

AWS – Pipeline CI/CD

jeudi 23 février 2023 14:02

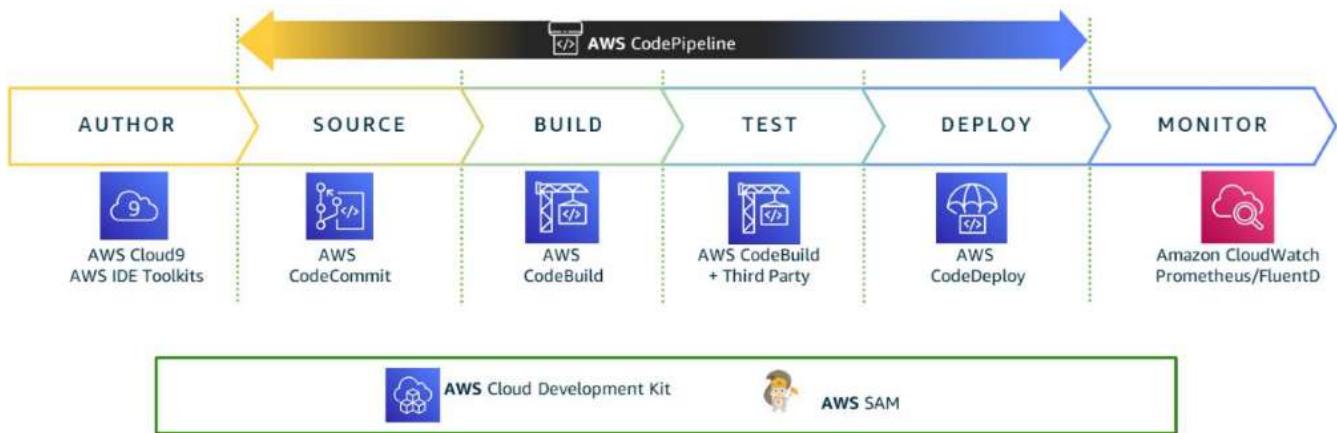
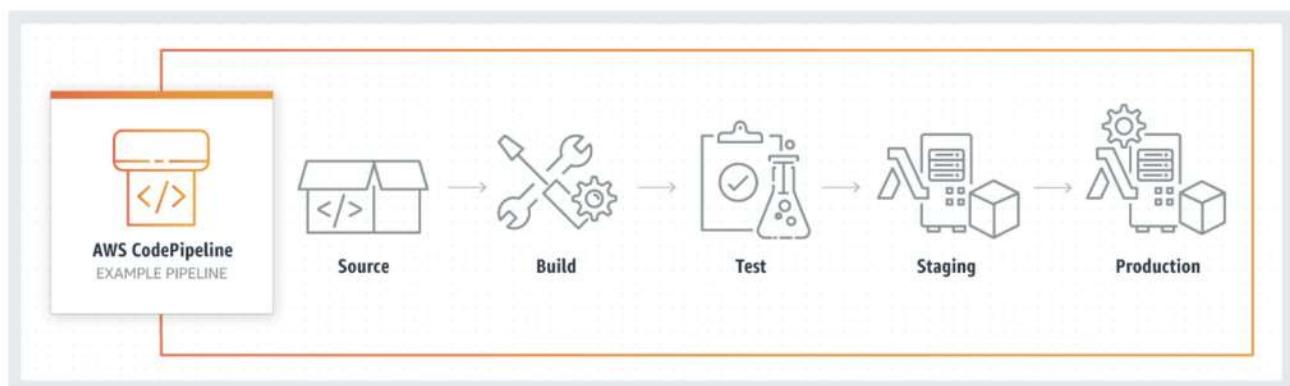
Formateur : Mohamed AIJJOU

https://docs.aws.amazon.com/whitepapers/latest/cicd_for_5g_networks_on_aws/cicd-on-aws.html

<https://fr.wikipedia.org/wiki/CI/CD>

En génie logiciel, **CI/CD** (parfois écrit CICD) est la combinaison des pratiques d'intégration continue et de livraison continue ou de déploiement continu. Le CI/CD comble le fossé entre les activités et les équipes de développement et d'exploitation en imposant l'automatisation de la création, du déploiement et de la surveillance des applications. Les pratiques DevOps modernes impliquent le développement continu, le test continu, l'intégration continue, le déploiement continu et la surveillance continue des applications logicielles tout au long de leur cycle de vie. La pratique CI/CD, ou pipeline CI/CD, constitue l'ossature des opérations DevOps modernes.

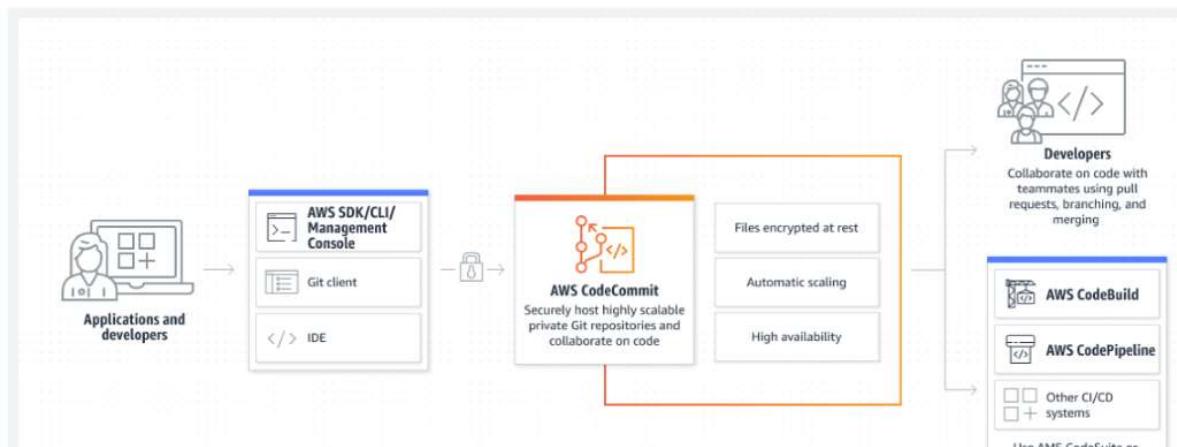
CI/CD can be pictured as a pipeline, where new code is submitted on one end, tested over a series of stages (source, build, test, staging, and production), and then published as production-ready code.



AWS Cloud9 : <https://aws.amazon.com/fr/cloud9/>

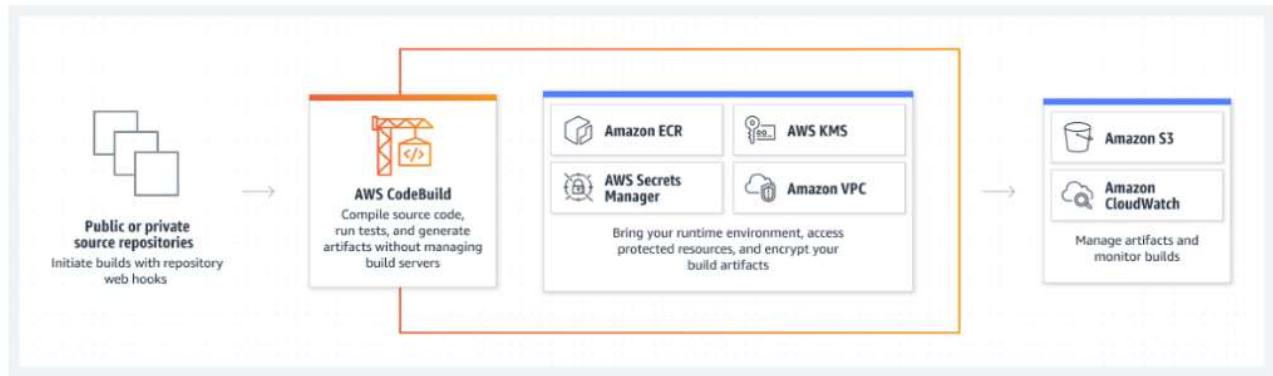
AWS Code Commit : <https://aws.amazon.com/fr/codecommit/>

AWS CodeCommit est un service de contrôle de sources géré, sécurisé et hautement évolutif qui héberge des référentiels Git privés.



