

**Sylvain Ard**

# **ECF Bachelor Devops**



**19/06/2024**

**studi**

# TABLE DES MATIERES

1. Introduction.....	2
1.1 Présentation de Studi .....	2
1.2 Travail de DevOps.....	3
2. Infrastructure as code .....	4
2.1 Présentation de l'infrastructure as code .....	4
2.2 Présentation de Terraform .....	6
2.3 Cahier des charges.....	7
2.4 Installation de terraform .....	7
2.5 Scripts terraform .....	8
3. CI/CD .....	25
3.1 Définition .....	25
3.2 Installation de NodeJS sur mon PC local .....	27
3.3 Création de l'application React .....	28
3.4 CI/CD proprement dit .....	30
4.Backups .....	36
5. Monitoring.....	38
5.1 Introduction au monitoring .....	38
5.2 Introduction à Centreon .....	39
5.3 Installation d'une instance Centreon pour surveiller les EC2 frontends et backends.....	40
6.Conclusion .....	59

# I. Introduction

## I.1 Présentation de Studi

Studi est une plateforme française de formation en ligne spécialisée dans l'enseignement supérieur et professionnel. Elle offre une variété de programmes de formation pour aider les apprenants à acquérir de nouvelles compétences et à progresser dans leur carrière.

### Caractéristiques Principales

1. Programmes Variés :
  - Offre des formations diplômantes et certifiantes dans divers domaines tels que le commerce, la gestion, le marketing, les ressources humaines, l'informatique, et plus encore.
2. Flexibilité :
  - Les cours sont accessibles en ligne, permettant aux apprenants d'étudier à leur propre rythme et selon leur emploi du temps.
3. Accompagnement Personnalisé :
  - Propose un suivi individualisé avec des tuteurs et des coachs pour aider les étudiants à réussir leur parcours de formation.
4. Interactivité :
  - Utilise des outils pédagogiques interactifs comme des vidéos, des quiz, des forums de discussion, et des travaux pratiques pour enrichir l'expérience d'apprentissage.
5. Reconnaissance Officielle :
  - Les formations proposées sont reconnues par l'État français et peuvent conduire à des diplômes ou des certifications reconnus sur le marché du travail.

### Avantages de Studi

1. Accessibilité :
  - Permet à chacun d'accéder à des formations de qualité depuis n'importe où, à tout moment.
2. Formation Professionnalisante :
  - Les programmes sont conçus pour répondre aux besoins du marché du travail et sont souvent créés en partenariat avec des entreprises.

### 3. Mise à Jour Continue :

- Les contenus de formation sont régulièrement mis à jour pour suivre les évolutions des secteurs professionnels et des technologies.

### 4. Communauté d'Apprenants :

- Favorise l'échange et le networking entre les étudiants grâce à une communauté active et des événements réguliers.

En résumé, Studi est une plateforme de formation en ligne flexible et accessible, offrant des programmes variés et reconnus, avec un accompagnement personnalisé pour assurer la réussite des apprenants.

## I.2 Travail de DevOps

DevOps combine développement logiciel (Dev) et opérations informatiques (Ops) pour améliorer l'efficacité et la vitesse de livraison des logiciels.

### Objectifs du DevOps

1. Automatisation : Automatiser le déploiement, les tests et la gestion des infrastructures.
2. CI/CD : Mettre en place des pipelines pour intégration et déploiement continus.
3. Collaboration : Faciliter la collaboration entre les équipes de développement et d'opérations.
4. Surveillance : Surveiller les performances des applications et de l'infrastructure.

### Responsabilités d'un DevOps

1. Gestion des Infrastructures : Utiliser des outils comme Terraform pour déployer les infrastructures.
2. Automatisation des Builds et Déploiements : Configurer des pipelines CI/CD avec Jenkins ou GitLab CI.
3. Surveillance et Logging : Utiliser Prometheus, Grafana, ELK Stack pour suivre les performances.
4. Sécurité : Intégrer la sécurité dans le cycle de développement.
5. Conteneurisation : Utiliser Docker et Kubernetes pour gérer les applications.

## Compétences Clés

- Techniques : Outils de gestion, scripting (Python, Bash).
- Collaboration : Travail efficace avec les équipes.
- Résolution de Problèmes : Identification et solution rapide des problèmes.
- Sécurité : Principes de sécurité des applications et infrastructures.

## Avantages

1. Livraison Rapide : Réduction du temps de mise en production.
2. Qualité Améliorée : Détection précoce des bugs.
3. Réduction des Risques : Déploiements plus fréquents et plus petits.
4. Efficacité : Automatisation des tâches répétitives.

En résumé, DevOps optimise les processus de développement et de déploiement, améliore la collaboration et assure une livraison rapide et sécurisée des logiciels.

Au cours du bachelor Devops de l'organisme STUDI nous devons réaliser un TP nommé « Evaluation en cours de formation » pour septembre 2024

## 2. Infrastructure as code

### 2.1 Présentation de l'infrastructure as code

#### IaC (Infrastructure as Code)

Infrastructure as Code (IaC) est une pratique de gestion de l'infrastructure informatique via des fichiers de configuration au lieu de processus manuels. Voici ses principales caractéristiques :

#### Caractéristiques de l'IaC

1. Automatisation :
  - Utilisation de scripts et fichiers de configuration pour automatiser le déploiement, la gestion et la mise à jour de l'infrastructure.
2. Consistance :
  - Garantit que l'infrastructure est déployée de manière cohérente chaque fois, éliminant les erreurs humaines.

### 3. Versionnement :

- Les fichiers de configuration peuvent être versionnés, permettant de suivre les modifications et de revenir à des versions précédentes si nécessaire.

### 4. Évolutivité :

- Facilite la mise à l'échelle de l'infrastructure en automatisant l'ajout et la suppression de ressources.

## Outils Courants

- Terraform : Provisionnement et gestion de l'infrastructure multi-cloud.
- AWS CloudFormation : Gestion de l'infrastructure sur AWS.
- Ansible : Automatisation de la configuration et de la gestion des systèmes.
- Puppet/Chef : Automatisation de la configuration et de la gestion des infrastructures.

## Avantages de l'IaC

### 1. Rapidité :

- Déploiement et configuration rapides de l'infrastructure, réduisant le temps de mise en production.

### 2. Fiabilité :

- Réduction des erreurs humaines grâce à l'automatisation et à la répétabilité des déploiements.

### 3. Gestion Facilitée :

- Simplification de la gestion de l'infrastructure grâce à la centralisation et au versionnement des configurations.

### 4. Coûts Réduits :

- Réduction des coûts opérationnels grâce à l'automatisation et à une gestion plus efficace des ressources.

En résumé, l'IaC permet une gestion plus rapide, plus fiable et plus efficace de l'infrastructure informatique, en automatisant les processus et en utilisant des fichiers de configuration pour déployer et gérer les ressources.

## 2.2 Présentation de Terraform

### Terraform

Terraform est un outil d'Infrastructure as Code (IaC) open source créé par HashiCorp, utilisé pour provisionner, gérer et versionner des ressources d'infrastructure de manière efficace et automatisée.

### Caractéristiques Principales

1. Provisionnement Multi-Cloud :
  - Terraform permet de gérer des infrastructures sur plusieurs fournisseurs de cloud (AWS, Azure, Google Cloud) et des services locaux avec une syntaxe unifiée.
2. Déclarations de Configuration :
  - Les infrastructures sont définies dans des fichiers de configuration en utilisant le langage HCL (HashiCorp Configuration Language), permettant une gestion claire et lisible de l'infrastructure.
3. Planification et Prévisualisation :
  - La commande `terraform plan` permet de prévisualiser les changements qui seront apportés à l'infrastructure avant de les appliquer, réduisant les risques d'erreurs.
4. Gestion des États :
  - Terraform maintient un fichier d'état qui conserve les informations sur les ressources provisionnées, assurant la cohérence entre les configurations et l'infrastructure réelle.
5. Modules et Réutilisabilité :
  - Les configurations peuvent être modulaires, facilitant la réutilisation de code et l'organisation des infrastructures complexes.

### Avantages de Terraform

1. Automatisation Complète :
  - Automatise le déploiement et la gestion de l'infrastructure, réduisant les efforts manuels et les erreurs humaines.

## 2. Scalabilité :

- Facilement scalable pour gérer des infrastructures de toutes tailles, des petites configurations aux environnements complexes et distribués.

## 3. Cohérence et Fiabilité :

- Garantit des déploiements cohérents et reproductibles grâce à des configurations déclaratives et au suivi des états.

## 4. Support Multi-Cloud :

- Gère plusieurs fournisseurs de cloud avec une seule interface, facilitant la gestion d'infrastructures hybrides et multi-cloud.

En résumé, Terraform est un outil puissant pour l'automatisation et la gestion d'infrastructures, offrant des fonctionnalités robustes pour le déploiement multi-cloud, la gestion d'état et la réutilisabilité des configurations.

## 2.3 Cahier des charges

Nous devons réaliser deux instances Front-end reliées par un load-balancer (un load-balancer est un service qui répartit la charge automatiquement entre plusieurs instances) et de même deux instances Back-End reliées par un load-balancer. Les front-end devaient accueillir une application ReactJS « Hello World » déployé par CI/CD et les instances back-end un « Hello-Word » en Java.

J'ai créé ces 5 instances par le logiciel Terraform (logiciel d'laas) sur la plateforme cloud « AWS ».

## 2.4 Installation de terraform

AWS CLI est une interface en ligne de commande pour AWS

J'ai téléchargé AWS CLI à l'adresse : [https://awscli.amazonaws.com/AWSCLIV 2.msi](https://awscli.amazonaws.com/AWSCLIV2.msi)

Puis j'ai lancé ce programme

Puis j'ai ouvert un cmd et ai fait la commande « aws configure » pour configurer AWS CLI

J'y ai mis mon « AM Access Key ID », mon « AM Secret Access Key », mon « Default region name » (us-east-1) et mon « Default output format » (json)



Puis j'ai téléchargé terraform sur

[https://releases.hashicorp.com/terraform/1.8.5/terraform\\_1.8.5\\_windows\\_amd64.zip](https://releases.hashicorp.com/terraform/1.8.5/terraform_1.8.5_windows_amd64.zip)

J'ai dézippé le fichier « terraform.exe » dans un répertoire « C:\terraform », puis je suis allé dans Panneau de configuration / Système / Paramètres système avancés, j'ai cliqué sur « Variables d'environnement », dans « Variables systèmes » j'ai cliqué sur « Path » puis « Modifier »

J'ai cliqué sur « Nouveau » et j'ai ajouté « C:\terraform » puis j'ai cliqué sur « OK » sur toutes les boîtes de dialogue.

## 2.5 Scripts terraform

Mon script Terraform est composé de 3 fichiers :

- main.tf : le programme principal
- variables.tf : la déclaration des variables
- terraform.tfvars : le contenu des variables (secret)

Voici le contenu de main.tf commenté :

```
provider "aws" {
  region = "us-east-1" # Remplacez par votre région AWS
}
```

**Cette section configure le fournisseur AWS et spécifie la région (us-east-1) où les ressources seront déployées.**

```
# Groupe de sécurité pour le front-end
resource "aws_security_group" "frontend_sg" {
  name      = "frontend-sg"
  description = "Allow HTTP and SSH traffic"
  vpc_id    = var.vpc_id

  ingress {
    from_port = 80
```

```

    to_port    = 80
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port = 22
    to_port   = 22
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port = 0
    to_port   = 0
    protocol  = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

```

**Ce groupe de sécurité permet le trafic HTTP (port 80) et SSH (port 22) entrant de n'importe où, et permet tout le trafic sortant. Il est associé à un VPC spécifique (`var.vpc_id`).**

```

# Groupe de sécurité pour le back-end
resource "aws_security_group" "backend_sg" {
  name      = "backend-sg"
  description = "Allow HTTP and SSH traffic"
  vpc_id    = var.vpc_id

  ingress {
    from_port = 80

```

```

to_port    = 80
protocol   = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}

```

```

ingress {
  from_port = 22
  to_port   = 22
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

```

```

egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
}

```

**Ce groupe de sécurité est similaire à celui du front-end, permettant également le trafic HTTP et SSH entrant et tout le trafic sortant.**

```

# Groupe de sécurité pour l'instance RDS
resource "aws_security_group" "rds_sg" {
  name      = "rds-sg"
  description = "Allow MySQL traffic"
  vpc_id    = var.vpc_id

  ingress {
    from_port = 3306
    to_port   = 3306

```

```

protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}

```

```

egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
}

```

**Ce groupe de sécurité permet le trafic MySQL entrant (port 3306) de n'importe où et tout le trafic sortant.**

```

# Paire de clés SSH
resource "aws_key_pair" "deployer_key" {
  key_name   = var.key_name
  public_key = file("~/ssh/${var.key_name}.pub")
}

```

**Cette ressource crée une paire de clés SSH pour permettre l'accès aux instances EC2. La clé publique est lue à partir d'un fichier local.**

```

# Instances EC2 pour le front-end
resource "aws_instance" "frontend_instance" {
  count      = 2
  ami       = var.ami_id
  instance_type = "t2.micro"
  key_name   = aws_key_pair.deployer_key.key_name
  security_groups = [aws_security_group.frontend_sg.name]
}

