

INTRODUCTION AUX AZURE PIPELINES

DÉFINITION

Azure Pipelines est un service de **CI/CD** (Intégration Continue et Déploiement Continu) proposé par Microsoft dans le cadre de la suite **Azure DevOps**.

AVANTAGES

- **Simple et rapide** à mettre en place
- Fonctionne avec **n'importe quel langage** ou plateforme
- **1 800 minutes gratuites** de CI/CD par mois pour les projets publics
- Intégration **native** avec d'autres services **Azure DevOps**

CAS D'UTILISATION

- **Automatiser** le processus de **build** et de **déploiement** de vos applications
- Exécuter des **tests automatisés** afin de détecter les problèmes tôt dans le cycle de développement
- Faciliter la **collaboration** pour les développeurs, les testeurs et les opérations

EXPLORER AZURE PIPELINES

INTRODUCTION AUX AZURE PIPELINES

DÉFINITION

Azure Pipelines est un service de **cloud** qui permet d'automatiser et d'orchestrer des processus de **build**, **test** et **déploiement** pour des applications.

AVANTAGES

- **Flexibilité** pour s'adapter à divers langages et plateformes
- **Intégration étroite** avec d'autres services Azure, GitHub et autres
- Facilité de configuration et de **surveillance** grâce à la version **YAML**

CAS D'UTILISATION

- **Automatisation** du processus de **build** et de **test** lors des commits (CI)
- **Déploiement continu** (CD) vers des environnements de **staging** ou de **production**

COMPOSANTS DES AZURE PIPELINES

Les pipelines sont composés de plusieurs éléments clés :

- **Builds** : processus de compilation et de test d'un projet
- **Releases** : processus de déploiement d'un projet vers un environnement donné
- **Artifacts** : éléments compilés et prêts à être déployés
- **Agents** : machines virtuelles exécutant des tâches

BUILDS

DÉFINITION

Les **builds** sont des processus **automatisés** qui compilent, testent et génèrent des **artifacts** à partir d'un code source.

EXEMPLES D'UTILISATION

- **Compiler** une application **.NET** sur un agent **Windows**
- **Exécuter** des tests unitaires avec **npm** pour une application **Node.js**

CONFIGURATIONS COMMUNES

- Utilisation d'un fichier **YAML** pour définir les étapes des tâches
- Choix d'un **agent approprié** pour l'exécution des tâches
- Génération et **stockage des artifacts**

RELEASES

DÉFINITION

Les **releases** sont des processus **automatisés** de déploiement des **artifacts** générés lors de la phase de **build**.

EXEMPLES D'UTILISATION

- Déployer une application Web sur Azure App Service
- Mettre à jour des services Kubernetes à l'aide de Helm

CONFIGURATIONS COMMUNES

- Utilisation de **modèles de déploiement préconfigurés**
- Configuration des **étapes de déploiement** et des **environnements**

INTRODUCTION AUX AZURE PIPELINES

YAML PIPELINES

Dans cette section, nous allons explorer les **YAML Pipelines** d'Azure pour déployer des applications et automatiser vos **builds** et **releases**.

INTRODUCTION AU FORMAT YAML

SYNTAXE DE BASE

Le **YAML** est un format de données facile à lire pour les humains, couramment utilisé pour la configuration des fichiers et le partage de données. Il est **sensible à l'indentation**.

prénom: John nom: Doe age: 30 langages:

- Python
- Java
- JavaScript

- Les fichiers YAML commencent généralement par `---` et se terminent par `...`
- L'indentation est réalisée avec des espaces, pas des tabulations
- Les listes sont représentées par des tirets `-`

ANATOMIE D'UN FICHIER YAML

```
cle: valeur
liste:
  - element1
  - element2
objet:
  cle1: valeur1
  cle2: valeur2
```

CRÉATION D'UN FICHIER YAML POUR AZURE PIPELINES

EXEMPLE DE FICHIER YAML

```
trigger:  
- main  
  
pool:  
  vmImage: 'ubuntu-latest'  
  
steps:  
- script: echo Bonjour, Azure Pipelines  
  displayName: 'Exécuter un script shell'
```

UTILISATION DE MODÈLES

Les **modèles** permettent de réutiliser des parties de code YAML dans plusieurs **pipelines**.

```
# mon_modele.yml
steps:
- script: echo Bonjour, Azure Pipelines
  displayName: 'Exécuter un script shell'

# azure-pipelines.yml
resources:
  repositories:
    - repository: templates
      type: git
      name: MyTemplates

jobs:
- template: mon_modele.yml@templates
```

VALIDATION ET DÉBOGAGE D'UN FICHIER YAML

OUTILS DE VALIDATION

- **YAMLLint**: un linter pour vérifier la syntaxe et la lisibilité des fichiers YAML.
- **Visual Studio Code**: avec l'extension "YAML", qui fournit la validation et l'autocomplétion.