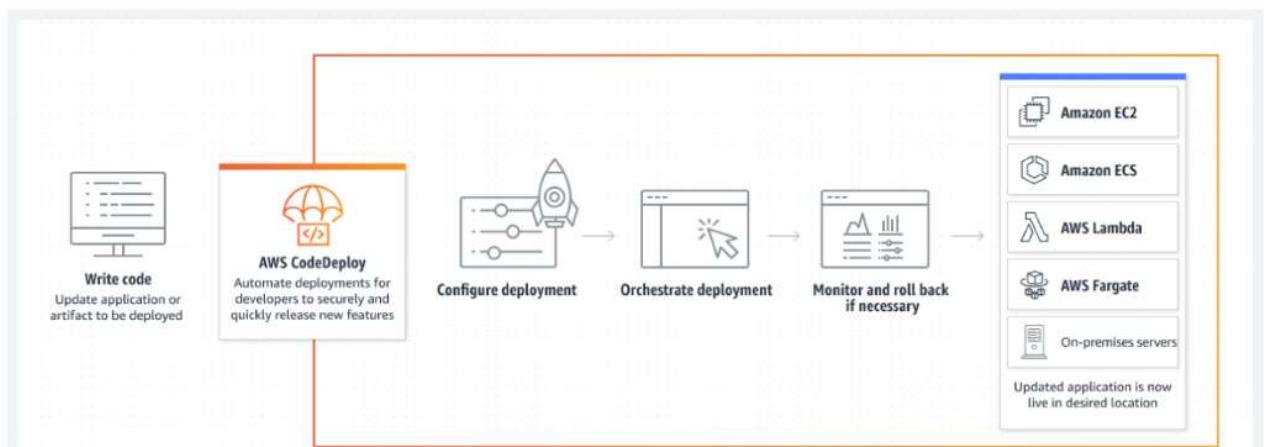
AWS Code Build : <https://aws.amazon.com/fr/codebuild/>

AWS CodeBuild est un service d'intégration entièrement géré qui compile votre code source, exécute des tests et produit des packages logiciels prêts à être déployés.



AWS Code Deploy : <https://aws.amazon.com/fr/codedeploy/>

AWS CodeDeploy est un service de déploiement entièrement géré qui automatise le déploiement de logiciels vers divers services de calcul, tels qu'Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), AWS Lambda et vos serveurs sur site. Utilisez AWS CodeDeploy pour automatiser les déploiements de logiciels, éliminant ainsi la nécessité d'opérations manuelles sujettes aux erreurs.



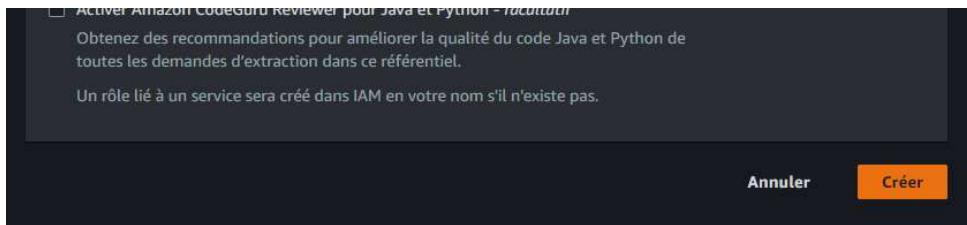
Code pipeline : connecte les autres services entre eux

1ère étape : créer un référentiel sur Code Commit

Créer un référentiel

Créez un référentiel sécurisé pour stocker et partager votre code. Commencez par taper un nom et une description pour votre référentiel. Les noms de référentiel sont inclus dans les URL de ce référentiel.

Paramètres de référentiel	
Nom du référentiel	<input type="text" value="benoit-m2i-devops"/> 100 caractères maximum. D'autres limites s'appliquent.
Créer - facultatif	<input type="text" value="Démo avec l'équipe pendant l'épreuve des poteaux"/> 1 000 caractères maximum.
Balises	<input type="button" value="Ajouter"/>
<input type="checkbox"/> Activer Amazon CodeGuru Reviewer pour Java et Python - facultatif	



Succès
Référentiel créé avec succès

Outils pour développeurs > CodeCommit > Référentiels > benoit-m2i-devops

benoit-m2i-devops

▼ Étapes de connexion

HTTPS SSH HTTPS (GRC)

Une fois le référentiel créé, il faut générer des informations d'identification git HTTPS sur notre compte dans la console IAM :

Informations d'identification Git HTTPS pour AWS CodeCommit (0)

Générez un nom d'utilisateur et un mot de passe que vous pouvez utiliser pour authentifier des connexions HTTPS sur des référentiels AWS CodeCommit. Vous pouvez disposer fois. [Learn more](#)

Actions ▾ Générer des informations d'identification

Nom d'utilisateur	Création
	Aucune information d'identification

Générer des informations d'identification

On récupère le fichier avec les éléments d'authentification et on le stock :

Générer des informations d'identification

Vos nouvelles informations d'identification sont disponibles.

Enregistrez votre nom d'utilisateur et votre mot de passe ou téléchargez le fichier d'informations d'identification.

C'est la seule fois où vous pouvez voir le mot de passe ou le télécharger. Vous ne pourrez pas le récupérer plus tard. Cependant, vous pouvez réinitialiser votre mot de passe à tout moment.

Vous pouvez utiliser ces informations d'identification lorsque vous vous connectez à partir de votre ordinateur local ou à partir d'outils qui nécessitent un nom d'utilisateur et un mot de passe statiques. [En savoir plus](#)

Nom d'utilisateur
 benoit-at-639962416620

Mot de passe
 CDyNq8tu8ui7xHAWrQSeANMdVtmU44D+07qV/aV8eMo= [Masquer](#)

[Télécharger les informations d'identification](#) [Fermer](#)

On rentre dans notre dossier cloné : on créer un fichier et on push le fichier sur notre code commit en respectant les étapes :

```
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp
$ git clone https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/benoit-m2i-dev
Cloning into 'benoit-m2i-devops'...
warning: You appear to have cloned an empty repository.
```

Git add .

Git commit -m "mon message"

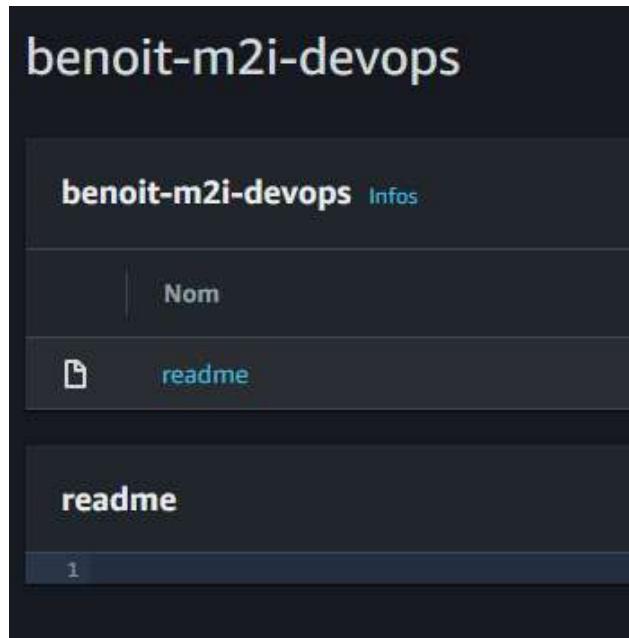
Git push

```
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp
$ cd benoit-m2i-devops/
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ ls
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ touch readme
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git add .

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git commit -m "Paris est magique !"
[master (root-commit) 56c0366] Paris est magique !
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 221 bytes | 221.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/benoit-m2i-devops
 * [new branch]      master -> master
```

On voit que notre fichier apparaît bien :



A partir de là : nous allons récupérer les fichier de notre projet, les rajouter dans notre dossier, et les push :

```
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git add .

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   app.js
    new file:   cron.yaml
    new file:   fichier1
    new file:   index.html
    new file:   package.json

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
```

```

Administrator@LIL-JTJ3KN3 MINGW64 ~/Documents/infrastructure as code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git commit -m "ajout des fichiers du projet web"
[master a5e7fef] ajout des fichiers du projet web
5 files changed, 146 insertions(+)
create mode 100644 app.js
create mode 100644 cron.yaml
create mode 100644 fichier1
create mode 100644 index.html
create mode 100644 package.json

Administrator@LIL-JTJ3KN3 MINGW64 ~/Documents/infrastructure as code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.95 KiB | 1.96 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/benoit-m2i-devops
  56c9366..a5e7fef master -> master

Administrator@LIL-JTJ3KN3 MINGW64 ~/Documents/infrastructure as code/ci_cd/test_tp/benoit-m2i-devops (master)
$ |

```

Autre façon de faire : copier notre projet, supprimer le dossier .git , puis faire un `git init` et un `git remote add origin` "lien d'un dépôt distant" pour lier notre dépôt local à no

```

Administrator@Salle_3_Z MINGW64 ~/Documents/cloud/tp_spring_demo
$ git init
Initialized empty Git repository in C:/Users/Administrateur/Documents/cloud/tp_spring_demo/.git/

Administrator@Salle_3_Z MINGW64 ~/Documents/cloud/tp_spring_demo (master)
$ git remote add origin https://git-codecommit.us-east-2.amazonaws.com/v1/repos/projet-m2i-mohamed

```

```

138 git init
139 git remote add origin https://git-codecommit.us-east-2.amazonaws.com/v1/repos/projet-m2i-mohamed
140 git status
141 git add .
142 git status
143 git commit -m"first commit"
144 git status
145 git push -u origin master
146 history

```

Git push -u origin master permet de s'assurer qu'on push bien sur la branche **master**

On peut vérifier nos branches dans le menu branche : ici nous avons uniquement la branche master :

Nom de branche	Date du dernier commit	Message de validation
master	Il y a 4 minutes.	ajout des fichiers du projet web

On peut également accéder au menu notification pour créer des alertes (peut remonter par un service sns si configuré ou un chatbot slack)

Créer une règle de notification

Les règles de notification définissent un abonnement aux événements qui se produisent au niveau de vos ressources. Lorsque ces événements ont lieu, des notifications sont envoyées aux cibles que vous désignez. Vous pouvez gérer vos préférences de notification dans Paramètres. [Infos](#)

Paramètres de règles de notification

Nom de la notification

Type de détail

Sélectionnez le niveau de détail de votre choix pour les notifications. [En savoir plus sur les notifications et la sécurité](#)

Complet

Comprend toutes les informations supplémentaires sur les événements fournis par la ressource ou la fonction de notifications.