```

```
user_data = <<-EOF
  #!/bin/bash
  sudo yum update -y
  sudo yum install nginx -y
  sudo systemctl enable nginx
  sudo systemctl start nginx

  # Configure Nginx to serve the React application
  sudo cat > /etc/nginx/conf.d/default.conf <<EOL
  server {
    listen 80;
    server_name _;

    root /usr/share/nginx/html;
    index index.html;

    location / {
      try_files $uri $uri/ /index.html;
    }
  }
  EOL

  sudo systemctl restart nginx
  EOF

tags = {
  Name = "frontend-instance-${count.index}"
}
}
```

Cette ressource crée deux instances EC2 pour le front-end en utilisant une AMI spécifiée par `var.ami_id`. Elle utilise la paire de clés SSH définie précédemment et le groupe de sécurité du front-end. Le script `user_data` configure Nginx pour servir une application React.

```
# Instances EC2 pour le back-end
resource "aws_instance" "backend_instance" {
  count      = 2
  ami       = var.ami_id
  instance_type = "t2.micro"
  key_name   = aws_key_pair.deployer_key.key_name
  security_groups = [aws_security_group.backend_sg.name]

  user_data = <<-EOF
    #!/bin/bash
    sudo yum update -y
    sudo yum install java-11-amazon-corretto -y
    sudo yum install maven -y
    sudo yum install nginx -y
    sudo systemctl enable nginx
    sudo systemctl start nginx

    # Create a simple Spring Boot application
    mkdir -p /home/ec2-user/springboot-app
    cd /home/ec2-user/springboot-app

    # Create Spring Boot application files
    sudo tee /home/ec2-user/springboot-app/pom.xml > /dev/null <<EOL
    <project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>demo</name>
<description>Demo project for Spring Boot</description>
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.5.4</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<properties>
  <java.version>11</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
```

```

        </plugin>
    </plugins>
</build>
</project>
EOL

```

```

mkdir -p /home/ec2-user/springboot-app/src/main/java/com/example/demo
sudo tee /home/ec2-user/springboot-
app/src/main/java/com/example/demo/DemoApplication.java > /dev/null <<EOL
package com.example.demo;

```

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

```

```

@SpringBootApplication
public class DemoApplication {

```

```

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

```

```

    @RestController
    class HelloController {
        @GetMapping("/")
        public String hello() {
            return "Hello World!";
        }
    }
}
EOL

```



```
# Build and run the Spring Boot application
```

```
sudo mvn package
```

```
sudo nohup java -jar target/demo-0.0.1-SNAPSHOT.jar &
```

```
# Configure Nginx to proxy requests to the Spring Boot application
```

```
sudo tee /etc/nginx/conf.d/default.conf > /dev/null <<EOL
```

```
server {
```

```
    listen 80;
```

```
    server_name _;
```

```
    location / {
```

```
        proxy_pass http://localhost:8080;
```

```
        proxy_set_header Host $host;
```

```
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
    }
```

```
}
```

```
EOL
```

```
sudo systemctl restart nginx
```

```
EOF
```

```
tags = {
```

```
    Name = "backend-instance-${count.index}"
```

```
}
```

```
}
```

**Cette ressource crée deux instances EC2 pour le back-end. Le script `user_data` installe Java, Maven, et Nginx, et configure une application Spring Boot simple. Nginx est configuré pour rediriger les requêtes vers l'application Spring Boot.**

```
# Load Balancer pour le front-end
resource "aws_elb" "frontend_elb" {
  name          = "frontend-elb"
  availability_zones = ["us-east-1d"]
  security_groups = [aws_security_group.frontend_sg.id]

  listener {
    instance_port    = 80
    instance_protocol = "HTTP"
    lb_port          = 80
    lb_protocol       = "HTTP"
  }

  health_check {
    target          = "HTTP:80/"
    interval        = 30
    timeout         = 5
    healthy_threshold = 2
    unhealthy_threshold = 2
  }

  instances = aws_instance.frontend_instance[*].id
}
```

**Ce Load Balancer équilibre la charge entre les instances front-end. Il vérifie la santé des instances en envoyant des requêtes HTTP toutes les 30 secondes.**

```

# Load Balancer pour le back-end
resource "aws_elb" "backend_elb" {
  name          = "backend-elb"
  availability_zones = ["us-east-1d"]
  security_groups = [aws_security_group.backend_sg.id]

  listener {
    instance_port    = 80
    instance_protocol = "HTTP"
    lb_port          = 80
    lb_protocol       = "HTTP"
  }

  health_check {
    target          = "HTTP:80/"
    interval        = 30
    timeout         = 5
    healthy_threshold = 2
    unhealthy_threshold = 2
  }

  instances = aws_instance.backend_instance[*].id
}

```

**Ce Load Balancer équilibre la charge entre les instances back-end et vérifie leur santé de la même manière que celui du front-end.**

```

# Instance RDS
resource "aws_db_instance" "default" {
  allocated_storage = 5
  storage_type      = "gp2"
  engine            = "mysql"
}

```

```

engine_version    = "8.0"
instance_class    = "db.t3.micro"
identifier        = "mydb-instance"
username          = var.db_username
password          = var.db_password
parameter_group_name = "default.mysql8.0"
skip_final_snapshot = true
publicly_accessible = true
vpc_security_group_ids = [aws_security_group.rds_sg.id]

tags = {
  Name = "mydb"
}
}

```

**Cette ressource crée une instance RDS MySQL avec 5 Go de stockage. Elle utilise les identifiants de base de données fournis par des variables (`var.db_username` et `var.db_password`). L'instance est accessible publiquement et associée à un groupe de sécurité RDS.**

```

# AWS Backup Vault
resource "aws_backup_vault" "rds_backup_vault" {
  name = "rds-backup-vault"
}

```

**Ce coffre-fort de sauvegarde AWS est utilisé pour stocker les sauvegardes RDS.**

```

# IAM Role for AWS Backup
resource "aws_iam_role" "backup_role" {
  name = "backup-role"

  assume_role_policy = jsonencode({

```

```

Version = "2012-10-17"
Statement = [
  {
    Action = "sts:AssumeRole"
    Effect = "Allow"
    Principal = {
      Service = "backup.amazonaws.com"
    }
  }
]
})
}

```

**Ce rôle IAM permet à AWS Backup d'assumer ce rôle pour effectuer des opérations de sauvegarde.**

```

resource "aws_iam_role_policy_attachment" "backup_role_policy" {
  role      = aws_iam_role.backup_role.name
  policy_arn = "arn:aws:iam::aws:policy/service-role/AWSBackupServiceRolePolicyForBackup"
}

```

**Cette ressource attache une politique au rôle IAM, permettant à AWS Backup de gérer les sauvegardes.**

```

# AWS Backup Plan
resource "aws_backup_plan" "rds_backup_plan" {
  name = "rds-backup-plan"

  rule {
    rule_name      = "rds-12hour-backup"
    target_vault_name = aws_backup_vault.rds_backup_vault.name
  }
}

```

```

schedule      = "cron(0 */12 * * ? *)" # Cron expression for every 12 hours

lifecycle {
  delete_after = 30 # Number of days to retain the backup
}
}
}

```

**Ce plan de sauvegarde définit une règle pour sauvegarder l'instance RDS toutes les 12 heures et conserver les sauvegardes pendant 30 jours.**

```

# AWS Backup Selection
resource "aws_backup_selection" "rds_backup_selection" {
  name      = "rds-backup-selection"
  iam_role_arn = aws_iam_role.backup_role.arn
  plan_id    = aws_backup_plan.rds_backup_plan.id

  resources = [
    aws_db_instance.default.arn
  ]
}

```

**Cette ressource associe l'instance RDS au plan de sauvegarde, en utilisant le rôle IAM pour les opérations de sauvegarde.**

Un **Virtual Private Cloud (VPC)** est un service fourni par AWS qui vous permet de lancer des ressources AWS dans un réseau virtuel isolé. Voici quelques points clés pour comprendre ce qu'est un VPC :

1. **Isolation** : Le VPC offre un espace réseau isolé dans lequel vous pouvez définir vos propres adresses IP, sous-réseaux, et configurations de routage.

2. **Sous-réseaux** : Vous pouvez diviser votre VPC en sous-réseaux publics et privés pour organiser vos ressources. Les sous-réseaux publics ont accès à l'Internet, tandis que les sous-réseaux privés n'en ont pas.
3. **Contrôle du trafic** : Vous pouvez utiliser des tables de routage et des passerelles pour contrôler le trafic entrant et sortant de votre VPC. Cela inclut la configuration des routes vers Internet, d'autres VPC, ou des connexions VPN.
4. **Sécurité** : Avec un VPC, vous pouvez utiliser des groupes de sécurité et des listes de contrôle d'accès réseau (NACL) pour contrôler l'accès à vos ressources.

### Groupe de sécurité

Un **groupe de sécurité** (Security Group) est une couche de sécurité qui agit comme un pare-feu virtuel pour contrôler le trafic entrant et sortant de vos instances. Voici quelques points importants sur les groupes de sécurité :

1. **Règles d'Ingress** : Ce sont des règles qui contrôlent le trafic entrant vers vos instances. Vous pouvez définir quelles adresses IP ou plages d'adresses IP sont autorisées à se connecter à vos instances sur des ports spécifiques.
2. **Règles d'Egress** : Ce sont des règles qui contrôlent le trafic sortant de vos instances. Vous pouvez définir quelles adresses IP ou plages d'adresses IP vos instances peuvent contacter sur des ports spécifiques.
3. **Stateless vs Stateful** : Les groupes de sécurité sont stateful, ce qui signifie que si vous autorisez une connexion entrante, la réponse de cette connexion est automatiquement autorisée. En revanche, les listes de contrôle d'accès réseau (NACL) sont stateless, ce qui signifie que vous devez explicitement autoriser le trafic dans les deux sens.
4. **Portée** : Les groupes de sécurité peuvent être appliqués à des instances EC2, des interfaces réseau, des points de terminaison de service, etc. Ils permettent de contrôler l'accès au niveau de ces ressources.

### Exemple pour illustrer

Supposons que vous ayez un site web hébergé sur une instance EC2 dans AWS :

- Vous créez un VPC pour isoler votre infrastructure.
- Dans ce VPC, vous créez deux sous-réseaux : un sous-réseau public pour le serveur web et un sous-réseau privé pour une base de données.

- Vous créez un groupe de sécurité pour le serveur web avec des règles d'ingress qui permettent le trafic HTTP (port 80) et HTTPS (port 443) de n'importe où, et des règles d'ingress pour SSH (port 22) uniquement depuis votre adresse IP.
- Vous créez un groupe de sécurité pour la base de données qui permet uniquement le trafic entrant depuis le serveur web sur le port de la base de données (par exemple, MySQL sur le port 3306).

Ainsi, le VPC vous donne un contrôle total sur le réseau et la sécurité de vos ressources AWS, tandis que les groupes de sécurité vous permettent de définir des règles précises pour le trafic entrant et sortant vers vos instances.

## **Politique IAM (Identity and Access Management)**

Une politique IAM (Identity and Access Management) est un document JSON qui définit les permissions d'accès aux ressources AWS. Les politiques IAM permettent de spécifier les actions qu'un utilisateur, groupe, ou rôle IAM peut effectuer sur des ressources AWS spécifiques. Les politiques sont essentielles pour la gestion de la sécurité et du contrôle d'accès dans un environnement AWS.

### **Structure d'une Politique IAM**

Une politique IAM est composée de plusieurs éléments :

1. Version :
  - Indique la version du langage de politique. La version la plus courante est "2012-10-17".
2. Statement (Déclaration) :
  - Une politique peut contenir une ou plusieurs déclarations (statements), chacune définissant un ensemble de permissions. Chaque déclaration comprend les éléments suivants :
    - Effect : Spécifie si la déclaration accorde ou refuse l'accès ("Allow" ou "Deny").
    - Action : Spécifie les actions que la politique permet ou refuse, telles que s3:PutObject, ec2:StartInstances, etc.
    - Resource : Spécifie les ressources sur lesquelles les actions sont autorisées ou refusées, identifiées par leur ARN (Amazon Resource Name).



- Condition (facultatif) : Ajoute des conditions supplémentaires qui doivent être remplies pour que la politique soit appliquée. Par exemple, restreindre l'accès à partir d'une plage d'adresses IP spécifique ou à une certaine période.

## Rôle IAM (Identity and Access Management)

Un **rôle IAM** dans AWS (Amazon Web Services) est une identité IAM qui possède des permissions spécifiques, mais contrairement à un utilisateur IAM, il n'est pas associé à une seule personne ou application. Un rôle IAM est destiné à être assumé par toute entité de confiance qui en a besoin, comme une instance EC2, une fonction Lambda, ou même des utilisateurs d'autres comptes AWS. Voici une explication plus détaillée des rôles IAM :

### Caractéristiques des Rôles IAM

#### 1. Permissions Délégables :

- Les rôles IAM permettent de déléguer des permissions à des entités AWS ou à des utilisateurs. Par exemple, vous pouvez créer un rôle que les instances EC2 peuvent assumer pour obtenir des permissions d'accès aux buckets S3.

#### 2. Assumption de Rôle :

- Lorsqu'une entité de confiance (comme un service AWS, une application, ou un utilisateur d'un autre compte) assume un rôle, elle obtient temporairement les permissions associées à ce rôle.
- L'assumption de rôle se fait via des mécanismes comme `sts:AssumeRole` qui génère des informations d'identification temporaires (access keys, secret keys, session tokens).

#### 3. Politiques de Confiance :

- Une politique de confiance est un document JSON qui spécifie quelles entités peuvent assumer le rôle. Elle définit la relation de confiance entre le rôle et les entités de confiance.
- Par exemple, une politique de confiance peut permettre à une fonction Lambda ou à un service EC2 d'assumer le rôle.

#### 4. Politiques de Permissions :

- En plus de la politique de confiance, un rôle a des politiques de permissions attachées qui définissent ce que le rôle peut faire, c'est-à-dire les actions qu'il peut effectuer sur quelles ressources.

Pour créer les instances j'ai lancé successivement les commandes :

