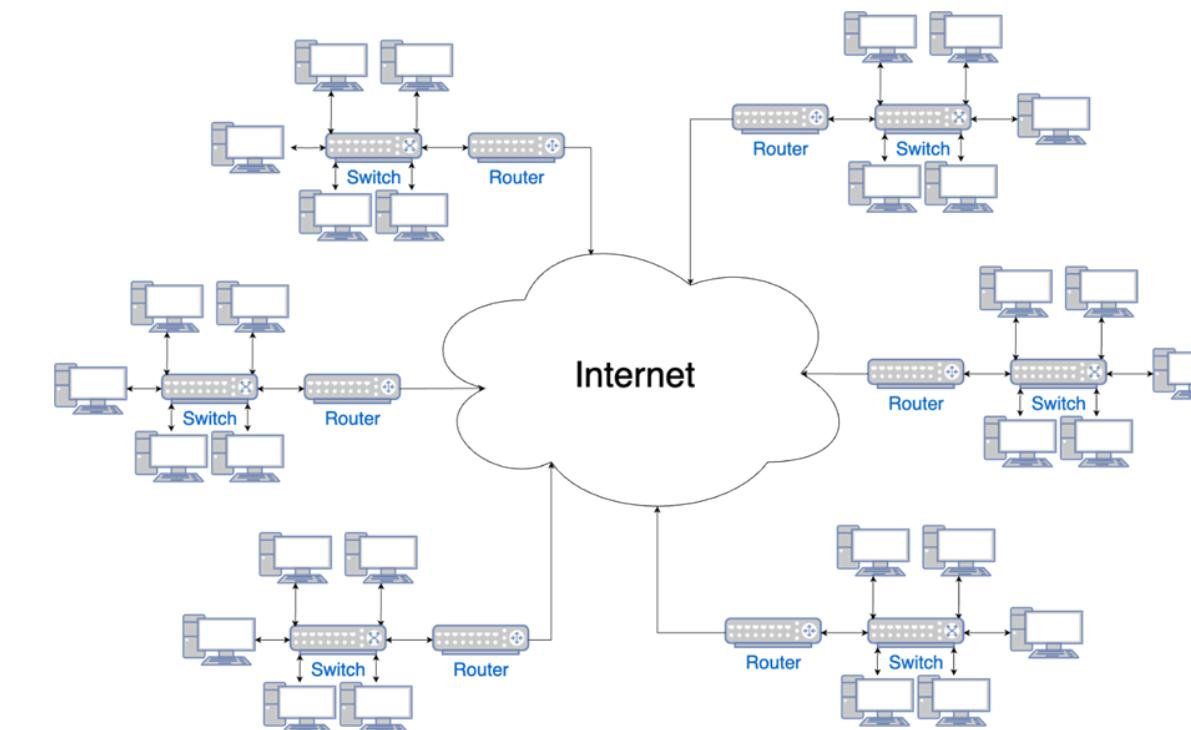


Figure 7-9. Connecting many networks together via the Internet

# Chapitre 7 : Comment mettre en place un réseau

- Passerelles pour réseaux privés
- Réseaux privés virtuels (VPC) dans le nuage
- Outils de mise en réseau orchestrée

Table 7-2. The route table

Destination	Target
10.0.0.0/16	VPC Foo
10.1.0.0/16	VPC Bar
0.0.0.0/0	NAT gateway
Table 7-2. Example route table	

# Chapitre 7 : Comment mettre en place un réseau

Auteur : Badr TAJINI

148

- Orchestrer la mise en réseau dans Kubernetes
- Créeation d'un VPC dans AWS
- Exemple de déploiement
- Principaux enseignements