Basique

Ne comprend que les informations fournies dans les événements de la ressource.

Événements qui déclenchent des notifications

Comments	Approvals	Pull request	Branches and tags
<input type="checkbox"/> On commits	<input type="checkbox"/> Status changed	<input type="checkbox"/> Source updated	<input type="checkbox"/> Created
<input type="checkbox"/> On pull requests	<input type="checkbox"/> Rule override	<input type="checkbox"/> Created	<input type="checkbox"/> Deleted
		<input type="checkbox"/> Status changed	<input type="checkbox"/> Updated
		<input type="checkbox"/> Merged	

Ne pas oublier de créer un bucket S3 qui va accueillir nos fichiers (faire attention à la région) :

Nom	Région AWS
benoit-bucket-project-spring	UE (Irlande) eu-west-1
bucket-tp-sdk-benoit	UE (Irlande) eu-west-1

Création projet code build :

Configuration du projet

Nom du projet

 Un nom de projet doit comporter entre 2 et 255 caractères. Il peut contenir les lettres A-Z et a-z, les chiffres 0-9 et les caractères spéciaux - et _.

Description - facultatif

Badge de génération - facultatif
 Activer le badge de génération

Activer la limite de génération simultanée - *facultatif*
 Limitez le nombre de générations simultanées autorisées pour ce projet.
 Limiter le nombre de générations simultanées que ce projet peut démarrer

▼ Configuration supplémentaire
Balises

Clé	Valeur	
Name	demo-nodejs-benoit-m2i	<input type="button" value="Supprimer la balise"/>

Source

Source 1 - Principale

Fournisseur de la source

Référentiel

Type de référence
 Sélectionnez le type de référence de version de la source qui contient votre code source.
 Branche
 Balise Git
 ID de validation

Branche
 Sélectionnez une branche qui contient le code à générer.

ID de validation - facultatif
 Sélectionnez un ID de validation. Cela peut réduire la durée de votre génération.

master

Version de la source [Infos](#)

refs/heads/master

a5e7fe6 ajout des fichiers du projet web

▶ Configuration supplémentaire
Profondeur du clone Git, Sous-modules Git

Environnement

Image d'environnement

Image générée Utiliser une image générée par AWS CodeBuild

Image personnalisée Spécifier une image Docker

Système d'exploitation

Amazon Linux 2 ▾

ⓘ Les environnements d'exécution des langages de programmation sont maintenant inclus dans l'image standard d'Ubuntu 18.04, ce qui est recommandé pour les nouveaux projets CodeBuild créés dans la console. Consultez [Images Docker fournies par CodeBuild](#) pour plus d'informations .

Environnement(s) d'exécution

Standard ▾

Image

aws/codebuild/amazonlinux2-aarch64-standard:2.0 ▾

Version d'image

Toujours utiliser la dernière image pour cette version d'exécution ▾

Privilégié

Activer cet indicateur si vous souhaitez créer des images Docker ou pour que vos générations bénéficient de priviléges élevés

Rôle de service

Nouveau rôle de service Crée un rôle de service dans votre compte

Rôle de service existant Choisir un rôle de service existant dans votre compte

Nom du rôle

codebuild-demo-nodejs-benoit-m2i-service-role

Taper le nom de votre rôle de service

Artefacts

Ajouter un artefact

Artéfact 1 - Principal

Type

Amazon S3 ▾

Vous pouvez choisir Aucun artefact si vous exécutez des tests ou transmettez une image Docker à Amazon ECR.

Nom du compartiment

bucket-tp-sdk-benoit

Nom

Nom du dossier ou du fichier compressé dans le compartiment qui contiendra vos artefacts de sortie. Utilisez Artifacts packaging (Emballage des artefacts) sous Additional configuration (Configuration supplémentaire) pour choisir si vous souhaitez utiliser un dossier ou un fichier compressé. Si le nom n'est pas indiqué, le nom du projet est utilisé par défaut.

Activer la gestion sémantique des versions Utiliser le nom d'artefact spécifié dans le fichier buildspec

Chemin - facultatif

Chemin du dossier ou du fichier ZIP de sortie de génération.

Par exemple : MyPath/MyArtifact.zip.

Type d'espace de noms - facultatif

Type d'espace de noms - facultatif

Aucun(e)

Choisissez ID de génération pour insérer l'ID de la génération dans le chemin vers le fichier zip ou le dossier de sortie de génération (par exemple, MyPath/MyBuildID/MyArtifact.zip). Sinon, choisissez Aucun.

Emballage des artefacts

Aucun(e)
Les fichiers d'artefact seront chargés dans le compartiment.

Zip
AWS CodeBuild charge les artefacts dans un fichier compressé qui est placé dans le compartiment spécifié.

Journaux

CloudWatch

Journaux CloudWatch - facultatif
Cochez cette option pour charger les journaux de sortie de génération dans CloudWatch.

Nom du groupe

Nom du flux

S3

Journaux S3 - facultatif
Cochez cette option pour charger les journaux de sortie de génération dans S3.

[Annuler](#) [Créer un projet de génération](#)

✓ **Projet créé**
Vous avez créé le projet suivant avec succès : **demo-nodejs-benoit-m2i**

[Outils pour développeurs](#) > [CodeBuild](#) > [Projets de génération](#) > **demo-nodejs-benoit-m2i**

On peut maintenant démarrer la génération de notre projet :

demo-nodejs-benoit-m2i

[Notifier ▾](#) [Partager](#) [Modifier ▾](#) [Supprimer un projet de génération](#) [Démarrer la génération avec des remplacements](#) [Démarrer la génération](#)

Configuration

Fournisseur de la source AWS CodeCommit	Référentiel principal benoit-m2i-devops	Emplacement de chargement des artefacts bucket-tp-sdk-benoit	Badge de génération Désactivé
Générations publiques Désactivé			

On va créer notre fichier buildspec.yaml :

benoit-m2i-devops

[Notifier ▾](#) [master](#) [Créer une demande d'extraction](#) [URL du clone ▾](#)

benoit-m2i-devops [Infos](#)

[Ajouter un fichier ▾](#)
[Créer un fichier](#)
[Charger le fichier](#)

[Outils pour développeurs](#) > [CodeCommit](#) > [Référentiels](#) > **benoit-m2i-devops** > Fichier

Créer un fichier

benoit-m2i-devops [Infos](#)

```
1 #!
```

Valider les modifications dans master

Nom de fichier
Par exemple, file.txt
buildspec.yaml
benoit-m2i-devops/buildspec.yaml

Nom de l'auteur
Benoit

Documentation utile pour la création d'un buildspec :

<https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html>

Code du fichier :

Créer un fichier

benoit-m2i-devops Infos

```

1 version: 0.2
2 phases:
3   install:
4     runtime-versions:
5       nodejs: latest
6     commands:
7       - echo "installing something"
8   pre_build:
9     commands:
10      - echo "we are in the pre build phase"
11   build:
12     commands:
13       - echo "we are in the build block"
14       - echo "we will run some tests"
15       - echo "Congratulations" index.html
16   post_build:
17     commands:
18       - echo "we are in the post build phase"

```

Une fois le fichier complété : on valide

L'action effectué sera équivalente à un **git commit/git push**

On peut voir les actions effectué par le fichier dans la partie **validation** :

Validation 50b71d2a5368c699a99ba102b061ab85a67ada3b		
Détails		
Auteur Benoit test.test@testeurdebuild.buildcom	Authored date Maintenant	Committer Benoit test.test@testeurdebuild.buildcom
Date de validation Maintenant	Validation parent a5e7fef62716be8fb4330c91b05c465da58e9b9a	
Message de validation Added buildspec.yaml		

buildspec.yaml Ajouté

```

1 + version: 0.2
2 + phases:
3 +   install:

```

```

4 +     runtime-versions:
5 +         nodejs: latest
6 +     commands:
7 +         - echo "installing something"
8 +     pre_build:
9 +         commands:
10 +             - echo "we are in the pre build phase"
11 +     build:
12 +         commands:
13 +             - echo "we are in the build block"
14 +             - echo "we will run some tests"
15 +             - echo "Congratulations" index.html
16 +     post_build:
17 +         commands:
18 +             - echo "we are in the post build phase"

```

Dans code build : on démarre la génération

La génération a démarré
Vous avez démarré avec succès la génération suivante : demo-nodejs-benoit-m2i:22771e8a-0a9f-4fd6-9acd-0ea307e34143.