RÉSOLUTION DES ERREURS COURANTES

1. **Vérifier l'indentation:** respecter la hiérarchie des blocs.
2. Utiliser les **bonnes clés** et la **bonne syntaxe** pour les objets et listes.
3. S'assurer que les **chaînes** sont correctement échappées si nécessaire.

TRIGGERS

INTRODUCTION AUX TRIGGERS

Les **triggers** permettent d'automatiser l'exécution d'une **pipeline Azure** en fonction d'événements spécifiques tels que les modifications du code, les **tags** et les **pull requests**.

TYPES DE TRIGGERS

BRANCHES

Les **triggers de branches** déclenchent une pipeline lorsqu'une modification est détectée sur une branche spécifique du **référentiel Git**.

TAGS

Les **triggers de tags** déclenchent une **pipeline** lorsqu'un nouveau tag est créé dans le **référentiel Git**.

PULL REQUESTS

Les **triggers de pull requests** déclenchent une **pipeline** lorsqu'une pull request est créée ou mise à jour dans le **référentiel Git**.

CONFIGURATION DES TRIGGERS

EXEMPLE DE CONFIGURATION DE TRIGGERS

Pour configurer les **triggers** dans un fichier YAML de la **pipeline Azure**, ajoutez la section `trigger` avec les branches, tags ou pull requests à surveiller.

```
trigger:  
  branches:  
    include:  
      - main  
      - feat/*  
    exclude:  
      - doc/*  
  tags:  
    include:  
      - v*  
    exclude:  
      - v1.0.*  
  pr:  
    autoCancel: true  
    branches:
```

RESTRICTIONS ET CONDITIONS

Les **triggers** peuvent être restreints et conditionnés par divers critères, tels que l'exclusion de certaines **branches** ou de certains **chemins**, et l'utilisation de **filtres** pour n'inclure que certains types d'événements.

Critères de restriction	Exemples
Branches	- master - develop
Chemins	- /src - /docs
Filtres d'événements	- push - pull_request

VARIABLES

VARIABLES

Les **variables** sont des éléments de données que vous pouvez **utiliser** et **modifier** tout au long de votre **pipeline**.

TYPES DE VARIABLES

- Variables prédefinies
- Variables personnalisées

VARIABLES PRÉDÉFINIES

Les **variables prédéfinies** sont des variables fournies par **Azure Pipelines** pour faciliter le suivi et la gestion des informations relatives au pipeline.

Exemples :

- Build.DefinitionName
- Build.SourceBranch
- Build.BuildNumber

VARIABLES PERSONNALISÉES

Les **variables personnalisées** sont des variables que vous créez pour stocker et gérer des informations spécifiques à votre **pipeline**.

Exemple :

```
variables:  
  customVariable: 'Hello, World!'
```

UTILISATION DES VARIABLES

SYNTAXE

Pour accéder à une variable dans votre **pipeline**, utilisez la syntaxe suivante :

- `$ (VariableName)` pour les variables simples
- `$ [variables.VariableName]` pour les expressions complexes

EXEMPLES D'UTILISATION

- Afficher une variable prédéfinie :

```
steps:  
- script: echo $(Build.BuildNumber)
```

- Utiliser une variable personnalisée :

```
variables:  
  customVariable: 'Hello, World!'  
steps:  
- script: echo $(customVariable)
```

TÂCHES ET ÉTAPES

DÉFINITION D'UNE TÂCHE

Une **tâche** est une unité de travail dans une **pipeline Azure**. Les tâches sont des actions individuelles, telles que l'exécution d'un script ou une opération comme la **compilation du code**.

EXEMPLES DE TÂCHES COURANTES

- **Compilation** du code
- Exécution de **tests unitaires**
- Envoi d'une **notification par e-mail**

UTILISATION DE TÂCHES PERSONNALISÉES

Vous pouvez créer des **tâches personnalisées** pour s'adapter aux besoins spécifiques de votre projet. Pour cela, il faut suivre la **documentation officielle** d'Azure DevOps.

DÉFINITION D'UNE ÉTAPE

Une **étape** est une phase d'exécution dans une **pipeline Azure**, qui contient un ensemble de **tâches** liées.

EXEMPLES D'ÉTAPES COURANTES

- **Étape de compilation** : Regroupe les tâches de compilation du code et d'exécution des tests unitaires
- **Étape de déploiement** : Effectue le déploiement de l'application sur un environnement de test ou de production

UTILISATION D'ÉTAPES PERSONNALISÉES

Créez des **étapes personnalisées** pour organiser votre pipeline selon les besoins de votre projet et pour faciliter la compréhension et la maintenance de la pipeline.