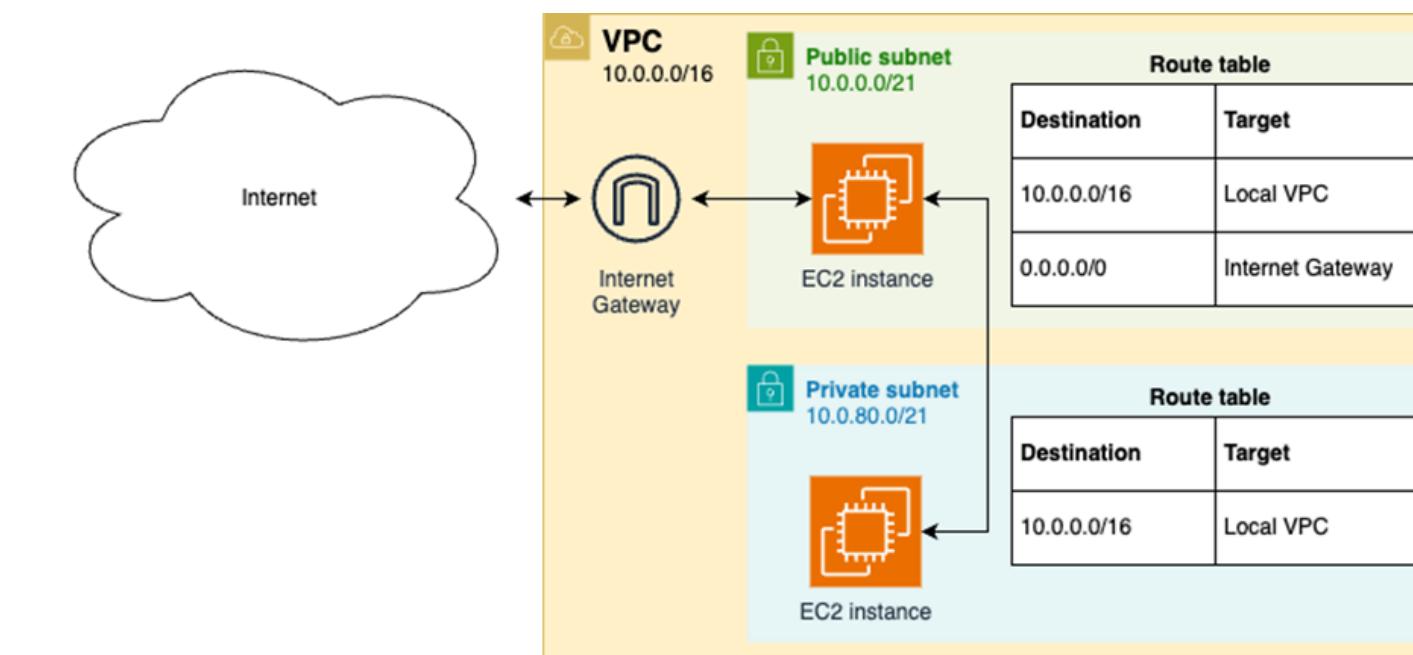


Figure 7-10. The diagram of the VPC

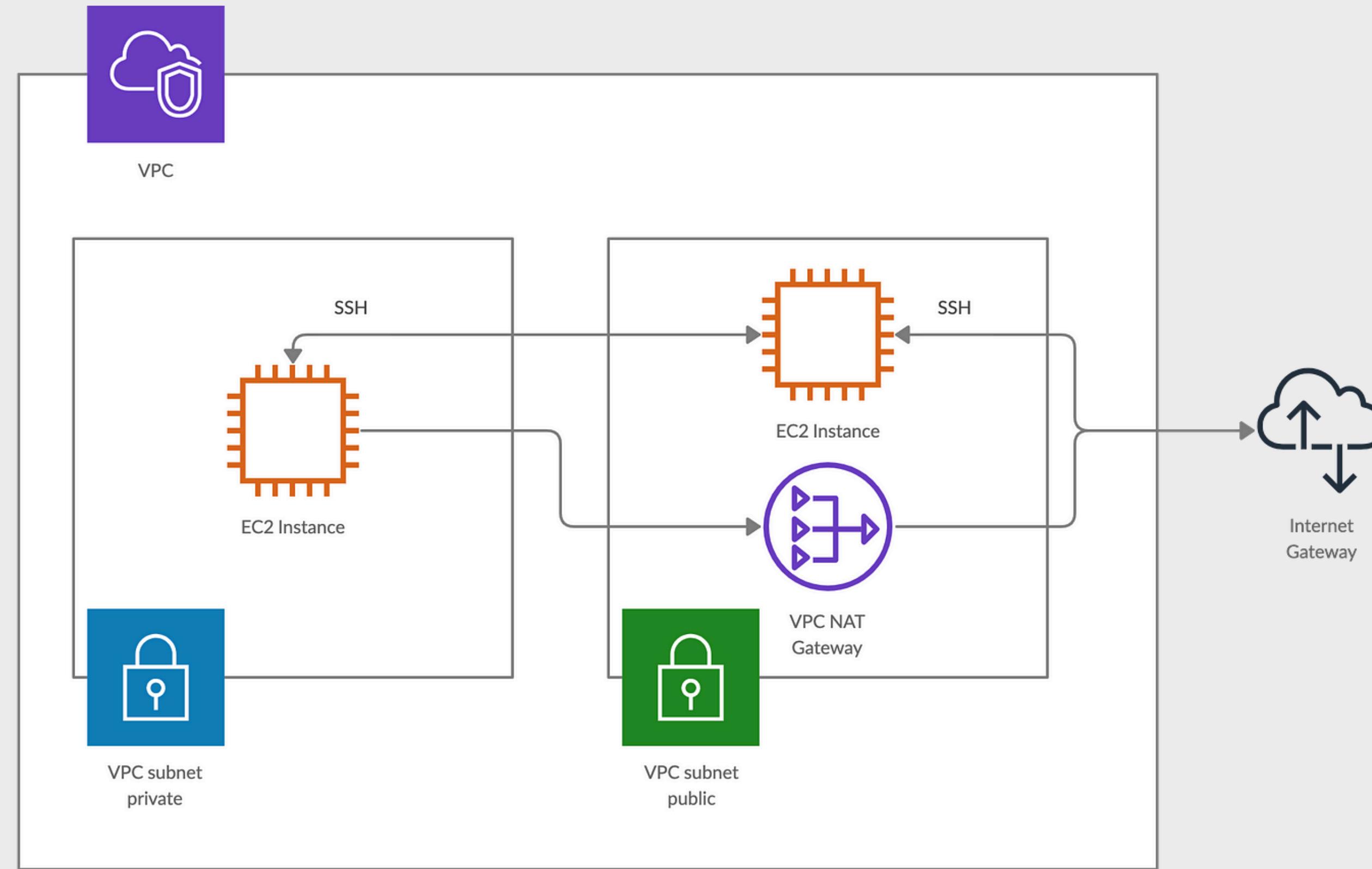
Table 7-3. Comparing the behavior of networking plugins for Kubernetes in various clouds

Cloud	Typical CNI plugin	Typical ingress controller	IP address management	Service communication	Ingress
AWS	<a href="#">Amazon VPC CNI plugin</a>	<a href="#">AWS Load Balancer Controller</a>	Assign IPs from the AWS VPC	Use AWS VPC routing	Deploy AWS Elastic Load Balancers
GCP	<a href="#">Cilium GKE plugin</a>	<a href="#">GKE ingress</a>	Assign IP addresses from Cloud VPC subnets	Use Cloud VPC routing	Deploy Cloud Load Balancers
Azure	<a href="#">Azure CNI plugin</a>	<a href="#">Nginx ingress controller</a>	Assign IP addresses from VNet subnets	Use VNet routing	Deploy Nginx

# Chapitre 7 :

## Comment mettre en place un réseau

- Déploiement d'une instance publique
- Déploiement d'une instance privée
- Variables de sortie pour les instances



# Chapitre 7 : Comment mettre en place un réseau

- Validation de la configuration du réseau
- Accès aux réseaux privés
- Modèle « château et maat » (Castle-and-Moat)
- Modèle de confiance zéro