```
# Initialiser Terraform  
terraform init
```

```
# Générer le plan et l'enregistrer dans un fichier nommé tfplan  
terraform plan -var-file="terraform.tfvars" -out=tfplan
```

```
# Appliquer le plan enregistré  
terraform apply "tfplan"
```

pour détruire mes instances j'ai lancé la commande :

```
# Détruire les ressources sans demande de confirmation  
terraform destroy -var-file="terraform.tfvars" -auto-approve
```

## 3. CI/CD

### 3.1 Définition

**CI/CD (Continuous Integration and Continuous Delivery/Deployment)**

**CI/CD** est une pratique de développement logiciel qui automatise l'intégration, la livraison et le déploiement du code pour améliorer la qualité et accélérer la mise en production.

**Continuous Integration (CI)**

- **Intégration Fréquente : Les développeurs intègrent leur code régulièrement (au moins une fois par jour).**
- **Build et Tests Automatisés : Chaque intégration déclenche une build et des tests automatiques pour détecter rapidement les erreurs.**
- **Feedback Rapide : Les développeurs reçoivent des retours immédiats sur l'état de leur code.**

### Continuous Delivery (CD)

- **Déploiement Automatisé : Automatisation du déploiement vers des environnements de pré-production.**
- **Prêt pour la Production : Le code est toujours dans un état déployable.**
- **Pipeline de Déploiement : Une série d'étapes automatisées pour tester et déployer le code.**

### Continuous Deployment

- **Déploiement en Production Automatisé : Chaque modification validée est automatiquement déployée en production.**
- **Monitoring et Rollback : Surveillance continue et mécanismes de retour en arrière en cas de problème.**

### Avantages

- **Détection Précoce des Bugs : Identification et correction rapide des erreurs.**
- **Livraison Plus Rapide : Réduction du temps de mise en production grâce à l'automatisation.**
- **Amélioration de la Qualité : Tests continus améliorant la qualité du code.**
- **Réduction des Risques : Déploiements fréquents et de petite taille réduisant les risques.**

### Outils Communs

- **CI/CD Servers : Jenkins, GitLab CI, CircleCI**
- **Version Control : Git**
- **Containerization : Docker, Kubernetes**
- **Infrastructure as Code : Terraform**

- Monitoring : **Prometheus, Grafana**

En résumé, CI/CD est une pratique clé pour livrer rapidement du code de haute qualité en automatisant les processus d'intégration, de test et de déploiement.

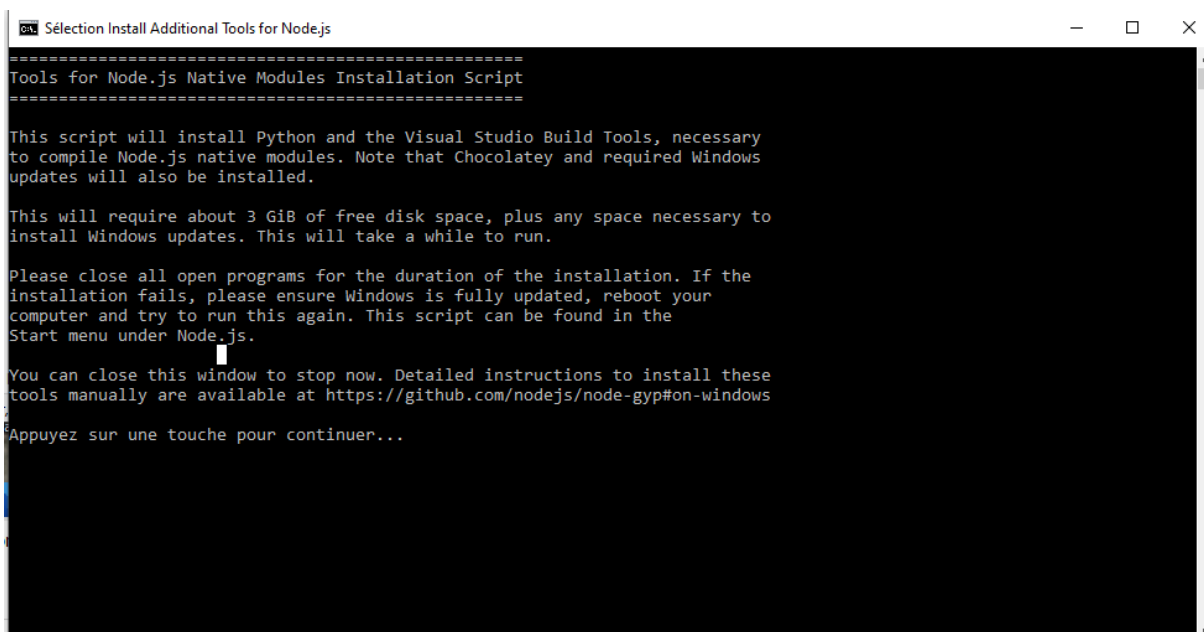
Je vais maintenant décrire les étapes que j'ai réalisées pour installer mon application ReactJS sur les instances frontend.

### 3.2 Installation de NodeJS sur mon PC local

Pour installer nodejs et npm je suis allé sur le site : <https://nodejs.org/en/>

J'ai cliqué sur le bouton « Download Node.js (LTS) »

J'ai lancé le programme « node-v20.14.0-x64.msi », j'ai cliqué sur « Next » sur le premier écran, puis j'ai accepté la licence et ai cliqué sur « Next », j'ai gardé l'emplacement d'installation par défaut et ai cliqué sur « Next », j'ai gardé les composants par défaut et ai cliqué sur « Next », j'ai coché « Automatically install the necessary tools (...) » et j'ai cliqué sur « Next », ensuite j'ai cliqué sur « Install » sur l'écran suivant. J'ai cliqué sur « Finish » sur le dernier écran. Une fenêtre MS-DOS s'est alors affichée :



```
Selection Install Additional Tools for Node.js
-----
Tools for Node.js Native Modules Installation Script
-----

This script will install Python and the Visual Studio Build Tools, necessary
to compile Node.js native modules. Note that Chocolatey and required Windows
updates will also be installed.

This will require about 3 GiB of free disk space, plus any space necessary to
install Windows updates. This will take a while to run.

Please close all open programs for the duration of the installation. If the
installation fails, please ensure Windows is fully updated, reboot your
computer and try to run this again. This script can be found in the
Start menu under Node.js.

You can close this window to stop now. Detailed instructions to install these
tools manually are available at https://github.com/nodejs/node-gyp#on-windows

Appuyez sur une touche pour continuer...
```

J'ai alors tapé une touche.

Cela a lancé PowerShell

```

Administrateur : Windows PowerShell
strapper.resources.dll...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\zh-Hant\vs_setup_
p_bootstrapper.resources.dll...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\es\vs_setup_bo
strapper.resources.dll...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\fr\vs_setup_bo
strapper.resources.dll...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\tr\vs_setup_bo
strapper.resources.dll...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\vs_setup_bootst
rapper.config...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\vs_setup_bootst
rapper.exe.config...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\detection.json...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\vs_setup_bootst
rapper.json...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\pt-BR\vs_setup_
bootstrapper.resources.dll...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\System.Runtime.
CompilerServices.Unsafe.dll...
Preparing: C:\Users\Sylvain\AppData\Local\Temp\chocolatey\142d4ce50ec4239f9197693eb5\vs_bootstrapper_d15\VSInstallerElev
ationService.Contracts.dll...
WARNING: Found 1 still running Visual Studio installer processes:
WARNING: [1128] setup
WARNING: Waiting for the processes to finish...
Visual Studio Installer has been installed.
Downloading visualstudio2019buildtools
from 'https://download.visualstudio.microsoft.com/download/pr/81bda3f8-b6f6-4caa-afe1-bfaaecb5ceb7/b792b2f59962c4f8765
fb9e2ccc10e1d7bc9e776fb6795ceaf6038e82a8bb1c/vs_BuildTools.exe'
Progress: 100% - Completed download of C:\Users\Sylvain\AppData\Local\Temp\chocolatey\visualstudio2019buildtools\16.11.3
6.0\vs_BuildTools.exe (3.79 MB).
Download of vs_BuildTools.exe (3.79 MB) completed.
Hashes match.
Installing visualstudio2019buildtools...

```

Plein de commandes ont alors été lancées, j'ai patienté.

Une fois terminé elle s'est automatiquement fermée.

J'ai alors tapé « node -v » dans une invite de commande cmd : cela m'a renvoyé « v20.14.0 » donc l'installation a marché.

Puis j'ai tapé « npm -v » dans la même invite et ai récupéré : 10.7.0 ce qui signifie que l'installation de npm a fonctionné.

### 3.3 Création de l'application React

J'ai navigué dans le dossier de mon dépôt git

J'ai tapé dans un cmd :

```
npx create-react-app hello-world-frontend
```

j'ai tapé « y » à la question puis « Entrée »

il a alors installé les dépendances

puis j'ai tapé :

```
cd hello-world-frontend
```

Puis je suis allé dans le dossier hello-world-frontend/src et ai remplacé l'ancien contenu du fichier « App.js » (l'appli) par ce contenu :

```
// src/App.js
```

```
import React from 'react';  
import './App.css';
```

```
function App() {  
  return (  
    <div className="App">  
      <header className="App-header">  
        <h1>Hello World!</h1>  
      </header>  
    </div>  
  );  
}
```

```
export default App;
```

j'ai modifié le contenu de App.test.js (le test de l'appli) par :

```
import { render, screen } from '@testing-library/react';  
import App from './App';
```

```
test('renders hello world text', () => {  
  render(<App />);  
  const linkElement = screen.getByText(/hello world/i);  
  expect(linkElement).toBeInTheDocument();  
});
```

### 3.4 CI/CD proprement dit

J'ai alors créé un dossier « .github/workflows » à l'intérieur de mon dossier git j'y ai mis à l'intérieur le code suivant :

**name:** CI/CD Pipeline

**on:**

**push:**

**branches:**

- main

**jobs:**

**copy:**

**runs-on:** ubuntu-latest

**steps:**

- **name:** Checkout code

**uses:** actions/checkout@v2

- **name:** Create .ssh directory

**run:** mkdir -p ~/.ssh

- **name:** Add EC2 Instance 1 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_1 }} >> ~/.ssh/known\_hosts

- **name:** Copy code to EC2 Instance 1

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_1 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

echo "\$KEY" > key.pem

chmod 600 key.pem

rsync -avz -e "ssh -i key.pem" hello-world-frontend/

\$USERNAME@\$HOST:/home/\$USERNAME/hello-world-frontend/

rm key.pem

- **name:** Add EC2 Instance 2 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_2 }} >> ~/.ssh/known\_hosts

- **name:** Copy code to EC2 Instance 2

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_2 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

echo "\$KEY" > key.pem

chmod 600 key.pem

rsync -avz -e "ssh -i key.pem" hello-world-frontend/  
\$USERNAME@\$HOST:/home/\$USERNAME/hello-world-frontend/  
rm key.pem

**test:**

**runs-on:** ubuntu-latest

**needs:** copy

**steps:**

- **name:** Create .ssh directory

**run:** mkdir -p ~/.ssh

- **name:** Add EC2 Instance 1 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_1 }} >> ~/.ssh/known\_hosts

- **name:** Test on EC2 Instance 1

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_1 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

echo "\$KEY" > key.pem

chmod 600 key.pem

ssh -i key.pem \$USERNAME@\$HOST "curl -fsSL  
[https://rpm.nodesource.com/setup\\_20.x](https://rpm.nodesource.com/setup_20.x) | sudo bash -"  
ssh -i key.pem \$USERNAME@\$HOST "sudo yum install -y nodejs"



```
ssh -i key.pem $USERNAME@$HOST "mkdir -p /home/$USERNAME/hello-world-frontend"
```

```
ssh -i key.pem $USERNAME@$HOST "cd /home/$USERNAME/hello-world-frontend && npm install"
```

```
ssh -i key.pem $USERNAME@$HOST "cd /home/$USERNAME/hello-world-frontend && npm test -- --watchAll=false"
```

```
rm key.pem
```

- **name:** Add EC2 Instance 2 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_2 }} >> ~/.ssh/known\_hosts

- **name:** Test on EC2 Instance 2

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_2 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

```
echo "$KEY" > key.pem
```

```
chmod 600 key.pem
```

```
ssh -i key.pem $USERNAME@$HOST "curl -fsSL https://rpm.nodesource.com/setup_20.x | sudo bash -"
```

```
ssh -i key.pem $USERNAME@$HOST "sudo yum install -y nodejs"
```

```
ssh -i key.pem $USERNAME@$HOST "mkdir -p /home/$USERNAME/hello-world-frontend"
```

