Lab2. Mise en place d'une pipeline CI/CD avec Jenkins

Partie 1 : Préparation de l'environnement de travail

Etape 1: Ouvrir le terminal et récupérer le répertoire sample-app à partir du github avec la commande :

git clone http://github.com/AbirKaldi/sample-app

Etape 2: Déplacer le répertoire sample-app dans votre propre compte github.

Partie 2 : Télécharger et exécuter l'image Jenkins Docker

Dans cette partie, vous allez télécharger l'image Jenkins Docker.Vous allez ensuite démarrer une instance de l'image et vérifier que le serveur Jenkins est en cours d'exécution.

Étape 1: Téléchargez l'image Jenkins Docker.

L'image Jenkins Docker est stockée ici : https://hub.docker.com/r/jenkins/jenkins.Au moment de la rédaction de ce laboratoire, ce site spécifie que vous utilisez la commande **docker pull jenkins/jenkins** pour télécharger le dernier conteneur Jenkins.Vous devriez obtenir un résultat similaire à ce qui suit :

```
devasc@labvm:~# docker pull jenkins/jenkins:lts
lts: Pulling from jenkins/jenkins
3192219afd04: Pulling fs layer
17c160265e75: Pulling fs layer
cc4fe40d0e61: Pulling fs layer
9d647f502a07: Pulling fs layer
d108b8c498aa: Pulling fs layer
1bfe918b8aa5: Pull complete
dafa1a7c0751: Pull complete
650a236d0150: Pull complete
cba44e30780e: Pull complete
52e2f7d12a4d: Pull complete
d642af5920ea: Pull complete
e65796f9919e: Pull complete
9138dabbc5cc: Pull complete
f6289c08656c: Pull complete
73d6b450f95c: Pull complete
a8f96fbec6a5: Pull complete
9b49ca1b4e3f: Pull complete
d9c8f6503715: Pull complete
20fe25b7b8af: Pull complete
Digest: sha256:717dcbe5920753187a20ba43058ffd3d87647fa903d98cde64dda4f4c82c5c48
Status: Downloaded newer image for jenkins/jenkins:lts
docker.io/jenkins/jenkins:lts
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$
```

Étape 2: Démarrez le conteneur Jenkins Docker.

Entrez la commande suivante sur **une ligne**.Vous devrez peut-être le copier dans un éditeur de texte si vous affichez une version PDF de ce laboratoire pour éviter les sauts de ligne.Cette commande démarrera le

conteneur Jenkins Docker, puis autorisera l'exécution des commandes Docker à l'intérieur de votre serveur Jenkins.

```
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ docker run --rm -u root -p 8080:8080 -v jenkins-data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -v "$HOME":/home --name jenkins_server jenkins/jenkins:lts
```

Les options utilisées dans cette commande **docker run** sont les suivantes :

- **—rm** - Cette option supprime automatiquement le conteneur Docker lorsque vous arrêtez de l'exécuter.
- **-u** - Cette option spécifie l'utilisateur. Vous voulez que ce conteneur Docker s'exécute en tant que root afin que toutes les commandes Docker entrées dans le serveur Jenkins soient autorisées.
- **-p** - Cette option spécifie le port sur lequel le serveur Jenkins s'exécutera localement.
- **-v** - Ces options lient les volumes de montage nécessaires pour Jenkins et Docker. Le premier **-v** spécifie où les données Jenkins seront stockées. Le second **-v** spécifie où obtenir Docker afin que vous puissiez exécuter Docker dans le conteneur Docker qui exécute le serveur Jenkins. Le troisième **-v** spécifie la variable PATH pour le répertoire de base.

Étape 3: Vérifiez que le serveur Jenkins est en cours d'exécution.

Le serveur Jenkins devrait maintenant être en cours d'exécution. Copiez le mot de passe admin qui s'affiche dans la sortie, comme indiqué dans la section suivante.