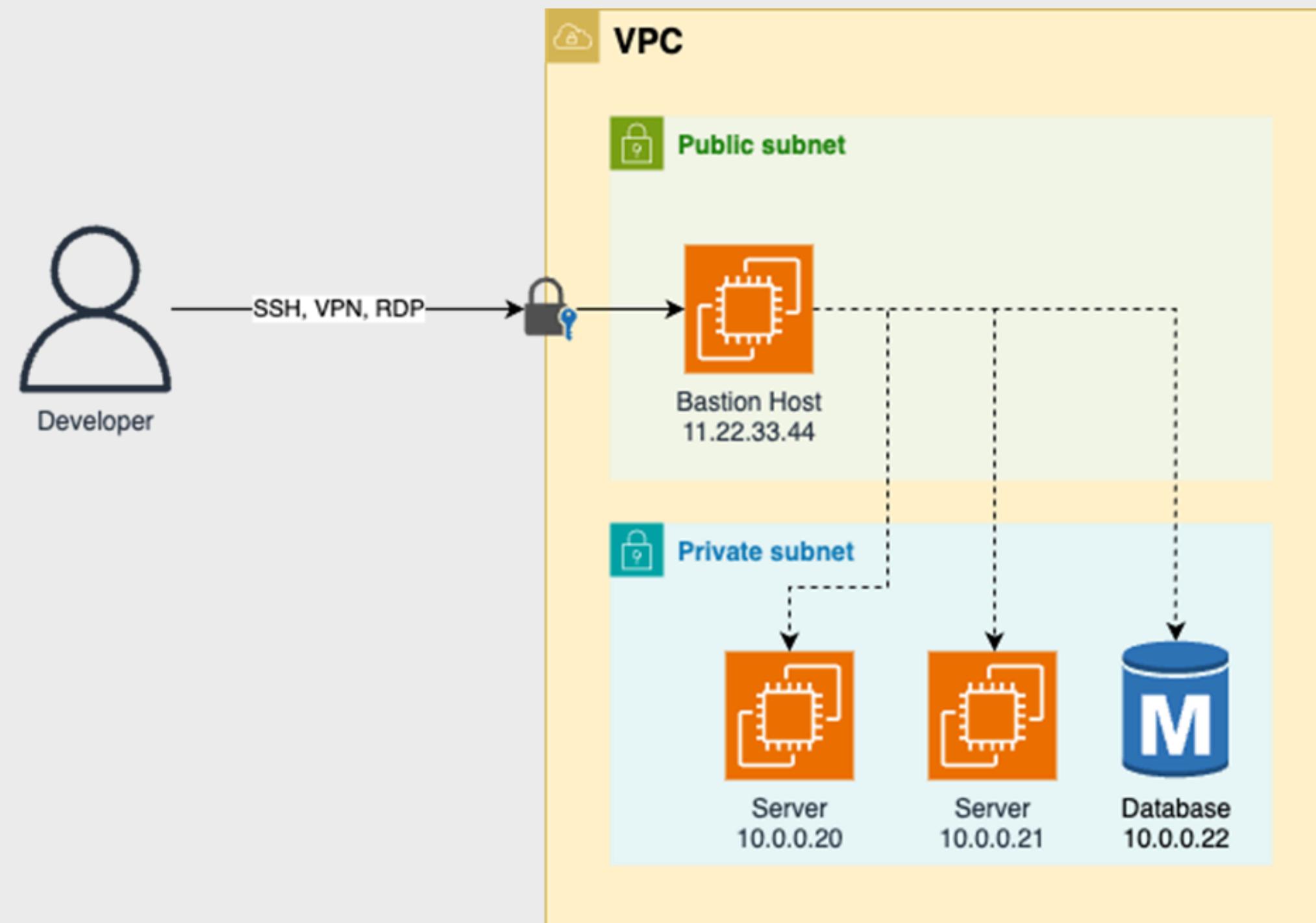
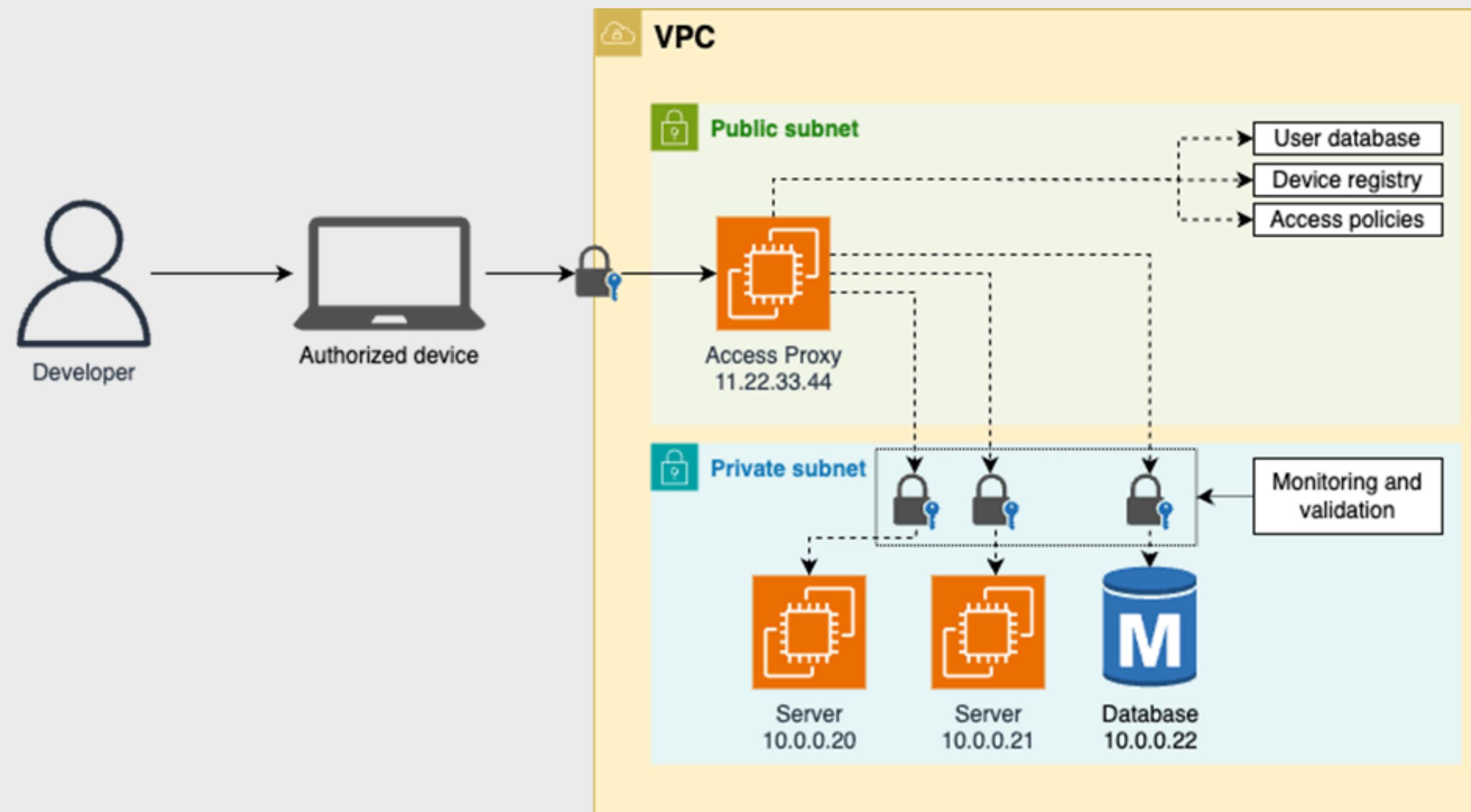


Figure 7-11. A castle-and-moat model with a bastion host as the sole access point

# Chapitre 7 : Comment mettre en place un réseau

Figure 7-12. Zero-trust  
architecture

- Architecture de confiance zéro (ZTA)
- BeyondCorp et la mise en œuvre de la confiance zéro
- Outils d'accès communs
- Applications de l'architecture de confiance zéro



# Chapitre 7 : Comment mettre en place un réseau

- Vue d'ensemble de SSH
- Composants de l'accès SSH
- Fonctionnalités de SSH
- Exemple pratique : Configuration d'un hôte SSH Bastion
- Notes de mise en œuvre