```
ssh -i key.pem $USERNAME@$HOST "cd /home/$USERNAME/hello-world-frontend && npm install"
```

```
ssh -i key.pem $USERNAME@$HOST "cd /home/$USERNAME/hello-world-frontend && npm test -- --watchAll=false"
```

```
rm key.pem
```

**build:**

**runs-on:** ubuntu-latest

**needs:** test

**steps:**

- **name:** Create .ssh directory

**run:** mkdir -p ~/.ssh

- **name:** Add EC2 Instance 1 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_1 }} >> ~/.ssh/known\_hosts

- **name:** Build on EC2 Instance 1

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_1 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

echo "\$KEY" > key.pem

chmod 600 key.pem

ssh -i key.pem \$USERNAME@\$HOST "cd /home/\$USERNAME/hello-world-frontend && npm run build"

rm key.pem

- **name:** Add EC2 Instance 2 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_2 }} >> ~/.ssh/known\_hosts

- **name:** Build on EC2 Instance 2

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_2 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

echo "\$KEY" > key.pem

chmod 600 key.pem

ssh -i key.pem \$USERNAME@\$HOST "cd /home/\$USERNAME/hello-world-frontend && npm run build"

rm key.pem

**deploy:**

**runs-on:** ubuntu-latest

**needs:** build

**steps:**

- **name:** Create .ssh directory

**run:** mkdir -p ~/.ssh

- **name:** Add EC2 Instance 1 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_1 }} >> ~/.ssh/known\_hosts

- **name:** Deploy to EC2 Instance 1

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_1 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

echo "\$KEY" > key.pem

chmod 600 key.pem

ssh -i key.pem \$USERNAME@\$HOST "sudo cp -r /home/\$USERNAME/hello-world-frontend/build/\* /usr/share/nginx/html/ && sudo systemctl restart nginx"

rm key.pem

- **name:** Add EC2 Instance 2 to known\_hosts

**run:** ssh-keyscan -H \${{ secrets.EC2\_FRONTEND\_HOST\_2 }} >> ~/.ssh/known\_hosts

- **name:** Deploy to EC2 Instance 2

**env:**

**HOST:** \${{ secrets.EC2\_FRONTEND\_HOST\_2 }}

**USERNAME:** \${{ secrets.EC2\_USER }}

**KEY:** \${{ secrets.EC2\_KEY }}

**run:** |

echo "\$KEY" > key.pem

chmod 600 key.pem

ssh -i key.pem \$USERNAME@\$HOST "sudo cp -r /home/\$USERNAME/hello-world-frontend/build/\* /usr/share/nginx/html/ && sudo systemctl restart nginx"

rm key.pem

puis j'ai pushé le tout sur mon dépôt Github :

git add .

git commit -m « application react »

git push -u origin dev

Je suis allé sur AWS rubrique EC2 et ai regardé les IP v4 de mes instances EC2 frontend :

Name	ID d'instance	État de l'instance	Type d'instance	Contrôle des statuts	Statut d'alarme	Zone de disponibilité	DNS IPv4 public	Adresse IPv4 publique
frontend-insta...	i-06388525721c433fe	En cours d'...	t2.micro	2/2 vérifications	Afficher les alarmes	us-east-1d	ec2-18-212-254-125.co...	18.212.254.125
backend-insta...	i-0f8e643f99c94c6a1	En cours d'...	t2.micro	2/2 vérifications	Afficher les alarmes	us-east-1d	ec2-3-88-140-44.comp...	3.88.140.44
backend-insta...	i-0b7f20fe6a1b7f966	En cours d'...	t2.micro	2/2 vérifications	Afficher les alarmes	us-east-1d	ec2-100-27-202-201.co...	100.27.202.201
frontend-insta...	i-0d38303d824fd7734	En cours d'...	t2.micro	2/2 vérifications	Afficher les alarmes	us-east-1d	ec2-34-224-31-208.co...	34.224.31.208

Puis j'ai généré une paire de clés SSH sur mon ordi Windows local, pour cela j'ai d'abord créé un dossier « .ssh » dans le dossier « C:\Users\Sylvain » j'ai lancé PowerShell et j'y ai lancé la commande suivante :

```
ssh-keygen -t rsa -b 2048 -f C:\Users\Sylvain\.ssh\my-key-pair
```

j'ai mis une passphrase vide

cela m'a créé une paire de clé dans C:\Users\Sylvain\.ssh

j'ai ensuite copié le contenu de « my-key-pair.pub » dans la variable « public\_key » de terraform.tfvars

ensuite je suis allé dans settings/secrets and variables dans mon dépôt  
github/actions/new repository secret

et j'ai ajouté :

? **EC2\_FRONTEND\_HOST\_1** : L'adresse IP publique de ma première instance EC2 frontend.

? **EC2\_FRONTEND\_HOST\_2** : L'adresse IP publique de ma deuxième instance EC2 frontend.

? **EC2\_USER** : Le nom d'utilisateur SSH (par exemple, ec2-user pour Amazon Linux 2, ubuntu pour Ubuntu, etc.).

? **EC2\_KEY** : Le contenu du fichier .pem de ma clé privée.

Ensuite j'ai fusionné la branche dev dans la branche main

# Mettre à jour les branches locales

```
git fetch origin
```

```
# Basculer vers la branche main
```

```
git checkout main
```

```
# Mettre à jour la branche main
```

```
git pull origin main
```

```
# Fusionner la branche dev dans la branche main
```

```
git merge dev
```

```
# Pousser les modifications sur le dépôt distant
```

```
git push origin main
```

## 4.Backups

Après m'être connecté à AWS j'ai recherché « AWS Backup »

Ensuite j'ai cliqué sur « Coffres de sauvegarde » puis sur « rds-backup-vault »

**Mon compte**

- Tableau de bord
- Tableau de bord des tâches
- [Nouveau](#)
- Coffres de sauvegarde**
  - Verrouillages de coffre de sauvegarde [Nouveau](#)
  - Plans de backup
  - Ressources protégées
  - Tâches
  - Tests de restauration [Nouveau](#)
  - Conservations légales [Nouveau](#)
  - Paramètres

**Coffres de sauvegarde (1) [Infos](#)**


Les coffres de sauvegarde sont des conteneurs dans lesquels vos sauvegardes sont stockées. Les sauvegardes peuvent être stockées.

Rechercher par nom de coffre-fort ou état de verrouillage du coffre-fort


Nom du coffre de backup ▲	État de verrouillage du coffre ▼	Poi
<a href="#">rds-backup-vault</a>	–	2

J’y ai alors vu mes backups :

Résumé

Nom du coffre-fort rds-backup-vault	ID de clé de chiffrement KMS 5012f0b3-4ecd-4fc5-a76a-fa5475c530e3 <a href="#">🔗</a>	Verrouillage de coffre –
ARN du coffre  <code>arn:aws:backup:us-east-1:533267397760:backup-vault:rds-backup-vault</code>	Date de création 11 juin 2024, 12:51 (UTC+02:00)	Période de conservation pour le verrouillage de coffre Période de conservation minimale: – Période de conservation maximale: –

Points de restauration (2) [Infos](#)

 [Tout désélectionner](#) [Actions ▼](#)

<input type="checkbox"/>	ID du point de récupération ▼	Statut ▼	Nom de la ressource ▼	ID de ressource ▼
<input type="checkbox"/>	awsbackup:job-819d61f0-9d80-ce67-2271-0d9edb05c12c	✔ Terminé(e)	mydb-instance	mydb-instance
<input type="checkbox"/>	awsbackup:job-da0fefba-489d-9388-7619-b8d1d80dc8e9	✔ Terminé(e)	mydb-instance	mydb-instance

Puis j’ai cliqué sur « Plans de backup » puis sur « rds-backup-plan » puis dans « règles de backup » sur « rds-12hour-backup », j’ai pu alors voir que les paramètres étaient bons

[AWS Backup](#) > [Plans de backup](#) > [rds-backup-plan](#) > rds-12hour-backup

rds-12hour-backup

[Supprimer](#) [Modifier](#)

Résumé

Nom de la règle de backup rds-12hour-backup	Fréquence Toutes les 12 heures 0 minutes après l'heure, toutes les 12 heures	Commencer dans 1Heure	Terminer dans 3heures
Transition vers le stockage à froid Non activé	Archiver les instantanés Amazon EBS Non activé	Période totale de conservation 1 mois	Coffre de sauvegarde <a href="#">rds-backup-vault</a>
Sauvegarde en continu Désactivé	Balises ajoutées à des points de récupération - <i>facultatif</i> –		

## 5. Monitoring

Pour le monitoring j'ai choisi la solution « Centreon » car Cloud Watch la solution d'Amazon est trop chère.

### 5.1 Introduction au monitoring

**Monitoring** est le processus de collecte, d'analyse et d'interprétation des données de performance et de disponibilité des systèmes informatiques pour assurer leur bon fonctionnement.

#### Objectifs du Monitoring

1. **Disponibilité** : S'assurer que les systèmes, applications et services sont disponibles et fonctionnent correctement.
2. **Performance** : Suivre les performances pour garantir que les systèmes répondent aux exigences de performance.
3. **Détection de Problèmes** : Identifier rapidement les problèmes potentiels avant qu'ils n'affectent les utilisateurs finaux.
4. **Optimisation** : Utiliser les données collectées pour améliorer l'efficacité et les performances des systèmes.

#### Composants du Monitoring

1. **Collecte de Données** :
  - Mesurer divers paramètres (CPU, mémoire, réseau, etc.) à partir des systèmes et applications.
2. **Alertes et Notifications** :
  - Envoyer des alertes aux administrateurs en cas de dépassement des seuils définis ou d'anomalies détectées.
3. **Rapports et Dashboards** :
  - Fournir des visualisations et des rapports sur les données de performance et d'état.
4. **Analyse des Tendances** :
  - Analyser les données historiques pour identifier les tendances et prévoir les besoins futurs.

## Outils Courants

- **Nagios** : Monitoring des infrastructures.
- **Prometheus** : Collecte de métriques et alertes.
- **Grafana** : Visualisation des données de monitoring.
- **Centreon** : Supervision complète des infrastructures.

## Avantages du Monitoring

1. **Réactivité** : Permet de réagir rapidement aux incidents.
2. **Prévention** : Identification proactive des problèmes avant qu'ils n'affectent les utilisateurs.
3. **Optimisation** : Amélioration continue des performances des systèmes.
4. **Transparence** : Visibilité claire de l'état et des performances des infrastructures.

En résumé, le monitoring est essentiel pour maintenir la disponibilité, la performance et la fiabilité des systèmes informatiques, permettant une gestion proactive et une optimisation continue.

## 5.2 Introduction à Centreon

**Centreon** est une solution open source de supervision informatique utilisée pour surveiller les réseaux, serveurs, applications, bases de données, et services cloud. Voici ses principales caractéristiques :

### Caractéristiques Clés

1. **Supervision Multi-Plateforme : Surveille divers équipements et services, y compris les réseaux, serveurs, et applications.**
2. **Alertes et Notifications : Envoie des alertes par email, SMS, ou intégrations tiers en cas de problème.**
3. **Rapports et Dashboards : Crée des rapports et tableaux de bord personnalisés en temps réel.**
4. **Extensibilité : Utilise des plugins pour étendre les capacités de supervision.**
5. **Découverte Automatique : Identifie et ajoute automatiquement des nouveaux équipements et services.**
6. **Analyse des Performances : Analyse les données de performance pour identifier les tendances et prévenir les problèmes.**



## Avantages

- Open Source : **Gratuit et modifiable, réduisant les coûts.**
- Interface Web Intuitive : **Facile à utiliser et configurer.**
- Scalabilité : **Adapté aux petites et grandes infrastructures.**
- Communauté Active : **Support et contributions de la communauté, avec options de support commercial disponibles.**

## Utilisation

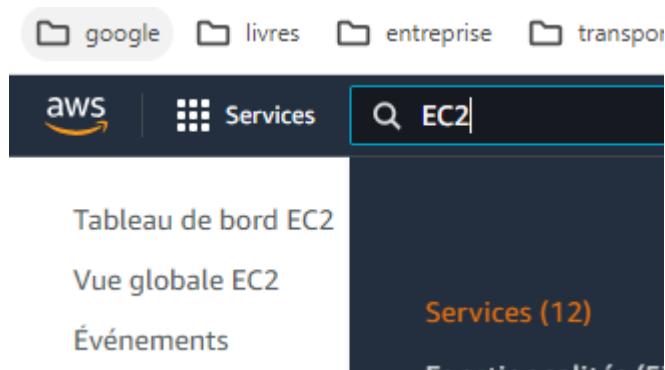
1. Installation : **Sur des distributions Linux comme CentOS et Debian, ou via Docker.**
2. Configuration : **Définir les hôtes et services à surveiller, les seuils, et les alertes.**
3. Surveillance : **Collecte et affiche des données en temps réel.**
4. Analyse : **Génère des rapports et identifie des tendances pour anticiper les problèmes.**

## Conclusion

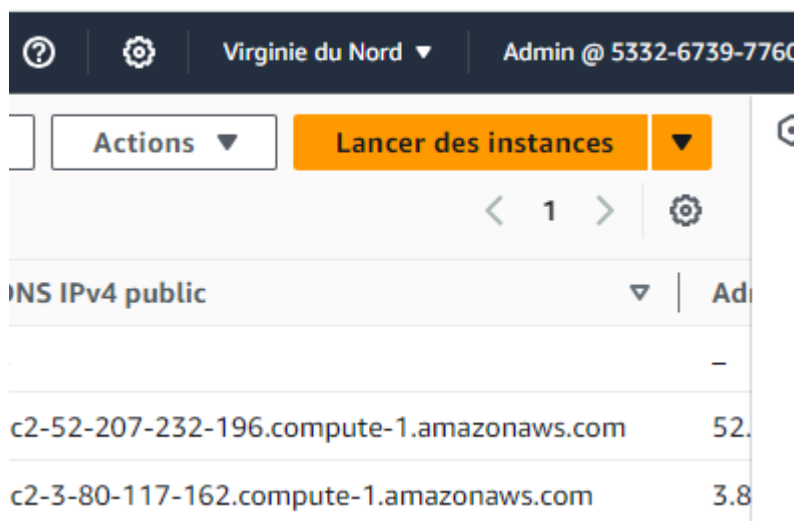
Centreon est une solution flexible et complète pour la supervision de l'infrastructure informatique, adaptée aux besoins des entreprises de toutes tailles.

## 5.3 Installation d'une instance Centreon pour surveiller les EC2 frontends et backends

Après m'être connecté à AWS, j'ai tapé « EC2 » dans la barre de recherche en haut :

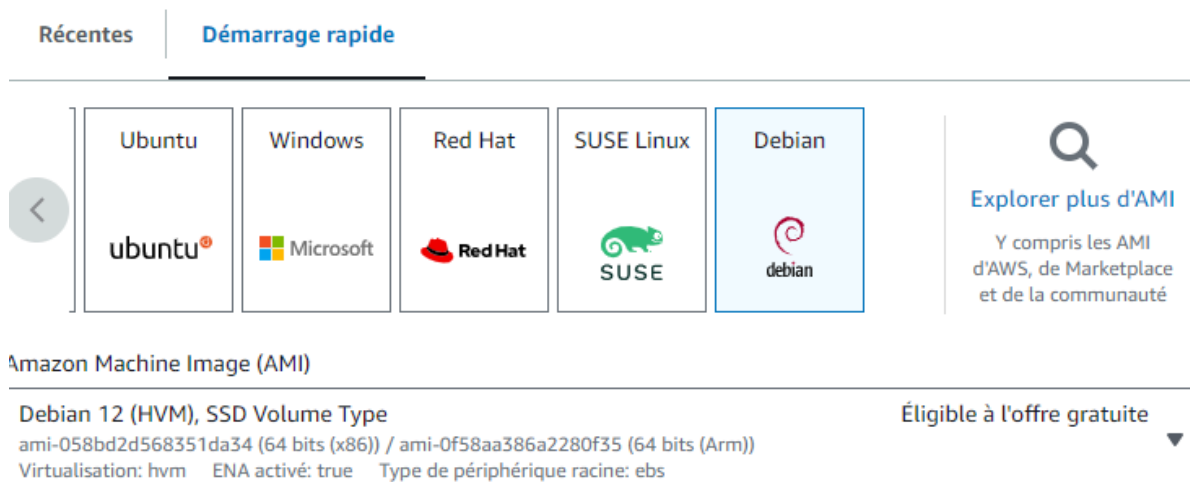


Puis j'ai cliqué sur le bouton « Lancer des instances » :



Dans le nom de l'instance j'ai mis « Monitoring »

J'ai choisi une image « Debian », pour cela j'ai fait défiler la liste sur la droite avec la flèche droite puis cliqué sur « Debian »



Dans « type d'instance » j'ai laissé « t2.micro » pour garder mon offre gratuite free tiers. Ensuite j'ai cliqué sur « Créer une paire de clés » et créé une paire de clé RSA pem que j'ai appelée « Paire\_monitoring »

## Créer une paire de clés



### Nom de la paire de clés

Les paires de clés vous permettent de vous connecter à votre instance en toute sécurité.

Paire\_monitoring

La longueur maximale du nom est de 255 caractères ASCII. Il ne peut pas inclure d'espaces avant ou après.

### Type de paire de clés



RSA

Paire de clés privée et publique chiffrée  
RSA



ED25519

Paire de clés privée et publique chiffrée  
ED25519

### Format de fichier de clé privée



.pem

À utiliser avec OpenSSH



.ppk

À utiliser avec PuTTY



Lorsque vous y êtes invité, stockez la paire de clés dans un emplacement sécurisé et accessible sur votre ordinateur. **Vous en aurez besoin ultérieurement pour vous connecter à votre instance.** [En savoir plus](#)

Annulez

Créer une paire de clés

Ensuite j'ai autorisé le trafic SSH, HTTP et HTTPS depuis n'importe où

Nous allons créer un nouveau groupe de sécurité appelé « **launch-wizard-2** » avec les règles suivantes :

- ☒ Autoriser le trafic SSH depuis  
Vous permet de vous connecter à votre instance.
- ☒ Autoriser le trafic HTTPS depuis l'Internet  
Pour configurer un point de terminaison, par exemple lors de la création d'un serveur web
- ☒ Autoriser le trafic HTTP depuis l'Internet  
Pour configurer un point de terminaison, par exemple lors de la création d'un serveur web

N'importe où  
0.0.0.0/0

⚠ Les règles avec la source 0.0.0.0/0 autorisent toutes les adresses IP à accéder à votre instance. Nous vous recommandons de définir des règles de groupe de sécurité pour autoriser l'accès à partir d'adresses IP connues uniquement.

Enfin j'ai cliqué sur « Lancer l'instance » à droite

Ensuite j'ai attendu que l'instance se crée.

J'ai ensuite tapé « CloudShell » dans AWS et ouvert CloudShell

Dans « actions » j'ai cliqué sur « charger un fichier »

J'ai chargé ma clé privée

Puis j'ai fait un chmod 400 sur ma clé privée pour la protéger