Outils pour développeurs > CodeBuild > Projets de génération > demo-nodejs-benoit-m2i > demo-nodejs-benoit-m2i:22771e8a-0a9f-4fd6-9acd-0ea307e34143

On peut voir les détails de la phase dans le sous-menu "détails de la phase" :

Nom	Statut
SUBMITTED	Opération réussie
QUEUED	Opération réussie
PROVISIONING	Opération réussie
DOWNLOAD_SOURCE	Opération réussie
INSTALL	Opération réussie
PRE_BUILD	Opération réussie
BUILD	Opération réussie
POST_BUILD	Opération réussie
UPLOAD_ARTIFACTS	Opération réussie
FINALIZING	Opération réussie
COMPLETED	Opération réussie

Le détail du journal est disponible dans la partie Journaux de génération :

Journaux de génération Détail de la phase Rapports Variables d'environnement Détails de la génération

Affichage des 56 dernières lignes du journal de génération. [Afficher le journal complet](#)

[▲ Afficher les journaux précédents](#)

```
1 [Container] 2023/02/23 15:54:29 Waiting for agent ping
2 [Container] 2023/02/23 15:54:30 Waiting for DOWNLOAD_SOURCE
3 [Container] 2023/02/23 15:54:34 Phase is DOWNLOAD_SOURCE
4 [Container] 2023/02/23 15:54:34 CODEBUILD_SRC_DIR=/codebuild/output/src945028934/src/git-codecommit.eu-west-1.amazonaws.com/test-m2i-nodejs
5 [Container] 2023/02/23 15:54:34 YAML location is /codebuild/output/src945028934/src/git-codecommit.eu-west-1.amazonaws.com/test-m2i-nodejs
6 [Container] 2023/02/23 15:54:34 Not setting HTTP client timeout for source type codecommit
7 [Container] 2023/02/23 15:54:34 Processing environment variables
8 [Container] 2023/02/23 15:54:34 Resolved 'nodejs' runtime alias 'latest' to '12'.
9 [Container] 2023/02/23 15:54:34 Selecting 'nodejs' runtime version '12' based on manual selections...
10 [Container] 2023/02/23 15:54:34 Running command echo "Installing Node.js version 12 ..."
11 [Container] 2023/02/23 15:54:34 Installing Node.js version 12 ...
12 [Container]
```

Après l'étape du code build, nous devons déployer notre code avec **Code Deploy** : pour cela nous avons besoin d'une nouvelle machine :

Détails	Sécurité	Mise en réseau	Stockage	Vérifications de statut	Surveillance	Balises
▼ Résumé de l'instance Informations						
ID d'instance				Adresse IPv4 publique		
i-0469347cb5ccf82e8 (node_benoit_application_test_m2i)				34.243.112.152 adresse ouverte		
Adresse IPv6				État de l'instance		
-				En attente		
Type de nom d'hôte				Nom DNS de l'IP privé (IPv4 uniquement)		
Nom de l'adresse IP: ip-172-31-31-226.eu-west-1.compute.internal				ip-172-31-31-226.eu-west-1.compute.internal		
Réponse à un nom DNS de ressource privée				Type d'instance		
IPv4 (A)				t2.micro		

On va devoir installer un agent code deploy sur notre machine EC2 pour assurer la communication :

<https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-cli.html>

Nous allons utiliser ce lien pour installer l'agent sur notre machine :

<https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html>

Données utilisateur - *optional* [Informations](#)

Enter user data in the field.

```
#!/bin/bash

# Installing CodeDeploy Agent
sudo yum update
sudo yum install ruby

# Download the agent (replace the region)
wget https://aws-codedeploy-eu-west-3.s3.eu-west-3.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
sudo service codedeploy-agent status
```

```
#!/bin/bash
# Installing CodeDeploy Agent
sudo yum update -y
sudo yum install ruby -y

# Download the agent (replace the region)
```

```

wget https://aws-codedeploy-eu-west-3.s3.eu-west-3.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto
sudo service codedeploy-agent status

```

```

Complete!
I, [2023-02-24T09:32:54.808997 #6434] INFO -- : Update check complete.
I, [2023-02-24T09:32:54.809095 #6434] INFO -- : Stopping updater.
[ec2-user@ip-172-31-16-93 ~]$ sudo service codedeploy-agent status
The AWS CodeDeploy agent is running as PID 6519

```

Nous allons ensuite créer un rôle :

Sélectionner une entité de confiance [Infos](#)

Type d'entité approuvée

- Service AWS
Autorisez les services AWS tels qu'EC2, Lambda ou autre à effectuer des actions dans ce compte.
- Compte AWS
Autorisez les entités d'autres comptes AWS qui appartiennent à vous à un tiers à effectuer des actions dans ce compte.
- Identité Web
Permet aux utilisateurs fédérés par le fournisseur d'identité web externe spécifié d'assumer ce rôle pour effectuer des actions dans ce compte.
- Fédération SAML 2.0
Autorisez les utilisateurs fédérés avec SAML 2.0 à partir d'un répertoire d'entreprise à effectuer des actions dans ce compte.
- Stratégie d'approbation personnalisée
Créez une stratégie d'approbation personnalisée pour permettre à d'autres utilisateurs d'effectuer des actions dans ce compte.

Cas d'utilisation

Autorisez un service AWS comme EC2, Lambda ou autres à effectuer des actions dans ce compte.

Cas d'utilisation courants

EC2

Allows EC2 instances to call AWS services on your behalf.

Lambda

Allows Lambda functions to call AWS services on your behalf.

Politiques des autorisations (Sélectionnée(s) 1/889) [Infos](#)

Choisissez une ou plusieurs stratégies à attacher à votre nouveau rôle.

<input type="text"/> <i>Filtrer les stratégies par nom de propriété ou de stratégie et appuyer sur Entrée.</i>		
<input style="border: 1px solid #ccc; padding: 2px 10px; margin-right: 10px;" type="button" value="S3"/>	<input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="Effacer les filtres"/>	
	<input type="checkbox"/> Nom de la politique	Type
	<input type="checkbox"/> KinesisFirehoseServicePolicy-KDS-S3-pAV05-eu-west-3	Gérées ...
	<input type="checkbox"/> KinesisFirehoseServicePolicy-PUT-S3-jew5D-eu-west-3	Gérées ...
	<input type="checkbox"/> KinesisFirehoseServicePolicy-PUT-S3-uowQl-eu-west-3	Gérées ...
	<input type="checkbox"/> s3	Gérées ...
	<input type="checkbox"/> s3-sdk	Gérées ...
	<input type="checkbox"/> S3 mfA	Gérées ...

<input type="checkbox"/>	<input type="checkbox"/> s3_read_write	Gérées ...
<input type="checkbox"/>	<input type="checkbox"/> strategie_s3	Gérées ...
<input type="checkbox"/>	<input type="checkbox"/> AmazonDMSRedshiftS3Role	Gérées ...
<input checked="" type="checkbox"/>	<input type="checkbox"/> AmazonS3FullAccess	Gérées ...

Informations du rôle

Nom du rôle

Saisissez un nom explicite pour identifier ce rôle.

64 caractères maximum. Utilisez des caractères alphanumériques, ainsi que les caractères « +=,.@-_ ».

Description

Ajoutez une brève explication de ce rôle.

1 000 caractères maximum. Utilisez des caractères alphanumériques, ainsi que les caractères '+=,.@-_ '.

Étape 1 : sélectionner des entités de confiance

```

1  [
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": [
7                  "sts:AssumeRole"
8              ],
9              "Principal": {
10                  "Service": [
11                      "ec2.amazonaws.com"
12                  ]
13              }
14          }
15      ]
16  ]

```

Second role pour code deploy :

Ajouter des autorisations Infos

Politiques des autorisations (Sélectionnée(s) 1/889) Infos
Choisissez une ou plusieurs stratégies à attacher à votre nouveau rôle.

Filtrer les stratégies par nom de propriété ou de stratégie et appuyer sur

"codedeploy"

Nom de la politique

AmazonEC2RoleforAWSCodeDeploy

Nom du rôle	Entités de confiance
<input type="checkbox"/> benoit-role-code-deploy-pipeline-m2i	Service AWS: ec2
<input type="checkbox"/> codebuild-demo-nodejs-benoit-m2i-service-role	Service AWS: codebuild
<input type="checkbox"/> role_s3_ec2_benoit_m2i	Service AWS: ec2

Type d'entité approuvée

<input checked="" type="radio"/> Service AWS Autorisez les services AWS tels qu'EC2, Lambda ou autre à effectuer des actions dans ce compte.	<input type="radio"/> Compte AWS Autorisez les entités d'autres comptes AWS qui appartiennent à vous à un tiers à effectuer des actions dans ce compte.	<input type="radio"/> Identité Web Permet aux utilisateurs fédérés par le fournisseur d'identité web externe spécifié d'assumer ce rôle pour effectuer des actions dans ce compte.
<input type="radio"/> Fédération SAML 2.0 Autorisez les utilisateurs fédérés avec SAML 2.0 à partir d'un répertoire d'entreprise à effectuer des actions dans ce compte.	<input type="radio"/> Stratégie d'approbation personnalisée Créez une stratégie d'approbation personnalisée pour permettre à d'autres utilisateurs d'effectuer des actions dans ce compte.	