N'entrez aucune commande dans cette fenêtre de serveur. Si vous arrêtez accidentellement le serveur Jenkins, vous devrez entrer à nouveau la commande **docker run** à partir de l'étape 2 ci-dessus. Après l'installation initiale, le mot de passe admin est affiché comme indiqué ci-dessous.

```
<output omitted>
*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required.An admin user has been created and a password generated.
Please use the following password to proceed to installation:

77dc402e31324c1b917f230af7bfebf2<--Your password will be different

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
<output omitted>
2020-05-12 16:34:29.608+0000 [id=19] INFO hudson.WebAppMain$3#run: Jenkins is fully up and running
```

Remarque : Si vous perdez le mot de passe, ou qu'il ne s'affiche pas comme indiqué ci-dessus, ou que vous devez redémarrer le serveur Jenkins, vous pouvez toujours récupérer le mot de passe en accédant à la ligne de commande du conteneur Jenkins Docker. Créez une deuxième fenêtre de terminal dans VS Code et entrez les commandes suivantes afin de ne pas arrêter le serveur Jenkins.:

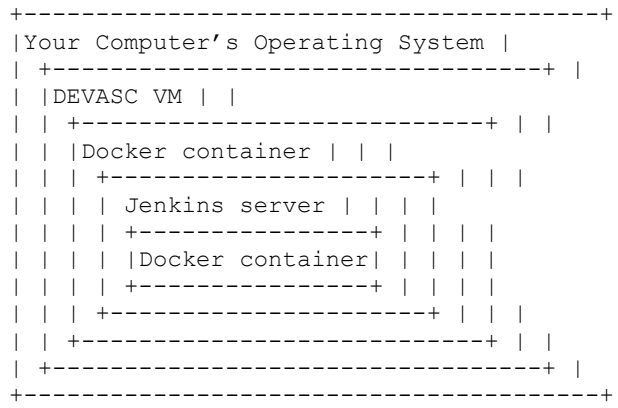
```
devasc@labvm:~ # docker exec -it jenkins_server /bin/bash
root@19d2a847a54e:/# cat /var/jenkins_home/secrets/initialAdminPassword
77dc402e31324c1b917f230af7bfebf2
```

```
root@19d2a847a54e:/# exit
exit
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$
```

Remarque : Votre identifiant de conteneur (19d2a847a54e surligné ci-dessus) et votre mot de passe seront différents.

Étape 4: Étudiez les niveaux d'abstraction actuellement en cours d'exécution sur votre ordinateur.

Le diagramme ASCII suivant montre les niveaux d'abstraction dans cette implémentation Docker Inside-Docker (dind).Ce niveau de complexité n'est pas inhabituel dans les réseaux et les infrastructures cloud d'aujourd'hui.



Partie 3 : Configurer Jenkins

Dans cette partie, vous allez terminer la configuration initiale du serveur Jenkins.

Étape 1: Ouvrez un onglet du navigateur web.

Accédez à **http://localhost:8080/** et connectez-vous à l'aide de votre mot de passe copié.

Étape 2: Installez les plugins Jenkins recommandés.

Cliquez sur **Install suggested plugins** et attendez que Jenkins télécharge et installe les plugins. Dans la fenêtre du terminal, vous verrez les messages de journal au fur et à mesure que l'installation se poursuit. Assurez-vous que vous ne fermez pas cette fenêtre de terminal. Vous pouvez ouvrir une autre fenêtre de terminal pour accéder à la ligne de commande.

Étape 3: Ignorer la création d'un nouvel utilisateur administrateur.

Une fois l'installation terminée, la fenêtre **Create First Admin User** s'affiche.Pour l'instant, cliquez sur **Skip and continue as admin** en bas.

Étape 4: Ignorer la création d'une configuration d'instance.

Dans la fenêtre **Instance Configuration**, ne modifiez rien.Cliquez sur **Save and Finish** en bas.

Étape 5: Commencez à utiliser Jenkins.

Dans la fenêtre suivante, cliquez sur **Start using Jenkins**. Vous devriez maintenant être sur le tableau de bord principal avec un **Welcome to Jenkins!** message.