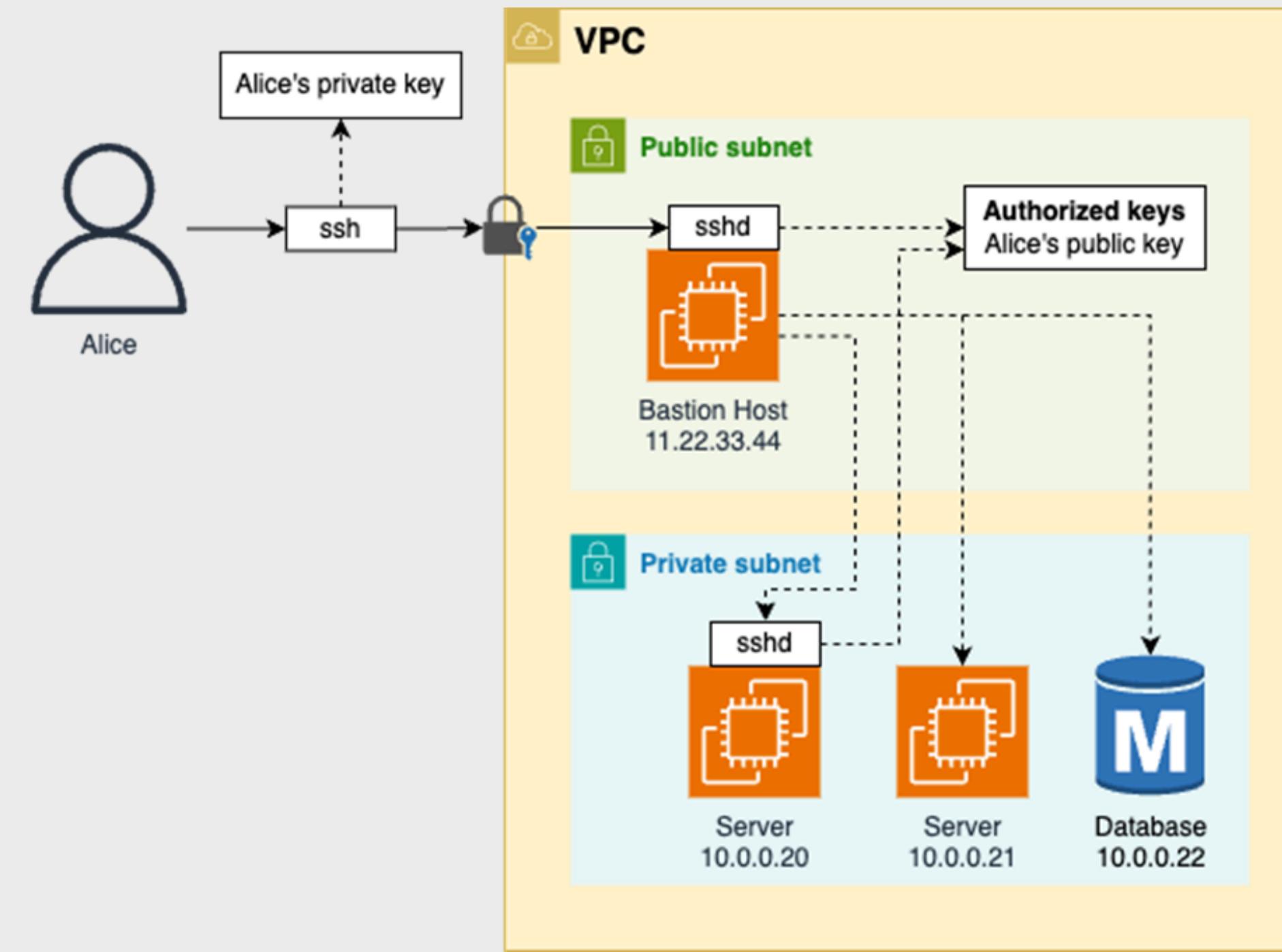


Figure 7-13. Using SSH  
to connect to a bastion  
host

# Chapitre 7 : Comment mettre en place un réseau

- Accès aux instances via SSH
- Sauter l'hôte : Connexion à des instances privées
- Utilisation de la redirection de port
- Exercices d'apprentissage pratique

Figure 7-14. The system log for an EC2 instance

System log

```
ci-info: +-----+
ci-info: | ssh-rsa | 0f:63:a3:0b:e9:b6:af:f0:51:af:06:20:b7:83:dc:37:3a:ed:23:1c:ff:b3:54:5
ci-info: +-----+
<14>Aug 1 13:14:27 cloud-init: #####
<14>Aug 1 13:14:27 cloud-init: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>Aug 1 13:14:27 cloud-init: 256 SHA256:CNHBiFQhZCPMHFKGt8amkllW/GC6CzEpGhcMZ2x5XTM root
<14>Aug 1 13:14:27 cloud-init: 256 SHA256:v+MXP6xY/03lGxlyywpBhEmr+qFwS0H2ASy77XPodNY root
<14>Aug 1 13:14:27 cloud-init: -----END SSH HOST KEY FINGERPRINTS-----
<14>Aug 1 13:14:27 cloud-init: #####
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBP2q33wv9Z7nn8dam0X
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILnw0u3po0o1QfpZivmhYHlsAcFSYV7CMVRZ0/0R36xK root@ip-10
-----END SSH HOST KEY KEYS-----
[ 12.724849] cloud-init[2203]: Cloud-init v. 22.2.2 finished at Thu, 01 Aug 2024 13:14:27

Amazon Linux 2023.4.20240401
Kernel 6.1.82-99.168.amzn2023.x86_64 on an x86_64 (-)

ip-10-0-1-26 login:
```

# Chapitre 7 : Comment mettre en place un réseau

- Utilisation du protocole de bureau à distance (RDP)
- Utilisation des réseaux privés virtuels (VPN)
- Mise en place d'un VPN
- Comparaison des méthodes d'accès

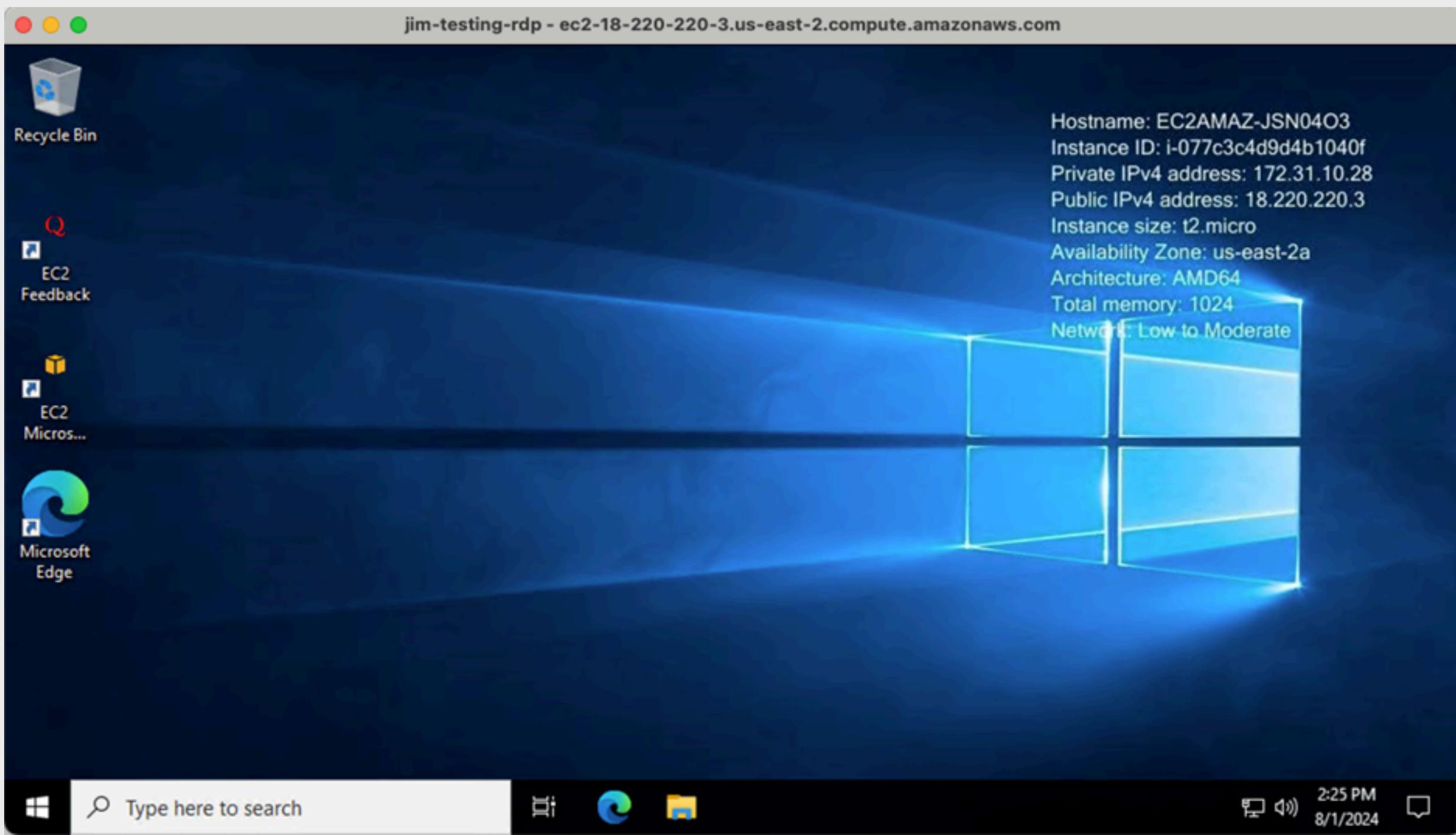
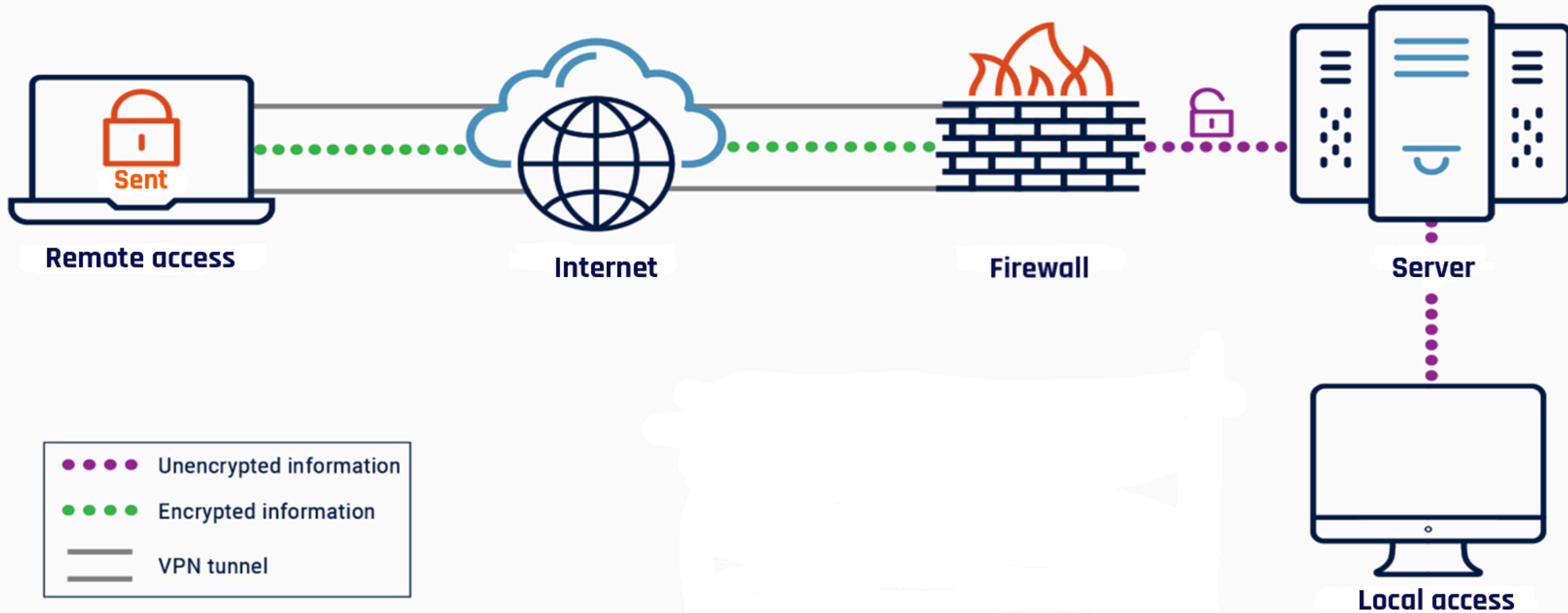


