jegan@tektutor.org
http://www.tektutor.org

Lab Setup for MicroServices Training

Install CentOS 7.6 64-bit OS - Server with GUI

All participants and trainer shall be given admin rights on the trainer and participants lab machines.

SSH Server shall be installed and trainer/participant shall be able to access any training VM from another training VM from their respective terminals

For instance, assuming 10.19.41.110 is the training machine ip then, from another training machine ip, we should be able to connect via ssh as shown below with user - train and password - amdocs@123

```
ssh train@10.19.41.110
```

Configure wget proxy

```
sudo vim /etc/wgetrc

https_proxy = http://10.19.16.165:8080/
http_proxy = http://10.19.16.165:8080/
ftp_proxy = http://10.19.16.165:8080/
use_proxy = on
```

Install JDK 1.8 (Install from terminal)

sudo yum install java-1.8.0-openjdk-devel

JDK 1.8 setup can be tested by checking in the terminal as train user

```
javac -version
javac 1.8.0_212

which javac
/usr/bin/javac
```

```
ls -l /usr/bin/javac
lrwxrwxrwx. 1 root root 23 May 31 07:57 /usr/bin/javac ->
/etc/alternatives/javac

ls -l /etc/alternatives/javac
lrwxrwxrwx. 1 root root 70 May 31 07:57 /etc/alternatives/javac ->
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64/bin/javac
```

The text highlighted in bold in the above line is the JAVA_HOME path
The JAVA_HOME environment variable shall be exported in the /home/train/.bashrc file as shown below

export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64

The minor version of JDK 1.8 may vary, which is perfectly fine.

Setup Maven 3.6.2 and configure Maven
http://apachemirror.wuchna.com/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz

Maven shall be extracted and kept at /home/train/Downloads/maven-3.6.1 and M2_HOME environment variable shall be exported in /home/train/.bashrc file as shown below.

```
export M2_HOME=/home/train/Downloads/maven-3.6.1
export PATH=$JAVA_HOME/bin:$M2_HOME/bin:$PATH
export HTTP_PROXY=http://10.19.16.165:8080
export HTTPS_PROXY=http://10.19.16.165:8080
```

In terminal, to apply the new changes done in bashrc file, it is important to execute the below command on the terminal

```
source /home/train/.bashrc
```

Maven setup can be tested by checking
```
mvn --version
```

We should get this kind of output

```
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555;
2019-04-04T15:00:29-04:00)
Maven home: /home/train/Downloads/apache-maven-3.6.1
Java version: 1.8.0_212, vendor: Oracle Corporation, runtime:
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
```

```
OS name: "linux", version: "3.10.0-862.el7.x86_64", arch: "amd64", family:
"unix"

Java version: 1.8.0_212, vendor: Oracle Corporation, runtime:
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7_6.x86_64/jre
```

The text highlighted in JDK 1.8 may vary depending on the minor version of JDK 1.8 installed, which is perfectly fine.

Proxy shall be configured for Maven setup to work normally
The settings.xml file under /home/train/Downloads/apache-maven-3.6.1/conf/settings.xml

```
<proxies>
    <proxy>
      <id>proxy-conf</id>
      <active>true</active>
      <protocol>http</protocol>
      <host>10.19.16.165</host>
      <port>8080</port>
      <nonProxyHosts>localhost | 10.19.41.110</nonProxyHosts>
   </proxy>
</proxies>
```

The <proxies> and its corresponding closing tag </proxies> exists already, so it is important that the <proxy></proxy> and the configurations shown above only are inserted below the existing <proxies> tag section.

The IP address `10.19.16.165` is the amdocs training network proxy and port 8080 is the amdocs training proxy server's port which is common configuration that shall be done in all training lab machines. However, `10.19.41.110` is the ip address of the training machine, which will vary for every single training machine.

To verify if the proxy configuration works fine, on the terminal try this below command

```
mvn archetype:generate -DgroupId=org.tektutor -DartifactId=tektutor-helloworld-app
-Dversion=1.0.0  -DarchetypeArtifactId=maven-archetype-quickstart
-DinteractiveMode=false
```

We should get the below output

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
```

```
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.1:generate (default-cli) @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.1:generate (default-cli) @ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:3.1.1:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype:
maven-archetype-quickstart:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: /home/train/
[INFO] Parameter: package, Value: org.tektutor
[INFO] Parameter: groupId, Value: org.tektutor
[INFO] Parameter: artifactId, Value: tektutor-helloworld-app
[INFO] Parameter: packageName, Value: org.tektutor
[INFO] Parameter: version, Value: 1.0.0
[INFO] project created from Old (1.x) Archetype in dir: /home/train/tektutor-helloworld-app
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 21.865s
[INFO] Finished at: Mon Sep 09 22:20:49 EDT 2019
[INFO] Final Memory: 14M/180M
[INFO] ------------------------------------------------------------------------
```

Now you could try to build the maven project created as shown below