Partie 4 : Utiliser Jenkins pour exécuter une version de votre application

L'unité fondamentale de Jenkins est le job (également connu sous le nom de projet).Vous pouvez créer des jobs qui effectuent diverses tâches, notamment les suivantes :

- Récupérez du code à partir d'un référentiel de gestion de code source tel que GitHub.
- Créez une application à l'aide d'un script ou d'un outil de construction.
- Emballer une application et l'exécuter sur un serveur

Étape 1: Créez une nouvelle tâche.

a. Cliquez sur le lien **Create a job** directement en dessous de la page **Welcome to Jenkins!** message. Vous pouvez également cliquer sur **New Item** dans le menu de gauche.

b. Dans le champ **Entrez un nom d'élément**, renseignez le nom **BuildAppJob**.

c. Cliquez sur **Freestyle project** comme type de tâche.Dans la description, l'abréviation SCM signifie gestion de la configuration logicielle, qui est une classification des logiciels qui est responsable du suivi et du contrôle des modifications apportées aux logiciels.

d. Scroll to the bottom and click **OK**.

Étape 2: Configurez le Jenkins BuildAppJob.

Vous êtes maintenant dans la fenêtre de configuration où vous pouvez entrer des détails sur votre travail.Les onglets situés en haut ne sont que des raccourcis vers les sections ci-dessous.Cliquez sur les onglets pour explorer les options que vous pouvez configurer.Pour ce travail simple, il suffit d'ajouter quelques détails de configuration.

a. Cliquez sur l' onglet **General**, ajoutez une description pour votre travail.Par exemple, "**Mon premier emploi Jenkins.**"

b. Cliquez sur l'onglet **Source Code Management** et cliquez sur le bouton radio **Git**.Dans le champ URL du référentiel, ajoutez votre lien de référentiel GitHub pour l'exemple d'application en prenant soin de saisir votre nom d'utilisateur sensible à la casse.Assurez-vous d'ajouter l'extension .git à la fin de votre URL. Par exemple :

```
https://github.com/github-nom_utilisateur/sample-app.git
```

c. Pour **Credentials** cliquez sur le bouton **Add** et choisissez **Jenkins**.

d. Dans la boîte de dialogue **Add Credentials**, renseignez votre nom d'utilisateur et votre mot de passe GitHub, puis cliquez sur **Add**.

Remarque : vous recevrez un message d'erreur indiquant que la connexion a échoué. Cela est dû au fait que vous n'avez pas encore sélectionné les informations d'identification.

e. Dans la liste déroulante des **Credentials** où il est actuellement indiqué **None**, choisissez les informations d'identification que vous venez de configurer.

f. Après avoir ajouté l'URL et les informations d'identification correctes, Jenkins teste l'accès au référentiel.Vous ne devriez pas avoir de message d'erreur.Si vous le faites, vérifiez votre URL et vos informations d'identification.Vous devrez les **Add** à nouveau car il n'y a aucun moyen à ce stade de supprimer ceux que vous avez entrés précédemment.

g. En haut de la fenêtre de configuration de **BuildAppJob**, cliquez sur l'onglet **Build**.

h. Pour la liste déroulante **Add build step**, choisissez **Execute shell**.

i. Dans le champ **Commande**, entrez la commande que vous utilisez pour exécuter le script build pour sample-app.sh.

```
bash./sample-app.sh
```

j. Cliquez sur le bouton **Save** (Enregistrer).Vous êtes retourné au tableau de bord Jenkins avec le **BuildAppJob** sélectionné.

Étape 3: Dis à Jenkins de construire l'application.

Sur le côté gauche, cliquez sur **Build Now** pour démarrer le travail.Jenkins will download your Git repository and execute the build command **bash ./sample-app.sh**.Votre construction devrait réussir parce que vous n'avez rien changé dans le code depuis la partie 3 lorsque vous avez modifié le code.

Étape 4: Accédez aux détails de construction.

Sur la gauche, dans la section **Build History**, cliquez sur votre numéro de build qui devrait être le **#1** sauf si vous avez créé l'application plusieurs fois.

Étape 5: Affichez la sortie de la console.

