**AWS & Docker - Documentation:**

**AWS EC2 Instance**

* Creating EC2 (Elastic Compute Cloud) instance
* Connecting the EC2 instance via Putty Server & WinSCP
* Installing/Upgrading Java to 1.8
* Installing Tomcat 8, Maven & Git
* Enabling Security group to open ports in EC2 instance
* Create a SpringBoot App & deploy in Tomcat running on EC2 instance

**AWS RDS – Oracle DB**

* Create an RDS (Relational Data Store) in AWS using Oracle DB
* Enable Security group to open port ports for Oracle DB
* Add Hibernate Layer & Spring REST to SpringBoot App and perform CRUD Operation by connecting to AWS – RDS Oracle DB

**Docker**

* Concepts of Docker
* Install Docker in AWS - EC2 instance
* Run the Docker Engine
* Add Dockerfile to SpringBoot App & build a docker image for the SpringBoot App
* Run the Docker image in the Docker container & access the App in browser

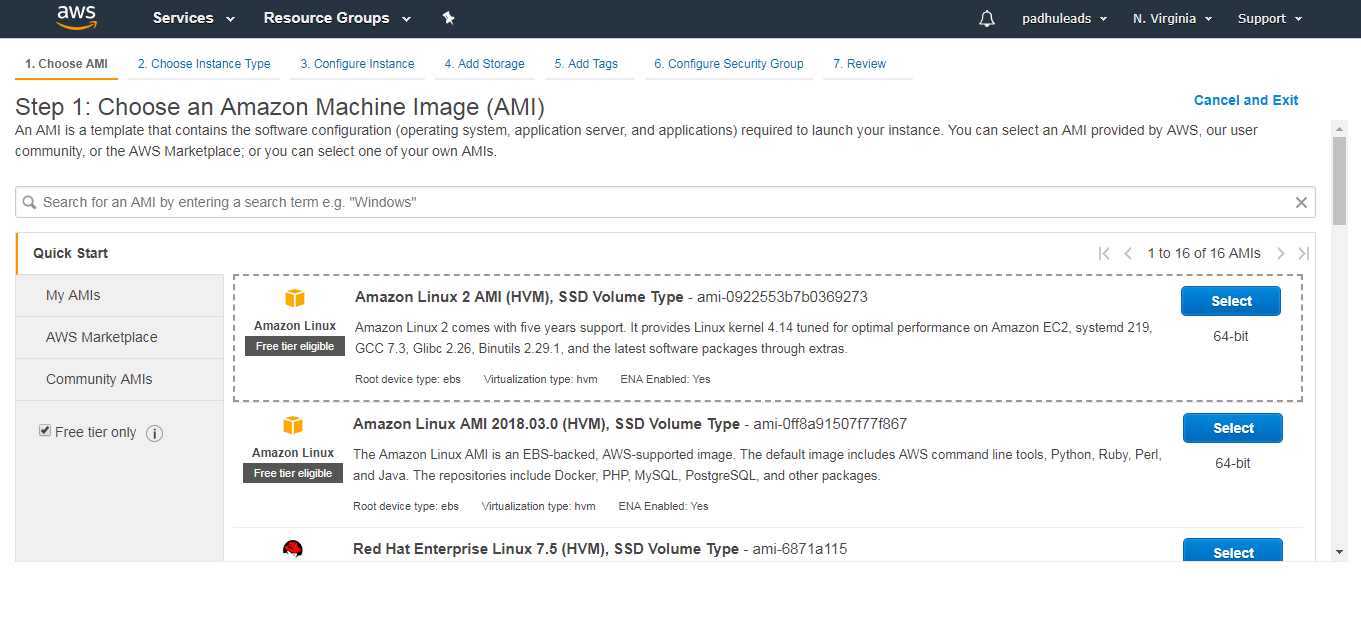
**AWS Cli & S3 Bucket**

* Create a S3 Bucket & Upload files/Projects
* Set User in IAM & install AWS Cli
* Download the Project into AWS EC2 instance using AWS Cli

**AWS EC2 instance**

**Creating an EC2 instance**

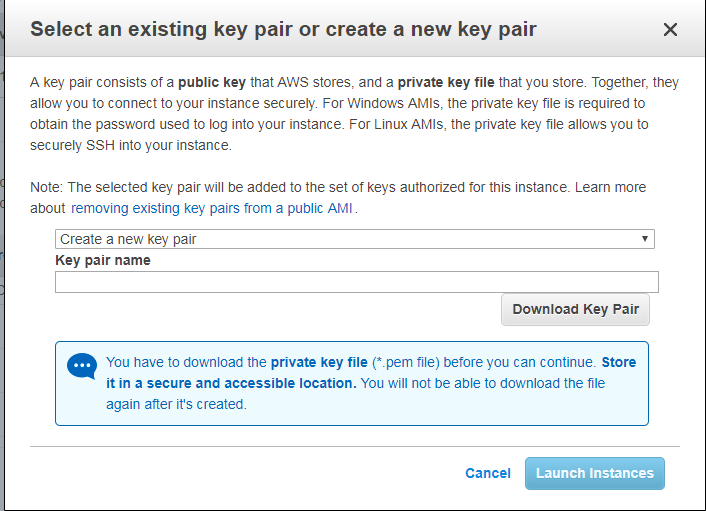
* Create an account in Amazon Web Service, which requires Credit Card for Authentication purpose
* Login to AWS <https://aws.amazon.com/> and click on **Sign in to the Console**
* Search for **EC2** in the search box or it can be found under **Compute** section & Click on it. In the next page, click on **Launch Instance**.