Avantages	Exemples
Modularité	Regroupement de tâches spécifiques
Lisibilité	Nom d'étape clair et descriptif
Maintenance facilitée	Modifications centrées sur une étape

ORCHESTRATION DES TÂCHES ET DES ÉTAPES

ENCHAÎNEMENT ET PARALLÉLISME

Les **tâches** et les **étapes** peuvent être exécutées en séquence ou en parallèle selon les besoins de votre pipeline.

Exemple d'enchaînement de tâches:

```
étape1-tâche1 -> étape1-tâche2 -> étape2-tâche1
```

Exemple de parallélisme entre étapes:

```
étape1-tâche1 -> étape1-tâche2  
étape2-tâche1
```

Contrôle de flux

Utilisez des **conditions** pour déterminer quelles tâches ou étapes doivent être exécutées ou sautées.

Exemple de condition:

```
steps:  
- script: echo 'Exécution conditionnelle'  
  condition: eq(variables['Build.SourceBranch'], 'refs/heads/main')
```

Opérateur	Description
eq (x, y)	Retourne vrai si x égal y
ne (x, y)	Retourne vrai si x différent de y
and (x, y)	Retourne vrai si x et y sont vrais
or (x, y)	Retourne vrai si x ou y est vrai

INTÉGRATIONS AVEC D'AUTRES SERVICES AZURE

AZURE REPOS

Azure Repos est un service d'hébergement de contrôle de version **Git** et **TFVC** (Team Foundation Version Control) dans **Azure DevOps**.

DÉFINITION

- **Service de contrôle de version**
- **Git et TFVC**
- Partie intégrante d'**Azure DevOps**

EXEMPLES D'INTÉGRATION

- Utiliser un **repo Azure** pour stocker le code source de votre application
- Utiliser les **PR (Pull Requests)** pour gérer et suivre les modifications du code
- Automatiser les **builds** et les **releases** à partir des commits sur le repo

AZURE BOARDS

Azure Boards est un service de gestion de projet et de suivi du travail pour **Azure DevOps**.

DÉFINITION

- **Gestion de projet agile**
- Suivi du travail : **éléments de travail**, états, itérations
- Partie intégrante d'**Azure DevOps**

EXEMPLES D'INTÉGRATION

- Associer des éléments de travail (issues, tâches) aux commits du **repo**
- Utiliser les tableaux **Kanban** pour suivre l'avancement du projet
- Générer des **rapports de projet** à partir des données de suivi

AZURE ARTIFACTS

Azure Artifacts est un service de gestion des paquets pour **Azure DevOps**.

DÉFINITION

- **Gestion des paquets** et de leurs **dépendances**
- Stockage et distribution des paquets : **NuGet**, **npm**, **Maven**, etc.
- Partie intégrante d'**Azure DevOps**

EXEMPLES D'INTÉGRATION

- **Héberger** vos paquets internes dans un **feed privé**
- Utiliser des paquets dans vos **pipelines** de build et de release
- Gérer les **dépendances** de votre projet avec des paquets hébergés dans **Azure Artifacts**

SÉCURITÉ ET AUTORISATIONS

AUTHENTIFICATION ET AUTORISATION

Il est important de gérer les **accès** et de contrôler les **autorisations** pour un usage sécurisé des **Azure Pipelines**.

CONCEPTS CLÉS

- **Authentification** : Processus de vérification de l'identité d'un utilisateur ou d'un système
- **Autorisation** : Processus d'attribution des droits d'accès aux ressources

EXEMPLES D'UTILISATION ET DE CONFIGURATION

AUTHENTIFICATION

- Utilisation d'**Azure Active Directory (AAD)**
- Utilisation d'**identifiants de service** pour l'authentification automatique

AUTORISATIONS

- **Gestion des rôles d'accès** pour les utilisateurs et les groupes
- Configuration des **accès** aux pipelines, **artefacts** et **environnements**

GESTION DES AUTORISATIONS

RÔLES D'ACCÈS

Les **rôles d'accès** sont utilisés pour contrôler les **permissions** dans les **Azure Pipelines**.

- Project Collection Administrators
- Project Administrators
- Build Administrators
- Release Administrators
- Contributors

EXEMPLE DE CONFIGURATION DES AUTORISATIONS

1. Accédez à la page des **Pipelines** dans **Azure DevOps**
2. Cliquez sur le menu déroulant "..." à côté du pipeline
3. Sélectionnez "**Sécurité**"
4. Recherchez un utilisateur ou un groupe dans la liste
5. Modifiez les autorisations en fonction des besoins