Cas d'utilisation

Autorisez un service AWS comme EC2, Lambda ou autres à effectuer des actions dans ce compte.

Cas d'utilisation courants

- EC2
Allows EC2 instances to call AWS services on your behalf.
- Lambda
Allows Lambda functions to call AWS services on your behalf.

Cas d'utilisation pour d'autres services AWS :

CodeDeploy	<input checked="" type="radio"/> CodeDeploy Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.
CodeDeploy for Lambda	<input type="radio"/> CodeDeploy for Lambda Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.
CodeDeploy - ECS	<input type="radio"/> CodeDeploy - ECS Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

Liste des rôles à avoir pour notre pipeline :

IAM > Rôles

Rôles (120) Infos

Un rôle IAM est une identité que vous pouvez créer et qui dispose d'autorisations spécifiques avec des informations d'identification valides pendant de courtes durées. Les rôles peuvent être endossés par des entités de confiance.

3 correspondances

Nom du rôle	Entités de confiance
<input type="checkbox"/> benoit-role-code-deploy-pipeline-m2i	Service AWS: ec2
<input type="checkbox"/> role-code-deploy-pipeline-benoit-m2i	Service AWS: codedeploy
<input type="checkbox"/> role_s3_ec2_benoit_m2i	Service AWS: ec2

Outils pour développeurs

AWS CodeDeploy

Automatisez les déploiements de code pour maintenir la disponibilité des applications

AWS CodeDeploy est un service de déploiement entièrement géré qui automatisé les déploiements de logiciels pour des services de calcul comme Amazon EC2, AWS Lambda, ainsi que vos serveurs sur site. AWS CodeDeploy vous permet de lancer rapidement et facilement de nouvelles fonctionnalités et d'éviter les temps d'arrêt pendant le déploiement d'une application, tout en gérant la complexité de la mise à jour de vos applications.

Créer un déploiement AWS CodeDeploy

Démarrez avec AWS CodeDeploy en créant votre première application de déploiement.

[Créer une application](#)

On va commencer par créer l'application : pour ce TP, nous allons choisir EC2 comme plateforme de calcul

Outils pour développeurs > CodeDeploy > Applications > Créer une application

Créer une application

Configuration de l'application

Nom de l'application
Entrer un nom d'application

100 caractères maximum

Plateforme de calcul
Choisir une plateforme de calcul

[Annuler](#) [Créer une application](#)



Application créée

Pour créer un déploiement, vous devez d'abord créer un groupe de déploiement.

Création du déploiement :

Créer un groupe de déploiement

Application

Application
benoit-application-test
Type de calcul
EC2/Sur site

Nom du groupe de déploiement

Entrer un nom de groupe de déploiement

100 caractères maximum

Rôle de service

Entrer un rôle de service
Saisissez un rôle de service doté d'autorisations CodeDeploy pour accorder à AWS CodeDeploy l'accès à vos instances cibles.

[X](#)

Type de déploiement

Choisir comment déployer votre application

Sur place

Met à jour les instances dans le groupe de déploiement avec les révisions d'application les plus récentes. Lors d'un déploiement, chaque instance sera mise hors ligne.

Bleu/vert

Remplace les instances dans le groupe de déploiement par de nouvelles instances et déploie la révision d'application la plus récente dans ces instances. Une fois terminé, les instances dans

déploiement, chaque instance sera mise hors ligne brièvement pendant sa mise à jour.

plus récente dans ces instances. Une fois les instances dans l'environnement de remplacement enregistrées avec un équilibrEUR de charge, l'enregistrement des instances de l'environnement d'origine est annulé et celles-ci peuvent être mises hors service.

On sélectionne instance EC2 + dans les tag, on utilise le nom de notre machine :

Configuration de l'environnement

Sélectionner une combinaison de groupes Amazon EC2 Auto Scaling, d'instances Amazon EC2 et d'instances sur site à ajouter à ce déploiement

Groupes Auto Scaling Amazon EC2

Instances Amazon EC2
Instance correspondante unique 1. [Cliquez ici pour en savoir plus](#)

Vous pouvez ajouter jusqu'à 3 groupes de balises pour des instances EC2 à ce groupe de déploiement.
Un groupe de balises : Le déploiement aura lieu dans toute instance identifiée par le groupe de balises.
Plusieurs groupes de balises : Le déploiement aura lieu uniquement dans les instances identifiées par tous les groupes de balises.

Groupe de balises 1

Clé	Valeur - facultatif	Supprimer la balise
<input type="text"/> Name	<input type="text"/> node-benoit-main-pipeline	X

[Ajouter une balise](#)

[+ Ajouter un groupe de balises](#)

Instances sur site

Correspondance d'instances

Instance correspondante unique 1. [Cliquez ici pour en savoir plus](#)

Configuration de l'agent avec AWS Systems Manager [Infos](#)

AWS Systems Manager installe l'agent CodeDeploy sur toutes les instances et le met à jour en fonction de la fréquence configurée.



Nous vous recommandons de configurer l'installation et les mises à jour de votre agent CodeDeploy avec AWS Systems Manager.

AWS Systems Manager offre plus de contrôle sur les mises à jour et restaurations de version de l'agent CodeDeploy que sur l'installation à l'aide d'autres méthodes. [En savoir plus](#)

Installation de l'agent AWS CodeDeploy

- Jamais
- Une seule fois
- Maintenant et planifier des mises à jour

Configuration de l'agent avec AWS Systems Manager [Infos](#)

AWS Systems Manager installe l'agent CodeDeploy sur toutes les instances et le met à jour en fonction de la fréquence configurée.



Nous vous recommandons de configurer l'installation et les mises à jour de votre agent CodeDeploy avec AWS Systems Manager.

AWS Systems Manager offre plus de contrôle sur les mises à jour et restaurations de version de l'agent CodeDeploy que sur l'installation à l'aide d'autres méthodes. [En savoir plus](#)

Installation de l'agent AWS CodeDeploy

- Jamais
- Une seule fois
- Maintenant et planifier des mises à jour



Succès

Groupe de déploiement créé

Outils pour développeurs > CodeDeploy > Applications > benoit-application-test > benoit-application-test

benoit-application-test

Détails du groupe de déploiement

Nom du groupe de déploiement benoit-application-test	Nom de l'application benoit-application-test
Type de déploiement Sur place	ARN de rôle de service arn:aws:iam::639962416620:role/role-code-deploy-pipeline-benoit-m2i
Restauration activée faux	Planificateur de mise à jour de l'agent Apprenez à planifier une mise à jour dans AWS Systems Manager [?]

On retourne dans code commit pour créer le fichier **appspec.yaml** :

Documentation fichier appspec :

<https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file.html>

Exemple fichier appspec :

Here is an example of a correctly spaced AppSpec file:

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

Pour créer ce fichier : on va retourner dans notre dépôt sur vscode, faire un **git pull** et créer un dossier **script** et un fichier **appspec**

```
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git pull
remote: Counting objects: 3, done.
Unpacking objects: 100% (3/3), 596 bytes | 37.00 KiB/s, done.
From https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/benoit-m2i-devops
  a5e7fef..50b71d2 master      -> origin/master
Updating a5e7fef..50b71d2
Fast-forward
 buildspec.yaml | 18 ++++++-----+
 1 file changed, 18 insertions(+)
 create mode 100644 buildspec.yaml

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ ls
app.js  buildspec.yaml  cron.yaml  fichier1  index.html  package.json  readme

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$
```



On complète notre fichier :

```
! appspec.yaml U X
benoit-m2i-devops > ! appspec.yaml > {} hooks > [ ] ApplicationStop > {} 0 > runas
1   version: 0.0
2   os: linux
3   files:
4     - source: /index.html
5       destination: /var/www/html/
6   hooks:
7     BeforeInstall:
8       - location: scripts/install_dependencies
9         timeout: 300
10        runas: root
11       - location: scripts/start_server
12         timeout: 300
13         runas: root
14     ApplicationStop:
15       - location: scripts/stop_server
16         timeout: 300
17         runas: root
```

```
version: 0.0
os: linux
files:
- source: /index.html
destination: /var/www/html/
hooks:
BeforeInstall:
- location: scripts/install_dependencies
timeout: 300
runas: root
- location: scripts/start_server
timeout: 300
runas: root
ApplicationStop:
- location: scripts/stop_server
timeout: 300
runas: root
```

On va également créer nos fichiers dans le dossier scripts car ils sont appelé dans notre fichier **appspec.yaml** :

Fichier **install_dependencies** :

```

benoit-m2i-devops > scripts > $ install_dependencies
1  #!/bin/bash
2  yum install -y httpd

```

Fichier `start_server` :

```

benoit-m2i-devops > scripts > $ start_server
1  #!/bin/bash
2  service httpd start

