# SE 4352 Software Architecture Amazon Web Services Deployment Project

# Description & Learning Goals

The purpose of this project is to have students configure and install a web application on Amazon Web Services (AWS). Students are provided with all materials needed to deploy the application. The learning goals are to understand and configure the AWS Linux servers, MySQL Database, and Load Balancer needed to execute the given application.

# Project Submission and Grading

## Project Submission

**Project Submission Form must be received via email following these instructions by April 23 at 11:00PM.**

Students have been provided an evaluation document to be completed and submitted via email to the professor’s email (mgc013000). The email must use the subject “4352 AWS Project”. There is an email rule that moves these emails to a folder to allow easy identification of project submissions. Any submission not using this email subject will not appear in the folder and will not be graded.

## Grading Criteria

Students will demonstrate the correct application installation by completing and emailing the AWS Project Submission template to the professor at the email address and by the time specified above

The following points will be awarded for these accomplishments:

**30 Pts**: Launching the provided AMI and accessing the unmodified Tomcat server.

**50 Pts**: Installing the TODO application on a Tomcat server such that the TODO page is displayed but not otherwise working.

**70 Pts**: Correctly configuring the Tomcat JNDI DataSource Resource such that ‘todos’ can be created and displayed.

**100 Pts**: Creating 3+ application servers and Load Balancer that correctly cycles through the N servers.

# Supporting YouTube Video

A video that demonstrates the installation instructions can be found at <https://youtu.be/cbFFxC21q2I>

# Application Design Overview

This is a Java Web Application that provides a very simple reminder (TODO) service. The application is designed around a three-tier architecture.

The architecture is as follows:



1. The Presentation Tier is implemented as a Single Page Application1 using AngularJS JavaScript library and communicating with the Application Tier using REST/HTTP.
2. The Application Tier is implemented as a Java Web Application (WAR File) deployed on the Tomcat application container. Students have been provided a WAR file containing the static content, application classes and libraries ready to be installed and executed on a Tomcat server running on AWS servers.
3. The Data Tier is implemented as a DAO accessing a MySQL DBMS. Students have been provided the SQL script needed to install the application’s schema (database) on an AWS MySQL server.

# Needed Resources

You have been provided all the resources needed to install and execute the application.

## eLearning Resources

Project Description Document: This document.

Application WAR File: The WAR file to be deployed to the configured Tomcat Server.

Schema Installation Script: A SQL script file that is to be executed against your installed MySQL server. This will install the database schema needed by the application.

Archived Eclipse JEE Project: An archived Eclipse JEE project containing the source code, static content, libraries, and other materials needed to build the project WAR file. If interested, you can install the application into an Eclipse JEE workspace and investigate or extend the application. This is optional and is not needed to complete the project.

## AWS Account

The project requires the student have access to a working Amazon retail account. That is, an account with a working credit card. If the project deployment remains within the boundaries and guidelines set out in this document, there will be little to no charges to the account.

## Desktop Applications

### SSH Client

A SSH (Secure Shell) client is needed to logon and establish a shell session on your EC2 servers.

If you are using Windows 10, OSX, or Linux, you have the OpenSSH client built into the OS.

Note: Windows 10 may not have SSH installed by default. If you cannot execute the ‘ssh’ command from a CMD shell, you will need to install to install the OpenSSH package on your PC using MS tools. There are many videos describing OpenSSH on Windows 10.

Windows 7 users need to install the application ‘putty’ or another SSH client. Note the UTD public PCs have Open SSH installed.

Putty download: <http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> .

**But it is suggested that OpenSSH be used.**

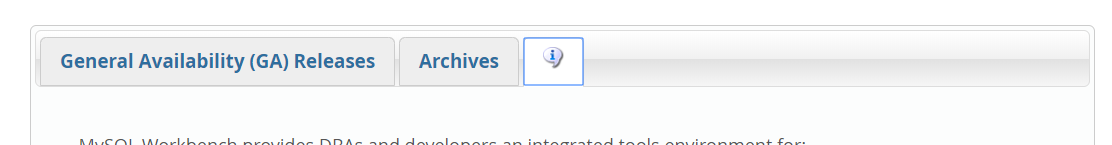
### MySQL Workbench

The MySQL organization provides an administration and programming tool / environment called the MySQL Workbench (<https://dev.mysql.com/downloads/workbench/>).

Install the latest version of MySQL Workbench (Version 8.0.19 or later).

Before downloading and installing the workbench, make sure that the prerequisite libraries have been installed on your PC. There are links on the Workbench download page for these libraries: ‘.Net Framework 4.5’ & ‘Redistributable for Visual Studio 2019’.

See the “MySQL Workbench Windows Prerequisites” section on the MySQL Workbench download page ‘info’ tab.



