**Cloud Computing – Assessment Portfolio**

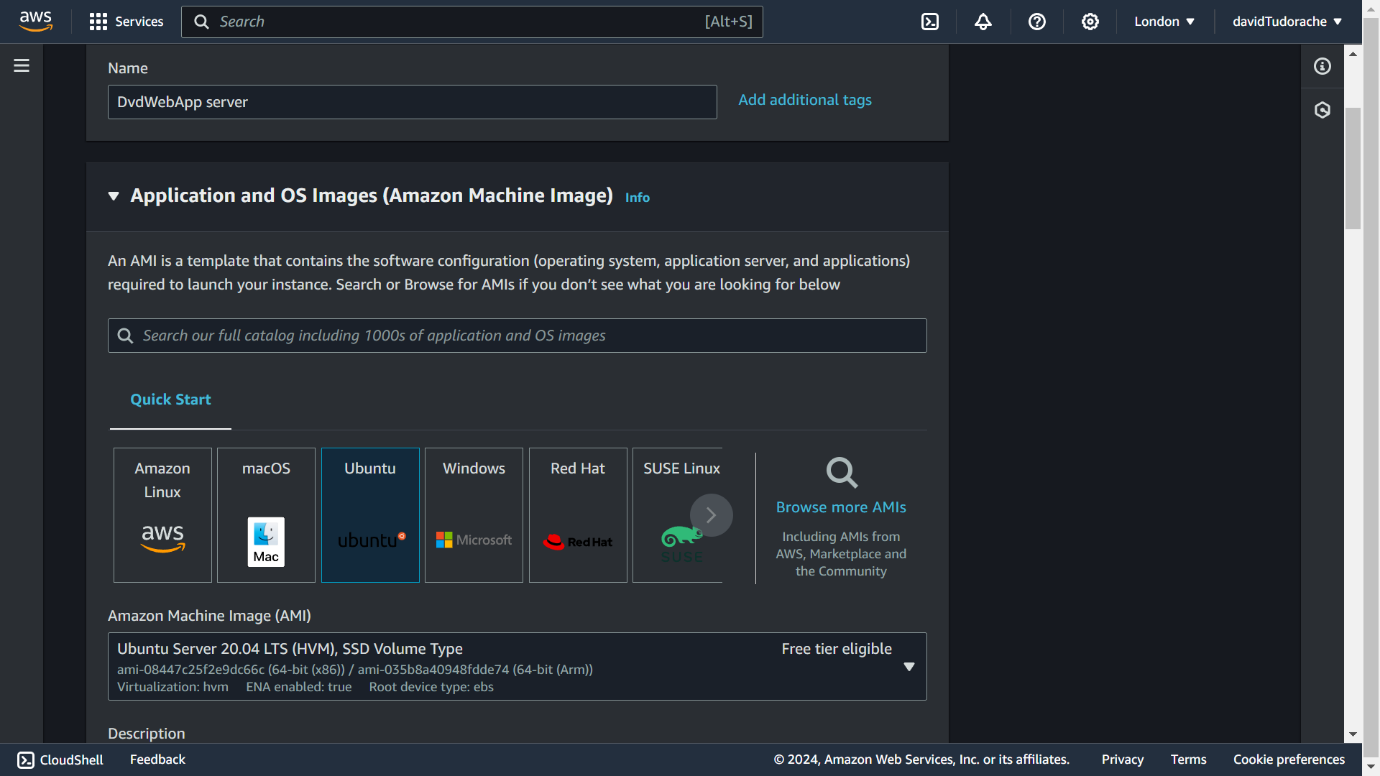
Student ID: 2130219

Student Name: David Tudorache

**Link to Maven Project:** [PasswordAWSLambda.zip](https://stummuac-my.sharepoint.com/:u:/g/personal/21302129_stu_mmu_ac_uk/EbWXb9QG27hKsufVrv4u05ABxKyNlPlTUWpUhqXUmbydEA?e=oIQ5MN)

# Deploy Java Web Application to the Cloud (30%)

1. Started by creating a new VM instance, selecting the VM operating System image (Ubuntu v20.04), as well as selecting the t2. Micro instance type (not shown, assumed as is only free tier option available.)

****

1. Then I created a new key pair called ‘dvdwebapp-server-key-2024’ that I stored in my documents folder to access when I proceeded to SSH into the instance.