```
aws Services Rechercher

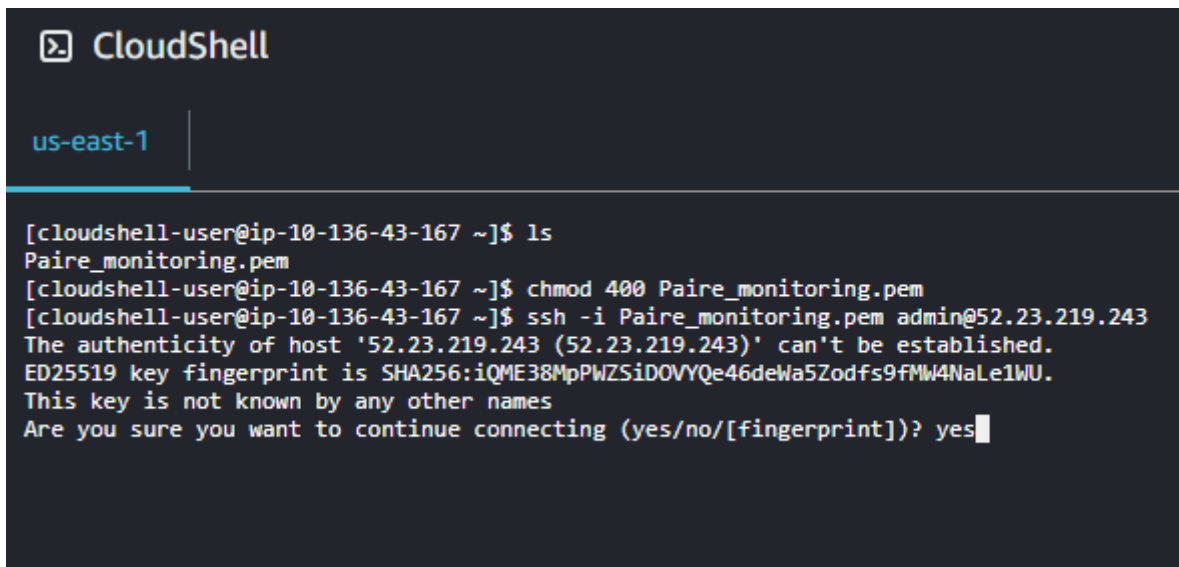
CloudShell

us-east-1

[cloudshell-user@ip-10-136-43-167 ~]$ ls
Paire_monitoring.pem
[cloudshell-user@ip-10-136-43-167 ~]$ chmod 400 Paire_monitoring.pem
[cloudshell-user@ip-10-136-43-167 ~]$
```

Puis j'ai fait : `ssh -i Paire_monitoring.pem admin@52.23.219.243`

J'ai répondu « yes » à la question



```
CloudShell
us-east-1

[cloudshell-user@ip-10-136-43-167 ~]$ ls
Paire_monitoring.pem
[cloudshell-user@ip-10-136-43-167 ~]$ chmod 400 Paire_monitoring.pem
[cloudshell-user@ip-10-136-43-167 ~]$ ssh -i Paire_monitoring.pem admin@52.23.219.243
The authenticity of host '52.23.219.243 (52.23.219.243)' can't be established.
ED25519 key fingerprint is SHA256:iQME38MpPWZSiDOVYQe46deWa5Zodfs9fMW4NaLe1WU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Et me voilà connecté à l'instance ! Après j'ai installé Centreon avec le tutorial <https://docs.centreon.com/fr/docs/installation/installation-of-a-central-server/using-packages/>

J'ai commencé par lancer la commande :

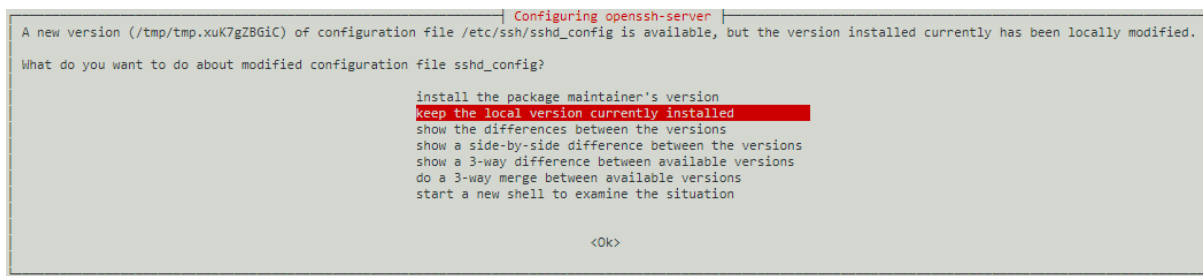
`sudo apt update`

puis :

`sudo apt upgrade -y`

Ceci pour mettre à jour Debian

Je suis tombé sur cette page :



```
Configuring openssh-server
A new version (/tmp/tmp.xuK7gZ8GiC) of configuration file /etc/ssh/sshd_config is available, but the version installed currently has been locally modified.
What do you want to do about modified configuration file sshd_config?

install the package maintainer's version
keep the local version currently installed
show the differences between the versions
show a side-by-side difference between the versions
show a 3-way difference between available versions
do a 3-way merge between available versions
start a new shell to examine the situation

<Ok>
```

J'ai gardé l'option par défaut dans le doute.

Je n'ai pas fait les deux commandes suivantes (`systemctl stop firewalld` et `systemctl disable firewalld`) car aucun firewall n'était installé

Puis j'ai lancé la commande suivante pour installer les dépendances :

`sudo apt update`

`sudo apt install lsb-release ca-certificates apt-transport-https software-properties-common wget gnupg2 curl`

J'ai tapé « Y » pour confirmer l'installation et ai pressé « Entrée »

Puis j'ai tapé la commande suivante pour installer le dépôt « Sury » :

```
sudo echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" | sudo tee
/etc/apt/sources.list.d/sury-php.list
```

Ensuite j'ai importé la clé du dépôt par :

```
sudo wget -O- https://packages.sury.org/php/apt.gpg | sudo gpg --dearmor | sudo tee
/etc/apt/trusted.gpg.d/php.gpg > /dev/null 2>&1
sudo apt update
```

Ensuite j'ai importé les dépôts centreon :

```
sudo echo "deb https://packages.centreon.com/apt-standard-24.04-stable/
$(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/centreon.list
sudo echo "deb https://packages.centreon.com/apt-plugins-stable/ $(lsb_release -sc)
main" | sudo tee /etc/apt/sources.list.d/centreon-plugins.list
```

ensuite j'ai importé la clé du dépôt par :

```
sudo wget -O- https://apt-key.centreon.com | sudo gpg --dearmor | sudo tee
/etc/apt/trusted.gpg.d/centreon.gpg > /dev/null 2>&1
sudo apt update
```

Ensuite j'ai installé Centreon par les commandes :

```
sudo apt install -y --no-install-recommends centreon-mariadb centreon
```

Ensuite j'ai défini le fuseau horaire de PHP :

```
sudo echo "date.timezone = Europe/Paris" >> /etc/php/8.1/mods-available/centreon.ini
```

Mais ça n'a pas marché :

```
admin@ip-172-31-42-123:~$ sudo echo "date.timezone = Europe/Paris" >>
/etc/php/8.1/mods-available/centreon.ini
```

```
-bash: /etc/php/8.1/mods-available/centreon.ini: Permission denied
```

J'ai alors édité le fichier centreon.ini par :

```
sudo nano /etc/php/8.1/mods-available/centreon.ini
```

et ai ajouté la ligne : date.timezone = Europe/Paris

puis j'ai redémarré php8-fpm par :

```
sudo systemctl restart php8.1-fpm
```

J'ai donc laissé tomber, une solution serait de créer l'instance dans la zone de Paris et non dans la zone us-east-1