Sur la gauche, cliquez sur **Console Output**.Vous devriez voir un résultat similaire à ce qui suit.Notez les messages de succès en bas ainsi que la sortie de la commande **docker ps -a**.Deux conteneurs docker sont en cours d'exécution : un pour votre application d'échantillonnage fonctionnant sur le port local 5050 et un pour Jenkins sur le port local 8080.

```
Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/BuildAppJob
using credential 0cf684ea-48a1-4e8b-ba24-b2fa1c5aa3df
Cloning the remote Git repository
Cloning repository https://github.com/github-user/sample-app
> git init /var/jenkins_home/workspace/BuildAppJob # timeout=10
Fetching upstream changes from https://github.com/github-user/sample-app
> git —version # timeout=10
using GIT_ASKPASS to set credentials
> git fetch —tags —progress — https://github.com/github-user/sample-app
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/github-user/sample-app # timeout=10
> git config —add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/github-user/sample-app # timeout=10
Fetching upstream changes from https://github.com/github-user/sample-app
using GIT_ASKPASS to set credentials
> git fetch —tags —progress — https://github.com/github-user/sample-app
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^ {commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^ {commit} # timeout=10
Checking out Revision 230ca953ce83b5d6bdb8f99f11829e3a963028bf
(refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 230ca953ce83b5d6bdb8f99f11829e3a963028bf # timeout=10
Commit message: "Changed port numbers from 8080 to 5050"
> git rev-list —no-walk 230ca953ce83b5d6bdb8f99f11829e3a963028bf # timeout=10
[BuildAppJob] $ /bin/sh -xe /tmp/jenkins1084219378602319752.sh
+ bash./sample-app.sh
Sending build context to Docker daemon 6.144kB

Step 1/7 : FROM python
---> 4f7cd4269fa9
```

```
Step 2/7 : RUN pip install flask
---> Using cache
---> 57a74c0dff93
Step 3/7 : COPY ./static /home/myapp/static/
---> Using cache
---> aee4eb712490
Step 4/7 : COPY ./templates /home/myapp/templates/
---> Using cache
---> 594cdc822490
Step 5/7 : COPY sample_app.py /home/myapp/
---> Using cache
---> a001df90cf0c
Step 6/7 : EXPOSE 5050
---> Using cache
---> eae896e0a98c
Step 7/7 : CMD python3 /home/myapp/sample_app.py
---> Using cache
---> 272c61fddb45
Successfully built 272c61fddb45
Successfully tagged sampleapp:latest
9c8594e62079c069baf9a88a75c13c8c55a3aeaddde6fd6ef54010953c2d3fbb
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9c8594e62079 sampleapp "/bin/sh -c 'python …" Less than a second ago Up Less than a second 0.0.0.0:5050->5050/tcp samplerunning
e25f233f9363 jenkins/jenkins:lts "/sbin/tini -- /usr/…" 29 minutes ago Up 29 minutes 0.0.0.0:8080->8080/tcp, 50000/tcp jenkins_server
Finished: SUCCESS
```

Étape 6: Ouvrez un autre onglet du navigateur Web et vérifiez que l'exemple d'application est en cours d'exécution.

Saisissez l'adresse locale, **localhost: 5050**.Vous devriez voir le contenu de votre index.html affiché en couleur de fond bleu acier clair avec **Vous m'appelez de 172.17.0.1** affiché en tant que H1.

Partie 5 : Utiliser Jenkins pour tester une construction

Dans cette partie, vous allez créer une deuxième tâche qui teste la construction pour vous assurer qu'elle fonctionne correctement.

Remarque: Vous devez arrêter et supprimer le conteneur docker **samplerunning**.

```
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ docker stop samplerunning
samplerunning
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ docker stop samplerunning
samplerunning
```

Étape 1: Commencez une nouvelle tâche pour tester votre sample-app.

a. Retournez à l'onglet du navigateur Web Jenkins et cliquez sur le lien **Jenkins** dans le coin supérieur gauche pour revenir au tableau de bord principal.

b. Cliquez sur le lien **New Item** pour créer un nouvel emploi.

c. Dans le champ Entrez un nom d'élément, renseignez le nom **TestAppJob**.

d. Cliquez sur **Freestyle project** comme type de tâche.

e. Faites défiler jusqu'en bas et cliquez sur **OK**.