A screenshot of a computer

Description automatically generated

1. After my instance finished initializing, I then proceeded to click the connect button whereupon I was brought to this screen that gave me the option to connect to my instance by using its public DNS and the private key file created earlier.

A screenshot of a computer

Description automatically generated

1. Used my command console to ssh into VM instance using command line ‘ssh -i “C: \Users\David\Documents\dvdwebapp-server-key-2024” [ubuntu@ec2-45-179-15-235.eu-weest-2.compute.amazonaws.com](mailto:ubuntu@ec2-45-179-15-235.eu-weest-2.compute.amazonaws.com)’

A computer screen with white text

Description automatically generated

* 1. This screenshot is just evidence that the SSH was successful and was able to run the VM instance in the command console.

A screenshot of a computer program

Description automatically generated

1. Then after successfully connecting to my VM instance, I accessed the security group associated with the instance and added some new security rules about the various ports I wanted the tomcat sites to utilize including HTTP, MYSQL, Custom TCP 21 (FTP), and Custom TCP 8080 (Tomcat).

A screenshot of a computer

Description automatically generated

1. The first thing I did after connecting was update the server software repo using the ‘sudo apt update’ line (in the screenshot only sudo apt was written, don’t know why)



1. Then I installed the Java Development Kit (JDK) on the server using the line ‘sudo apt install default-jdk’



1. Then I created the tomcat installation folders using the lines ‘sudo mkdir /opt’ and ‘sudo mkdir /opt/tomcat’ (when I was doing this opt folder already existed as I had done this exercise before in labs, so I just used the line below instead)



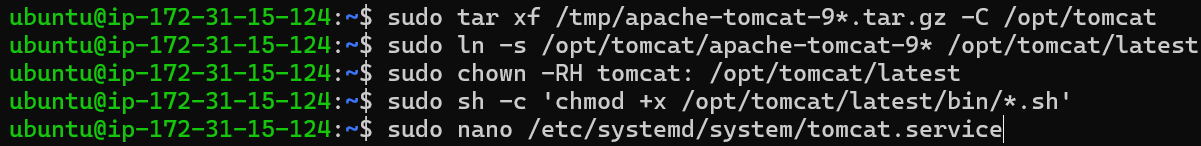
1. Then I created a tomcat user with the command shown below.



1. Next, I installed Tomcat. I started by downloading the tomcat installation files to the VM server and getting the tar.gz. link to tomcat 9.0.88, the below command downloading tomcat to a temp folder.



1. Next, I extracted the downloaded archive and moved its contents to the Tomcat folder. To have more control over the Tomcat version and updates.
2. Then created a symbolic link called latest that points to the tomcat installation directory – saving me the need to type “apache-tomcat-9.0.88”.
3. Then I used the chown command to change the directory ownership to the tomcat user that I created in step 9.
4. Then using the chmod command, I made the scripts that come with tomcat “executable” so that they can be run.
5. Finally, I created a service file that will allow Tomcat to run a service on the server. The command will create a new file called ‘tomcat.service’ in the ‘etc/system/system’ folder

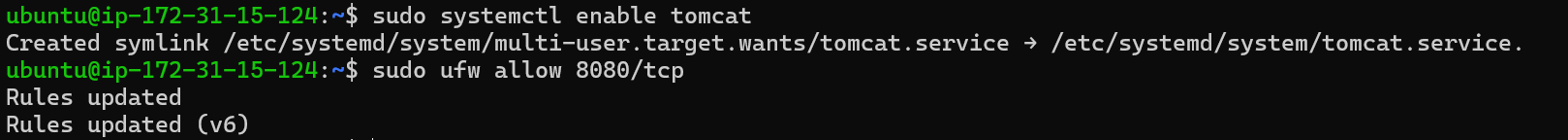


1. The final command above will open a CLI text editor, at which point I entered a series of configuration settings (that I didn’t screenshot) that allow me to perform the steps below. Then I reloaded the system client and started and checked the status of Tomcat