Ensuite j'ai fait la commande suivante pour démarrer les services au démarrage du système :

```
sudo systemctl enable php8.1-fpm apache2 centreon cbd centengine gorgoned  
centreontrapd snmpd snmptrapd
```

J'ai ensuite lancé les commandes suivantes pour relancer mariadb :

```
sudo systemctl enable mariadb  
sudo systemctl restart mariadb
```

J'ai lancé alors la commande suivante pour sécuriser mariadb :

```
sudo mariadb-secure-installation
```

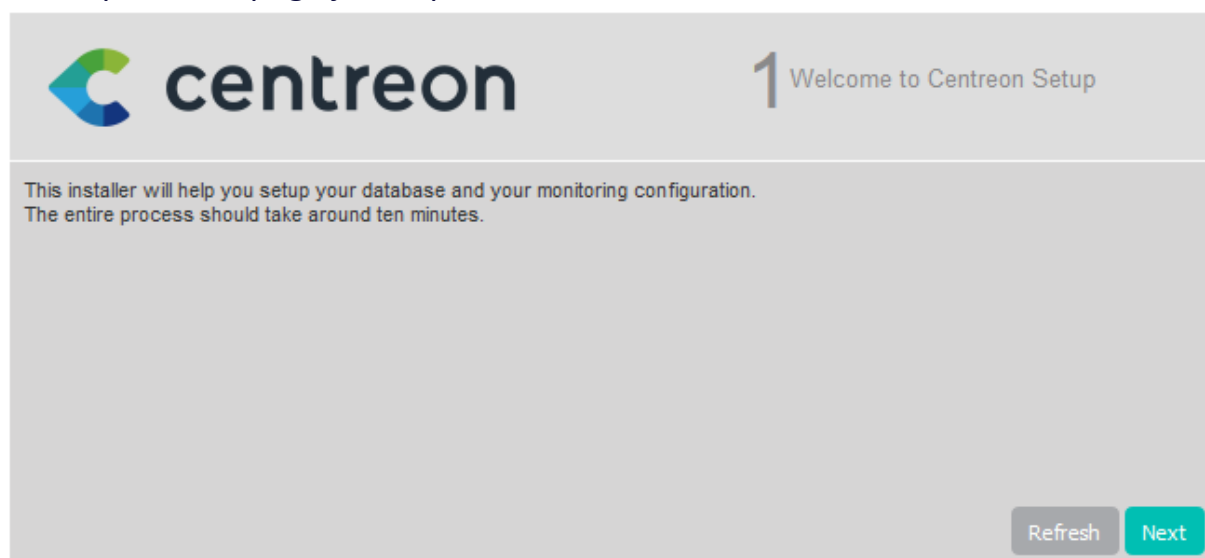
J'ai alors défini et redéfini le mot de passe root, répondu « y » à toutes les questions sauf « Disallow root login remotely ? » comme indiqué dans le tutorial

J'ai démarré le service apache avec :

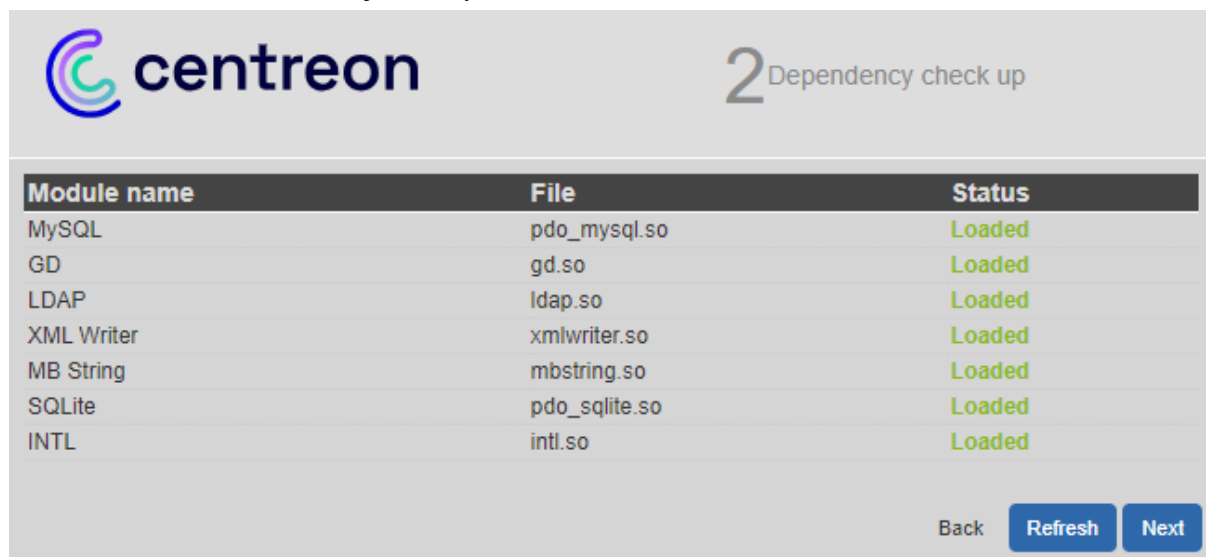
```
sudo systemctl start apache2
```

Puis je me suis connecté à l'adresse DNS de mon instance dans la liste des instances EC2 d'AWS pour passer à l'installation Web

Sur la première page j'ai cliqué sur « Next »



Puis sur l'écran suivant j'ai cliqué sur « Next »



Module name	File	Status
MySQL	pdo_mysql.so	Loaded
GD	gd.so	Loaded
LDAP	ldap.so	Loaded
XML Writer	xmlwriter.so	Loaded
MB String	mbstring.so	Loaded
SQLite	pdo_sqlite.so	Loaded
INTL	intl.so	Loaded

Back Refresh Next

Puis sur l'écran suivant j'ai cliqué sur « Next »

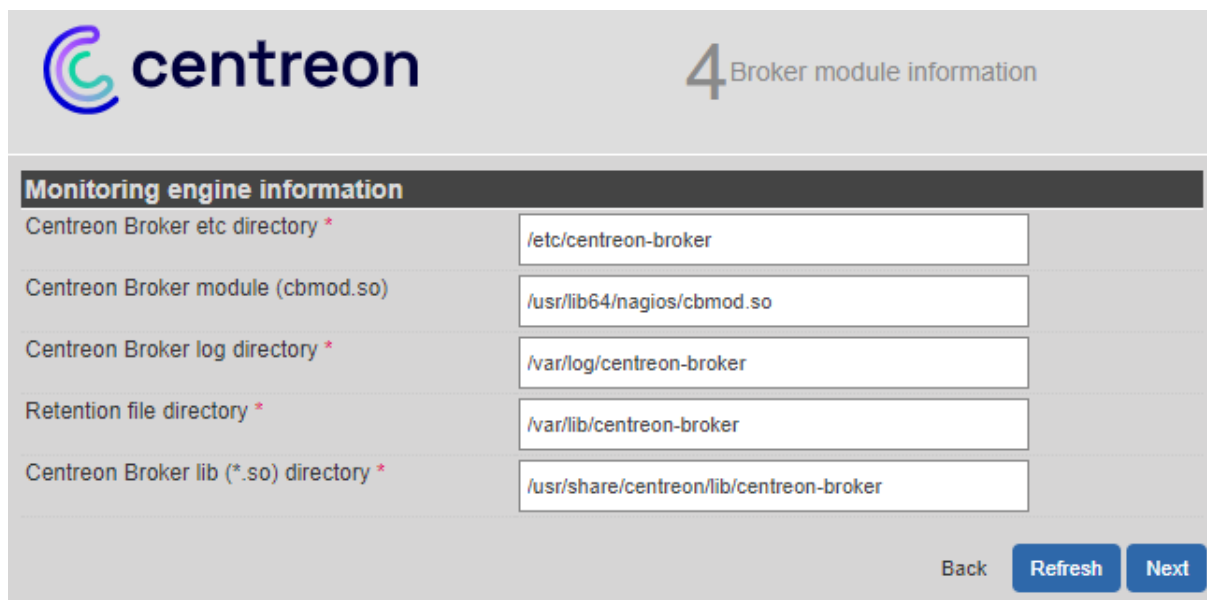


Monitoring engine information	
Centreon Engine Stats binary *	<input type="text" value="/usr/sbin/centenginestats"/>
Centreon Engine var lib directory *	<input type="text" value="/var/lib/centreon-engine"/>
Centreon Engine Connector path	<input type="text" value="/usr/lib64/centreon-connector"/>
Centreon Engine Library (*.so) directory *	<input type="text" value="/usr/lib64/centreon-engine"/>
Centreon Plugins Path *	<input type="text" value="/usr/lib/centreon/plugins/"/>

Back Refresh Next

Puis sur l'écran suivant j'ai cliqué sur « Next » :



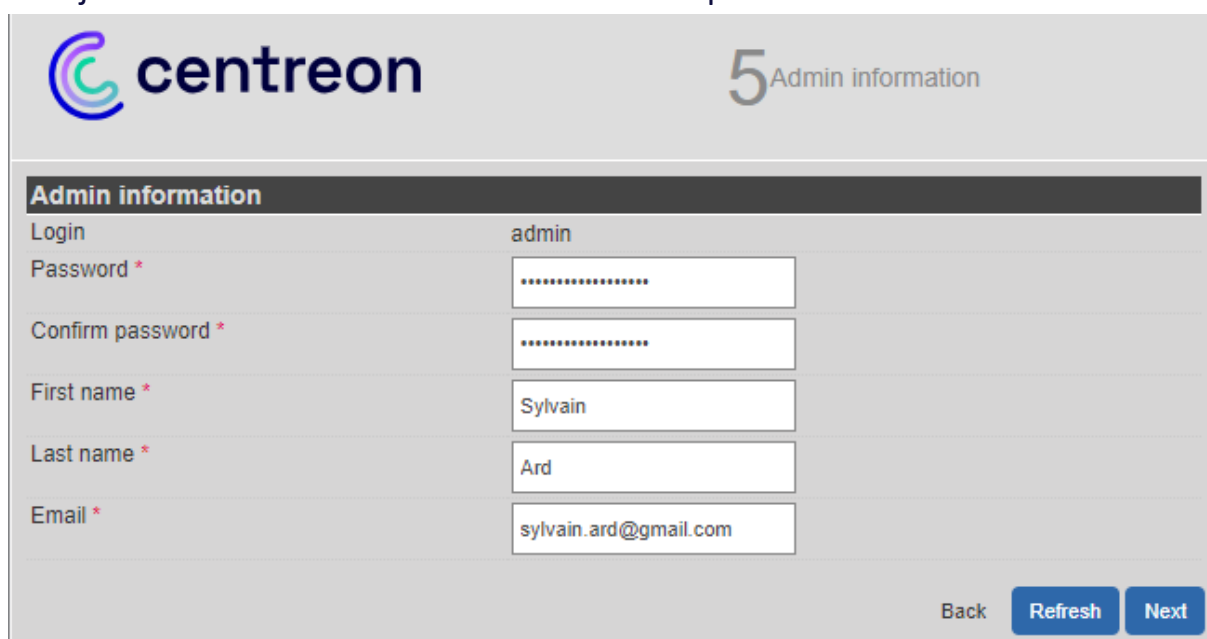


The screenshot shows the Centreon web interface at step 4, 'Broker module information'. The header includes the Centreon logo and the step number. The main content area is titled 'Monitoring engine information' and contains a table of configuration fields. At the bottom right, there are 'Back', 'Refresh', and 'Next' buttons.