Étape 2: Configurez le Jenkins TestAppJob.

a. Ajoutez une description pour votre travail.Par exemple, "Mon premier test Jenkins."

b. Laissez Source Code Management définie sur **None**.

c. Cliquez sur l'onglet **Build Triggers** et cochez la case **Build after other projects are built**.Pour les **Projects to watch**, renseignez le nom **BuildAppJob**.

Étape 3: Écrivez le script de test qui doit s'exécuter après une version stable de BuildAppJob.

a. Cliquez sur l'onglet **Build**.

b. Cliquez sur **Add build step** et choisissez **Execute shell**.

c. Entrez le script suivant.La commande **if** doit être sur une seule ligne, y compris le **; then**.Cette commande **grep** la sortie renvoyée par la commande cURL pour voir si **vous m'appelez à partir de 172.17.0.1** est retourné.Si true, le script se termine avec un code de 0, ce qui signifie qu'il n'y a pas d'erreurs dans la **BuildAppJob**.If false, the script exits with a code of 1 which means the **BuildAppJob** failed.

```
if curl http://172.17.0.1:5050/ | grep "You are calling me from 172.17.0.1"; then
exit 0
else
exit 1
fi
```

d. Cliquez sur **Save**, puis sur le lien **Back to Dashboard** sur le côté gauche.

Étape 4: Dis à Jenkins d'exécuter le travail BuildAppJob à nouveau.

a. Actualisez la page Web avec le bouton d'actualisation de votre navigateur.

b. Vous devriez maintenant voir vos deux tâches répertoriées dans un tableau.Pour le travail **BuildAppJob**, cliquez sur le bouton build à l'extrême droite (une horloge avec une flèche).

Étape 5: Vérifiez que les deux tâches sont terminées.

Si tout se passe bien, vous devriez voir l'horodatage pour la mise à jour de la colonne **Last Success** pour **BuildAppJob** et **TestAppJob**.Cela signifie que votre code pour les deux tâches s'est exécuté sans erreur.Mais vous pouvez également le vérifier par vous-même.

Remarque : Si les horodatages ne sont pas mis à jour, assurez-vous que l'activation de l'actualisation automatique est activée en cliquant sur le lien en haut à droite.

a. Cliquez sur le lien pour **TestAppJob**.Sous **Permaliens**, cliquez sur le lien correspondant à votre dernière version, puis cliquez sur **Console Output**.Vous devriez voir un résultat similaire à ce qui suit :

```
Started by upstream project "BuildAppJob" build number 13
causé à l'origine par:
Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/TestAppJob
[TestAppJob] $ /bin/sh -xe /tmp/jenkins1658055689664198619.sh
+ grep You are calling me from 172.17.0.1
+ curl http://172.17.0.1:5050/
% Total % Received % Xferd Average Speed Time Time Time Current
```

```
Dload Upload Total Spent Left Speed

0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0
100 177 100 177 0 0 29772 0 --:--:-- --:--:-- --:--:-- 35400
<h1>You are calling me from 172.17.0.1</h1>
+ exit 0
Finished: SUCCESS
```

b. Il n'est pas nécessaire de vérifier que votre exemple d'application est en cours d'exécution car **TestAppJob** l'a déjà fait pour vous.Cependant, vous pouvez ouvrir un onglet de navigateur pour **172.17.0. 1:5050** pour voir qu'il est en cours d'exécution.