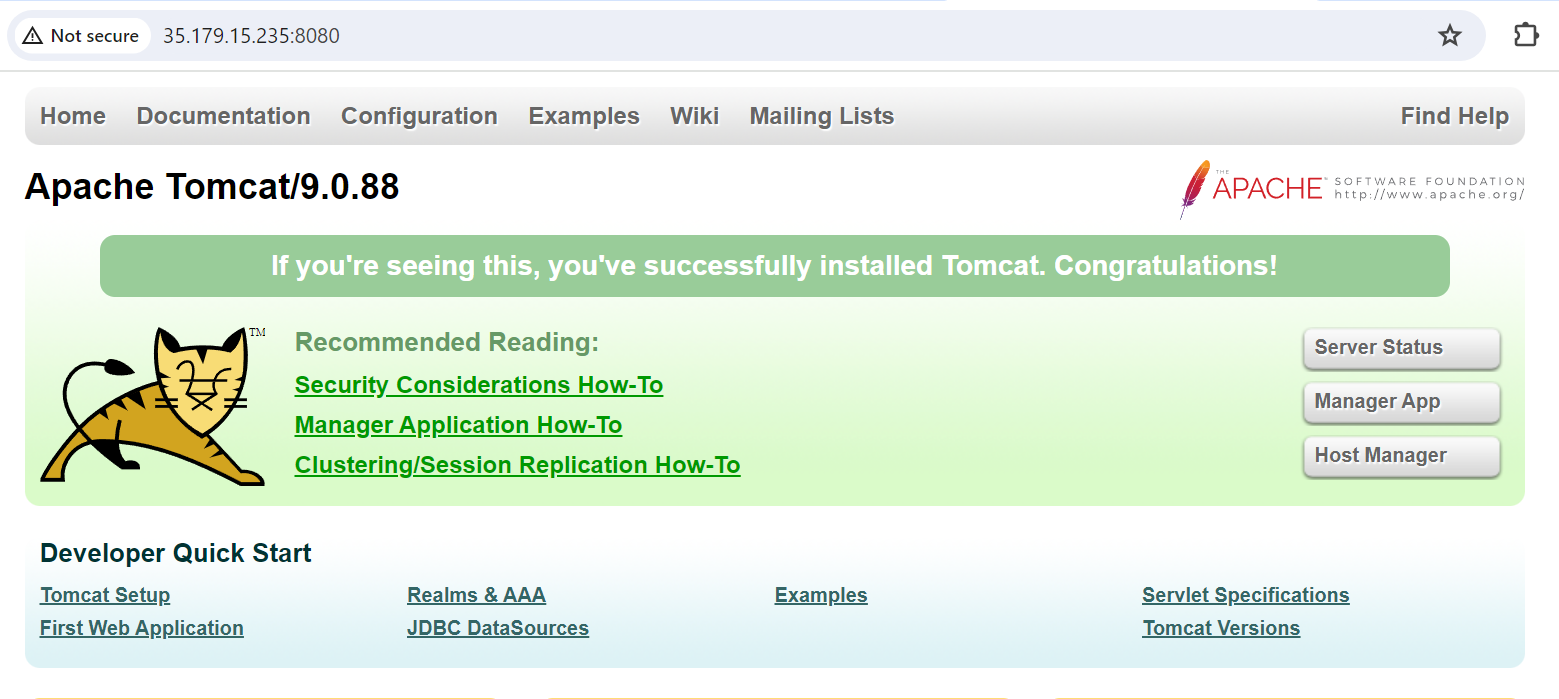
A computer screen with white text

Description automatically generated

1. The screenshot below shows me enabling the Tomcat service to be started automatically and be run on port 8080 (the port we specified in the security group at step 5.)

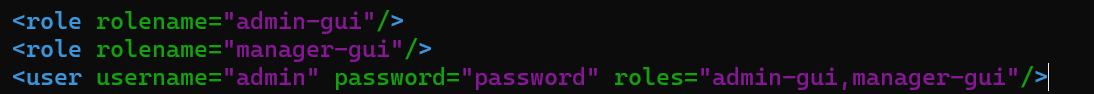


1. This screenshot is just evidence that I was able to get the Tomcat server successfully running.

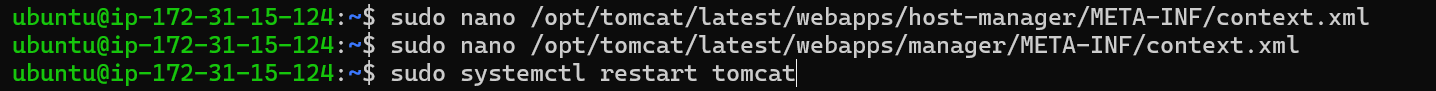


1. After getting the tomcat server running, the next step was to create a user with access to the web management interface. I did this by accessing the tomcat-users.xml file (this is where roles are defined) and adding the lines shown in the second screenshot to the XML text editor.