Monitoring engine information	
Centreon Broker etc directory *	/etc/centreon-broker
Centreon Broker module (cbmod.so)	/usr/lib64/nagios/cbmod.so
Centreon Broker log directory *	/var/log/centreon-broker
Retention file directory *	/var/lib/centreon-broker
Centreon Broker lib (*.so) directory *	/usr/share/centreon/lib/centreon-broker

Back Refresh Next

Puis j'ai mis mes coordonnées et mon mot de passe sur l'écran suivant :

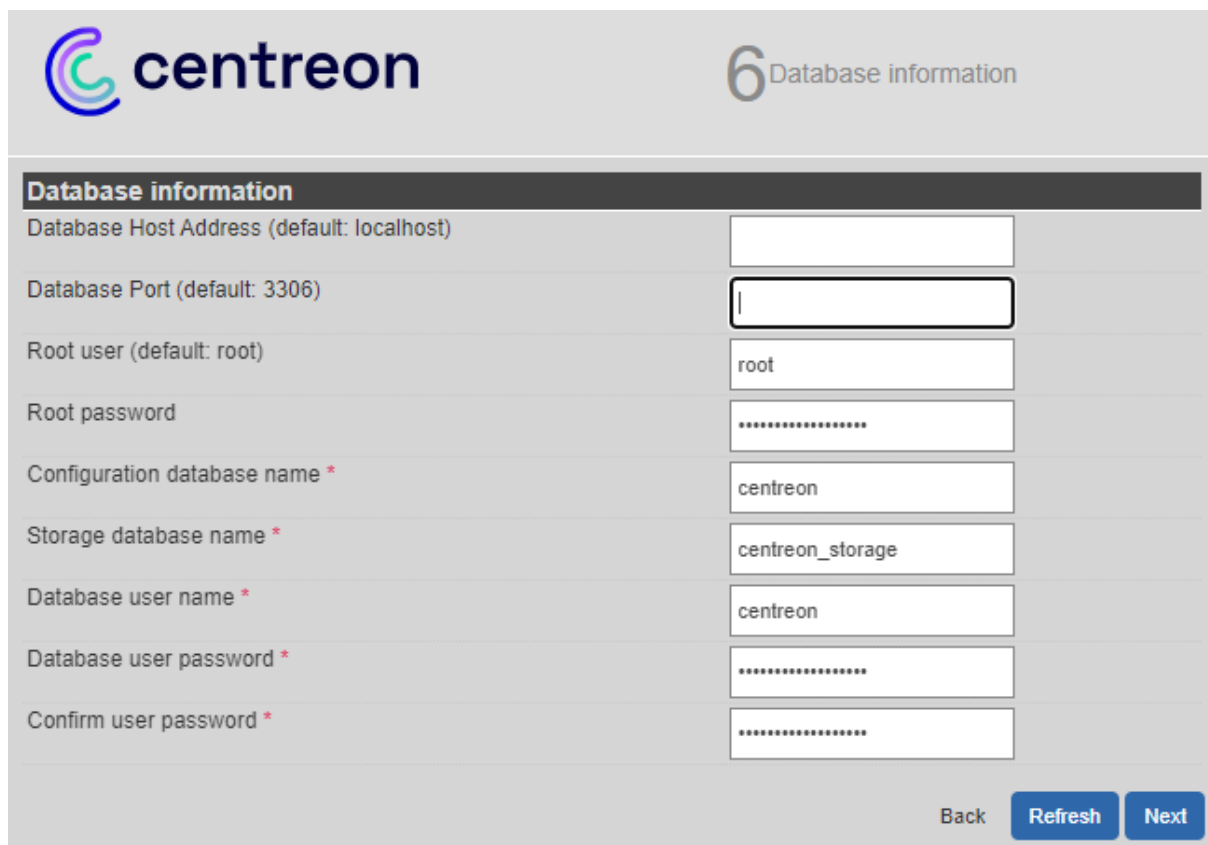


The screenshot shows the Centreon web interface at step 5, 'Admin information'. The header includes the Centreon logo and the step number. The main content area is titled 'Admin information' and contains a form for user registration. At the bottom right, there are 'Back', 'Refresh', and 'Next' buttons.

Admin information	
Login	admin
Password *	.....
Confirm password *	.....
First name *	Sylvain
Last name *	Ard
Email *	sylvain.ard@gmail.com

Back Refresh Next

Puis j'ai mis les coordonnées de la base de données (l'hôte et le port sont laissés vide pour garder leur valeur par défaut), le mot de passe root est celui défini par mariadb-secure-installation, le database user password est le mot de passe de l'utilisateur centreon qui sera créé à l'installation), j'ai laissé les autres champs par défaut :



**centreon** 6 Database information

**Database information**

Database Host Address (default: localhost)

Database Port (default: 3306)

Root user (default: root)

Root password

Configuration database name \*

Storage database name \*

Database user name \*

Database user password \*

Confirm user password \*

Back Refresh Next

L'installation de la base de données de Centreon a réussi comme indiqué dans ce panneau :



**centreon** 7 Installation

Currently installing database and generating cache... please do not interrupt this process.

Step	Status
Setting up configuration file	OK
Configuration database	OK
Storage database	OK
Creating database user	OK
Setting up basic configuration	OK
Partitioning database tables	OK
Generating application cache	OK

Next

J'ai alors cliqué sur « Next » et suis tombé sur l'écran d'installation des modules :



Module	Author	Version	
Centreon License Manager	Centreon	24.04.0	✓
Centreon IT Edition Extensions	Centreon	24.04.0	✓
Centreon Monitoring Connector Manager	Centreon	24.04.0	✓
Centreon Auto Discovery	Centreon	24.04.0	✓

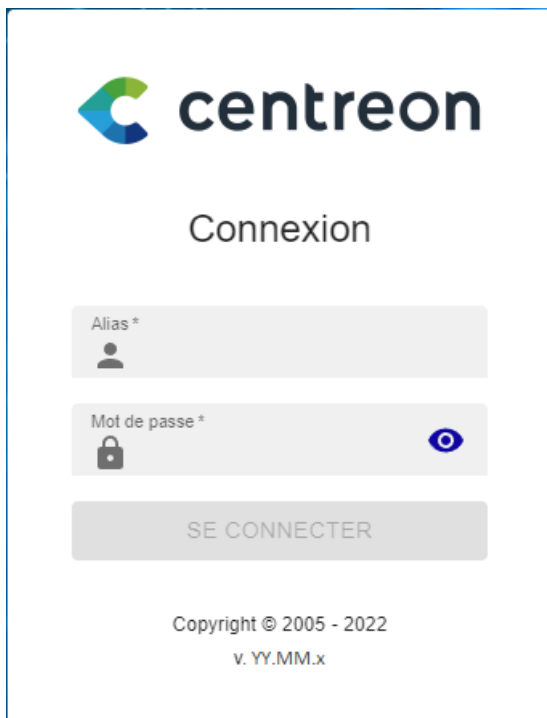
Widget	Author	Version	
Servicegroup Monitoring	Centreon	24.04.0	✓
Host Monitoring	Centreon	24.04.0	✓
NtopNG	Centreon	24.04.0	✓
HTTP Loader	Centreon	24.04.0	✓
Engine-status	Centreon	24.04.0	✓
Live Top 10 Memory Usage	Centreon	24.04.0	✓
Single Metric	Centreon	24.04.0	✓
Hostgroup Monitoring	Centreon	24.04.0	✓
Tactical Overview	Centreon	24.04.0	✓
Live Top 10 CPU Usage	Centreon	24.04.0	✓
Grid-map	Centreon	24.04.0	✓
Graph Monitoring	Centreon	24.04.0	✓
Global Health	Centreon	24.04.0	✓
Service Monitoring	Centreon	24.04.0	✓

Refresh Install

J'ai alors cliqué sur « Install » puis « Next »

Puis l'écran de fin s'est affiché et j'ai cliqué sur « Finish »

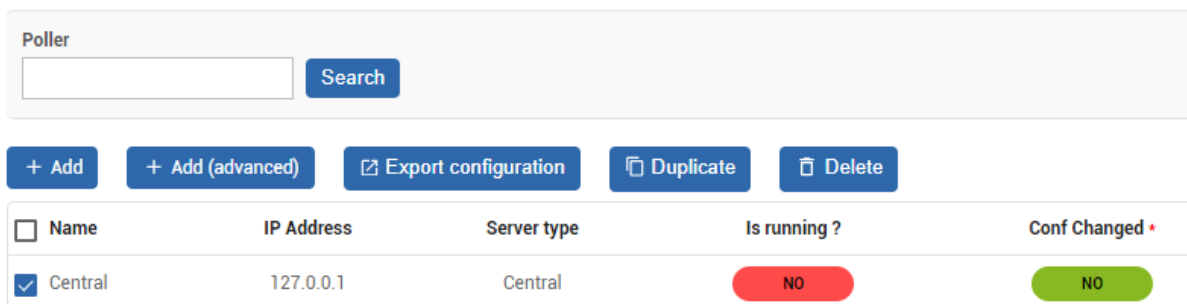
Sur l'écran suivant je me suis connecté avec le login « admin » et le mot de passe défini précédemment



The image shows the Centreon login interface. At the top is the Centreon logo, which consists of a stylized 'C' in blue and green followed by the word 'centreon' in a dark blue sans-serif font. Below the logo is the word 'Connexion' in a smaller, dark blue font. There are two input fields: the first is labeled 'Alias \*' with a person icon on the left, and the second is labeled 'Mot de passe \*' with a lock icon on the left and an eye icon on the right. Below these fields is a grey button with the text 'SE CONNECTER'. At the bottom, there is a copyright notice: 'Copyright © 2005 - 2022' and 'v. YY.MM.x'.

Dans la fenêtre de Centreon j'ai cliqué sur la roue dentée (configuration) puis « Pollers »/ « Pollers », j'ai coché « Central » puis j'ai cliqué sur « Export configuration »

Configuration > Pollers



The image shows the 'Pollers' configuration page in Centreon. At the top, there is a search bar with the label 'Poller' and a 'Search' button. Below the search bar are five buttons: '+ Add', '+ Add (advanced)', 'Export configuration' (with an external link icon), 'Duplicate' (with a copy icon), and 'Delete' (with a trash icon). Below these buttons is a table with the following columns: 'Name', 'IP Address', 'Server type', 'Is running?', and 'Conf Changed'. The table contains one row for 'Central' with IP address '127.0.0.1' and server type 'Central'. The 'Is running?' column has a red 'NO' button, and the 'Conf Changed' column has a green 'NO' button.

<input type="checkbox"/> Name	IP Address	Server type	Is running ?	Conf Changed <span style="color: red;">*</span>
<input checked="" type="checkbox"/> Central	127.0.0.1	Central	NO	NO

J'ai coché « Move export files » puis cliqué sur « Export »

The screenshot shows the Nagios Core web interface. On the left, a sidebar menu includes 'Custom Views', 'Dashboards', and 'Polling instances'. The main content area is titled 'Polling instances' and shows a list of pollers. One poller, 'Central', is selected. Below the poller list, there is an 'Actions' section with several checkboxes: 'Generate Configuration Files' (checked), 'Run monitoring engine debug (-v)' (checked), 'Move Export Files' (checked), 'Restart Monitoring Engine' (unchecked), and 'Post generation command' (unchecked). The 'Restart Monitoring Engine' action has a 'Method' dropdown menu set to 'Reload'. At the bottom right of the interface, there is a blue 'Export' button.

Puis dans CloudShell j'ai tapé la commande suivante pour démarrer/redémarrer le processus de collecte :


```
sudo systemctl restart cbd centengine
```

puis j'ai redémarré le gestionnaire de tâche :

```
sudo systemctl restart gorgoned
```

puis j'ai démarré les services de supervision passive :

```
sudo systemctl start snmptrapd centreontrapd
```

Puis j'ai cliqué sur cet icône (Administration) :  puis sur « Extensions »/ « Manager »

Puis j'ai suivi ce tutorial : <https://www.youtube.com/watch?v=m8aXs53C9eg>

Ensuite j'ai créé un utilisateur IAM pour Centreon, pour cela j'ai tapé « IAM » en haut et ai cliqué sur « Utilisateurs » à gauche, puis j'ai créé un utilisateur nommé « centreon » avec les paramètres de l'écran suivant :

Nom d'utilisateur

centreon

Le nom d'utilisateur peut comporter jusqu'à 64 caractères. Caractères valides : A-Z, a-z, 0-9 et +, -, @, \_ - (tiret)

☒ Fournir aux utilisateurs l'accès à la console de gestion AWS - *facultatif*  
Si vous fournissez à une personne l'accès à la console, c'est aux *bonne pratique* de gérer leur accès dans IAM Identity Center.

**Informations** Fournissez-vous à une personne un accès à la console ?

Type d'utilisateur

☐ Spécifier un utilisateur dans Identity Center - *recommandé*  
Pour accorder à une personne l'accès à la console, nous vous recommandons d'utiliser Identity Center. Grâce à cet outil, vous centralisez la gestion de l'accès des utilisateurs à leurs comptes AWS et à leurs applications cloud.

☒ Je souhaite créer un utilisateur IAM  
La création d'utilisateurs IAM est recommandée uniquement en cas de besoin d'accès par programmation à AWS CodeCommit ou Amazon Keyspaces via des clés d'accès ou des informations d'identification spécifiques à un service, ou en cas de besoin d'un accès d'urgence à un compte via des informations d'identification de secours.

Mot de passe de la console

☒ Mot de passe généré automatiquement  
Vous pouvez afficher le mot de passe après avoir créé l'utilisateur.

☐ Mot de passe personnalisé  
Saisissez un mot de passe personnalisé pour l'utilisateur.

☐ Afficher le mot de passe

☐ Les utilisateurs doivent créer un nouveau mot de passe lors de la prochaine connexion (recommandé)  
Les utilisateurs obtiennent automatiquement la politique IAMUserChangePassword les autorisant à modifier leur propre mot de passe.

**Info** En cas de création d'un accès par programmation à AWS CodeCommit ou Amazon Keyspaces via des clés d'accès ou des informations d'identification spécifiques à un service, vous pourrez les générer après avoir créé cet utilisateur IAM. [En savoir plus](#)

Annuler Suivant

Dans « Régler les autorisations » j'ai cliqué sur « Attacher directement les politiques »  
J'ai alors ajouté les politiques « [AmazonEC2ReadOnlyAccess](#) » et  
« [CloudWatchReadOnlyAccess](#) »  
J'ai ensuite fait « Suivant » pour aboutir sur ce panneau récapitulatif :

Vérifier et créer

Vérifiez vos choix. Après avoir créé l'utilisateur, vous pouvez afficher et télécharger le mot de passe généré automatiquement, si cette option est activée.

