TP4 – Architecture distribuée, déploiemennt sur environnement de test et test de performance

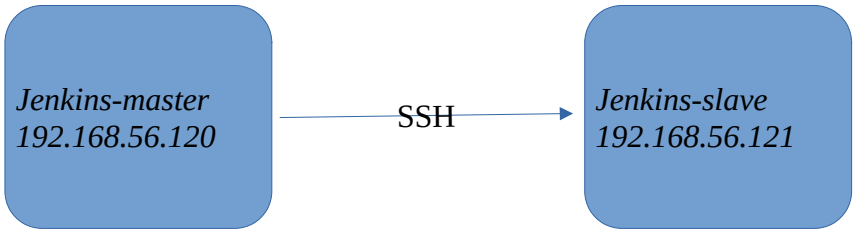
Brahim HAMDI

Objectifs :

- Créer une architecture distribuée
- Configurer et exécuter un test de performance dans un environnement de tests automatiques

Connexion SSH jenkins maître-esclave

1. Maintenant que le nœud slave est prêt, on va tester la connexion ssh depuis jenkins master vers slave. Jenkins maître utilise ssh pour communiquer avec le slave linux.



- Pour tester cette connexion, taper la commande suivante depuis jenkins maître (VM Jenkins-master) :

  ssh vagrant@192.168.56.121

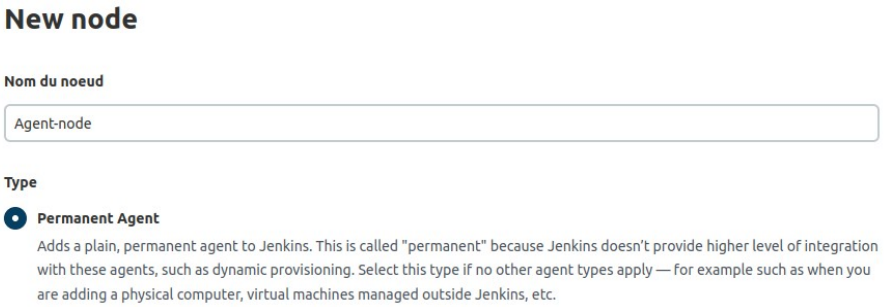
  Le mot de passe est « *vagrant* » aussi.

```
brahim@Training:~/jenkins-dist$ vagrant ssh Jenkins-master
Last login: Sat Sep  2 16:30:03 2023 from 10.0.2.2
vagrant@Jenkins-master:~$ ssh vagrant@192.168.56.121
vagrant@192.168.56.121's password:
Last login: Sun Sep  3 13:26:14 2023 from 10.0.2.2
vagrant@Jenkins-slave:~$
vagrant@Jenkins-slave:~$ exit
logout
Connection to 192.168.56.121 closed.
vagrant@Jenkins-master:~$
vagrant@Jenkins-master:~$
```

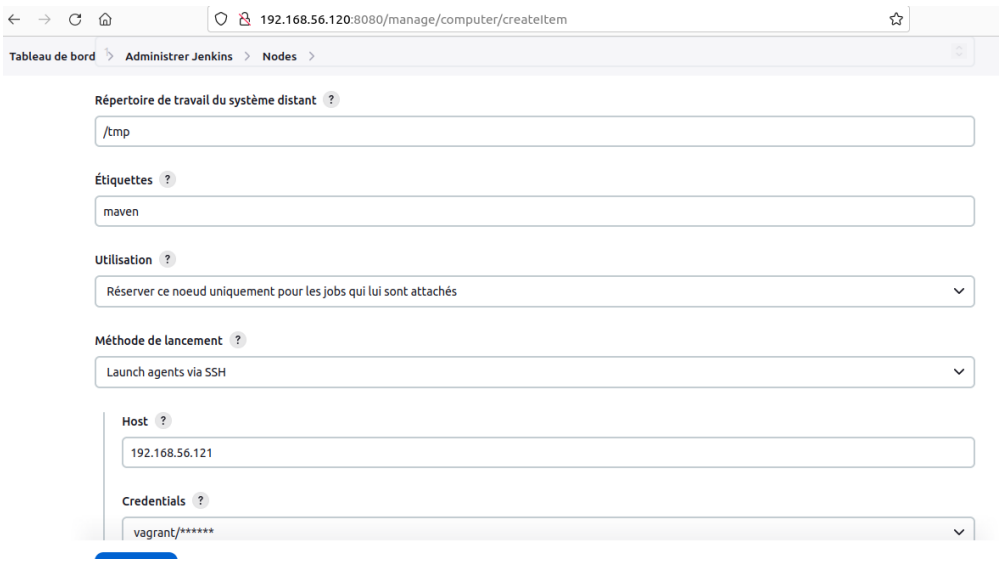
Enregistrement du nouveau nœud sur jenkins maître