Figure 7-15. Using RDP  
to manage a Windows  
server remotely

# Chapitre 7 : Comment mettre en place un réseau

- Configuration d'un VPN pour l'accès à un réseau privé
- Communication de services dans les réseaux privés
- Outils et approches de découverte de services



# Chapitre 7 :

## Comment mettre en place un réseau

- Compromis dans les outils de découverte de services
- Protocole de communication des services

Table 7-4. Comparing service discovery tools

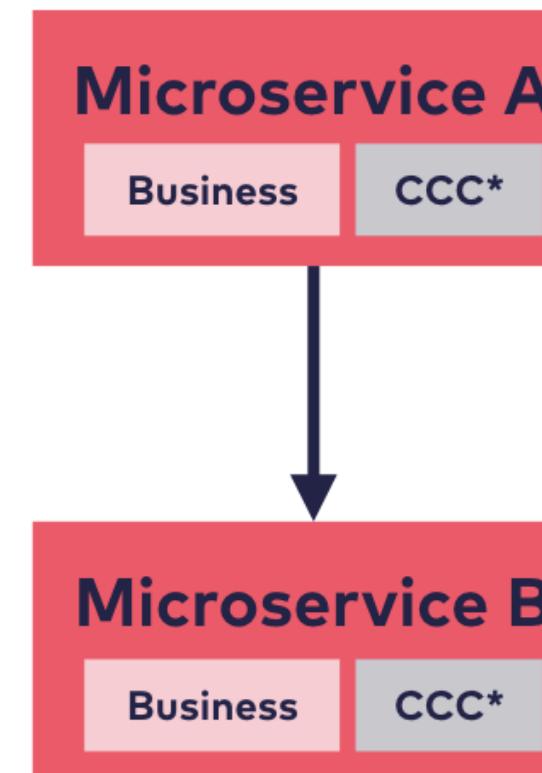
	Configuration files	Load balancers	DNS	Registry + library	Local proxy
Minimize manual error	Poor	Moderate	Very strong	Very strong	Very strong
Update speed	Poor	Very strong	Moderate	Very strong	Very strong
Scalability	Poor	Moderate	Strong	Very strong	Very strong
Transparent to app code	Moderate	Moderate	Very strong	Poor	Very strong
Minimize latency overhead	Very strong	Poor	Moderate	Very strong	Strong
Minimize CPU & memory overhead	Very strong	Very strong	Very strong	Very strong	Poor
Minimize infrastructure overhead	Very strong	Moderate	Strong	Poor	Poor

