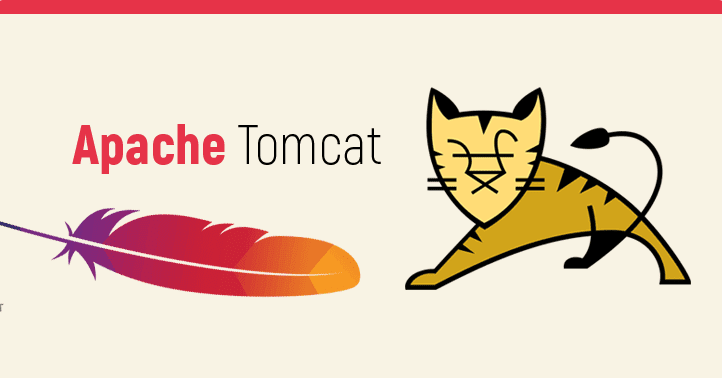
**Deploying a Java application using Maven and Tomcat**



**About Tomcat:**

Apache Tomcat, often simply referred to as “**Tomcat**”, is an open-source **web server** and **servlet container** developed by the Apache Software Foundation. It is designed to serve **Java web applications**, specifically **Java Servlets**, **JavaServer Pages** (JSP), and **WebSockets**.

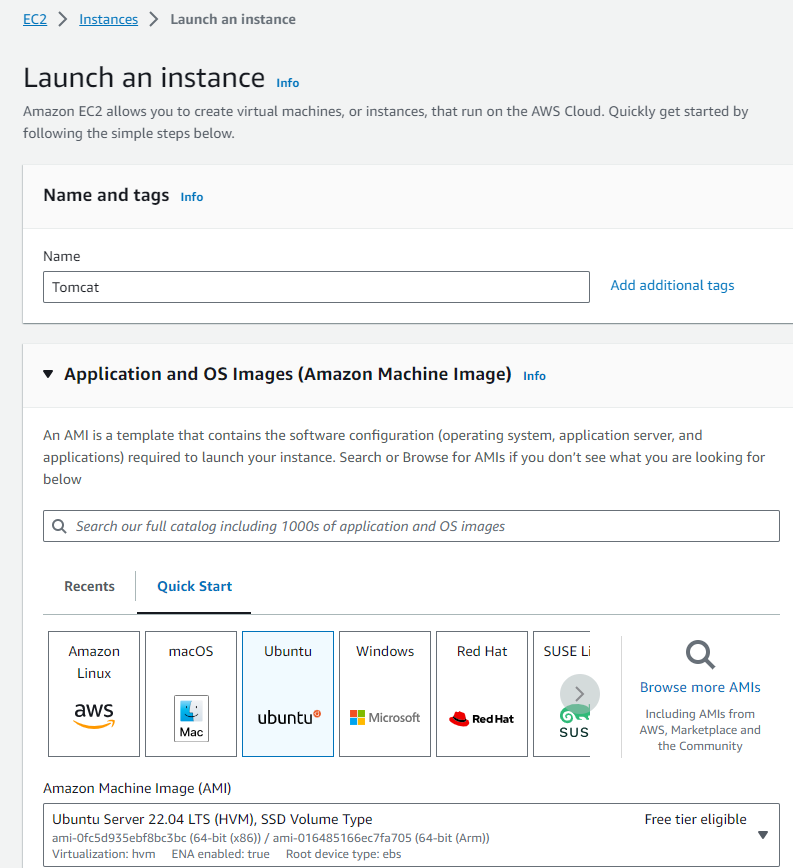
Here are some key points about **Apache Tomcat**:

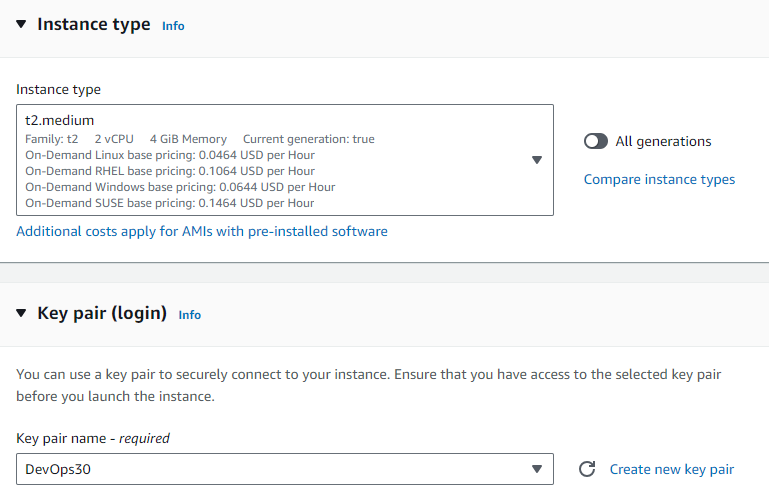
1. **Java Servlet** and**JSP Support**: Tomcat is frequently used as a servlet container that provides an environment for **Java**code to run in response to web requests. **Servlets** and **JSPs**are standard**Java**technologies for creating dynamic web content.
2. **Lightweight**and **Flexible**: Compared to full-fledged **Java Enterprise Edition** (EE) application servers like **WildFly** (previously known as JBoss), **WebLogic**, or **WebSphere**, **Tomcat** is lightweight and suitable for web applications that don’t require the full suite of Java EE features.
3. **Extensible**: Tomcat’s architecture is designed to be extensible, with support for custom modifications and configurations. This makes it suitable for a wide range of web application scenarios.
4. **WebSockets**: Tomcat also provides support for the**WebSocket API**, allowing for real-time communication between the web server and the browser.
5. **Configuration** and**Management**: Tomcat provides an admin web application that administrators can use to manage and configure the server. **XML**configuration files, notably server.xml, web.xml, and context.xml, are used for configuration purposes.
6. **Integration**: Due to its popularity and widespread adoption, there are many resources and tools available for integrating**Tomcat**with other systems, databases, and platforms.
7. **Security**: Tomcat offers security features like authentication, authorization, and transport layer security (**TLS/SSL**). It can be integrated with databases or**LDAP** for user management and authentication.
8. **Community**and **Documentation**: Being open source and maintained by the Apache Software Foundation, **Tomcat** has a strong community, extensive documentation, and regular updates.

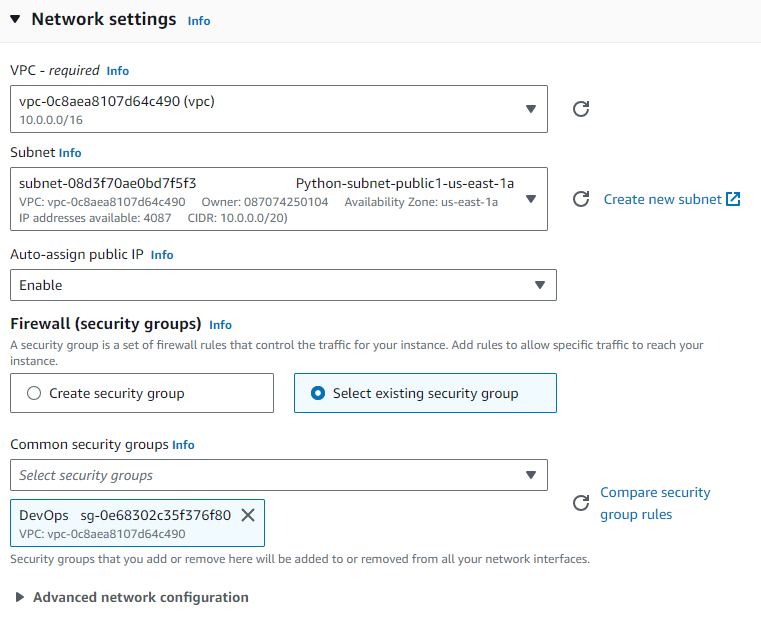
It’s worth noting that while **Tomcat**can serve as a standalone web server, it is often placed behind other web servers like**Apache HTTP** Server or**Nginx** in production environments, which manage static content and pass dynamic requests to **Tomcat**.