Dans cette partie nous allons enregistrer le nouveau nœud sur jenkins maître.

2. Naviguer vers la vue "***Administrer Jenkins > Gérer les nœuds***" et cliquer sur le bouton ***Créer un nœud***.
   - Ajouter le nom du nœud et cocher *permanent Agent*, puis cliquer sur le bouton Create.



3. Sur la nouvelle interface, entrer les détails suivants :
   - **Name** : Nom du nœud à afficher dans Jenkins
   - **# d'exécuteurs** : combien de travaux doivent pouvoir s'exécuter simultanément sur cet agent, généralement défini sur le nombre de CPUs disponibles.
   - **Répertoire racine distant** : */tmp* ou */home/jenkins* par exemple
   - **Labels** : une liste des labels associés à ce nœud. Cela peut être utilisé pour limiter les tâches à ce nœud ou à ce type de nœud.
   - **Méthode de lancement** : Lancer l'agent via SSH
   - **Hôte** : le nom d'hôte ou l'adresse IP du système qui exécutera l'agent



- **Credentials** : C'est ici que nous spécifions le nom d'utilisateur/mot de passe (vagrant/vagrant).
- **Stratégie de vérification de la clé de l'hôte** : Pas de vérification.

◦ Cliquerz sur **Enregistrer**. Le nouveau nœud apparaître dans la liste des nœuds.

4. La liste des machines du cluster Jenkins est affiché sur la nouvelle interface :

- maître
- Agent-node (esclave)

◦ Cliquez sur *Agent-node*, puis *Voir les logs pour plus de détails* pour voir le journal et vérifier si jenkins master a pu contacter le nœud slave.

Déléguer l’exécution du pipeline au slave node

Dans cette partie on va configurer la section « agent » dans le pipeline (jenkinsfile) pour déléguer des jobs au nœud slave.

5. Dans un premier temps, on va déléguer tout le pipeline au nœud slave. Ci-dessous tout le pipeline (la section agent est en gras) :

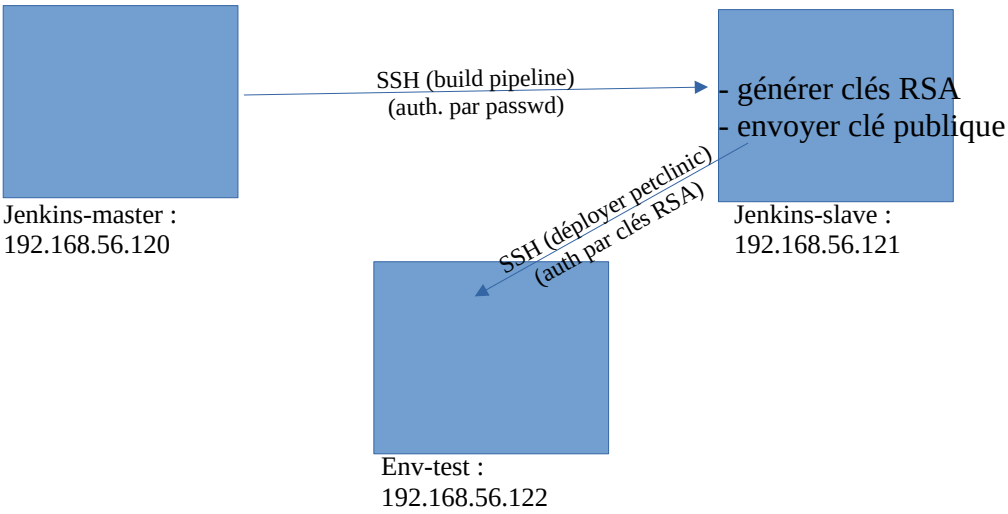
```
pipeline {
   agent {
                label 'Agent-node'
                }

   tools {
      maven "M3"
   }

   stages {

     …
```