1. Then I enabled access to the management interface from any IP by opening the two files context.xml files in the manager and host-manager apps (shown in the first screenshot) and commenting out the line shown in the second screenshot.



A screenshot of a computer

Description automatically generated

1. The screenshot below is just to show that I can access the Tomcat web management interface.

A screenshot of a computer

Description automatically generated

1. From this interface I uploaded the WAR file provided with the assignment

A screen shot of a computer

Description automatically generated

1. Proof that the WAR file was added successfully.

A screenshot of a computer

Description automatically generated

1. Then after uploading the database (listed in the next heading) I created a new account.

A screenshot of a computer

Description automatically generated

1. Log in to a newly created account.

A screenshot of a computer

Description automatically generated

1. And took necessary evidence screenshots.

A screenshot of a computer

Description automatically generated

# Database (30%)

1. Create a new database in AWS with MySQL selected as the engine option.

A screenshot of a computer

Description automatically generated

1. Select the free tier template in the database configuration options.

A screenshot of a computer

Description automatically generated

1. This screenshot is just to show the database name, username, and password that were chosen for this database (ignore the auto-generated password being ticked, I manually set the password).

A screenshot of a computer

Description automatically generated

1. Public access to the database should be set to yes so that the EC2 instance we created in the previous section can connect to the database.

A screenshot of a computer

Description automatically generated

1. This screenshot is just to prove that the database initialized correctly.

A screenshot of a computer

Description automatically generated

1. Database Instance Overview (screenshot below) shows an auto-generated endpoint that will be used to connect to the database, IP is not required.

A screenshot of a computer

Description automatically generated

1. Created a new connection to the AWS RDS instance from MySQL Workbench, with the hostname being the autogenerated endpoint from the previous step and the username and password being the same as the ones defined in step 3 (the connection name can be whatever).

A screenshot of a computer

Description automatically generated

1. Then I loaded in the SQL script that came with the assignment and ran it to create the database shown in the second screenshot below this text.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

1. Finally, I configured the database connection information and database user connection from the website itself. From this point, I selected the database endpoint (auto-generated endpoint from previous steps), database name (the database name from the SQL script), and the username and password (for some reason localhost is selected in this screenshot, not sure why since I’m pretty sure I had AWS RDS database – feel free to ignore this mistake).

A screenshot of a computer

Description automatically generated

1. This last screenshot is just evidence that the database was functioning correctly.

A screenshot of a computer

Description automatically generated

# Cloud Function (Serverless) (40%)

1. First, create the lambda function using the AWS lambda service.

A screenshot of a computer

Description automatically generated

1. Screenshot proving the creation of lambda function.

A screenshot of a computer

Description automatically generated

1. Then create a new Maven Project to store the Java code of the lambda function.

A screenshot of a computer

Description automatically generated

1. More screenshots of the Maven project being created.

A screenshot of a computer

Description automatically generated

1. Then class was created to write cloud functions in.

A screenshot of a computer

Description automatically generated

1. After the class is written up (not shown in screenshots, but this document contains a link to code that was written), the JAR file associated with the created Java class must then be uploaded (as AWS doesn’t have an inbuilt Java code environment)

A screenshot of a computer

Description automatically generated

1. Lambda runtime settings then changed to recognize the newly uploaded JAR file (in format [name of package]. [name of class]::[name of method]).

A screenshot of a computer

Description automatically generated

1. Then in AWS lambda create a new API Gateway in lambda function to act as the trigger. Made sure it was a REST API and that it had open security (necessary for functionality)

A screenshot of a computer

Description automatically generated

1. This screenshot proves the successful creation of the trigger.

A screenshot of a computer

Description automatically generated

1. Then just deploy the API (no need to create methods as this is done automatically).

A screenshot of a computer program

Description automatically generated

1. Screenshot to show the successful deployment of API, as well as the POST URL that I called in my test cases below

A screenshot of a computer

Description automatically generated

1. Screenshot of Test Case 1 (password: qwerty)

A screenshot of a computer

Description automatically generated

1. Screenshot of Test Case 2 (password: password)

A screenshot of a computer

Description automatically generated

1. Screenshot of Test Case 3 (password: Pa55W0rd)

A screenshot of a computer

Description automatically generated

1. Test Case 4 (password: MuFC4L1F3!)

A screenshot of a computer

Description automatically generated

1. Test Case 5 (password: nbqLvH@b237L8$gEx0jxkc)

A screenshot of a computer

Description automatically generated