**Détails de l'utilisateur**

Nom d'utilisateur centreon	Type de mot de passe de la console Autogenerated	Demander la réinitialisation du mot de passe Non
-------------------------------	---	---

**Résumé des autorisations**

Nom	Type	Utilisé comme
<a href="#">AmazonEC2ReadOnlyAccess</a>	Gérées par AWS	Stratégie des autorisations
<a href="#">CloudWatchReadOnlyAccess</a>	Gérées par AWS	Stratégie des autorisations

**Balises - facultatif**  
Les identifications sont des paires clé-valeur que vous pouvez ajouter aux ressources AWS pour vous aider à identifier, organiser ou rechercher des ressources. Choisissez les identifications que vous voulez associer à cet utilisateur.

Aucune identification associée à la ressource.

[Ajouter une nouvelle identification](#)  
Vous pouvez ajouter jusqu'à 50 identifications supplémentaires.

Annuler Précédent Créer un utilisateur

J'ai fait alors « créer un utilisateur »  
J'ai copié collé les logins et mots de passe pour m'en souvenir

J'ai fait alors « Revenir à la liste des utilisateurs » puis j'ai cliqué sur l'utilisateur « centreon », j'ai alors cliqué à droite sur « Créer une clé d'accès »

J'ai cliqué sur « Interface en ligne de commande (CLI) et coché « Je comprends la recommandation ci-dessus et je souhaite procéder à la création d'une clé d'accès. » puis sur « Suivant » . Dans « Description de la clef d'identification » j'ai mis « centreon »

Puis j'ai copié en lieu sûr les credentials

J'ai arrêté le tutorial là car il ne correspondait pas à mon interface j'ai repris ce tutorial : <https://docs.centreon.com/fr/pp/integrations/plugin-packs/procedures/cloud-aws-ec2/>

j'ai lancé les commande suivante :

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt-get install unzip
unzip awscliv2.zip
sudo ./aws/install
```

j'ai eu en retour :

```
admin@ip-172-31-42-123:~$ sudo ./aws/install
```

Puis je suis allé sur roue dentée (configuration) puis « Monitoring Connector Manager »

J'ai tapé « EC2 » dans « Keyword » puis j'ai cliqué sur le carré « Amazon EC2 » puis j'ai cliqué dessus j'ai vu que c'était payant.

Je suis donc allé sur <https://www.centreon.com/free-trial/>

J'ai rempli mes coordonnées

J'ai alors reçu un token par mail

J'ai lu alors ce tutoriel : <https://docs.centreon.com/fr/docs/administration/licenses/>

Je suis donc allé dans Administration/Parameters/Centreon UI dans Proxy j'ai cliqué sur « Test Internet Connection », j'ai eu « Connection successful »

Ensuite je suis allé dans Administration/Extensions/Manager

J'ai cliqué sur « Add token » et y ai collé mon token

Voilà ma licence était installée

J'ai donc recommencé

Je suis allé sur roue dentée (configuration) puis « Monitoring Connector Manager »

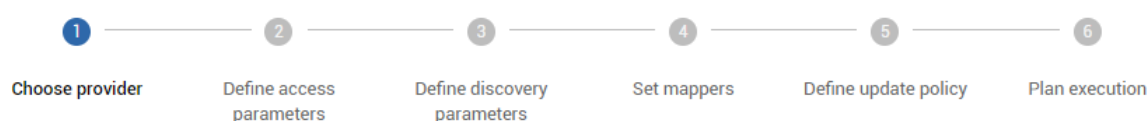
J'ai tapé « EC2 » dans « Keyword » puis j'ai passé ma souris sur « Amazon EC2 » , cliqué sur la croix verte et appuyé sur « Apply » sur la fenêtre qui s'est affichée

Puis j'ai lancé la commande suivante pour installer le plugin :

```
sudo apt install centreon-plugin-cloud-aws-ec2-api
```

J'ai alors repris la vidéo

Ensuite je suis allé dans Configuration/Hosts/Discovery, j'ai cliqué sur « Amazon AWS EC2 » puis sur « Next »



### Choose a name

Job's name  
Amazon AWS EC2

### Choose a provider

Search provider

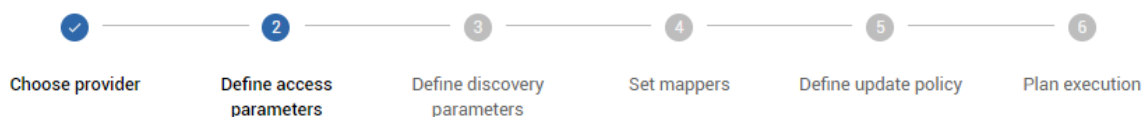


Next

J'ai cliqué sur le petit plus en face de « Choose credentials » et y ai mis mes credentials pour l'utilisateur « centreon »

J'ai laissé les champs « proxy » vides et suis passé à la suite en cliquant sur « Next »





### Choose a monitoring server

Monitoring server \*

Central

### Define a proxy

Url

Port

User

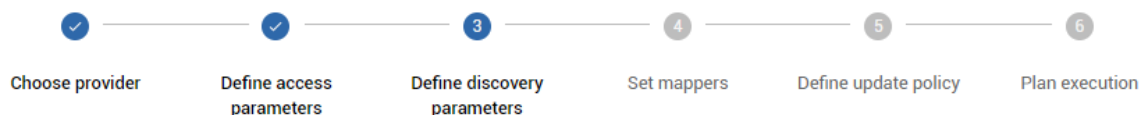
Password

### Choose credentials

Credentials

+

La découverte d'instances se fait par régions et comme toutes mes instances sont créées dans la région « us-east-1 » j'ai mis « us-east-1 » dans « region »



### AWS

Assume Role

Region \*

us-east-1

### Additional parameters

Extra Options

Timeout \*

10

J'ai cliqué sur « Next » sur l'écran suivant puis sur « Automatic analysis » sur l'écran suivant puis sur « Next », j'ai laissé coché « execute immediately » et cliqué sur « Finish »

J'ai attendu 10 secondes puis ai cliqué sur l'icône « rafraîchir » ce qui m'a montré qu'il a découvert 5 instances ce qui est normal :

Configuration > Hosts > Discovery

Search Search Status Running Scheduled Failed Finished Sea...

+ ADD ↻

Status	Name	Provider	Creation date ↓	Last execution	Duration	Discovered items
✓	Amazon AWS EC2	Amazon AWS EC2	06/11/2024 4:36 PM	06/11/2024 4:36 PM	3s	5

Puis je suis allé sur Configuration/Hosts/Hosts et j'ai retrouvé mes 5 instances AWS  
Puis j'ai coché les 2 instances backend et les 2 instances frontend selon les ids retrouvés dans ma liste d'EC2 sur AWS et j'ai mis « Deploy services » dans la liste déroulante « More actions »

Si je clique sur la roue dentée en face d'une instance backend ou frontend je peux voir les services :

More actions... Add 36

Host	Service	Scheduling	Template	Status	Options
<input type="checkbox"/> i-004670e08cd53d766	Ec2-Cpu-Credit	15 min / 1 min	--> Cloud-Aws-Ec2-Cpu-Credit-Api-custom --> Cloud-Aws-Ec2-Cpu-Credit-Api --> generic-active-service-custom --> ...	ENABLED	<span>⊗</span> 1
<input type="checkbox"/>	Ec2-Cpu-Usage	15 min / 1 min	--> Cloud-Aws-Ec2-Cpu-Usage-Api-custom --> Cloud-Aws-Ec2-Cpu-Usage-Api --> generic-active-service-custom --> ...	ENABLED	<span>⊗</span> 1
<input type="checkbox"/>	Ec2-Diskio	15 min / 1 min	--> Cloud-Aws-Ec2-Diskio-Api-custom --> Cloud-Aws-Ec2-Diskio-Api --> generic-active-service-custom --> generic-active-service	ENABLED	<span>⊗</span> 1
<input type="checkbox"/>	Ec2-Network	15 min / 1 min	--> Cloud-Aws-Ec2-Network-Api-custom --> Cloud-Aws-Ec2-Network-Api --> generic-active-service-custom --> ...	ENABLED	<span>⊗</span> 1
<input type="checkbox"/>	Ec2-Status	15 min / 1 min	--> Cloud-Aws-Ec2-Status-Api-custom --> Cloud-Aws-Ec2-Status-Api --> generic-active-service-custom --> generic-active-service	ENABLED	<span>⊗</span> 1

More actions... Add 36

Ensuite je suis allé dans Configuration/Pollers/Pollers, j'ai coché « Central », j'ai cliqué sur « Export configuration », j'ai coché les premières cases puis j'ai cliqué sur « Export »

Configuration > Pollers > Export configuration

#### | Configuration Files Export

##### Polling instances

? Pollers •

Central ×

##### Actions

? ☒ Generate Configuration Files

? ☒ Run monitoring engine debug (-v)

? ☒ Move Export Files

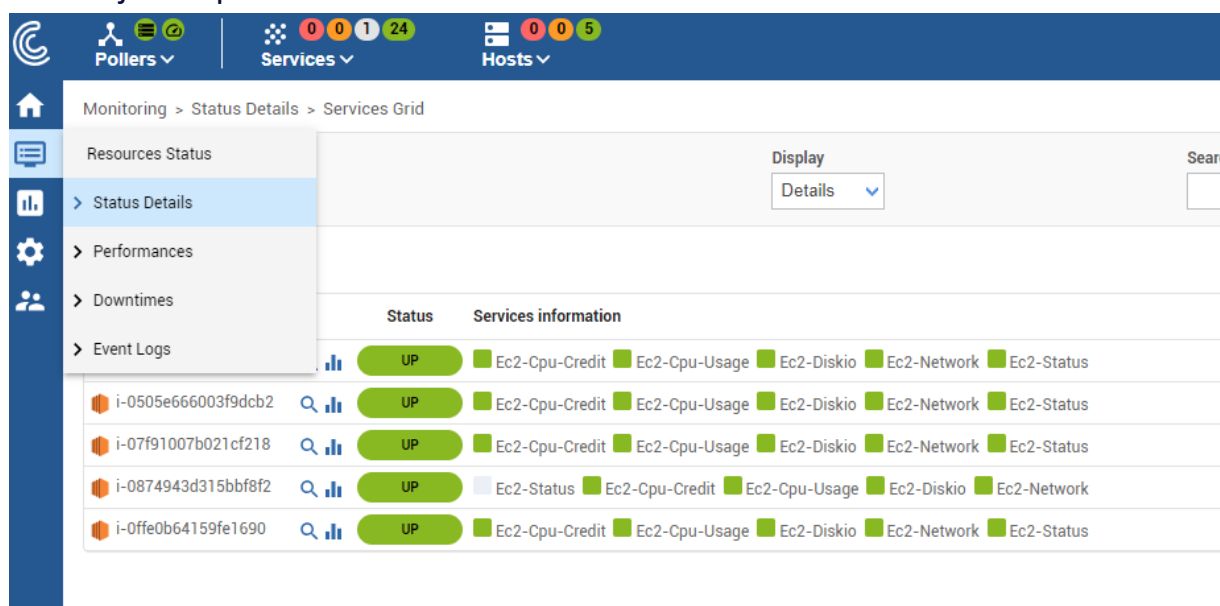
? ☒ Restart Monitoring Engine

Method Reload ▼

? ☐ Post generation command

Export

Puis j'ai cliqué sur Monitoring (  ) / Status details / Services grid puis sur Display details j'ai cliqué sur « All »




Nous pouvons voir que nous supervisons :

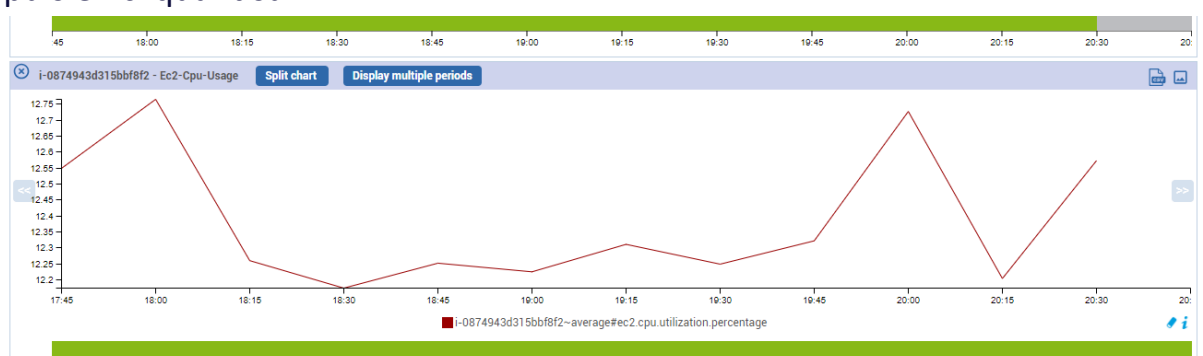
- EC2-Cpu-Credit
- EC2-CPU-Usage
- EC2-Diskio
- EC2-Network
- EC2-Status

En cliquant sur les différentes métriques on a des détails

J'ai voulu après faire des graphiques, pour cela je suis allé sur la page Web :

<https://docs.centreon.com/fr/docs/metrology/chart-management/>

Exemple de graphique obtenu en cliquant sur Monitoring/Status details/Services grid puis en cliquant sur  :



## 6. Conclusion

Ce travail m'a permis de m'initier à la création d'instances dans AWS avec Terraform (notamment des load balancers), de réaliser un CI/CD complet avec git et de voir comment installer Centreon. Durant ce travail je n'aurai pas vu la sécurisation à fond (firewall, seLinux etc) ni la dockerisation. J'espère les voir durant le dossier professionnel.