6. Sauvegardez et lancez un nouveau build. Dans la sortie de la console, il doit signaler que tout le pipeline est envoyé au nœud slave.

Déploiement de l’application sur l’environnement de tests

Vous aurez besoin de copier votre package jar sur le serveur de tests (VM env-test) et par la suite lancer l’application à distance. L’une des méthodes les plus facile pour automatiser ces tâches est l’utilisation du protocole SSH.

Dans ce qui suit, nous allons ajouter un stage « deploy » au pipeline afin de déployer l’application « petclinic » sur l’environnement de production (la VM env-test).



1. Puisque qu'on va déployer l'application sur le serveur de test (VM env-test) à partir de Jenkins Slave (VM Jenkins-slave) en utilisant SSH, on doit configurer la connexion entre les 2 machines :
   - 1ère étape : Générer une paire de clés SSH sur jenkins-slave.

```
vagrant@Jenkins-master:~$
logout
brahim@Training:~/jenkins-dist$ vagrant ssh Jenkins-slave
Last login: Thu Aug 31 07:36:23 2023 from 192.168.56.120
vagrant@Jenkins-slave:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/vagrant/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vagrant/.ssh/id_rsa
Your public key has been saved in /home/vagrant/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:D5137BfxYK23xi840Pb6St1rPL9bybE1GKYRf+o8gOE vagrant@Jenkins-slave
The key's randomart image is:
+---[RSA 3072]----+
|            .    |
|             o  . |
|          . . +oo.|
|         ..o.+o=oo|
|        SEo+.o+++|
|         o..*+ooO|
|          .o.*+B+|
|           .o ==+|
|            o=o=B|
+----[SHA256]-----+
vagrant@Jenkins-slave:~$ ls .ssh/
authorized_keys  id_rsa  id_rsa.pub
vagrant@Jenkins-slave:~$
```

   - 2ème étape : Copier la clé publique sur le serveur de test.

```
vagrant@Jenkins-slave:~$ ssh-copy-id -i /home/vagrant/.ssh/id_rsa.pub vagrant@192.168.56.122
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/vagrant/.ssh/id_rsa.pub"
The authenticity of host '192.168.56.122 (192.168.56.122)' can't be established.
ED25519 key fingerprint is SHA256:0hxT9SloiG5jCZRdWc/j3aallRBhQfNqQBOjexK9/mI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
vagrant@192.168.56.122's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'vagrant@192.168.56.122'"
and check to make sure that only the key(s) you wanted were added.

vagrant@Jenkins-slave:~$
vagrant@Jenkins-slave:~$
```

- 3ème étape : Vérifiez la connexion en ssh avec authentification par clés.

```
vagrant@Jenkins-slave:~$ ssh vagrant@192.168.56.122
Last login: Sun Sep  3 13:44:17 2023 from 10.0.2.2
vagrant@env-test:~$
vagrant@env-test:~$ exit
logout
Connection to 192.168.56.122 closed.
vagrant@Jenkins-slave:~$
logout
brahim@Training:~/jenkins-dist$
```