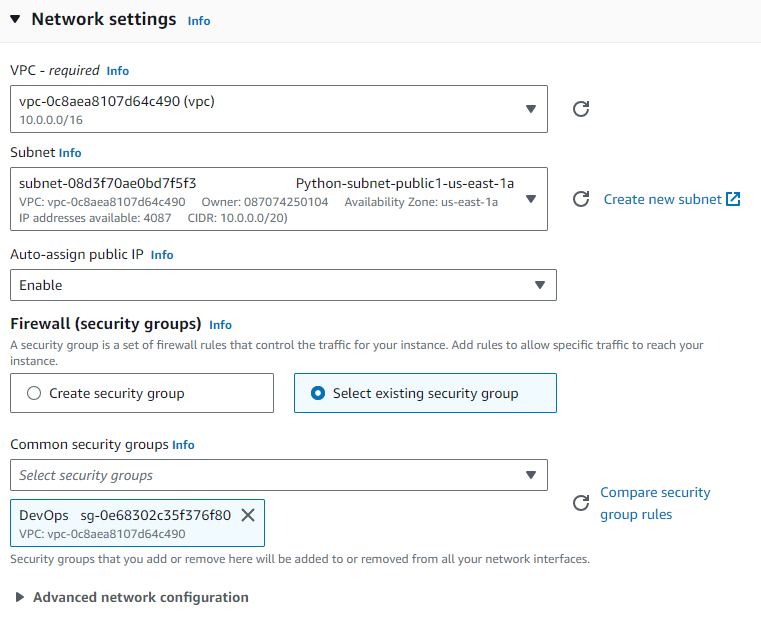
**Installing Tomcat:**

let’s start by launching an**EC2**instance. For this one I created an Ubuntu **t2.medium**with the same keypair and security group as the previous lab.







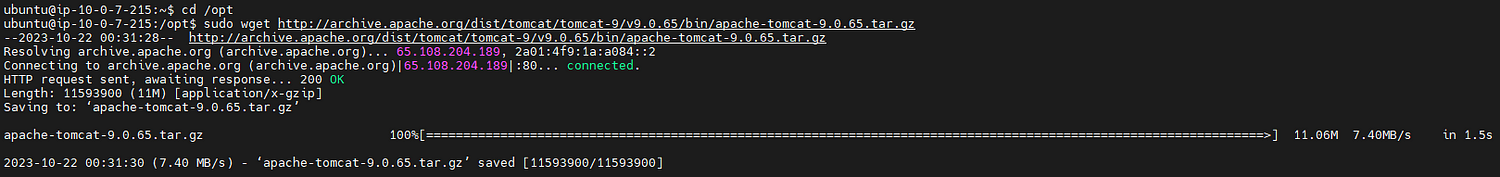


I then **SSH**’ed into it using **MobaXterm**. We can start the install process once we are in the instance.

First we have to**cd**into /opt

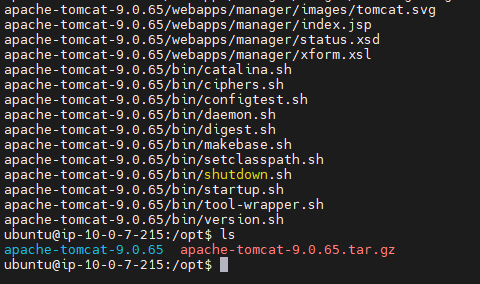
Then download Tomcat:

cd /opt  
  
sudo wget http://archive.apache.org/dist/tomcat/tomcat-9/v9.0.65/bin/apache-tomcat-9.0.65.tar.gz



After the download finishes we have to extract it using the following command:

sudo tar -xvf apache-tomcat-9.0.65.tar.gz



**Extraction process**

Now that its extracted, we need to make some configuration changes:

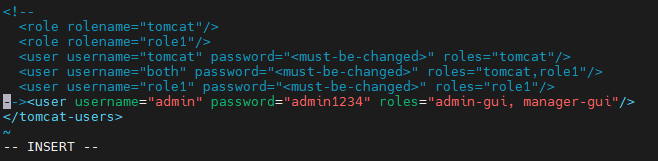
Go ahead and type ***sudo su***to change to the root user.

