TP2 : JENKINS

**1- Installation de Jenkins**

Suivre le tutorial : *Installing and Running Jenkins as a Standalone Application*

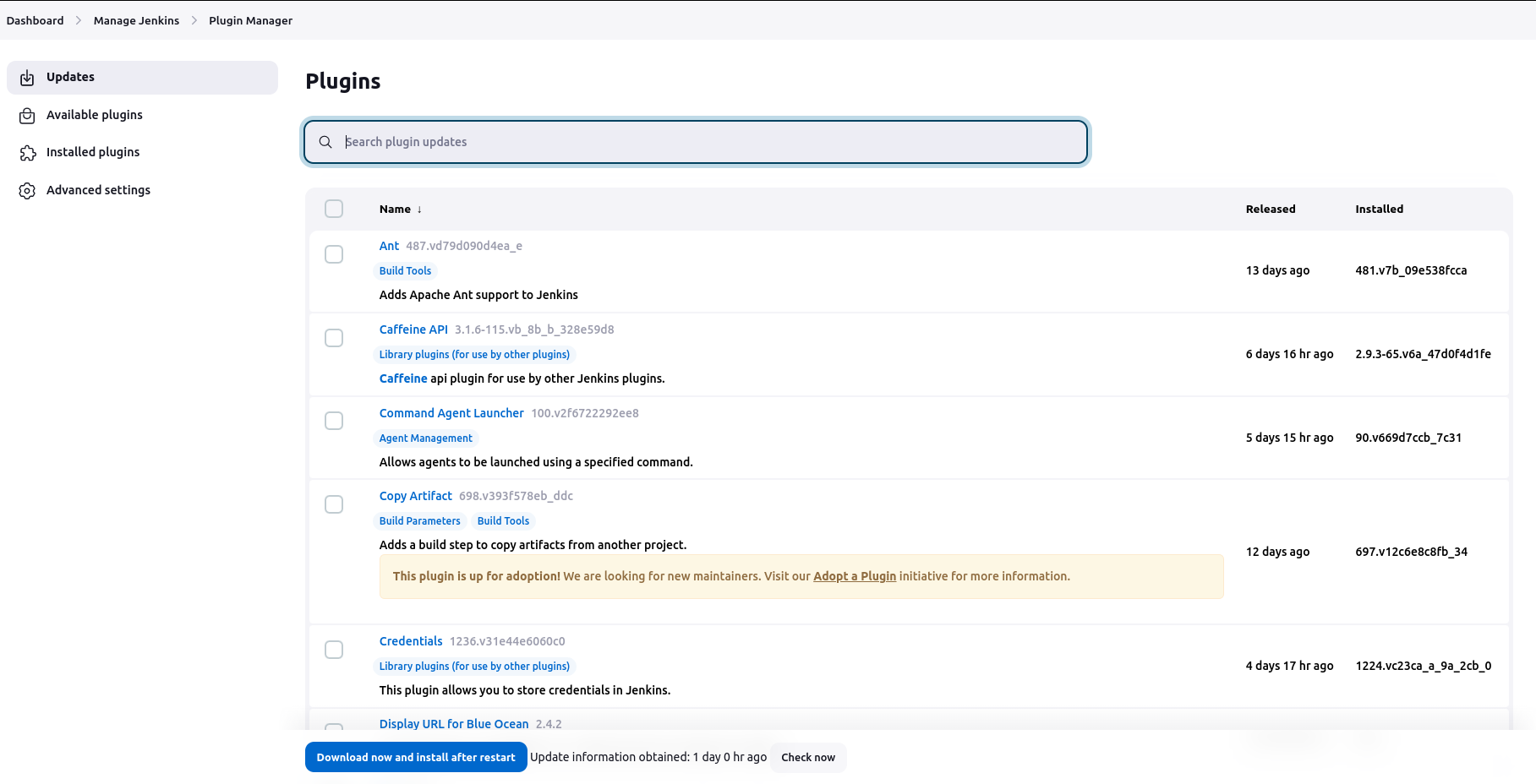
**2- Installation de plugins**

Une grande richesse de Jenkins est son système de plugins, ainsi que le grand nombre de plugins existant.