Partie 6 : Créer un pipeline dans Jenkins

Bien que vous puissiez actuellement exécuter vos deux tâches en cliquant simplement sur le bouton Créer maintenant pour **BuildAppJob**, les projets de développement logiciel sont généralement beaucoup plus complexes.Ces projets peuvent grandement bénéficier de l'automatisation des builds pour l'intégration continue des modifications de code et de la création continue de builds de développement prêts à être déployés.C'est l'essence de la CI/CD.Un pipeline peut être automatisé pour s'exécuter en fonction d'une variété de déclencheurs, y compris périodiquement, en fonction d'un sondage GitHub pour les modifications, ou à partir d'un script exécuté à distance.Cependant, dans cette partie, vous allez écrire un pipeline dans Jenkins pour exécuter vos deux applications chaque fois que vous cliquez sur le bouton Pipeline **Build Now**.

Étape 1: Créez un travail Pipeline.

a. Cliquez sur le lien **Jenkins** en haut à gauche, puis sur **Nouvel élément**.

b. Dans le champ **Enter an item name** , tapez **SamplePipeline**.

c. Sélectionnez **Pipeline** comme type de tâche.

d. Faites défiler jusqu'en bas et cliquez sur **OK**.

Étape 2: Configurez le travail SamplePipeline.

a. En haut, cliquez sur les onglets et examinez chaque section de la page de configuration.Notez qu'il existe un certain nombre de façons différentes de déclencher une construction.Pour le travail **SamplePipeline**, vous le déclencherez manuellement.

b. Dans la section **Pipeline**, ajoutez le script suivant.

```
node {
stage('Preparation') {
catchError(buildResult: 'SUCCESS') {
sh 'docker stop samplerunning'
sh 'docker rm samplerunning'
}
}
stage ('Build') {
build 'BuildAppJob'
}
stage('Results') {
build 'TestAppJob'
}
}
```

Ce script effectue les opérations suivantes :

- Il crée une construction de nœud unique par opposition à un nœud distribué ou multi-nœud.Les configurations distribuées ou multi-nœuds sont destinées à des pipelines plus volumineux que celui que vous construisez dans ce laboratoire et dépassent le cadre de ce cours.
- Dans la phase de **préparation**, **SamplePipeline** vérifiera d'abord que toutes les instances précédentes du conteneur docker **BuildAppJob** sont arrêtées et supprimées.Mais s'il n'y a pas encore de conteneur en cours d'exécution, vous obtiendrez une erreur.Par conséquent, vous utilisez la fonction **catchError** pour attraper les erreurs et renvoyer une valeur "SUCCESS".Cela permettra de faire en sorte que le pipeline passe à l'étape suivante.
- Dans l'étape **Build**, **SamplePipeline** construira votre **BuildAppJob**.
- Dans l'étape **Résultats**, **SamplePipeline** va construire votre **TestAppJob**.

c. Cliquez sur **Save** et vous serez retourné au tableau de bord Jenkins pour la tâche **SamplePipeline**.

Étape 3: Exécutez SamplePipeline.

Sur la gauche, cliquez sur **Build Now** pour exécuter le travail **SamplePipeline**.Si vous avez codé votre script Pipeline sans erreur, la **Stage View** doit afficher trois zones vertes avec le nombre de secondes que chaque étape a pris pour construire.Si ce n'est pas le cas, cliquez sur Configurer sur la gauche pour revenir à la configuration **SamplePipeline** et vérifier votre script Pipeline.

Étape 4: Vérifiez la sortie SamplePipeline.

Cliquez sur le lien de construction le plus récent sous **Permaliens**, puis cliquez sur **Console Output**.Vous devriez voir un résultat similaire à ce qui suit :

```
Started by user admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/SamplePipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Preparation)
[Pipeline] catchError
[Pipeline] {
[Pipeline] sh
+ docker stop samplerunning
samplerunning
[Pipeline] sh
+ docker rm samplerunning
samplerunning
[Pipeline] }
[Pipeline] // catchError
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] {(Build)
[Pipeline] build (BuildAppJob de construction)
Scheduling project: BuildAppJob
Starting building: BuildAppJob #15
[Pipeline] }
[Pipeline] // stage
```

```
[Pipeline] stage
[Pipeline] { (Results)
[Pipeline] build (Building TestAppJob)
Scheduling project: TestAppJob
Starting building: TestAppJob #18
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```