```

Fichier `stop_server` :

```

benoit-m2i-devops > scripts > $ stop_server
1  #!/bin/bash
2  isExistApp = `pgrep httpd`
3  if [[ -n $isExistApp ]]; then
4      service httpd stop
5  fi

```

```

#!/bin/bash
isExistApp = `pgrep httpd`
if [[ -n $isExistApp ]]; then
    service httpd stop
fi

```

On push nos modifications sur le dépôt AWS Commit :

```

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git add .
warning: in the working copy of 'scripts/install_dependencies', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'scripts/start_server', LF will be replaced by CRLF the next time Git touches it

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

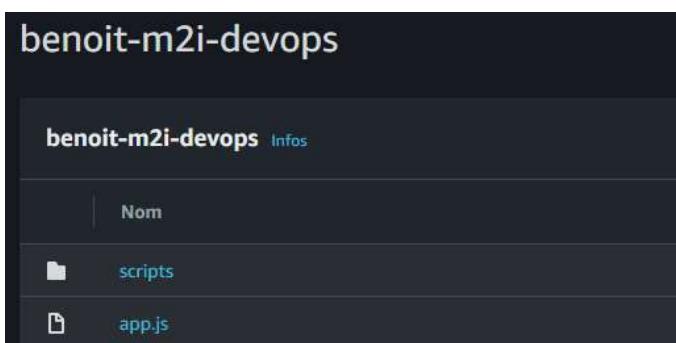
Changes to be committed:
(use "git restore --staged <file>" to unstage)
  new file: appspec.yaml
  modified:  README
  new file: scripts/install_dependencies
  new file: scripts/start_server
  new file: scripts/stop_server

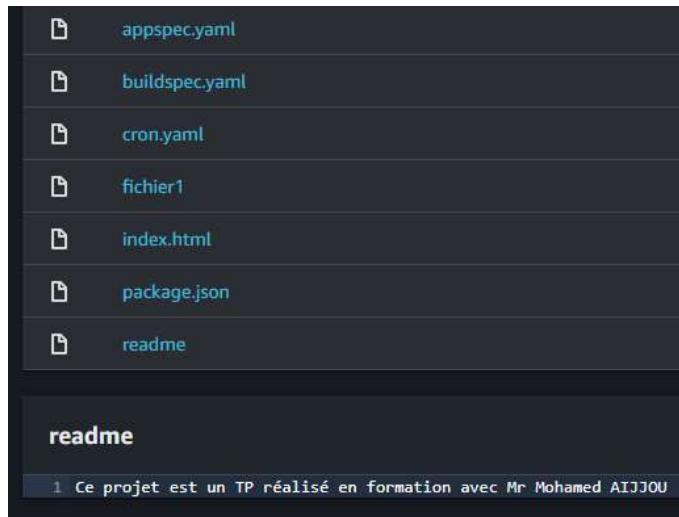
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git commit -m "add scripts + appspec.yaml"
[master 5369df8] add scripts + appspec.yaml
5 files changed, 27 insertions(+)
create mode 100644 appspec.yaml
create mode 100644 scripts/install_dependencies
create mode 100644 scripts/start_server
create mode 100644 scripts/stop_server

Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/test_tp/benoit-m2i-devops (master)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 878 bytes | 878.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/benoit-m2i-devops
  50b71d2..5369df8  master -> master

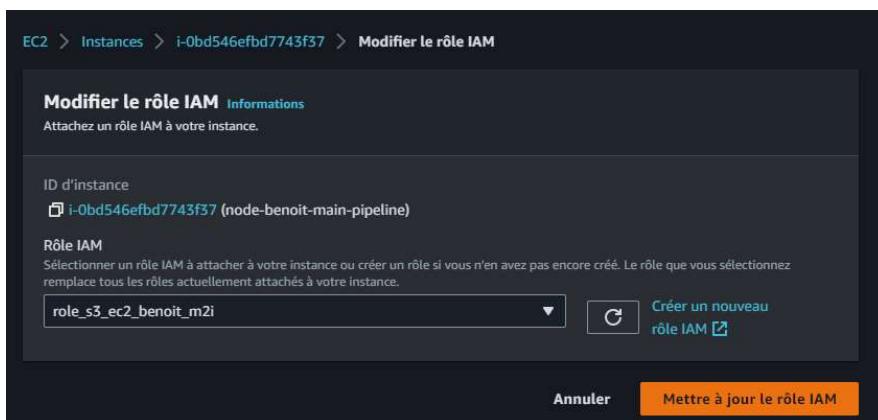
```

Vérification du push des fichiers :

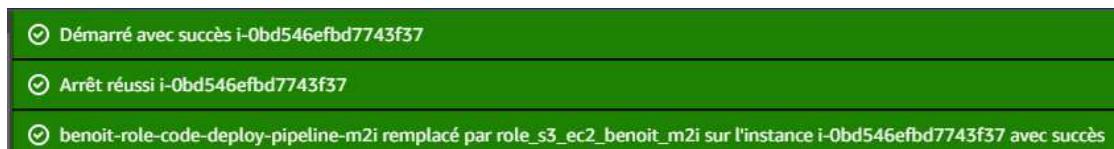




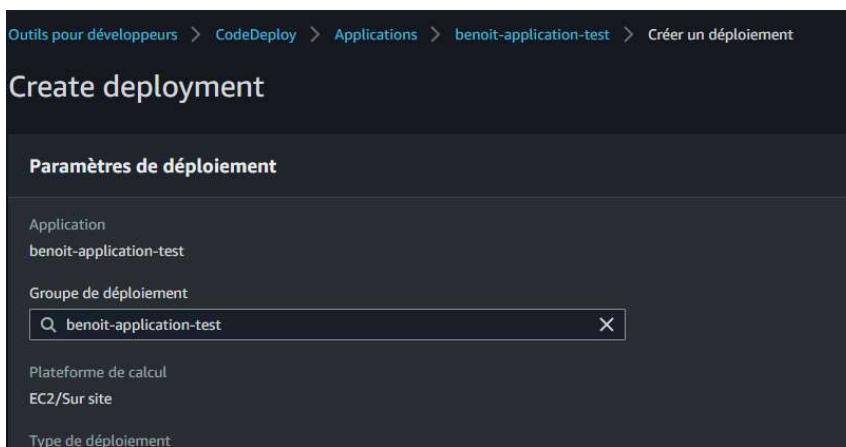
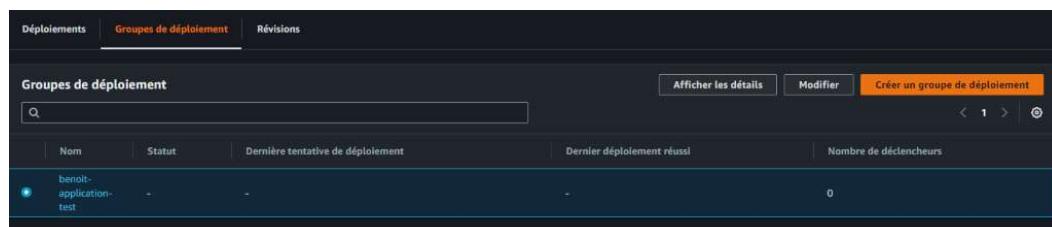
On modifie le rôle IAM de notre instance pour lui ajouter le rôle S3 que nous avons créer :



Pour que la machine prenne en compte ce changement de rôle : on stop l'instance et on la redémarre :



Ensuite, dans code deploy : on créer un groupe de déploiement :



Sur place

Type de révision

Mon application est stockée dans Amazon S3

Mon application est stockée dans GitHub

Emplacement de la révision
Copier et coller le compartiment Amazon S3 dans lequel votre révision est stockée

s3://benoit-bucket-project-spring/benoit-project-spring-m2i

Aucune révision précédente

On donne l'emplacement de notre S3 :

Emplacement de la révision
Copier et coller le compartiment Amazon S3 dans lequel votre révision est stockée

s3://benoit-bucket-project-spring/benoit-project-spring-m2i

Aucune révision précédente

Type de fichier de révision

On précise le type de fichier :

Type de fichier de révision

.zip

On crée notre déploiement :

Succès
Déploiement créé

Outils pour développeurs > CodeDeploy > Déploiements > d-SQG2HTEOM

On peut voir le déroulement de notre déploiement dans le menu "view events"

ID d'instance	Durée	Statut	Événement le plus récent	Événements
i-0137f138c1d24500f	-	En cours	-	View events

arn:aws:ec2:eu-west-1:639962416620:instance/i-0137f138c1d24500f

Détails du déploiement

Application project-spring-benoit-m2i	ID de déploiement d-EZQCFQEOM	Statut En cours
Configuration de déploiement CodeDeployDefault.AllAtOnce	Groupe de déploiement group-deploy-project-spring-benoit-m2i	Initié par Action utilisateur
Description du déploiement 1er déploiement le vendredi après-midi... Tada !		

