# TP Jenkins

### Étape 1 : Configuration Globale des Outils (Ant & Maven)

Avant de créer les jobs, il faut indiquer à Jenkins où trouver les installations d'Ant et de Maven.

1. Allez dans Gérer Jenkins (Manage Jenkins) dans le menu de gauche.
2. Cliquez sur Outils (Tools) - *Note: Dans certaines versions plus anciennes ou configurations différentes, cela pourrait être sous "Global Tool Configuration"*.
3. Descendez jusqu'à la section **Ant**.
    - Cliquez sur Installations d'Ant... (Ant installations...).
    - Cliquez sur Ajouter Ant (Add Ant).
    - Donnez un **Nom** (ex: Ant_1.10.x).
    - Décochez Installer automatiquement (Install automatically) si Ant est déjà installé sur la machine.
    - Dans ANT_HOME, entrez le chemin absolu vers le répertoire d'installation d'Ant (ex: /usr/share/ant sur Linux, C:\apache-ant-1.10.13 sur Windows).
    - Cliquez sur Enregistrer (Save) en bas de la page.
4. Descendez jusqu'à la section **Maven**.
    - Cliquez sur Installations de Maven... (Maven installations...).
    - Cliquez sur Ajouter Maven (Add Maven).
    - Donnez un **Nom** (ex: Maven_3.9.x).
    - Décochez Installer automatiquement (Install automatically) si Maven est déjà installé.
    - Dans MAVEN_HOME, entrez le chemin absolu vers le répertoire d'installation de Maven (ex: /opt/maven sur Linux, C:\apache-maven-3.9.6 sur Windows).
    - Cliquez sur Enregistrer (Save) en bas de la page.
5. *Optionnel mais recommandé :* Vérifiez/configurez le **JDK** dans la même section Outils si vos projets nécessitent une version spécifique.

### Étape 2 : Création et Configuration du Job pour le Projet Ant

1. Retournez à la page d'accueil de Jenkins.
2. Cliquez sur Nouveau Item (New Item) dans le menu de gauche.
3. Entrez un nom pour votre job (ex: MonProjet-Ant-Build).
4. Sélectionnez Freestyle project.
5. Cliquez sur OK. Vous êtes maintenant sur la page de configuration du job.
6. **Description :** (Optionnel) Ajoutez une description du projet (ex: "Build du projet X utilisant Ant").
7. **Gestion de code source** (Source Code Management) :
    - Sélectionnez Git.

- Dans Repository URL, collez l'URL de votre dépôt Git pour le projet Ant (ex: https://github.com/votre-user/projet-ant.git ou git@github.com:votre-user/projet-ant.git).
- Credentials : Sélectionnez None si le dépôt est public. Si privé, cliquez sur Add > Jenkins pour ajouter vos identifiants SSH (avec clé privée) ou HTTPS (avec nom d'utilisateur/mot de passe ou token).
- Branches to build : Laissez */main ou */master (ou spécifiez la branche que vous voulez construire).

8. **Déclencheurs de build** (Build Triggers) :
   - Pour ce TP, laissez vide pour déclencher manuellement. (Plus tard, vous pourrez explorer Polling SCM, Build periodically, GitHub hook trigger...).
9. **Environnement de Build** (Build Environment) :
   - Laissez vide pour ce TP simple.
10. **Build Steps** :
    - Cliquez sur Ajouter une étape au build (Add build step).
    - Sélectionnez Invoquer Ant (Invoke Ant).
    - Version d'Ant : Choisissez le nom que vous avez donné à votre installation Ant à l'Étape 1 (ex: Ant_1.10.x).
    - Targets : Entrez les cibles (targets) Ant que vous voulez exécuter, séparées par des espaces. Par exemple :
      - clean compile test package (si ces cibles existent dans votre build.xml).
      - Laissez vide pour exécuter la cible par défaut définie dans votre build.xml.
    - Cliquez sur Avancé... (Advanced...).
    - Build File : Laissez vide si votre build.xml est à la racine du projet. Sinon, spécifiez le chemin relatif depuis la racine (ex: chemin/vers/mon_build.xml).
    - Properties : Vous pouvez passer des propriétés à Ant ici (ex: ma.propriete=valeur).
11. **Actions à la suite du build** (Post-build Actions) :
    - (Optionnel mais utile) Cliquez sur Ajouter une action après le build (Add post-build action).
    - Sélectionnez Archiver les artefacts (Archive the artifacts).
    - Dans Fichiers à archiver (Files to archive), spécifiez les fichiers générés par le build que vous voulez conserver. Utilisez les motifs Ant (wildcards). Exemples :
      - dist/**/*.jar (tous les JARs dans le répertoire dist et ses sous-répertoires)
      - build/libs/*.war (le fichier WAR dans build/libs)
      - target/mon-app.jar (un JAR spécifique dans target)
      - *Adaptez ceci en fonction de la sortie de votre build Ant.*
12. Cliquez sur Enregistrer (Save).