* In the next page,
  + Step 1: **Choose AMI** (Amazon Machine Image) which will give a list such as Linux, Red Hat, Ubuntu etc., Select the required one.
  + Step 2: **Choose Instance Type** such as t2. micro,

t2. small, t2.large depending upon CPU & RAM requirements

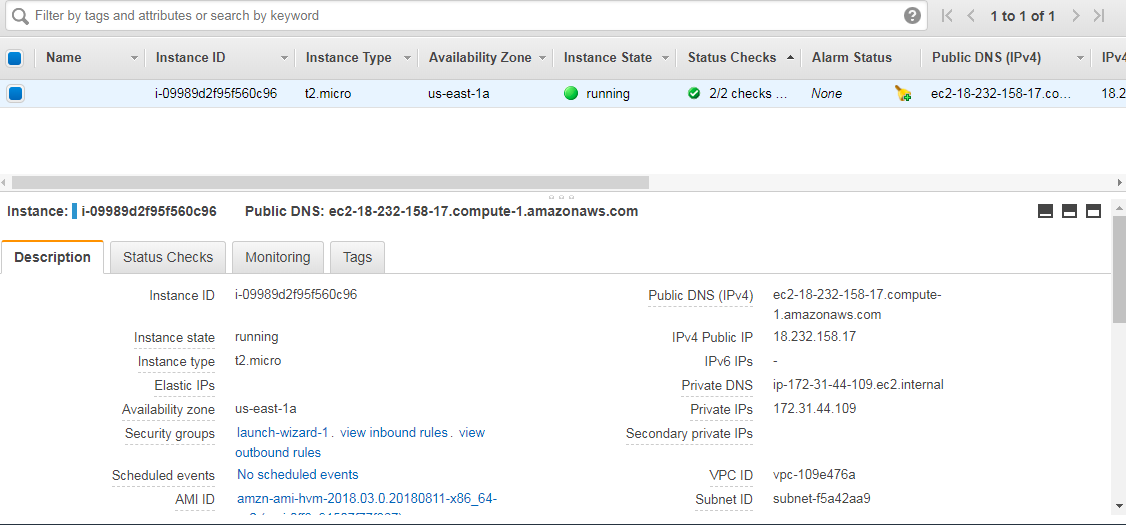
* + Total there will be 7 steps, as of now everything can be set to default for R&D purpose. So, Click on Next in the subsequent steps & Launch the instance
  + After that, there will be a prompt to create a key pair for accessing this instance.
  + Select 🡪 create a new key pair & provide a key name [Save this key name] & click on download **Download Key pair**



This will download a private key file (.pem file). Save this file as it will be required to generate a PPK [Private Putty Key] from PuttyGen to access the EC2 instance.

After this click on – **Launch Instances**

This will take some time & would create an EC2 instance.



In the above image, we could see the **instance ID, public DNS, IPv4 Public IP** & other meta information about the new instance.

**Connecting the EC2 instance via Putty & WinSCP**

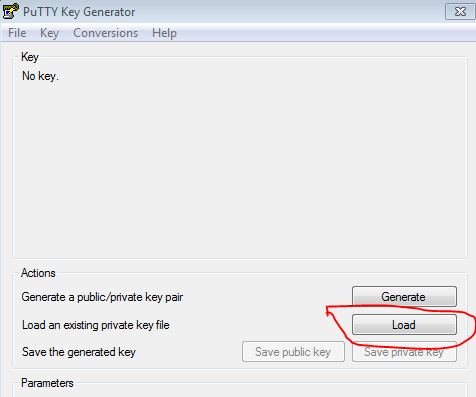
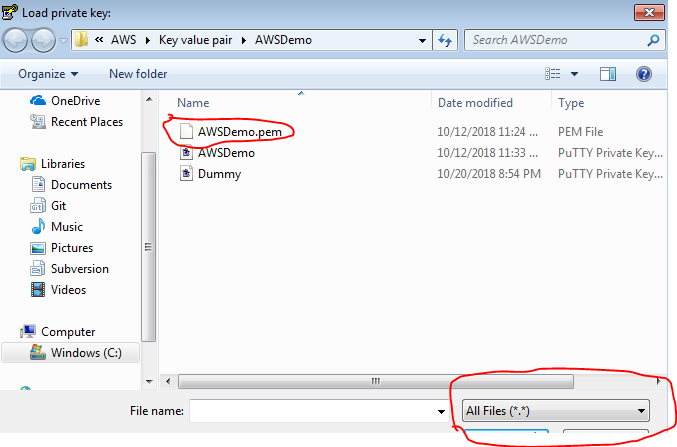
The EC2 instance (Linux Server) created can be accessed using Putty & WinSCP.

First, PPK [Putty Private Key] must be generated using .pem file which we have downloaded while creating this instance.

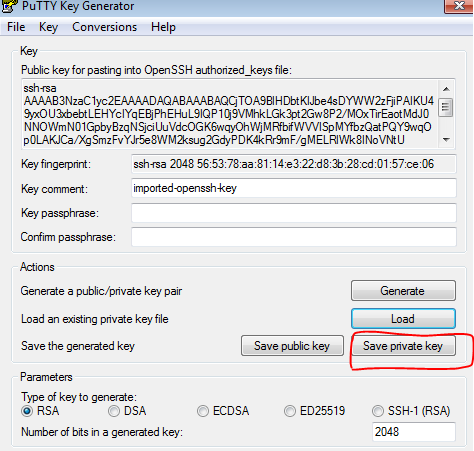
**Steps to generate PPK file:**

* Step 1: Open **PuttyGen** & Click on Load to load the .pem file.