# Chapitre 7 :

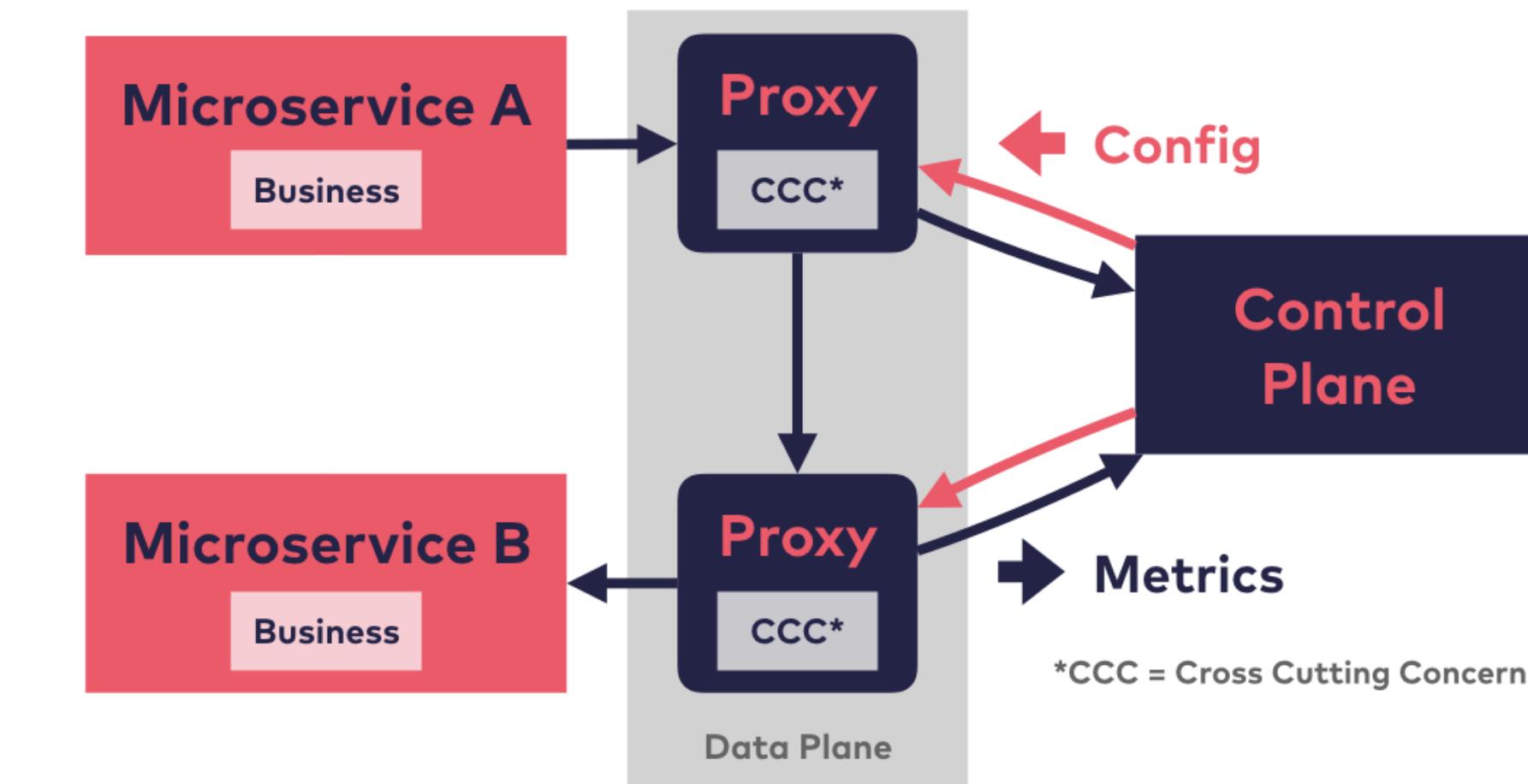
## Comment mettre en place un réseau

- Bibliothèques de sérialisation
- Facteurs clés dans le choix d'un protocole de communication
- Service Mesh

### Microservices



### Microservices + Service Mesh



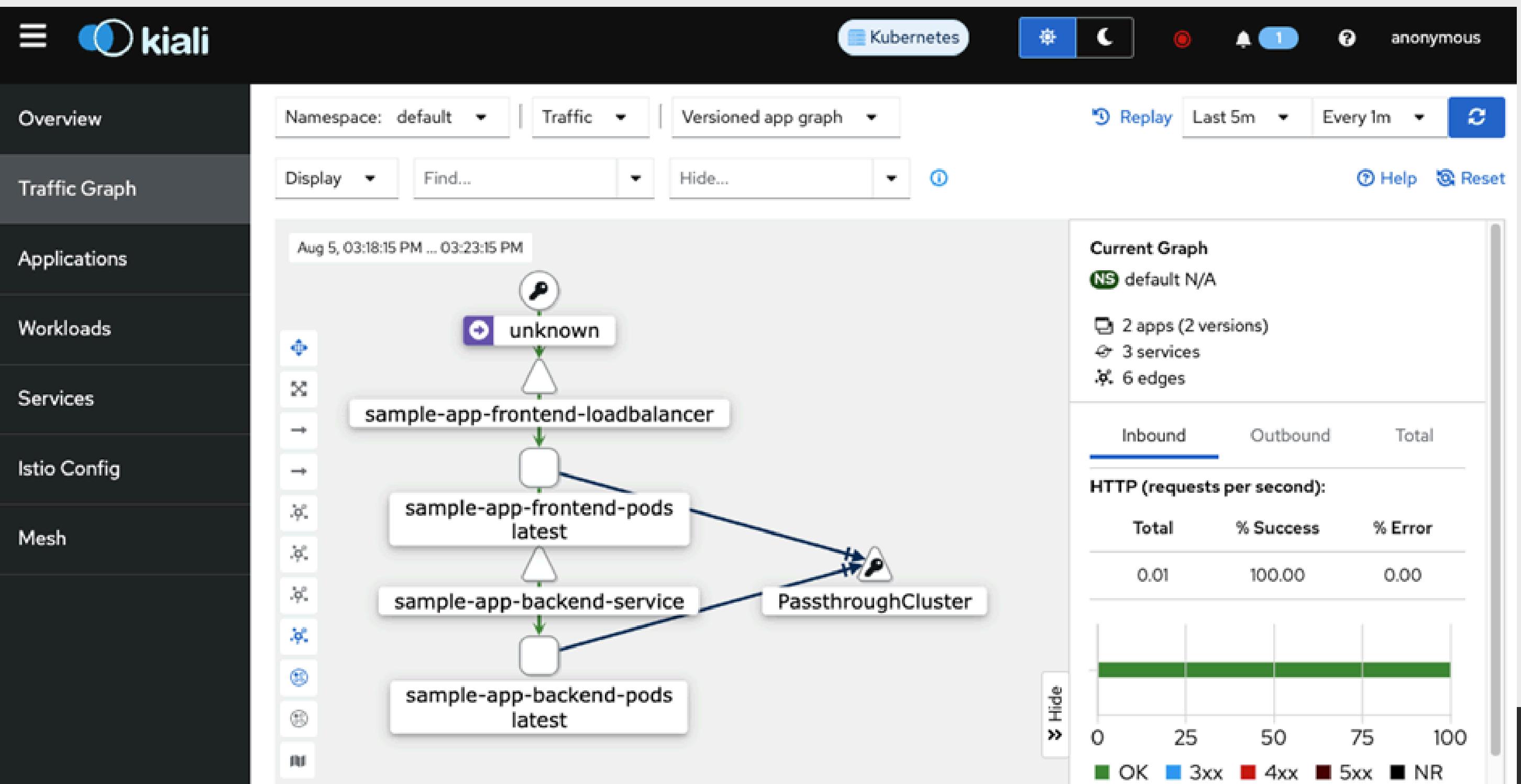
**Business** = Business Logic, Business Metrics