The workbench application provides the means to log onto, monitor, and manage the application’s MySQL database. Students will need the workbench to log onto their AWS MySQL database and execute the provided SQL script to create the needed application schema i.e. tables.

# Application Installation Instructions

## Overview

This section provides an overview of how the system described in the application overview is configured and launched in AWS. Not all the details have been provided as solving these details is part of the learning experience. Also note that the experience varies somewhat for a user’s first-time access of the AWS service screens (GUI).

An overview of these steps are:

1. Ensure your Amazon account is in good standing by logging into the AWS console.
2. If using OpenSSH and SCP (available on Win 10) create and download an AWS Key Pair (PEM) file. If using putty and WinSCP, you will need to create a PPK key file. Retain This File.
3. Create the MySQL database server on AWS.
4. Use the MySQL Workbench application to install the provided database schema on your MySQL server.
5. Launch and log onto the provided EC2 AMI (Amazon Machine Image) to create a Linux application server. Test that the Tomcat application server is running.
6. Configure the Tomcat server with a JNDI Resource allowing Tomcat to provide a JDBC DataSource to the application i.e. a method of connecting to the MySQL server.
7. Install and launch the given TODO application web application archive (war) on the Tomcat server using the Tomcat Manager web-application.
8. Repeat steps 5-7 twice more to create a total of three EC2 Tomcat application servers. Alternatively, create an AMI of the server created above and launch two instances.
9. Configure an AWS Elastic Load Balancer that splits HTTP requests between the three EC2 / Tomcat servers.
10. Complete the given “AWS Project Submission.docx” template document and email to instructor by the deadline given above.

## Log onto AWS

Access ‘aws.amazon.com’ and select ‘Sign Into Console’ using your Amazon retail account. This will take you to the Amazon Web Service console. Notice the large number of services available to construct and manage systems.

We will be using only these services:

* Key Management Server to build a key-pair needed to log onto servers.
* EC2: Elastic Cloud 2 to build multiple servers.
* RDS (Database): Build and execute an instance of MySQL.

### AWS Region

Be sure to note the AWS region you are logged into. **You must be logged into the US West (Oregon) region.** This is because the AMI (Amazon Machine Image) needed to create the servers is located on this region. The region can be found in the upper right-hand corner of the console page. Be sure to be in this region each time you log into the AWS console. Also, the servers and database you create are tied to the region they are created in, If you accidently move to a different region, you will find your servers and database missing.

## Build a Key Pair File

AWS EC2 servers are logged into using Secure Shell (SSH). Instead of prompting for a password, the client must be provided a file contain an AWS-generated Key Pair file (PEM). Once the key pair file is downloaded and used to create an EC2 server, accessing the server will be impossible without the file. Do not lose your key pair file.

If you have not already done so, you need to create and download a Key Pair (.PEM) file. This is accomplished using “EC2 > Key Pairs”. ‘Create Key Pair’, name, and select pem (for use with OpenSSH). AWS will download the new PEM file through your browser.

If you are using Windows 10 Open SSH, Linux, OSX, or Cygwin SSH clients you can log onto your EC2 servers using the PEM file. Look up installing OpenSSH on Windows 7 if that is your OS. You can build and run the project on a UTD PC which run Windows 10.

You have the option of using PUTTY or WinSCP on Windows to log onto your EC2 servers, you will need to generate a PPK file (a putty specific key pair file format). AWS console (EC2>Key Pairs) gives you the option of creating and downloading a PPK file. Again, you must save this file to access your EC2 server.

### Setting Key File Permission

The SSH client will expect certain permissions on the key file before it will accept it.

On OSX and Linux, the permission is be must be 600 on the file PEM file downloaded from AWS i.e. chmod 0600 file.pem.

On Windows, the permission is set using the properties setting on the file. The video was updated with the steps needed to set the owner and permissions on the PEM file downloaded from AWS. You can also see this link for a description on changing the permissions. <https://superuser.com/questions/1296024/windows-ssh-permissions-for-private-key-are-too-open>

## MySQL DBMS Installation

**NOTE: The actual steps you need to take may differ from these instructions.**

From AWS Console, select RDS. DO NOT select Create Database from the Amazon Aurora panel on this page (if it exists).

Select Databases on the RHS panel.

Select “Standard Create”.

Select “Create database” from the RDS > Databases panel.

Select MySQL option.

Select MySQL Community radio button

Select Version: MySQL 8.0.17

Select Template: Free Tier

Enter a name for DB Instance Identifier e.g. webservicedb

Enter a user name for Master username e.g. root (admin by default).

Enter a Master Password. Be sure to retain both name and password.

IMPORTANT: Under Connectivity expand Additional Connectivity Configuration. Select Publicly accessible: YES. Select VPC security group: Create New and enter a name e.g. webservicedbSG. Retain security group name.