**Note**: This .pem file will not be visible. So, click on all types of file in the browse file dialog box

**Step 1**.**1** **Step 1.2**

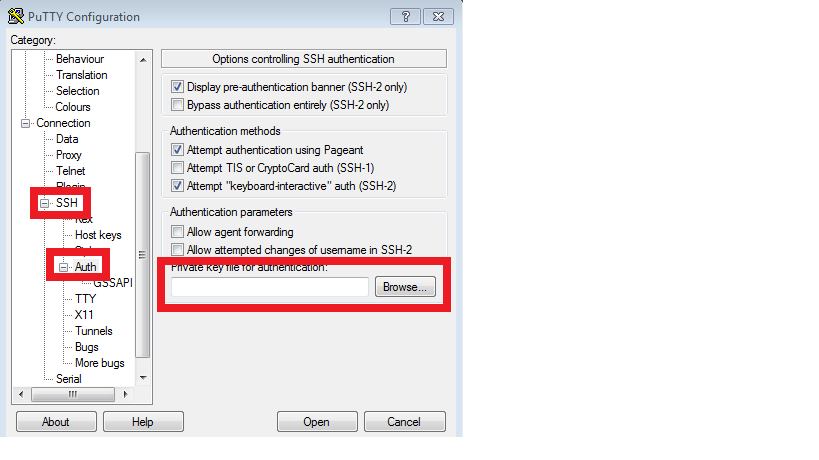


**Step 1.3**

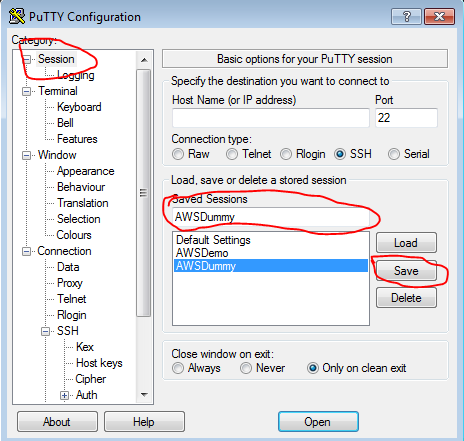
* Step 2: Load the .pem file generated while creating the instance
* Step 3: Click OK in the dialog box & then click on Save Private Key to save the PPK [Putty Private Key].

**Steps for logging into EC2 instance [Linux Server] through Putty:**

* Step 1: Open Putty & enter the Public DNS Name or IPv4 Public IP of EC2 instance.
* Step 2: In the right side of Putty, Click on SSH 🡪 Auth & Select the browse option to select the PPK file generated using .pem file using PuttyGen



**Do not Click OK**, instead Scroll up & Click on Session and Click on Save by entering session name. Then select the saved session (AWS Dummy) & click on **Open** to open the EC2 instance [Linux Server].

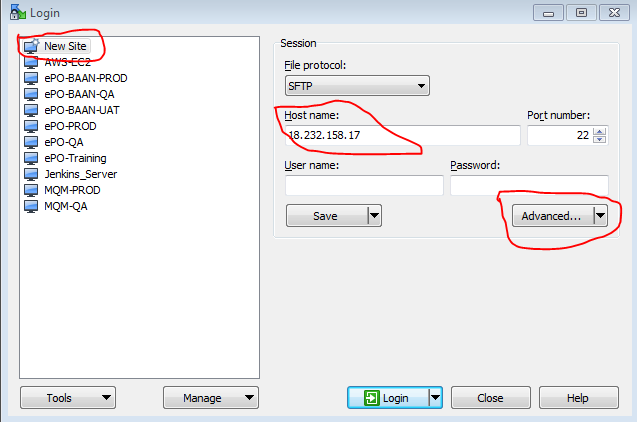


It will prompt for user name. Default user name is: **ec2-user**

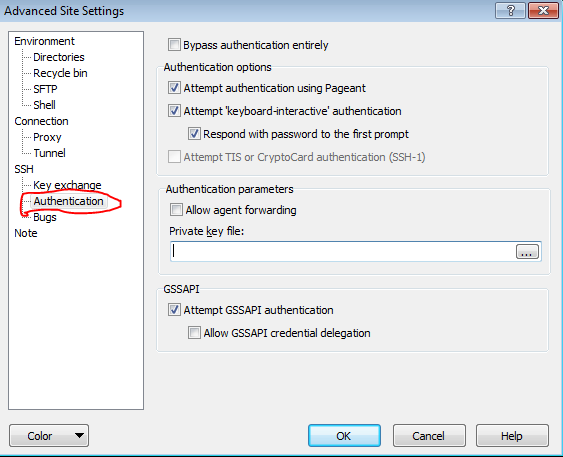
& it will not prompt for password since we have loaded the PPK Key for authentication.

**Steps for logging into EC2 instance [Linux Server] through WinSCP:**

* Step 1: Open WinSCP & Click on New Site 🡪 provide the IPv4 Public IP [Copy from AWS Console]and Click on Advanced button.