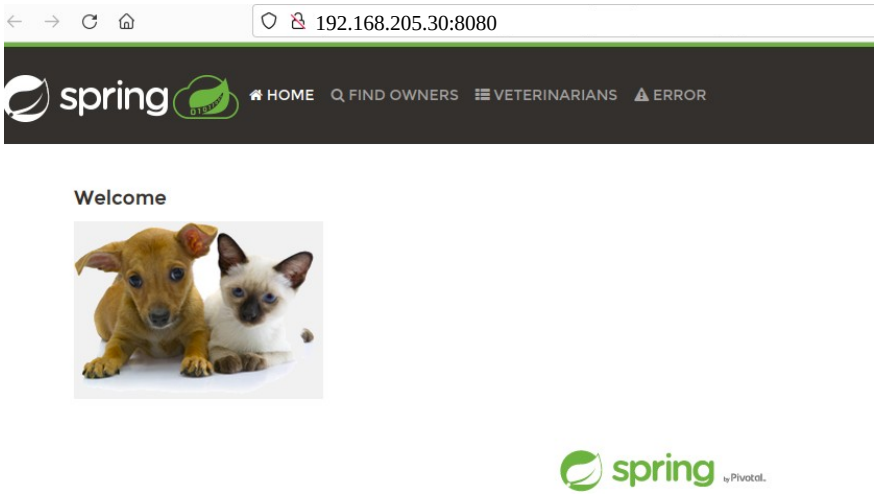
2. Copiez le package ***jar*** sur l'environnement de production (VM env-prod).
   - Lancez l'application à distance (en utilisant ssh).

…

```
    }
    stage('deploy-petclinic') {
        steps {
                script
            {
                if (isUnix())
                {
                    sh 'scp target/spring-petclinic-3.0.0-SNAPSHOT.jar vagrant@192.168.56.122:/tmp/'
                    sh 'ssh vagrant@192.168.56.122 java -jar /tmp/spring-petclinic-3.0.0-SNAPSHOT.jar &'
                    sh 'sleep 20'
                }
                else
                {
                    bat 'scp target/spring-petclinic-3.0.0-SNAPSHOT.jar vagrant@192.168.56.122:/tmp/'
                    bat 'ssh vagrant@192.168.56.122 java -jar /tmp/spring-petclinic-3.0.0-SNAPSHOT.jar &'
                    sh 'sleep 20'
                }
            }
```

…

3. Pour vérifier, tapez l'url http://192.168.56.122:8080 sur votre navigateur. Il doit afficher l'interface de votre application.

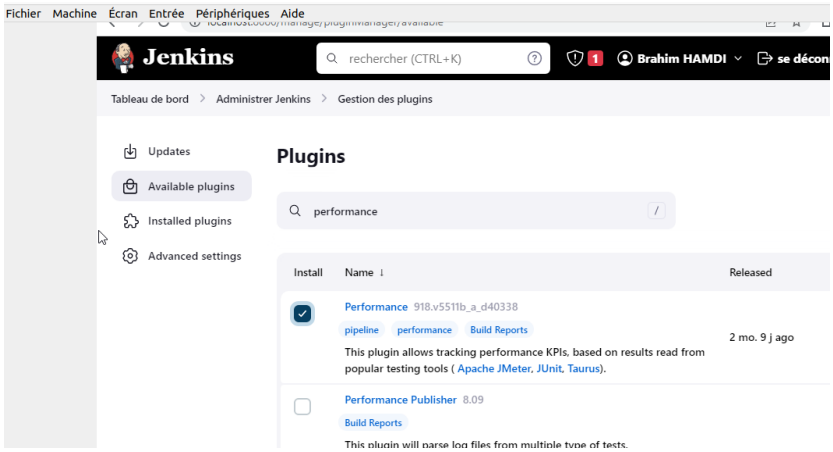


Test de performance avec JMeter

Apache JMeter peut être utilisé pour tester les performances de sites statiques, de sites dynamiques et d'applications Web complètes. Il peut également être utilisé pour simuler une charge importante sur un serveur, un groupe de serveurs, un réseau ou un objet, permettant des tests de résistance ou une analyse des performances globales sous différents types de charge.

On va faire toute le travail dans le projet de type pipeline, sachant qu'il est possible de faire le même travail en mode freestyle.

