Jenkinsfile

Un **Jenkinsfile** est un fichier texte au format Groovy qui décrit un pipeline Jenkins. Il détaille les différentes phases du processus d'intégration continue.

Jenkinsfile offre deux manières de coder : une syntaxe déclarative (declarative pipeline) basée sur des mots-clés et une syntaxe scriptée (scripted pipeline) basée sur le langage **Groovy**.

Voici un exemple de Jenkinsfile (declarative pipeline):

```
pipeline {
   agent any
--ou--
   agent {
      node {
         label 'build'
      }
   }
   tools {
      maven 'M2_HOME'
   }
   options {
      --Timeout counter starts after agent is allocated--
      timeout(time: 1, unit: 'SECONDS')
   }
   environment {
      APP_ENV  = "DEV"
   }
   stages {
      stage('Code Checkout') {
         steps {
            git branch: 'master',
            url: 'https://github.com/hwafa/atelier-jenkins.git',
            credentialsId:  'jenkins-example-github-pat'
         }
      }
      stage('Code Build') {
         steps {
             sh 'mvn install -Dmaven.test.skip=true'
         }
      }
   }
     post {
       always {
             echo "======always======"
       }
       success {
             echo "=====pipeline executed successfully ====="
       }
       failure {
             echo "======pipeline execution failed======"
       }
     }
   }
```

Structure du fichier Jenkinsfile

- Le bloc pipeline : c'est le premier bloc de type pipeline, il va contenir des directives, qui sont de type agent, tools, options, environment, post et stages.

```
pipeline {

---La déclaration du pipeline se fait dans un block pipeline---

}
```

- La directive agent : nous définissons sur quel agent va tourner notre pipeline. On peut ainsi indiquer n'importe quel agent avec la balise any, ou des spécifiques en indiquant un node portant un label.

```
pipeline {
   agent any
---Le reste du code---
}
```

Ou :

```
pipeline {
   agent {
      node {
         label 'ansible'
      }
   }
---Le reste du code---

}
```

- La directive tools : ce bloc permet d'indiquer quels outils utiliser. Ces outils doivent être installés et configurés au préalable.

```
pipeline {
   agent any
   tools {
      maven 'M2_HOME'
   }
---Le reste du code---
}
```

- La directive options : ce bloc contient toutes les options requises par le job. Certaines de ces options sont natives et d'autres apportées par les plugins. Ces options ont une portée sur l'ensemble du pipeline. Ces options peuvent aussi être définies au niveau des stages.

```
pipeline {
   agent any
   tools {
      maven 'M2_HOME'
   }
   options {
      --Timeout counter starts after agent is allocated--
      timeout(time: 1, unit: 'SECONDS')
   }
---Le reste du code---
   }
```

- La directive environnement : cette directive permet de définir des variables d'environnement utilisé dans tout le pipeline. Cette directive peut également être définie au niveau de chaque stage.

```
pipeline {
        agent any
        tools {
           maven 'M2_HOME'
        }
        options {
           timeout(time: 1, unit: 'SECONDS')
        }
        environment {
           APP_ENV  = "DEV"
        }
      ---Le reste du code---
        }
```

- La directive stages : cette section va permettre de définir tous les stages d'un pipeline. Il en faut au minimum un. Chaque stage va contenir des steps qui doit contenir au minimum un step. Les steps permettent de lancer des fonctions.

```
pipeline {
   agent any
   tools {
      maven 'M2_HOME'
   }
   options {
      timeout(time: 1, unit: 'SECONDS')
   }
   environment {
      APP_ENV  = "DEV"
   }
stages {
      stage('Code Checkout') {
         steps {
            git branch: 'master',
            url: 'https://github.com/hwafa/atelier-jenkins.git',
            credentialsId:  'jenkins-example-github-pat'
         }
      }
}
---Le reste du code---
   }
```

- La directive post : cette directive permet de définir un ou plusieurs steps qui sont exécutés à la fin de l'exécution d'un pipeline.

```
pipeline {
   agent any
   stages {
      stage('Example') {
         steps {
            echo 'Hello World'
         }
      }
  }
post {
       always {
            echo "======always======"
       }
       success {
            echo "=====pipeline executed successfully ====="
       }
       failure {
            echo "======pipeline execution failed======"
       }
    }
   }
```