Détails de la révision

Emplacement de la révision s3://benoit-bucket-project-spring/benoit-project-spring-m2i	Révision créée Il y a 12 minutes.	Description de la révision Application revision re		
Événement	Durée	Statut	Code d'erreur	Heure de début
ApplicationStop	moins d'une seconde	Réussi	-	fév 24, 2023 4:53 PM (UTC+1:00)
DownloadBundle	moins d'une seconde	Réussi	-	fév 24, 2023 4:53 PM (UTC+1:00)

Événement

Durée

Statut

Code d'erreur

ApplicationStop	moins d'une seconde	Réussi	-
DownloadBundle	1 seconde	Réussi	-
BeforeInstall	moins d'une seconde	Réussi	-
Install	moins d'une seconde	Réussi	-
AfterInstall	moins d'une seconde	Réussi	-
ApplicationStart	moins d'une seconde	Réussi	-
ValidateService	moins d'une seconde	Réussi	-

Statut du déploiement

Installation d'une application sur vos instances

1 instances sur 1 mises à jour Réussi

Rajout du code Pipeline :

Outils pour développeurs

AWS CodePipeline

Visualisez et automatisez les différentes étapes de votre processus de lancement de logiciel

AWS CodePipeline est un service d'intégration et de diffusion continues, qui permet un déploiement rapide et fiable de mises à jour d'applications et d'infrastructures. CodePipeline élaboré, teste et déploie votre code à chaque fois qu'un changement de code a lieu, en fonction des modèles de processus de lancement que vous avez définis.

[Créez un pipeline AWS CodePipeline](#)

Démarrez avec AWS CodePipeline en créant votre premier pipeline d'intégration continue et de livraison continue.

[Créer un pipeline](#)

Configuration Pipeline :

Paramètres de pipeline

Nom de pipeline
Saisissez le nom du pipeline. Vous ne pouvez plus modifier le nom du pipeline après sa création.

100 caractères maximum

Rôle de service
 Nouveau rôle de service
Créer un rôle de service dans votre compte
 Rôle de service existant
Choisir un rôle de service existant dans votre compte

ARN du rôle

X

▼ Paramètres avancés

Magasin d'artefacts

- Emplacement par défaut**
Créez un compartiment S3 dans votre compte.
- Emplacement personnalisé**
Sélectionnez un emplacement S3 dans la même région et le même compte que votre pipeline.

Clé de chiffrement

- Clé gérée par AWS par défaut**
Utilisez la clé principale client générée par AWS pour CodePipeline dans votre compte afin de chiffrer les données dans le magasin d'artefacts.
- Clé gérée par le client**
Pour chiffrer les données dans le magasin d'artefacts sous une clé gérée par le client AWS KMS, spécifiez l'ID de clé, son ARN de clé ou son ARN d'alias.

Ajouter l'étape source :

[Ajouter une étape source](#) Infos

Source

Fournisseur de source
Il s'agit de l'emplacement où les artefacts d'entrée pour votre pipeline sont stockés. Sélectionnez le fournisseur, puis indiquez les détails de connexion.

AWS CodeCommit ▾

Nom du référentiel
Sélectionnez un référentiel existant dans lequel vous avez transféré votre code source.

X

Nom de branche
Sélectionnez une branche du référentiel.

X

Options de détection de modifications
Sélectionnez un mode de détection pour démarrer automatiquement votre pipeline lorsqu'un changement se produit dans le code source.

Amazon CloudWatch Events (recommandé)
Utiliser Amazon CloudWatch Events pour démarrer automatiquement mon pipeline lorsqu'un changement se produit

AWS CodePipeline
Utiliser AWS CodePipeline pour vérifier régulièrement s'il y a des éventuelles modifications

Format d'artefact de sortie
Choisissez le format d'artefact de sortie.

CodePipeline par défaut
AWS CodePipeline utilise le format zip par défaut pour les artefacts dans le pipeline. Il n'inclut pas les métadonnées git relatives au référentiel.

Clone complet
AWS CodePipeline transmet les métadonnées relatives au référentiel qui permettent aux actions suivantes d'effectuer un clone git complet. Il est uniquement pris en charge pour les actions AWS CodeBuild.

Etape de génération :

Ajouter une étape de génération Infos

Génération - facultatif

Fournisseur de génération
Il s'agit de l'outil de votre projet de génération. Spécifiez les détails d'artefact de génération comme le système d'exploitation, le fichier de spécification de génération et les noms de fichiers de sortie.

AWS CodeBuild ▾

Région

Europe (Irlande) ▾

Nom de projet
Sélectionnez un projet de génération existant dans la console AWS CodeBuild, ou créez un nouveau projet de génération, puis revenez à cette tâche.

X ou [Créer un projet](#)

Variables d'environnement - facultatif
Choisissez la clé, la valeur et le type de vos variables d'environnement CodeBuild. Dans le champ de valeur, vous pouvez référencer les variables générées par CodePipeline. [En savoir plus](#)

[Ajouter une variable d'environnement](#)

Type de génération

Génération unique
Déclenche une seule génération.

Génération de lot
Déclenche plusieurs générations en une seule exécution.

[Annuler](#) [Précédent](#) [Ignorer l'étape de génération](#) [Suivant](#)

Etape de déploiement :

Ajouter une étape de déploiement Infos

Déploiement - facultatif

Fournisseur de déploiement
Sélectionnez le mode de déploiement sur les instances. Sélectionnez le fournisseur, puis indiquez les détails de configuration le concernant.

AWS CodeDeploy ▾

Région
Europe (Irlande) ▾

Nom de l'application
Choisissez une application déjà existante dans la console AWS CodeDeploy, ou créez une nouvelle application, puis revenez à cette tâche.

benoit-second-project-java-legacy

Groupe de déploiement
Sélectionnez un groupe de déploiement existant dans la console AWS CodeDeploy, ou créez un nouveau groupe de déploiement, puis revenez à cette tâche.

benoit-second-project-java-legacy

On a la possibilité de revoir toute les configurations de notre pipeline :

Révision Infos

Étape 1 : Définir les paramètres de pipelines

Paramètres de pipeline

Nom de pipeline
Pipeline_Benoit_Project_Java_Legacy

Emplacement de l'artefact
benoit-bucket-second-project-java

Nom de rôle de service
benoit_role_pipeline_project

Création Pipeline :

Succès
Félicitations ! Le pipeline Pipeline_Benoit_Project_Java_Legacy a été créé.

Il faudra configurer **deux permissions** dans notre rôle nouvellement créé : **AmazonS3FullAccess** et **AWS CodeDeployRole**

Permissions policies (2) Info

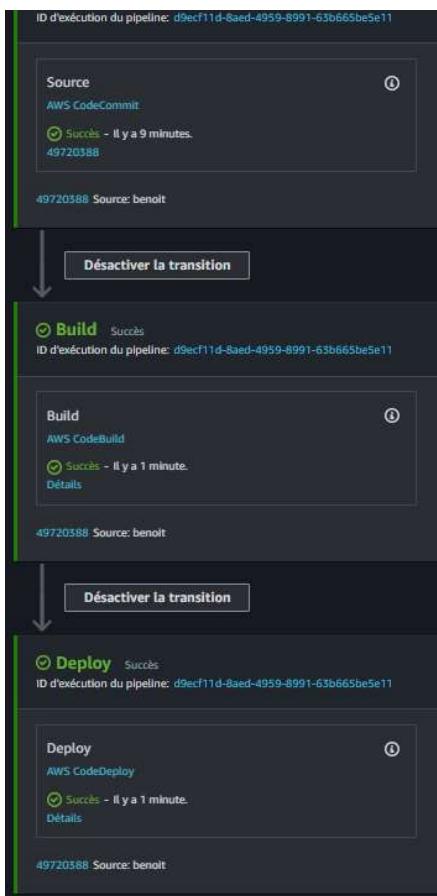
You can attach up to 10 managed policies.

Filter policies by property or policy name and p

<input type="checkbox"/>	Policy name <small>⤓</small>
<input type="checkbox"/>	<input type="checkbox"/> AmazonS3FullAccess
<input type="checkbox"/>	<input type="checkbox"/> AWSCodeDeployRole

Une fois les vérifications faites : on exécute le pipeline :

Source Succès



Notre Pipeline est fonctionnel

Etape de réalisation du pipeline en ligne de commande :

https://docs.aws.amazon.com/fr_fr/AmazonECR/latest/userguide/getting-started-cli.html

Création registry ecr :

Étape 3 : Créer un référentiel

Maintenant que vous disposez d'une image à transmettre à Amazon ECR, vous devez créer un référentiel afin de la contenir. Dans cet exemple, vous créez un référentiel nommé `hello-repository` dans lequel vous pourrez transmettre l'image `hello-world:latest`. Pour créer un référentiel, exécutez la commande suivante :

```
aws ecr create-repository \
--repository-name hello-repository \
--image-scanning-configuration scanOnPush=true \
--region region
```