1. Pour intégrer ***JMeter*** à Jenkins, nous utiliserons le plugin ***Performance***. Nous devons l'installer d'abord.



2. Vous devez installer l'outil Jmeter aussi, lancez les commandes suivantes sous Jenkins_slave (et pas Jenkins_master) :

```
brahim@Training:~/jenkins-dist$ vagrant ssh Jenkins-master
Last login: Wed Aug 30 17:14:51 2023 from 10.0.2.2
vagrant@Jenkins-master:~$ cd /opt/
vagrant@Jenkins-master:/opt$ sudo wget https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.6.2.tgz
--2023-08-30 17:16:03--  https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.6.2.tgz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 91109473 (87M) [application/x-gzip]
Saving to: 'apache-jmeter-5.6.2.tgz'

apache-jmeter-5.6.2.tgz            100%[===================================================================>]  86.89M  1.34MB/s    in 67s

2023-08-30 17:17:10 (1.30 MB/s) - 'apache-jmeter-5.6.2.tgz' saved [91109473/91109473]

vagrant@Jenkins-master:/opt$
vagrant@Jenkins-master:/opt$ sudo tar xzvf apache-jmeter-5.6.2.tgz
apache-jmeter-5.6.2/
apache-jmeter-5.6.2/LICENSE
apache-jmeter-5.6.2/licenses/
apache-jmeter-5.6.2/licenses/flot-axislabels/
apache-jmeter-5.6.2/licenses/flot-axislabels/flot-axislabels-0.8.3/
apache-jmeter-5.6.2/licenses/flot-axislabels/flot-axislabels-0.8.3/LICENSE
```

...

```
vagrant@Jenkins-master:/opt$ sudo chown -R vagrant:vagrant apache-jmeter-5.6.2/
vagrant@Jenkins-master:/opt$
```

   - Ajoutez cette commande à la dernière ligne du fichier */opt/apache-jmeter-5.6.2/bin/user.properties* : *jmeter.save.saveservice.output_format=xml.*

     Cette commande intègre la sortie de JMeter dans Jenkins.
     Enregistrez et fermez le fichier pour vous assurer que les modifications sont apportées.

3. Dans le fichier ***petclinic_test_plan.jmx*** (voir dans le dépôt petclinic), nous avons préparé avec Jmeter quelques plans de test : Home page, CSS, JS, Vets, Find owner, Find owner with lastname, Edit Owner, POST Edit Owner, New Pet, POST new pet et New visit.

   Pour voir comment préparer un plan de test vous pouvez consulter la documentation officielle de Jenkins : https://www.jenkins.io/doc/book/using/using-jmeter-with-jenkins/

   Dans la partie suivante, nous allons configurer le pipeline de Jenkins pour exécuter ces plans de test.

4. Nous avons maintenant tout le nécessaire pour exécuter JMeter depuis Jenkins.
   Pour exécuter Jmeter depuis Jenkins, ajoutez une nouvelle étape au pipeline qui exécute la commande shell suivante :

/opt/apache-jmeter-5.6.2/bin/jmeter -j jmeter.save.saveservice.output_format=xml -n -t petclinic_test_plan.jmx -l petclinic_test_plan.jtl

```
stage('Test performance - Jmeter') {
    steps {
        script
        {
            sh '/opt/apache-jmeter-5.6.2/bin/jmeter -j  jmeter.save.saveservice.output_format=xml -n -t petclinic_te
        }
    }
```

5. La dernière étape consiste à publier le rapport du test performance sous forme de graphes, tableaux, …
   - Ajoutez une nouvelle étape pour publier le rapport *petclinic_test_plan.jtl*.

```
stage('Pub perf Test') {
    steps {
        script
        {
            sh 'echo publier rapport de test performance ##############################################'
        }
    }
    post {
        success {
            perfReport errorFailedThreshold: 5, errorUnstableThreshold: 1, sourceDataFiles: 'petclinic_test_plan.jtl
        }
    }
```

Vous pouvez voir le rapport de test via la rubrique ***Performance Trend***.