**CCC\*** = Traffic Metrics, Routing, Retry, Timeout, Circuit Breaking, Encryption, Decryption, Authorization, ...

\*CCC = Cross Cutting Concerns

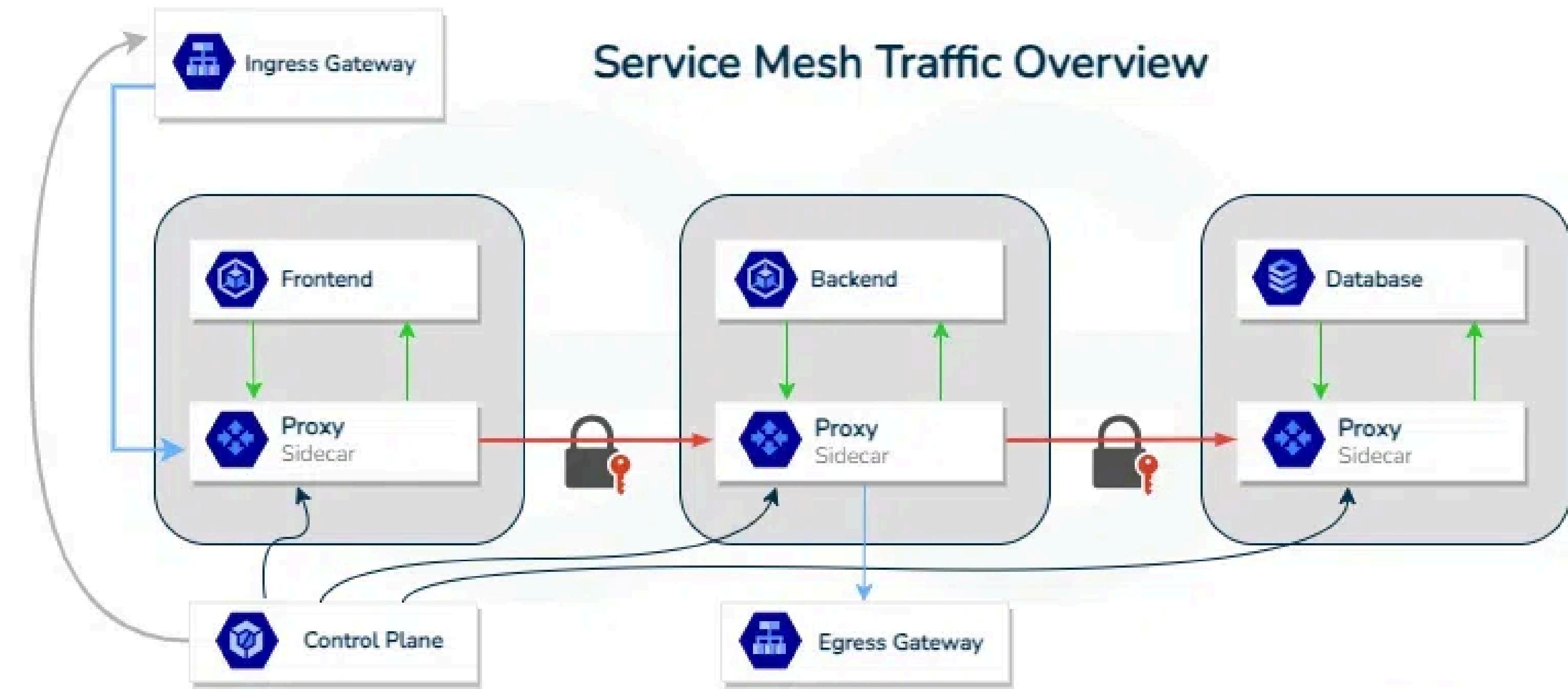
# Chapitre 7 : Comment mettre en place un réseau

- Vue d'ensemble d'Istio
- Mise en place et configuration
- Observabilité avec Istio
- Renforcer la sécurité
- Principaux avantages d'Istio



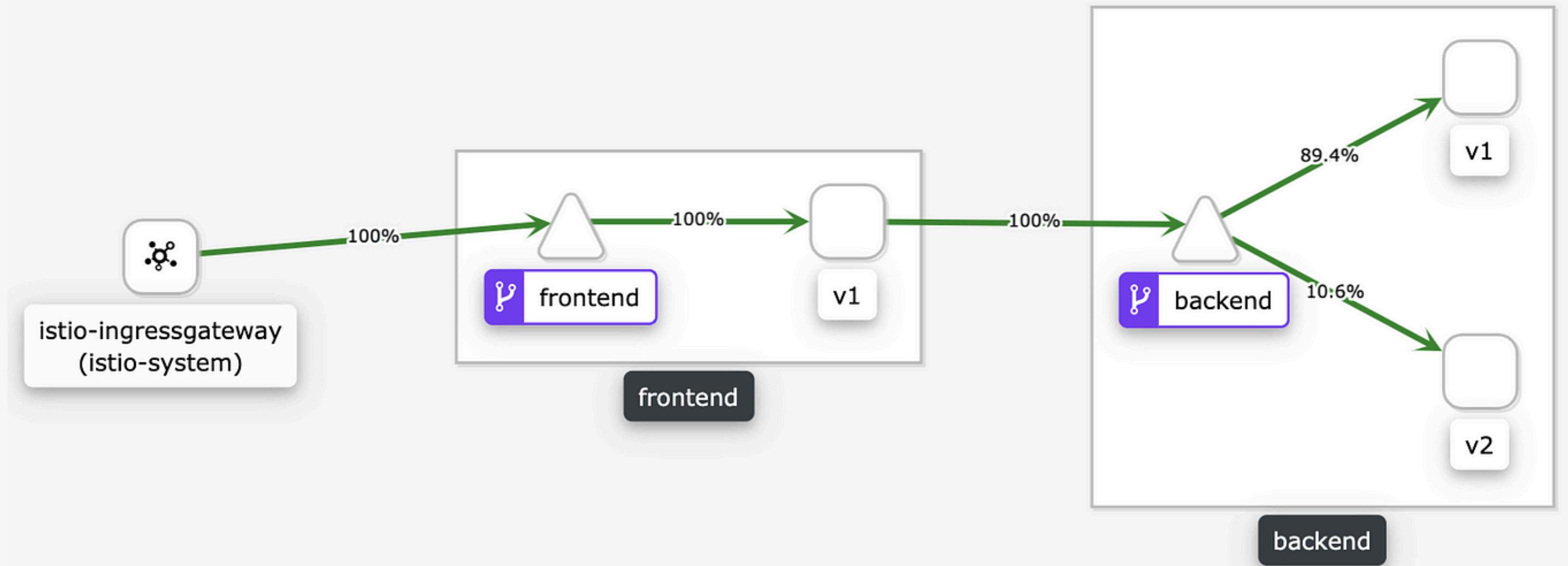
# Chapitre 7 : Comment mettre en place un réseau

- Appliquer et désactiver mTLS
- Rationalisation des configurations Kubernetes
- Politiques d'autorisation pour le Frontend et le Backend



# Chapitre 7 : Comment mettre en place un réseau

- Autorisation pour le Frontend
- Autorisation pour le backend
- Tests et résultats



# Chapitre 7 : Comment mettre en place un réseau

- Améliorations pratiques avec Istio
- Principaux enseignements sur les réseaux
- Architecture réseau complète en pratique