Capture d'écran du shell du formateur

```
Administrator@Salle_3_Z MINGW64 ~/Documents/aws/ecr
$ aws ecr create-repository \
--repository-name mohamed-depot \
--image-scanning-configuration scanOnPush=true \
--region us-east-2
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:us-east-2:639962416620:repository/mohamed-depot",
        "registryId": "639962416620",
        "repositoryName": "mohamed-depot",
        "repositoryUri": "639962416620.dkr.ecr.us-east-2.amazonaws.com/mohamed-depot",
        "createdAt": "2023-02-27T12:11:27+01:00",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": true
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
```

```
        }
    }
}
} }
```

Capture d'écran de mon shell :

```
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/ecr
$ aws ecr create-repository \
> --repository-name benoit-depot \
> --image-scanning-configuration scanOnPush=true \
> --region eu-west-1
{
  "repository": {
    "repositoryArn": "arn:aws:ecr:eu-west-1:639962416620:repository/benoit-depot",
    "registryId": "639962416620",
    "repositoryName": "benoit-depot",
    "repositoryUri": "639962416620.dkr.ecr.eu-west-1.amazonaws.com/benoit-depot",
    "createdAt": "2023-02-27T12:12:16+01:00",
    "imageTagMutability": "MUTABLE",
    "imageScanningConfiguration": {
      "scanOnPush": true
    },
    "encryptionConfiguration": {
      "encryptionType": "AES256"
    }
  }
}
```

Résultat dans ECR :

The screenshot shows the AWS ECR console interface. At the top, it says 'Amazon ECR > Référentiels'. Below that, there are tabs for 'Private' and 'Public', with 'Private' being selected. The main area is titled 'Référentiels privés (14)'. A search bar is present. A table lists the repositories, with columns for 'Nom du référentiel' (Repository Name) and 'URI'. One row is visible, showing 'benoit-depot' and its URI.

Nom du référentiel	URI
benoit-depot	639962416620.dkr.ecr.eu-west-1.amazonaws.com/benoit-depot

Deuxième étape : s'authentifier :

Étape 2 : Vous authentifier auprès de votre registre par défaut

Après avoir installé et configuré la AWS CLI, authentifiez la CLI Docker auprès de votre registre par défaut. Ainsi, la commande `docker` peut pousser et extraire des images avec Amazon ECR. La AWS CLI fournit une commande `get-login-password` permettant de simplifier le processus d'authentification.

Le `get-login-password` est la méthode préférée pour s'authentifier auprès d'un registre privé Amazon ECR lors de l'utilisation de AWS CLI. Vérifiez que vous avez configuré votre AWS CLI pour interagir avec AWS. Pour en savoir plus, consultez [Bases de la configuration AWS CLI](#) dans le [guide de l'utilisateur AWS Command Line Interface](#).

Lorsque vous transmettrez le jeton d'authentification Amazon ECR à la commande `docker login`, utilisez la valeur `AWS` pour le nom d'utilisateur et spécifiez l'URI de registre Amazon ECR auquel vous voulez vous authentifier. Si vous vous authentifiez auprès de plusieurs registres, vous devrez répéter la commande pour chacun d'eux.

⚠️ Important

Si vous recevez une erreur ou si la commande `get-login-password` n'est pas disponible, assurez-vous que vous utilisez la dernière version de l'AWS CLI. Pour en savoir plus sur l'installation ou sur la mise à niveau vers la dernière version de la AWS CLI, consultez [Installer la AWS Command Line Interface](#) dans le [Guide de l'utilisateur AWS Command Line Interface](#).

- `get-login-password` (AWS CLI)

```
aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

```
Administrateur@LIL-JTJ3KN3 MINGW64 ~/Documents/Infrastructure as Code/ci_cd/ecr
$ aws ecr get-login-password --region eu-west-1 | docker login --username AWS --password-stdin 639962416620.dkr.ecr.eu-west-1.amazonaws.com/benoit-depot
Login Succeeded
```

3ème étape : push une image :

```
docker tag nginx:latest 639962416620.dkr.ecr.eu-west-1.amazonaws.com/benoit-depot:latest
```

```
docker tag nginx:latest 639962416620.dkr.ecr.eu-west-1.amazonaws.com/benoit-depot:latest
```

```
docker push 639962416620.dkr.ecr.eu-west-1.amazonaws.com/benoit-depot:latest
```

```
docker push 639962416620.dkr.ecr.eu-west-1.amazonaws.com/benoit-depot:latest
```

Vérification de l'image sur le dépôt ECR

benoit-depot						
Images (1)						
<input type="text"/> Rechercher des images						
□	Balise d'image	Type d'artefact	Transmis à	Taille (Mo)	URI de l'image	Digest
□	latest	Image	27 février 2023, 12:47:48 (UTC+01)	56.89	<input type="button"/> Copier l'URI	<input type="text"/> sha256:7f797701ded5055676d656f11071f...

Suite à cette étape : création d'un projet codeCommit et d'un codeBuild sur lequel on ne générera pas d'artefact -> construction du **buildspec.yml**

Ne pas oublier de cocher la case dans la partie **environnement** du **codeBuild** pour que nos images Docker puissent utiliser des commandes privilégiées

Privilégié

Activez cet indicateur si vous souhaitez créer des images Docker ou pour que vos générations bénéficient de priviléges élevés.

Rôle de service

Choisir un rôle de service existant dans votre compte

arn:aws:iam::639962416620:role/service-role/role_service_arn_project_spring

Autoriser AWS CodeBuild à modifier ce rôle de service pour qu'il puisse être utilisé avec ce projet de génération

Buildspec pour docker :

<https://docs.aws.amazon.com/codebuild/latest/userguide/sample-docker.html>

Files

This sample uses these files.

`buildspec.yml` (in `(root directory name)`)

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
```

Code après les modifications :

```
benoit-project-docker-repository / buildspec.yml infos
1 version: 0.2
2
```

```

3 env:
4   variables:
5     AWS_DEFAULT_REGION: eu-west-1
6     AWS_ACCOUNT_ID: 639962416620
7     IMAGE_REPO_NAME: benoit-depot
8
9   phases:
10    install:
11      runtime-versions:
12        java: corretto8
13    pre build:
14      commands:
15        - echo Logging in to Amazon ECR...
16        - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
17        - IMAGE_TAG=$(date +%-Y-%m-%d.%H.%M.%S).$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | head -c 8)"
18        - echo Maven operation
19        - "mvn clean package"
20    build:
21      commands:
22        - echo Build started on `date`
23        - echo Building the Docker image...
24        - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
25        - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
26    post build:
27      commands:
28        - echo Build completed on `date`
29        - echo Pushing the Docker image...
30        - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG

```

Bloc env :

```

env:
  variables:
    AWS_DEFAULT_REGION: eu-west-1
    AWS_ACCOUNT_ID: 639962416620
    IMAGE_REPO_NAME: benoit-depot

```

Variable utilisé dans notre **buildspec.yml**

Il faut également modifier les politiques de notre rôle de sécurité **role_service_arn_project_spring_benoit**

Pour être sur de notre rôle : on vérifie dans "environnement" sur notre code build

On lui rajoute le rôle **allow-ecr**

role_service_arn_project_spring_benoit

Récapitulatif

Date de création	ARN
February 24, 2023, 15:01 (UTC+01:00)	arn:aws:iam::639962416620:role/service-role/role_service_arn_project_spring_benoit
Dernière activité	Durée maximale de la session
Il y a 4 heures	1 heure

[Autorisations](#)

[Relations d'approbation](#)

[Balises](#)

[Access Advisor](#)

[Révoquer les séances](#)

Politiques des autorisations (5) [Infos](#)

Vous pouvez attacher jusqu'à 10 politiques gérées.

Filtrez les stratégies par nom de propriété ou de stratégie et appuyer sur Entrée.

Nom de la politique [?](#)

Type

Description



allow-ecr

Gérées par le client

allow-ecr

Créer une stratégie

Une stratégie définit les autorisations AWS que vous pouvez attribuer à un utilisateur, un groupe ou un rôle.
[En savoir plus](#)

Éditeur visuel

JSON

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Action": [  
6         "ecr:BatchCheckLayerAvailability",  
7         "ecr:CompleteLayerUpload",  
8         "ecr:GetAuthorizationToken",  
9         "ecr:InitiateLayerUpload",  
10        "ecr:PutImage",  
11        "ecr:UploadLayerPart"  
12      ],  
13      "Resource": "*",  
14      "Effect": "Allow"  
15    }  
16  ]  
17 }
```

Une fois cette opération effectué : on démarre la génération : normalement notre build sera en success si toute les étapes on été respecté :



Vérification dans notre Elastic Container Registry :

The screenshot shows the AWS Elastic Container Registry (ECR) interface. The top navigation bar includes 'benoit-depot' and 'Opérations'. The main area displays the 'Images (2)' section. A search bar contains the placeholder 'Rechercher des images'. The table lists two images:

	Balise d'image	Type d'artefact	Transmis à	Taille (Mo)	URI de l'image
<input type="checkbox"/>	2023-02-27.15.17.52.a706318f	Image	27 février 2023, 16:18:38 (UTC+01)	90.80	<input type="button"/> Copier l'URI
<input type="checkbox"/>	latest	Image	27 février 2023, 12:47:48 (UTC+01)	56.89	<input type="button"/> Copier l'URI