Pour accéder aux écrans de gestion des devrait vous mener à <http://localhost:8080/pluginManager/>.

Nous allons choisir les plugins dont nous avons besoin pour faire fonctionner nos projets notamment : maven, Unit Test, Docker, …

**PS : Si vous n’arrivez pas c’est pas très grave passer à la suite du TP.**



**2- Configuration du GitHub - Jenkins**

Tout d’abord fork le projet : <https://github.com/labibaVinson/TP4IPS>

Puis appliquer le tuto suivant :

Suivre le tutoriel : *Integrating GitHub Webhooks with Jenkins to automate unit and integration test after GitHub events for CI/CD*

**3- Creer un Job**

* Dans Jenkins Cliquez sur [Nouveau Item] puis sur FreeStyle et saisissez un nom pour votre Job.
* Entrez l’url de votre projet au format https.
* Créez le crédential avec votre utilisateur + PAT

### Une fois sauvegardé, il suffit de faire un commit dans votre projet Git. Si vous cliquez ensuite sur le job, vous devriez voir les traces de son exécution.

**4- Jenkins pipeline**

Un fichier Jenkinsfile utilise une syntaxe appelée Jenkins Job DSL qui est fournie avec le plugin [Pipeline](https://plugins.jenkins.io/workflow-aggregator/). Voici un exemple de Jenkinsfile :

| **pipeline {** |  |
| --- | --- |
|  | **agent { docker { image 'python:3.7.2' } }** |
|  | **stages {** |
|  | **stage('build') {** |
|  | **steps {** |
|  | **sh 'pip install -r requirements.txt'** |
|  | **}** |
|  | **}** |
|  | **stage('test') {** |
|  | **steps {** |
|  | **sh 'python test.py'** |
|  | **}** |
|  | **post {** |
|  | **always {** |
|  | **junit 'test-reports/\*.xml'** |
|  | **}** |
|  | **}** |
|  | **}** |
|  | **}** |
|  | **}** |

### **Rappel :**

### **Le bloc pipeline**

Tous les pipelines possèdent un premier bloc de type pipeline.

C’est ce bloc qui va contenir des directives, qui sont pour les principales de type agent, tools, options, environment, post et stages. Nous verrons dans un autre billet le reste des directives.

#### **La directive Agent (requis)**

Dans le bloc agent où 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

#### **La directive stages (requise)**

Cette section va permettre de définir tous les stages d’un pipeline. Il en faut au minimum un. indiquant un node portant un label.

Chaque stage va contenir des steps qui doivent contenir au minimum un step.

#### **Le code des steps**

Composantes des stages, les steps permettent de lancer des fonctions, appelés step. Ces steps sont composés de celles fournies nativement avec Jenkins, appelés basic steps.

#### **La directive post (optionnel)**

Cette directive permet de définir un ou plusieurs steps qui sont exécutés à la fin de l’exécution d’un pipeline. Post peut prendre des post-condition permettent l’exécution des steps à l’intérieur de chaque condition en fonction de l’état d’exécution du pipeline : always, changed, fixed; regression, aborted,...

**1- Pipeline avec paramètre:**

Créer un job jenkins qui permet d’utiliser les paramètres de builds et d’afficher nos paramètres dans la console.( tester les différents types de paramètres)

**2- Pipeline avec variable d’environnement:**

Créer une pipeline Jenkins qui permet d’utiliser les paramètres dans les steps comme exemple ci-après:

pipeline{

agent any

environment {

VAR = "${params.VAR}"

}

stages {

stage("Display param") {

steps {

script {

if(VAR)

{

Echo “my value is true”

}

else if(VAR)

{

Echo “my value is false”

}

}

}

}

}

}

**3- Pipeline avec conditions:**

Créer une pipeline avec conditions comme dans l'exemple ci après :

1. When :

when {

expression {

params.ENVIRONMENT == 'development'

}

}

2- IF-ELSE :

node {

stage('Step1') {

if (env.BRANCH\_NAME == 'main') {

echo 'Hello from main branch'

} else {

sh "echo 'Hello from ${env.BRANCH\_NAME} branch!'"

}

}

}