**Étape 3 : Création et Configuration du Job pour le Projet Maven**

1. Retournez à la page d'accueil de Jenkins.
2. Cliquez sur Nouveau Item (New Item).
3. Entrez un nom pour votre job (ex: MonProjet-Maven-Build).
4. Sélectionnez Freestyle project.
5. Cliquez sur OK.
6. **Description :** (Optionnel) Ajoutez une description (ex: "Build du projet Y utilisant Maven").
7. **Gestion de code source** (Source Code Management) :
    - Configurez Git exactement comme à l'Étape 2, mais avec l'URL du dépôt Git de votre projet Maven. N'oubliez pas les Credentials si nécessaire et la branche (Branches to build).
8. **Déclencheurs de build** (Build Triggers) :
    - Laissez vide pour un déclenchement manuel.
9. **Environnement de Build** (Build Environment) :
    - Laissez vide.
10. **Build Steps** :
    - Cliquez sur Ajouter une étape au build (Add build step).
    - Sélectionnez Invoquer des cibles Maven de haut niveau (Invoke top-level Maven targets).
    - Version de Maven : Choisissez le nom que vous avez donné à votre installation Maven à l'Étape 1 (ex: Maven_3.9.x).
    - Goals : Entrez les phases ou goals Maven que vous voulez exécuter, séparés par des espaces. Exemples courants :
        - clean install (nettoie, compile, teste, package et installe dans le dépôt local)
        - clean package (nettoie, compile, teste, package)
        - verify (exécute les tests d'intégration)
    - Cliquez sur Avancé... (Advanced...).
    - POM : Laissez vide si votre pom.xml est à la racine du projet. Sinon, spécifiez le chemin relatif (ex: chemin/vers/mon_pom.xml).
    - Properties : Vous pouvez passer des propriétés système Maven ici (ex: -DskipTests=true).
11. **Actions à la suite du build** (Post-build Actions) :
    - (Optionnel) Cliquez sur Ajouter une action après le build (Add post-build action).
    - Sélectionnez Archiver les artefacts (Archive the artifacts).
    - Dans Fichiers à archiver (Files to archive), spécifiez les artefacts Maven. Exemples :
        - target/*.jar (le JAR principal généré dans le répertoire target)
        - target/*.war (le WAR généré dans target)

        - *Adaptez ceci en fonction de la sortie de votre build Maven (défini dans le pom.xml).*
12. Cliquez sur Enregistrer (Save).

**Étape 4 : Exécution et Vérification des Builds**

1. Depuis le tableau de bord Jenkins, vous devriez voir vos deux nouveaux jobs : MonProjet-Ant-Build et MonProjet-Maven-Build.
2. Pour lancer un build, cliquez sur l'icône "Lancer un build" (une flèche verte ou une horloge avec une flèche verte) à côté du nom du job.
3. Un nouveau build apparaîtra dans la section Historique des builds (Build History) pour ce job. Il aura un numéro (ex: #1).
4. Cliquez sur le numéro du build pour voir sa page de détails.
5. **Consulter la sortie de la console** :
    - Cliquez sur Sortie de la console (Console Output) dans le menu de gauche de la page du build.
    - Vous verrez les logs détaillés : récupération du code depuis Git, exécution des commandes Ant/Maven.
    - Recherchez les messages de succès (BUILD SUCCESSFUL pour Ant, BUILD SUCCESS pour Maven) ou les erreurs éventuelles.
6. **Vérifier les artefacts** :
    - Si le build a réussi et que vous avez configuré l'archivage, retournez à la page de résumé du build (#1).
    - Vous devriez voir un lien Artefacts (Artifacts) ou Derniers artefacts archivés (Last Successful Artifacts) sur la page principale du job ou la page du build spécifique. Cliquez dessus pour voir et télécharger les fichiers archivés.