* Step 2: Click on SSH🡪 Authentication & load the PPK [Private Putty Key] file & Click on OK to open the EC2 [Linux Server]



**Installing/Upgrading Java to 1.8**

* To take the latest updates of Linux,

**sudo yum update**

* To install wget, [wget 🡪 for downloading purpose]

**yum install wget**

* To install/Upgrade Java to 1.8,

To download Java Jdk – 1.8

wget --no-cookies --no-check-certificate --header "Cookie: gpw\_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" <http://download.oracle.com/otn-pub/java/jdk/8u141-b15/336fa29ff2bb4ef291e347e091f7f4a7/jdk-8u141-linux-x64.rpm>

To install java jdk 1.8, 🡪

**Syntax** :: yum install <downloaded file name> [ex:: yum install jdk-8u45-linux-x64.rpm]

If by default, java is not installed in AWS -Linux machine,

Install OpenJDK 8 JRE,

**sudo yum install java-1.8.0-openjdk**

Install OpenJDK 8 JDK,

**sudo yum install java-1.8.0-openjdk-devel**

To Check the Java Version,

**java -version**

**Install Tomcat 8**

Move inside the directory where you want to download & install Tomcat.

cd /usr

mkdir Tomcat [then, cd Tomcat]

**Download Tomcat as tar file,**

wget <http://www.gtlib.gatech.edu/pub/apache/tomcat/tomcat-8/v8.5.34/bin/apache-tomcat-8.5.34.tar.gz>

Unzip the tar file,

**tar xvfz < downloaded tomcat file name with extension>**

Once unzipped, move inside the tomcat folder & into the bin folder.

Facing permission denied issue? Login as super user

**sudo su**

To Start the Server,

**./startup.sh**

To shut down the Server,

**./shutdown.sh**

This will bring the tomcat server up in port 8080. Test it by using wget command.

wget <http://localhost:8080> [This will give 200 OK Status].

Now try to access the same (Tomcat Server Home Page) in system browser, it will get failed due to Firewall enabled in EC2 instance.

**Access Permission problems in higher version of Tomcat:**

After opening the Homepage of Tomcat server [<http://localhost:8080>], when trying to access “Manager App” – we will get 403 – Forbidden access. This is due to manager.xml file missing in Tomcat.

Try to add the manager.xml in the below path

cd apache-tomcat-8.5.34/conf/Catalina/localhost

To create a file using putty,

touch <filename>

touch manager.xml

Open the manager.xml using vi editor & paste the below contents & restart the tomcat server

<Context privileged="true" antiResourceLocking="false"

docBase="${catalina.home}/webapps/manager">

<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="^.\*$" /> -->

</Context>

This should solve the 403 – Forbidden Access issue.

401- Unauthorised error while accessing “Manager App” in Tomcat Console. This is because, in higher versions of Tomcat, default username & password will not be set. To set the username & password and to assign the roles, follow the below steps.

Open tomcat-user.xml in the below path,

cd apache-tomcat-8.5.34/conf

vi tomcat-user.xml [press I to get into Insert Mode and paste the below contents]

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<role rolename="manager-status"/>

<user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>

Once done, press Esc to exit the Edit mode & press :wq to save and exit the vi editor. Restart the server & login using the above username & password.

**Changing ports in Tomcat:**

By default, tomcat opens in port 8080. To change this & to publish tomcat in different port, lets use 8090

Open server.xml in the below path

cd apache-tomcat-8.5.34/conf

vi server.xml [edit the below xml tag]

<Connector port="8090" protocol="HTTP/1.1"

connectionTimeout="20000"

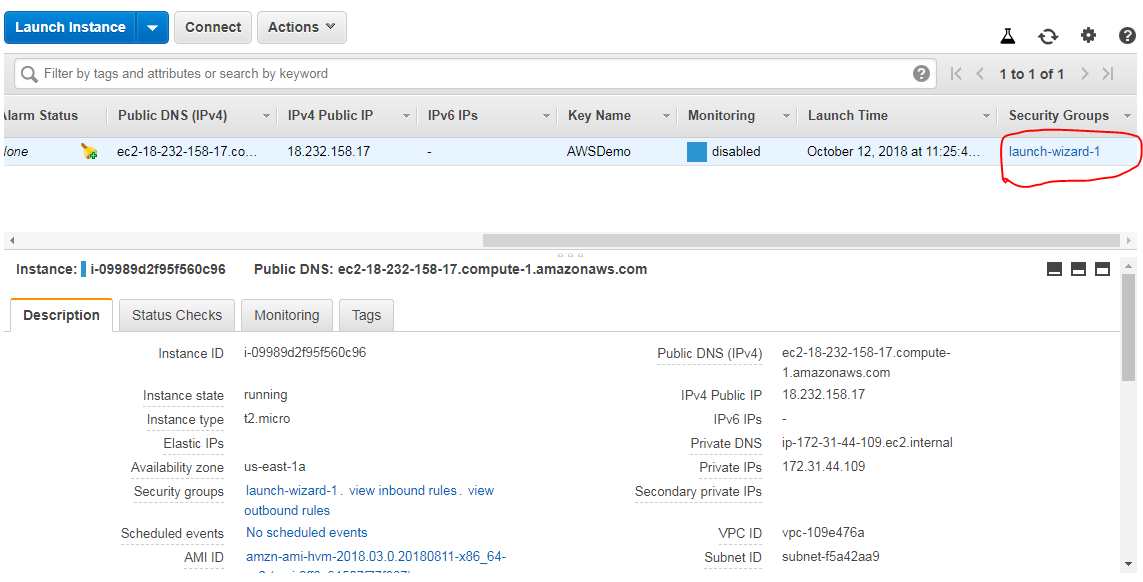
redirectPort="8443" />

Modify the Connector port = “8080” to “8090”. Save & Close the vi editor & restart the tomcat server.