```
cd tektutor-helloworld-app/
mvn clean install
```

The build should succeed without any download problem, if you see a Build Success it proves the maven proxy settings are configured correctly.

SpringToolSuite 4.3.2 IDE
https://download.springsource.com/release/STS4/4.3.2.RELEASE/dist/e4.12/spring-tool-suite-4-4.3.2.RELEASE-e4.12.0-linux.gtk.x86_64.tar.gz

Extract the tar gunzip file from the terminal as shown below

```
cd /home/train/Downloads
```

```
tar xvfz
spring-tool-suite-4-4.3.2.RELEASE-e4.12.0-linux.gtk.x86_64.tar.gz
```

Once the tar gunzip file is extracted, you should be able to see a folder sts-4.3.2.RELEASE under /home/train/Downloads path.

We need to configure Spring Tool Suite IDE Proxy to make sure it works fine. Hence do the below from terminal window

```
cd /home/train/Downloads
cd sts-4.3.2.RELEASE
./SpringToolSuite4
```

Once the SprintToolSuite4 IDE is launched, you need to configure the proxy settings by navigating to

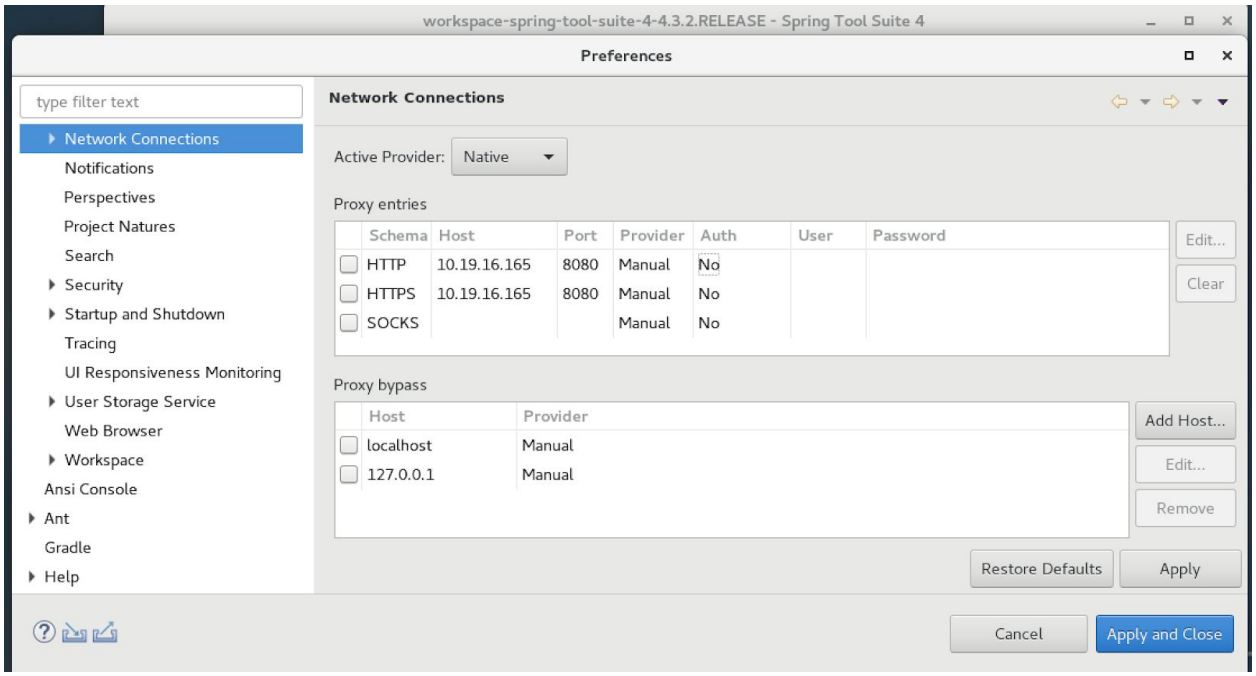
Window → Preferences Menu

In the Preferences Window left side, select Maven → User Settings

Make sure Global Settings is configured as below
/home/train/Downloads/apache-maven-3.6.1/conf/settings.xml

Make sure User Settings is configured as below
/home/train/Downloads/apache-maven-3.6.1/conf/settings.xml

Next, in the preferences window, navigate to General → Network Connections
Select the Http and click the Edit button to configure Http and Http as shown below and make sure the configurations are saved by clicking “Apply and Close” button.



In the Proxy bypass section shown above, make sure the IP Address of the lab machine IP is added. With this, the Proxy settings for SpringToolSuite4 IDE is complete.

Installing Docker and configuring proxy settings in CentOS

You can install docker from CentOS terminal with the below commands

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2

sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo

sudo yum install docker-ce
```

You can configure proxy in the /etc/sysconfig/docker file as shown below

```
HTTP_PROXY=http://10.19.16.165:8080/
HTTPS_PROXY=https://10.19.16.165:8080/
NO_PROXY=localhost,host.tuxfixer.com,10.19.41.10
```

In the above configuration, you need to replace 10.19.41.10 IP with the training VM IP.

Once the docker is installed, you can enable and start the docker service as shown below

```
sudo systemctl enable docker
sudo systemctl start docker
```

You can add train user to docker group to gain admin rights as shown below

```
sudo usermod -aG docker train
sudo su train
```

You can check if docker is installed properly as shown below

```
docker --version
docker pull ubuntu:16.04
docker images
```

Assuming the proxy is configured as instructed, the above command should download ubuntu docker image without any failure.