**Étape 5 : Création du Jenkinsfile pour le Projet Ant**

1. À la racine de votre projet Ant (dans votre copie locale), créez un nouveau fichier nommé exactement Jenkinsfile (sans extension).
2. Collez le contenu suivant dans ce fichier, en adaptant les parties commentées :

```
// Jenkinsfile (Declarative Pipeline) pour le projet Ant

pipeline {
    // Définit où le pipeline s'exécutera. 'any' signifie sur n'importe quel agent disponible.
    agent any

    // Définit les outils à mettre à disposition dans le PATH pendant l'exécution
    tools {
        // Assurez-vous que 'Ant_1.10.x' correspond EXACTEMENT au nom configuré dans Jenkins -> Outils
        ant 'Ant_1.10.x'
        // Ajoutez un JDK si votre projet en requiert un spécifique et qu'il est configuré dans Jenkins -> Outils
        // jdk 'AdoptOpenJDK-11'
```

```
    }

    stages {
        // Étape 1 : Récupération du code source
        stage('Checkout') {
            steps {
                // Nettoie l'espace de travail avant le checkout (nécessite Workspace Cleanup
Plugin)
                cleanWs()
                // Récupère le code depuis le dépôt configuré dans le job Jenkins
                // Les détails (URL, credentials) seront pris depuis la configuration du job
Pipeline
                checkout scm
                echo 'Code source récupéré.'
            }
        }

        // Étape 2 : Construction avec Ant
        stage('Build') {
            steps {
                echo "Début du build Ant..."
                // Exécute Ant. Utilise sh pour Linux/macOS, bat pour Windows.
                // La directive 'tools' a ajouté l'exécutable Ant au PATH.
                // Adaptez les cibles (targets) à votre build.xml
                // Si build.xml n'est pas à la racine : ant(buildFile: 'chemin/vers/build.xml',
targets: 'clean compile test package')
                sh 'ant clean compile test package' // Ou 'bat "ant clean compile test package"'
pour Windows
                echo 'Build Ant terminé.'
            }
        }

        // Étape 3 : Archivage des artefacts
        stage('Archive Artifacts') {
            steps {
                echo "Archivage des artefacts..."
                // Archive les fichiers produits par le build Ant
                // Adaptez le pattern à vos artefacts réels (ex: 'dist/**/*.jar', 'build/libs/*.war')
                archiveArtifacts artifacts: 'dist/**/*.jar', fingerprint: true
                echo "Artefacts archivés."
            }
        }
    }

    // Actions exécutées à la fin du pipeline, quel que soit le résultat (success, failure,
etc.)
    post {
        always {
            echo 'Pipeline terminé.'
            // Optionnel: Nettoyer l'espace de travail après le build
```

```
            // cleanWs()
        }
        success {
            echo 'Build réussi !'
            // Envoyer une notification de succès, etc.
        }
        failure {
            echo 'Build échoué !'
            // Envoyer une notification d'échec, etc.
        }
    }
}
```

1. **Adaptez** :
    - Le nom de l'outil Ant (ant 'Ant_1.10.x') si vous l'avez nommé différemment.
    - Ajoutez/modifiez la ligne jdk si nécessaire.
    - La commande sh 'ant ...' : modifiez les cibles (clean compile test package) selon votre build.xml. Si votre build.xml n'est pas à la racine, utilisez la syntaxe commentée ant(buildFile: '...', targets: '...'). Utilisez bat si votre agent Jenkins tourne sous Windows.
    - Le pattern dans archiveArtifacts pour correspondre aux fichiers que votre build Ant produit.
2. Ajoutez, commitez et pushez ce Jenkinsfile dans votre dépôt Git du projet Ant.