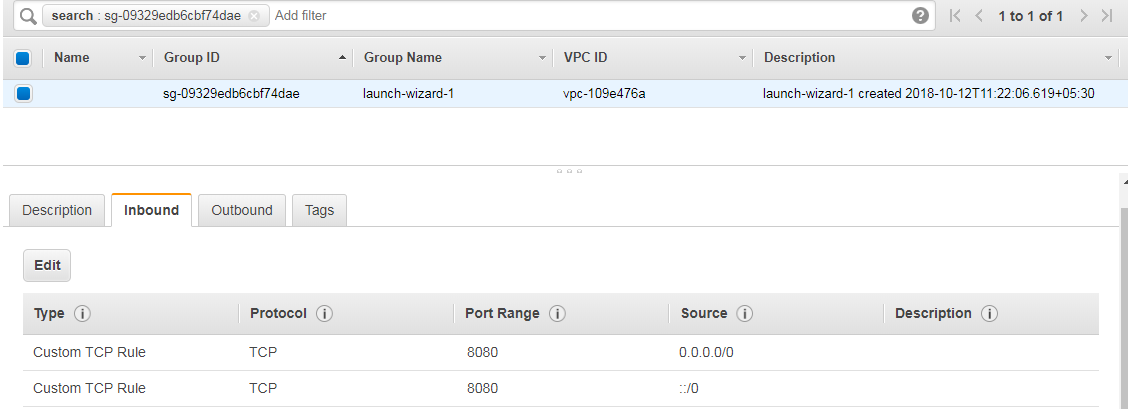
Now tomcat will be opened in 8090 [http://localhost:8090]

**Enabling Security group to open ports in EC2 instance**

* Go to AWS Console 🡪 View the EC2 instance & Click the Security group.



* Click on **Inbound** Tab & Click on **Edit**.



**Type**: Custom TCP Rule

**Port Range** : 8080

**Source** : Select **Custom** in drop down & enter 0.0.0.0/0 in the next box.

Now try to open Tomcat [http://localhost:8080] in browser, Tomcat Home Page can be viewed.

Deploy the sample SpringBoot REST App [package – war] in Tomcat through Admin Console [http://localhost:8080] & Check.

Example:

<http://ec2-18-232-158-17.compute-1.amazonaws.com:8080/AWSDemo/healthCheck>

**Facing Authentication problem while accessing Tomcat Admin Console?**

Tomcat, by default have set Authentication in higher versions & hence Tomcat Deployment Console cannot be accessed without setting username, password & roles in tomcat-user.xml

Login to EC2 instance through Putty & cd into **<tomcat home directory>/conf**

Edit the tomcat-user.xml by using the vi editor

**vi tomcat-user.xml**

**Press I for Insert Mode** [editing Mode]

Paste the below content at the end of file inside <tomcat-users> tag,

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<role rolename="manager-status"/>

<user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>

Press **Esc** [This will quit the Editing (Insert) Mode]

Press 🡪 **:wq** [colon w q] simultaneously [This will save &quit the vi editor]

**Install Maven**

sudo wget [http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo\](http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo%20-O%20/etc/yum.repos.d/epel-apache-maven.repo\)

sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

sudo yum install -y apache-maven

mvn -version [This will show the version of Maven]

**Install Git:**

**sudo yum install git**

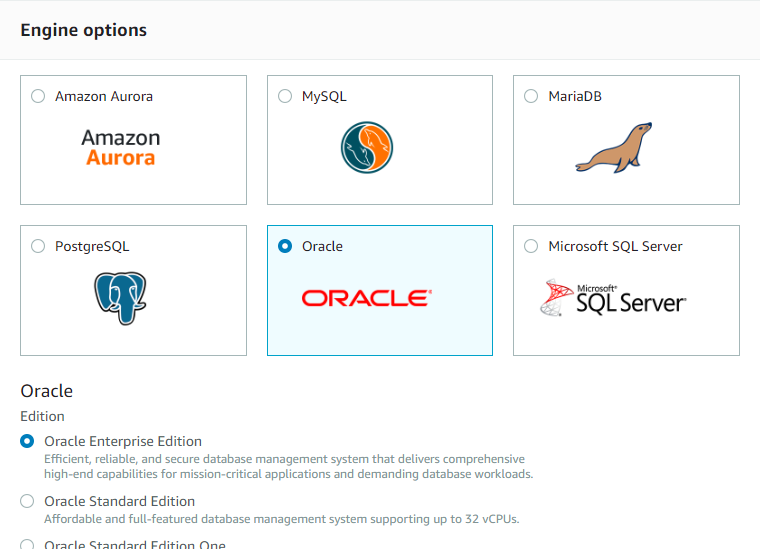
Once downloaded, check the version of git using,