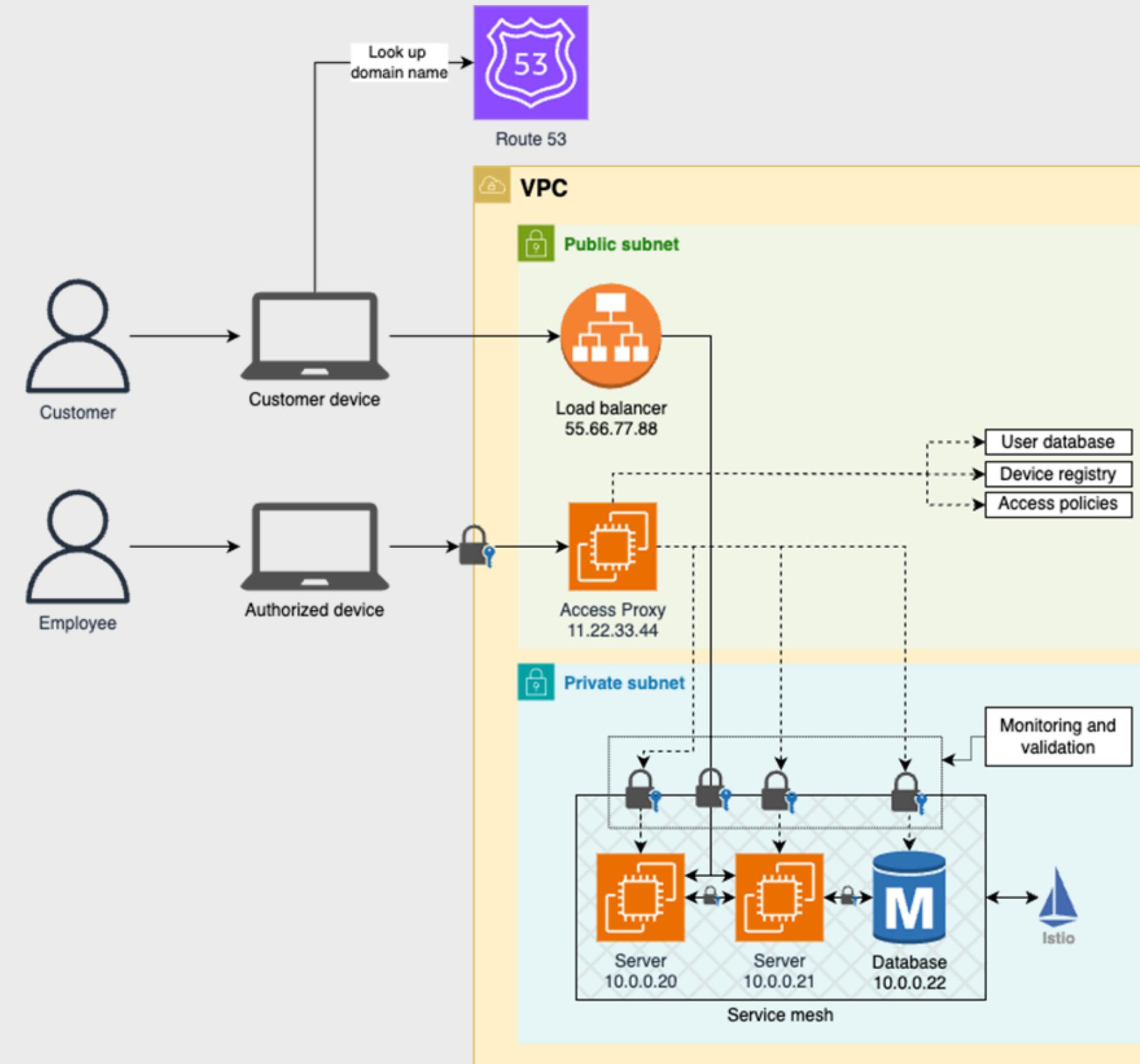


Figure 7-17. Full network architecture

# Thanks Folks!

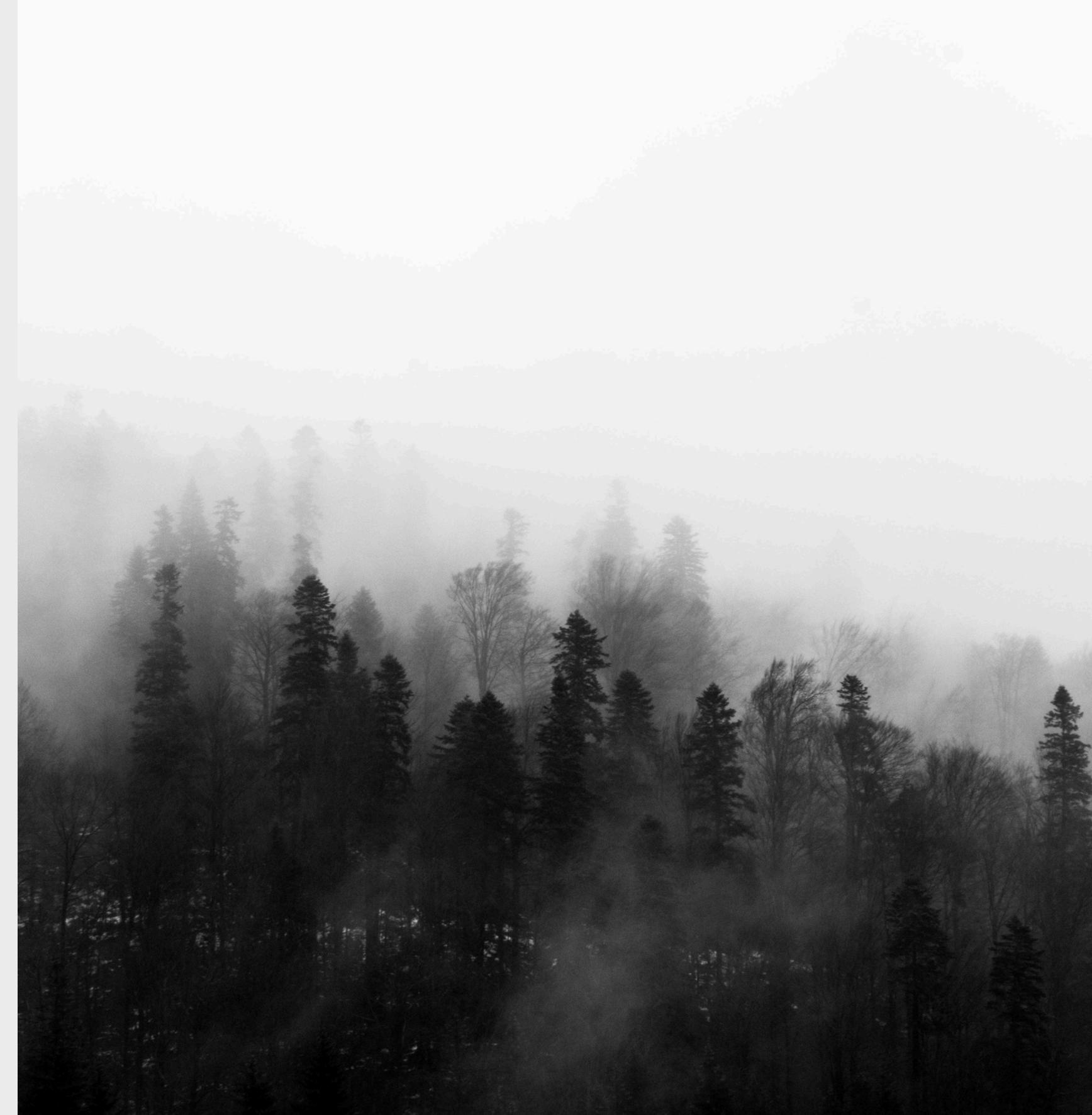


Auteur : **Badr TAJINI**

Année Universitaire : **2025-2026**

Ecole d'ingénieur : **ESIEE**

Merci de  
votre écoute !



Auteur : **Badr TAJINI**

Année Universitaire : **2025-2026**

Ecole d'ingénieur : **ESIEE**

# Références

- [1] Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations.*
- [2] References Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps : Building and Scaling High Performing Technology Organizations.*
- [3] References Kim, G., Behr, K., & Spafford, G. (2018). *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win.*
- [4] Arundel, J., & Domingus, J. (2019). *Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud.*
- [5] Tutorials : *GitHub Actions, Ansible, OpenTofu, Kubernetes*
- [6] Online TDD resources: practical examples and best practices - Link : <https://leanpub.com/tdd-ebook> or <https://github.com/grzesiek-galezowski/tdd-ebook>