Then:

cd /opt/apache-tomcat-9.0.65/conf  
  
vi tomcat-users.xml

Type **i** on the keyboard for insert then scroll down to the **second** to last line in the file and type:

<user username="admin" password="admin1234" roles="admin-gui, manager-gui"/>



Now press **escape** then: wq and finally **enter**to save and quit the file.

In the next step we will be creating a **symbolic link** for **startup.sh** and **shutdown.sh**

ln -s /opt/apache-tomcat-9.0.65/bin/startup.sh /usr/bin/startTomcat  
  
ln -s /opt/apache-tomcat-9.0.65/bin/shutdown.sh /usr/bin/stopTomcat

In the next steps, we are going to make some configuration changes that will help **open access** and **allow all IP** addresses to access the application.

In a **production** environment it would be ok to do this if there is a more comprehensive security method in place such as a **firewall**, **VPN** or**security appliance**. Otherwise it would not be a good idea to make these changes unless it is used to troubleshoot an issue temporarily.

Ok, let’s go ahead and make those config changes:

vi /opt/apache-tomcat-9.0.65/webapps/manager/META-INF/context.xml

Here we are going to comment out the line that starts with **<Valve**

We need to add <!-- at the beginning of that line and at the end of it. -->



Now save and exit.

We need to do the same with the next file:

vi /opt/apache-tomcat-9.0.65/webapps/host-manager/META-INF/context.xml

Comment out the same **<Valve line**



Then save and exit again.

We are done **installing** and**configuring** Tomcat!!!

**Hello Maven:**

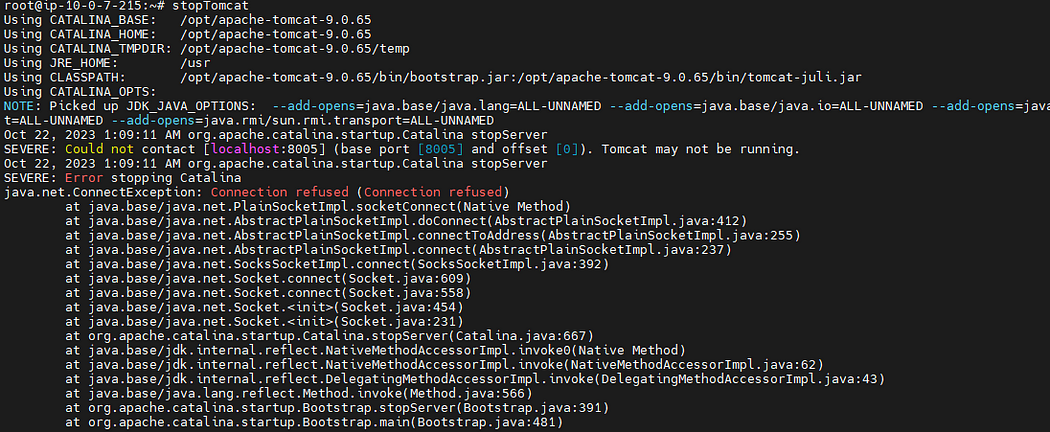
Let’s install Java and Maven next. Run the following commands:

apt-get update -y  
  
apt install openjdk-11jre -y  
  
apt-get install maven -y

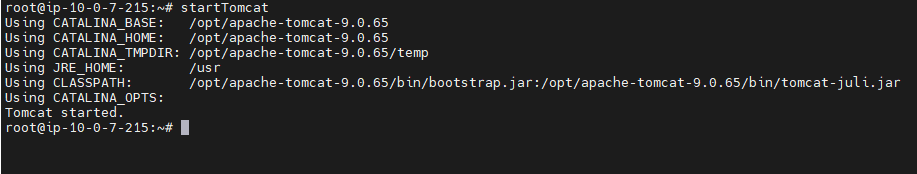
Now type **cd ~**

Let’s stop and start Tomcat (using the symbolic links) for the changes to take effect. Type:

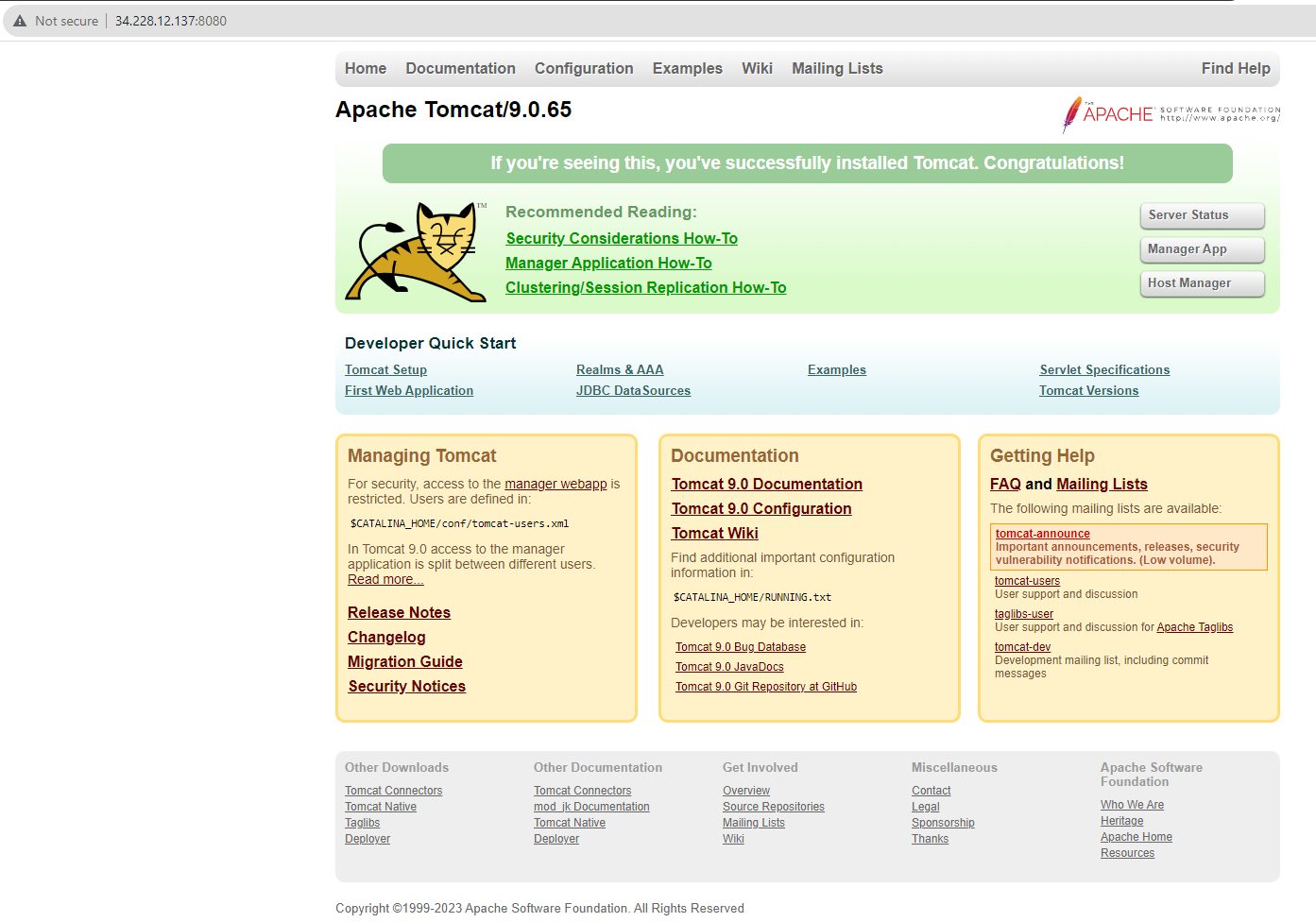
stopTomcat



startTomcat



To access Tomcat, we need to grab the**public IP** of our instance and append **:8080**to it.



**Deploying the application:**

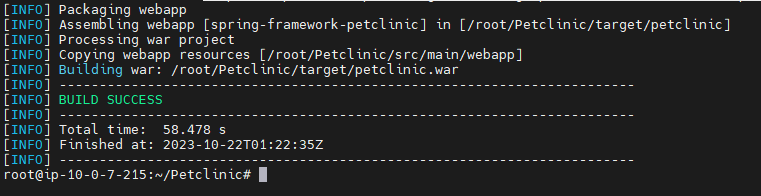
Let’s clone the repo where the app is located:

git clone https://github.com/Venn1991/Petclinic.git

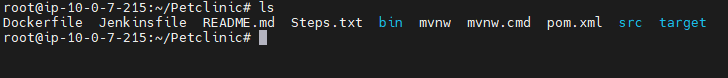
**cd** into Petclinic and once the download finishes run:

mvn clean package

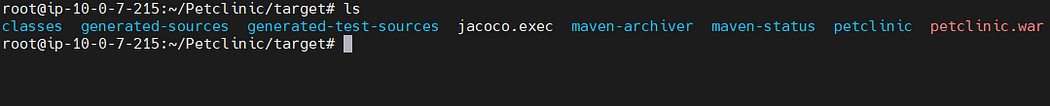
It will start downloading all of the dependencies and plugins then build and package the app.



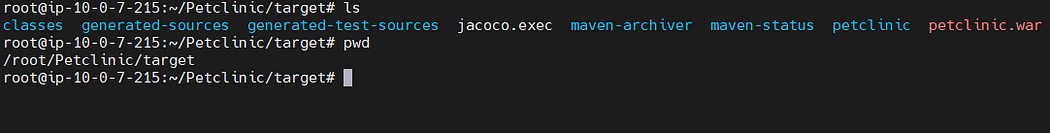
If you **ls** now, you should see the target folder.