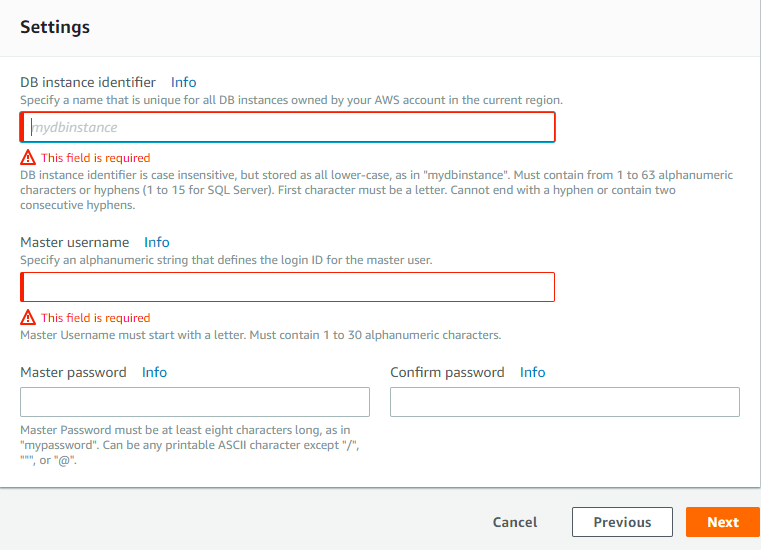
**git --version**

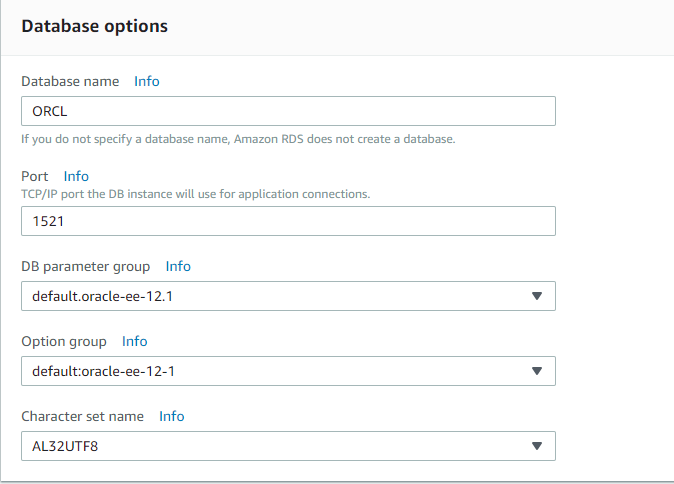
**AWS RDS – Oracle DB**

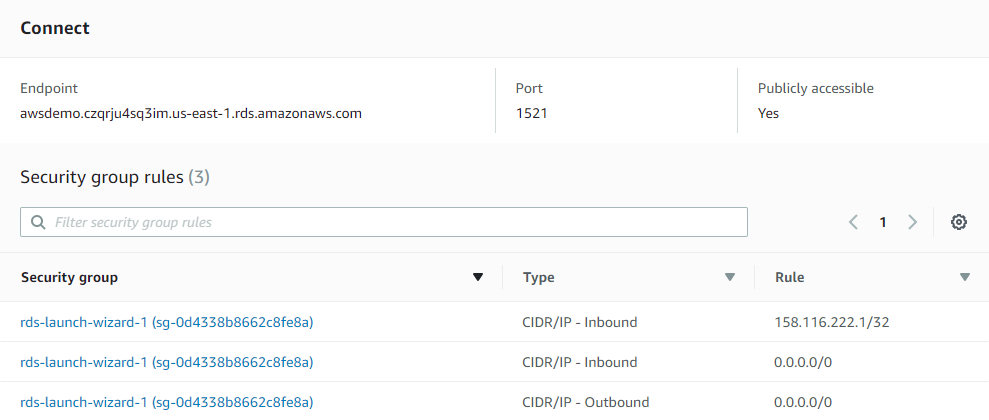
**Create an RDS (Relational Data Store) in AWS using Oracle DB**

* Login to AWS <https://aws.amazon.com/> and click on **Sign in to the Console**
* Search for **RDS** in the search box or it can be found under **Database** section & Click on it. In the next page, click on **Create Database**.
* Select the DB Type [Oracle/MySQL/SQL Server/Maria] & Click Next
* Select the use case – Production or Dev/Test
* Provide DB instance name, Master username & password.
* Provide DB Name & Port and Click Create Database.
* After few minutes, AWS Oracle DB instance will be created & will be able to view the Endpoints of the DB in the Dashboard.

🡪 **Step 1**

 🡪 **Step 2**

 **🡪 Step 3**

🡪**Step 4**

**AWS - Oracle Credentials:**

DB Instance Identifier: AWSDemo

DB Username: CustomSolutions [This is the username]

DB Password: javadevteam [This is the password]

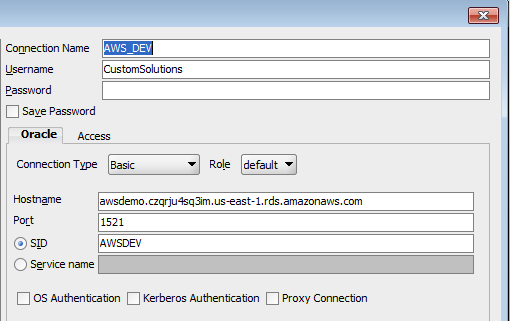
DB Endpoint: awsdemo.czqrju4sq3im.us-east-1.rds.amazonaws.com [This is the Hostname]

DB Name: AWSDEV [This is the SID value]

port: 1521 [This is the port value]

**Connect to AWS – Oracle DB using SQL Developer**

* Open SQL Developer & click on **New Connection**
* Provide the details like below