**Étape 6 : Création du Jenkinsfile pour le Projet Maven**

1. À la racine de votre projet Maven (dans votre copie locale), créez un nouveau fichier nommé Jenkinsfile.
2. Collez le contenu suivant, en adaptant :

```
// Jenkinsfile (Declarative Pipeline) pour le projet Maven

pipeline {
    agent any

    tools {
        // Assurez-vous que 'Maven_3.9.x' correspond EXACTEMENT au nom configuré dans Jenkins -> Outils
        maven 'Maven_3.9.x'
        // Ajoutez un JDK si nécessaire
        // jdk 'AdoptOpenJDK-11'
    }

    stages {
        stage('Checkout') {
            steps {
```

```
                cleanWs()
                checkout scm
                echo 'Code source récupéré.'
            }
        }

        // Étape 2 : Construction avec Maven
        stage('Build') {
            steps {
                echo "Début du build Maven..."
                // Exécute Maven. 'mvn' est dans le PATH grâce à la directive 'tools'.
                // Adaptez les goals Maven (ex: 'clean install', 'clean package', 'verify')
                // Si pom.xml n'est pas à la racine: sh "mvn -f chemin/vers/pom.xml clean package"
                sh 'mvn clean package' // Ou 'bat "mvn clean package"' pour Windows
                echo 'Build Maven terminé.'
            }
        }

        // Étape 3 : Archivage des artefacts
        stage('Archive Artifacts') {
            steps {
                echo "Archivage des artefacts..."
                // Archive les fichiers produits par le build Maven (typiquement dans 'target/')
                // Adaptez le pattern si nécessaire (ex: 'target/*.war', '**/target/*.jar')
                archiveArtifacts artifacts: 'target/*.jar', fingerprint: true
                echo "Artefacts archivés."
            }
        }
    }

    post {
        always {
            echo 'Pipeline terminé.'
            // cleanWs()
        }
        success {
            echo 'Build réussi !'
        }
```

```
        failure {
            echo 'Build échoué !'
        }
    }
}
```

1. **Adaptez** :
    - Le nom de l'outil Maven (maven 'Maven_3.9.x').
    - Ajoutez/modifiez la ligne jdk si nécessaire.
    - La commande sh 'mvn ...' : modifiez les goals (clean package) si besoin. Utilisez bat pour Windows. Spécifiez -f chemin/pom.xml si le POM n'est pas à la racine.
    - Le pattern dans archiveArtifacts (généralement target/*.jar ou target/*.war pour Maven).
2. Ajoutez, commitez et pushez ce Jenkinsfile dans votre dépôt Git du projet Maven.

**Étape 7 : Création des Jobs Pipeline dans Jenkins**

Créez un job pour chaque projet :

1. Retournez à la page d'accueil de Jenkins.
2. Cliquez sur Nouveau Item (New Item).
3. Entrez un nom pour votre job (ex: MonProjet-Ant-Pipeline).
4. Sélectionnez Pipeline.
5. Cliquez sur OK.
6. **Description :** (Optionnel) Ajoutez une description.
7. Descendez jusqu'à la section **Pipeline**.
8. **Definition :** Sélectionnez Pipeline script from SCM. C'est l'approche recommandée, Jenkins lira le Jenkinsfile depuis votre dépôt Git.
9. **SCM :** Sélectionnez Git.
10. **Repository URL :** Collez l'URL de votre dépôt Git pour le projet correspondant (Ant ou Maven).
11. **Credentials :** Sélectionnez les identifiants appropriés si le dépôt est privé.
12. **Branches to build (Branch Specifier) :** Entrez la branche où se trouve votre Jenkinsfile (ex: */main ou */master).
13. **Script Path :** Laissez la valeur par défaut Jenkinsfile. C'est le nom du fichier que Jenkins cherchera à la racine de la branche spécifiée.
14. (Optionnel) Configurez les Déclencheurs de build (Build Triggers) si vous voulez une exécution automatique (ex: Poll SCM ou via Webhooks). Pour ce TP, laissez vide pour déclencher manuellement.
15. Cliquez sur Enregistrer (Save).

16. Répétez les étapes 2 à 15 pour l'autre projet (ex: MonProjet-Maven-Pipeline), en utilisant l'URL du dépôt Git du projet Maven.

**Étape 8 : Exécution et Vérification des Pipelines**

1. Depuis le tableau de bord Jenkins, localisez vos nouveaux jobs Pipeline.
2. Cliquez sur Lancer un build (Build Now) à côté du nom de chaque job.
3. Un nouveau build apparaîtra dans l'Historique des builds (Build History). Cliquez sur le numéro du build (ex: #1).
4. Sur la page du build, vous verrez la **Vue des étapes** (Stage View), qui montre la progression à travers les différentes étapes définies dans le Jenkinsfile (Checkout, Build, Archive Artifacts). Les étapes vertes indiquent un succès, les rouges un échec.
5. Cliquez sur Sortie de la console (Console Output) dans le menu de gauche pour voir les logs détaillés, comme pour les jobs Freestyle. Vérifiez les messages Ant/Maven et recherchez BUILD SUCCESSFUL / BUILD SUCCESS.
6. Si le pipeline réussit et atteint l'étape d'archivage, vous trouverez un lien Artefacts (Artifacts) sur la page du build, vous permettant de télécharger les fichiers .jar ou .war produits.