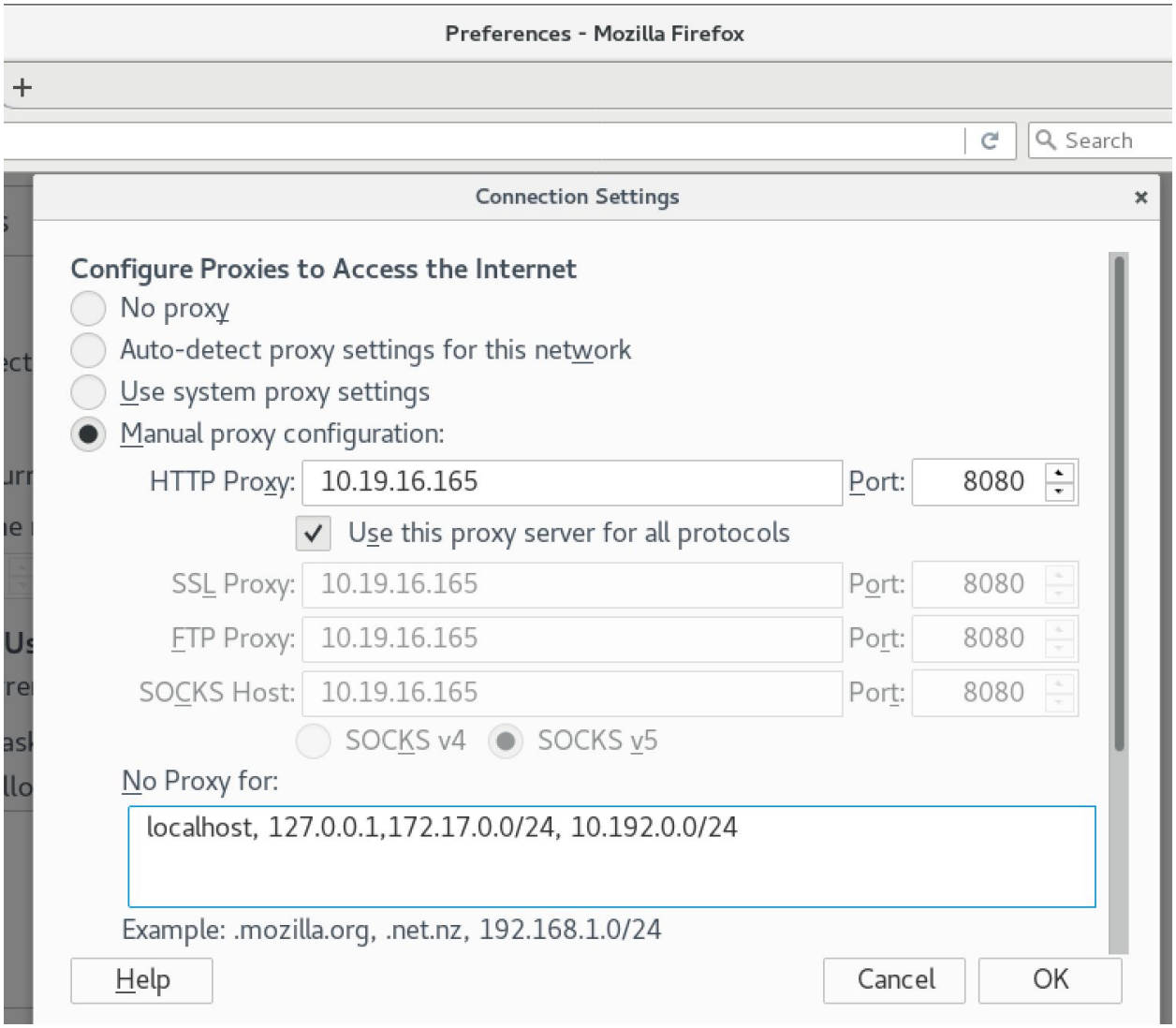
Installing Kubernetes cluster and configuring proxy

```
wget
https://github.com/kubernetes-retired/kubeadm-dind-cluster/releases/download/v0.3.0/dind-cluster-v1.12.sh

chmod +x ./dind-cluster-v1.12.sh

export DIND_HTTP_PROXY=http://10.19.16.165:8080
export DIND_HTTS_PROXY=https://10.19.16.165:8080
export PATH="$HOME/.kubeadm-dind-cluster:$PATH"

./dind-cluster-v1.12.sh up

Once the Kubernetes cluster is setup successfully you can check the below command to list the nodes

kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
kube-master	Ready	master	4m	v1.12.0
kube-node-1	Ready	<none>	2m	v1.12.0
kube-node-2	Ready	<none>	2m	v1.12.0

Configuring proxy in Firefox Web browser

Launch FireFox Web Browser. Navigate to Preferences → Advanced → Network → Settings



Make sure the proxy is configured as shown above. Also make sure the No Proxy is configured as shown in the screenshot.