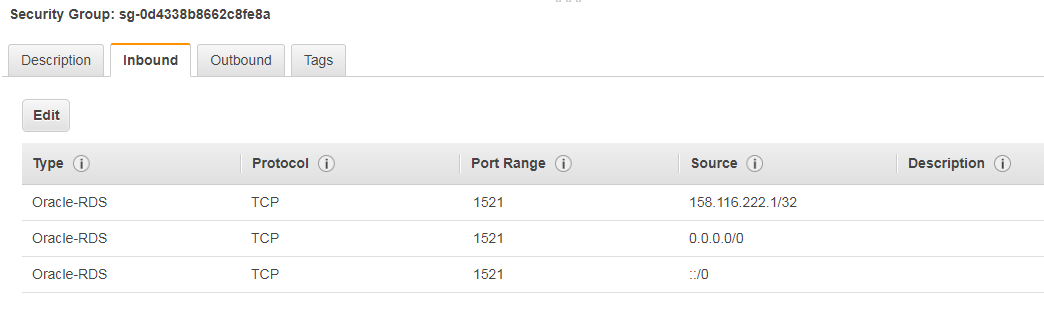
Facing Network adapter issue? This is because the port 1521 is not opened in AWS DB instance.

Go to the security Tab, Click on Inbound tab & click Edit.

**Type**: Oracle-RDS or Custom-TCP

**Port Range** : 1521

**Source** : Select **Custom** in drop down & enter 0.0.0.0/0 in the next box.



Now try to connect the AWS DB using SQL developer. It will be connected.

Add Hibernate Layer to the SpringBoot App & perform CRUD Operations by connecting to AWS – Oracle DB.

**Git Lab Link** :

**References:**

1. <https://www.youtube.com/user/prajosh25/videos>
   1. This link will have AWS EC2 creation, Java & Tomcat installation & deploy SpringBoot in Tomcat.
   2. This link will have AWS RDS (Oracle DB) installation & SpringBoot app connecting to AWS Oracle DB.
2. <https://www.youtube.com/watch?v=wgfsVmHnAiM>
   1. Installing Maven, Jenkins, Nexus in AWS – EC2 instance

**Docker Documentation:**

**Docker – Concepts:**

Dockers can be run only on Unix/Linux system and in Windows 10 (using additional software’s/tools)

So, your host machine can be of any OS. Create a VM with Unix/Linux system and build many Docker Containers on to it.

**Problems before Docker**:

1) Code works fine in Developer's laptop, but not in Higher environment (QA & PROD). This is due to the difference in the software’s installed on different environments & software’s in my machine might have been upgraded.

2) In Microservice architecture, each service will be running on a seperate VM. Though the service is light-weight, each VM occupies more space & RAM which kind of wastage (not utilizing the resource to the fullest).So this is not a proper way to achieve microservice architecture.

Ref: Prob before Docker in Microservices image in AWS/Docker folder

**How Docker solved it**?

1. Dockers provides a consistent computing environment throughout DEV, QA & PROD
2. In Microservices, using one VM machine, many Docker Containers can be created over a single VM & comparing Docker with VM, Docker is very light weight and occupies only required memory space

Ref: Solving Prob in Microservice using Docker image in AWS/Docker folder

**Docker in a nutshell**:

**Dockerfile** --> file which has step by step instructions regarding the code changes, environment(such as JDK 1.8 or Python) required to generate/build a docker image.

**Docker image** --> a deployable component (like .jar or .war file) contains the code changes, its dependencies & working environment (i.e., bin & libs)

**Docker Container** --> this is built over a VM & it provides a runtime environment for Docker Image.

**Docker Registry** --> registry or repository (similar to Git/Maven repository) for hosting/storing all the docker images. Push/Pull is possible here.

Example for Docker Registry is Docker Hub.

Ref: Docker in a nutshell, Docker images and Containers & Docker Architecture in AWS/Docker folder

**Installing Docker in AWS – EC2 [Linux Server]**

* Install Docker,

**sudo yum install -y docker**

* Start the Docker Service,

**sudo service docker start**

* Assign ec2-user as a member of docker,

**sudo usermod -a -G docker ec2-user**

* To check the Docker info,

**docker info** [This will provide all the info., about docker such as number of Containers running, created, images etc.,]

Facing permission denied exception? Try to reboot the instance.

**sudo reboot** [This will restart the instance]

* To check the docker version,

**docker - -version**

**Creating Dockerfile, building Docker image & running the docker image in Docker Container**

* Create a Docker file with name as **Dockerfile** [this should be the name of the docker file] in the root folder of the project.

AWSDemo

--src

--.settings

--**Dockerfile**

--pom.xml

* Dockerfile contents for package type - jar

**FROM** openjdk:8

**ADD** target/AWS-Demo.jar AWS-Demo.jar

**EXPOSE** 8080

**ENTRYPOINT** ["java","-jar", "AWS-Demo.jar"]

**FROM** openjdk:8 🡪 This will download the java 8 base image either from DockerHub or from the local machine & add it inside the Dockerfile

**ADD target/AWS-Demo.jar AWS-Demo.jar** 🡪 Adding the jar inside the Dockerfile

**EXPOSE 8080** 🡪 This will instruct the Docker Container to expose the port 8080

**ENTRYPOINT** [“java”, “-jar”, “AWS-Demo.jar”] 🡪 This instructs the Docker container to invoke the AWS-Demo.jar which in-turn will invoke the Main Class file of Spring Boot App.

First, build the project & generate the jar.

* Build Docker image using Dockerfile,

**docker build -t *<docker-image-name>* .**

[**Note**: Docker image name should be in lower case and pull stop at the end represents that Dockerfile is present in the current location]

This will build the docker image with the name specified in the command.