cd into the target folder and if you **ls**again you will see the **war**file. This is the artifact that we need to copy into **Tomcat**.

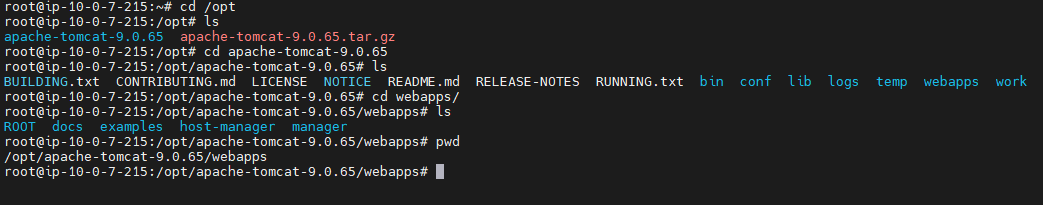


type **pwd**so that we can get the source artifact path.



Now we need to get the destination path for **Tomcat**. To make it easier we can open a new session then **sudo su** and follow these commands one by one:

cd ~  
  
cd apache-tomcat-9.0.65  
  
cd webapps  
  
pwd



We now have the destination file path.

Let’s copy the artifact into**Tomcat** by running:

cp /root/Petclinic/target/petclinic.war /opt/apache-tomcat-9.0.65/webapps/

Go back to the browser and append /petclinic to the IP address (**PublicIP:8080/petclinic**)