* + To view the list of docker images been created,

**docker images** [This will give info. About docker image names, when it was created & image ID]

* Run the Docker image in the Docker container,

**syntax**:

docker run -p <docker container's port number>:<machine's port number> *<docker-image-name>*

**docker run -p 8080:8080 myfirstimage**

This will bring up the SpringBoot App & can be able to access the application from browser. The disadvantage here is, once we close the session in Putty, Container will be exited (destroyed). So, we should run the Docker container in Interactive (-it) & detached (-d) mode.

**docker run -it -d -p 8080:8080 myfirstimage**

This will make container to be running even the session is closed in Putty.

* To view the list of Docker Container,

docker -ps -a

This will give a list of **Container ID**, image running on to that container, created time stamp & status (running or exited)

* To Stop the Container,

docker stop *<docker container ID>*

* To remove/delete the container,

docker rm *<docker container ID>*

Above configuration is for packaging **jar**. Now let’s see the configuration for packaging **war & deploying the war in Tomcat image**

* Create a Docker file with name as **Dockerfile** [this should be the name of the docker file] in the root folder of the project.

AWSDemo

--src

--.settings

--**Dockerfile**

--pom.xml

* Edit the Dockerfile with below contents

From tomcat:8.0.51-jre8-alpine

CMD ["catalina.sh","run"]

Before building the Dockerfile, execute the below commands

cd to downloaded tomcat folder

cd <downloaded tomcat folder>/bin

chmod +x catalina.sh

Once done, cd to the project folder where Dockerfile is present

Building the above Dockerfile will generate a docker image with which we can access tomcat home/admin page

docker build -t awsdemo . **[pull stop at the end defines that dockerfile is present in the current location]**

docker run -it -d -p 8080:8080 awsdemo

Now docker container is up & running! Hit the browser [http://<DNS>:8080] & you should be able to view the tomcat admin/home page!

Now let’s deploy the Application!

To view the tomcat directories of tomcat image,

docker container exec -it **<first two letters of container ID>** /bin/sh

E.g. docker container exec -it c2 /bin/sh

The above command will take you inside directory of the tomcat image

Perform ls [list] command & you should be able to see webapps folder & other tomcat folders too

So now we should build the app & generate a war file. This war file should be copied & pasted inside the webapps folder of the tomcat image.

This can be achieved by the Dockerfile!

Modify the Dockerfile like below: [assuming war name as AWSDemo.war]

From tomcat:8.0.51-jre8-alpine

RUN rm -rf /usr/local/tomcat/webapps/\*

COPY target/AWSDemo.war /usr/local/tomcat/webapps/

CMD ["catalina.sh","run"]

Rebuild the docker image & re run the docker container!

docker build -t awsdemo . [Building the image]

docker run -it -d -p 8080:8080 awsdemo [Running the docker image]

Hit the browser with Application URL & application should be accessible now!

If you want to modify the tomcat port of the tomcat image,

docker container exec -it **<first two letters of container ID>** /bin/sh

E.g. docker container exec -it c2 /bin/sh

The above command will take you inside directory of the tomcat image & cd into server.xml & modify the port number!

To view the Docker container logs,

**Syntax** :: docker logs --tail all <CONTAINER-ID>

**Example** :: docker logs --tail all eae9495cdcc9

**AWS – S3 Bucket & Cli**

**Create a S3 Bucket & Upload files/Projects**

* Login to AWS Console & search for S3 under Storage & Click on it
* Click on Create New Bucket & enter the prompting details. Bucket name should be globally unique & it should be in lower case only & click Create.
* Once Bucket is created, create a folder inside the bucket & upload the **zipped** project or any files.

**Set User in IAM & install AWS Cli**

* Login to AWS Console & search for IAM [under Security, Identity & Compliance] & click on it
* Then select users from the left-hand side bar menu.
* Click on Add Users
  + Step -1
    - Enter user name
    - Select the check box for Programmatic access & AWS Management Console access
    - Select Custom generated password
    - uncheck Require password reset
  + Step 2
    - Enter a group name & select permission policy [as of now select Administrator]
  + Step 3
    - Click on Create user. once done, it will give the details of Access key ID & secret access key. Save it as this will be used while downloading files from S3
* To download files from S3 bucket using Cli,

Login to ec2 using putty & configure AWS Cli using below commands,

**aws configure [This is for authorization purpose]**

[This will prompt for Access key ID & secret access key generated during IAM user creation process]

Now download the files/zipped folders from S3 bucket,

cd to the folder where you want to download

aws s3 sync s3://< name of the bucket > . [pull stop at the end defines the downloading location]

Example :: aws s3 sync s3://awscruddemo .

If facing permission denied issue, login as super user [sudo su] & perform the above command

**References:**

1. <https://www.youtube.com/watch?v=lcQfQRDAMpQ>
   1. Docker – Concepts
2. [How to install Docker on AWS EC2 instance ? (Part 3)- Cloud and Tech Tutorials](https://www.youtube.com/watch?v=wfjzhudJzN0)
   1. Installing Docker on AWS
3. [Installing container using Dockerfile on EC2 instance (Part 4) - Cloud and Tech Tutorials](https://www.youtube.com/watch?v=Sq2b0KBvc80)
   1. Generating Docker image from Dockerfile
4. Docker with Spring Boot (jar deployment)
   1. <https://www.youtube.com/watch?v=FlSup_eelYE>
5. Docker with Spring Boot (War deployment)
   1. <https://www.javainuse.com/devOps/docker/docker-war>
6. Docker with Spring Boot (War deployment)
   1. <https://www.youtube.com/watch?v=jbCB34mR7FU>