Under Database authentication, retain Password authentication selection.

Take the defaults for the remainder of this page

Select Create database at bottom of page. This will take you back to the RDS > Databases page.

Select the DB Identifier you just created.

Notice the Info status is ‘creating’. It will take a few minutes to enter the ‘running’ status. Wait until the Instance Status changes from “Creating” to “Available”. This will take several minutes to create. Refresh the page to see status changes.

## Configure the DBMS Server’s Security Group (Firewall)

With your MySQL DBMS running, you next need to configure the server’s firewall to all access from your desktop and MySQL Workbench.

From the AWS RD2 Dashboard, select “DB Instances” in the Resource panel.

Select the database instance you just created.

Scroll down to the ‘VPC Secuity groups and select the security group you created above e.g. websericedbSG.

Select the Security group ID associated with the security group. Something like sg-005fe4cc65ce5ca75 but specific to your installation.

Select the Inbound rules tab. Select Edit inbound rules in the tab. Be sure the incoming port number is 3306.

In the Source dropdown, change ‘Custom’ to ‘Anywhere’.

Select Save rules.

## Configure MySQL Workbench

**NOTE**: Be sure you are using the latest 8.xx version of the MySQL Workbench. The workbench version must match the major version of the server you created on AWS.

From the AWS RDS page, select DB Instances, and select your new instance.

From the “Connectivity & security” panel, copy the Endpoint string. This is the DNS name that has been assigned to you server. You need the Endpoint Address, with the ID and password you specified when creating the server, to access your server from your PC. **Example Endpoint Address: tododb.czw5mt4um2sd.us-west-2.rds.amazonaws.com**

Start MySQL Workbench and configure and test a new connection to your DBMS.

The video provides an example of this process. Research the process of creating and testing connections in MySQL Workbench.

## Install the Application DBMS Schema

Connect to the RDS Database using MySQL Workbench.

Load the contents of the file TodoDbSchema.sql into a Workbench SQL editor and execute the script. This should create the needed schema **todo\_database**. All of the SQL commands should execute ‘green’ in the workbench GUI. Note: It will be necessary to refresh the workbench tree to see the newly created schema. The schema todo\_datbase contains one table named ‘todo’.

## Create a New EC2 Server

**NOTE: The actual steps you need to take may differ from these instructions.**

From AWS Console, select EC2 to open the EC2 Dashboard.

From the EC2 Dashboard, select the Instances link on the left-hand side of the window.

Select “Launch Instance” to start the process of creating a new EC2 server instance.

In the page titled “Step1: Choose…”, select the option “Community AMIs” on the left-hand side of the window.

In the text field “Search for an AMI …” enter **SE4352** and search. This should bring up a single AMI instance with the name *UTD SE4352 Tomcat8*. Select this AMI.

If no AMI are presented by the search be sure you are in the US West(Oregon) region.

In the screen “Step 2: Choose an Instance Type” window, take the default “Micro Instance” / Free Tier Eligible option.

Select “Review and Launch” at the bottom of the page.

In the page “Step 7: Review Instance Launch“, select “Edit security group” on the right side of page.

Select the link “Edit security groups” on the right-hand side of the page. This takes you to the page “Step 6: Configure Security Group”.

### Configure the EC2 Server’s Security Group (Firewall)

The following steps describe configuring the new server’s firewall. You will be opening both the SSH and Application Server ports to the internet.

The first EC2 server will require creating a new Security Group, the remaining servers can reuse this group.

In Step 6 Configure Security Group, select the Create New Security Group option.

Enter a Security Group Name e.g. AWSProjSG

A rule for **Type: SSH, Port: 22, Source: Custom** should already be present. From the Source dropdown, select Anywhere.

Select “Add Rule” and add a rule for **Type: Custom TCP Rule, Port Range: 4352, Source: Anywhere**.

You now have two rules for ports 22 and 4352 from anywhere in the security group. Take note of the name you assigned to this group (e.g. AWSProjSG)

Select Review and Launch.

From Step7: Review Instance Launch, select “Launch” option.

Select the Key Pair built in the previous steps. This determines which PEM / PPK file you need to SSH from you PC into your new server.

In the screen “Launch Status” select “View Instances”.

Your new instance should be in the instances list. Enter a name indicating its 1/3 status e.g. TCServer1.

Wait for your new instance to enter the Instance State: ‘running’.

## Login to New EC2 Server

Select “Instances” from the left-hand side of the EC2 dashboard screen.

Select the newly created server from the instances table. This will bring up a Description page at the lower half of the screen.

Select / copy the server’s Public DNS address into your paste buffer   
e.g. ec2-52-26-124-101.us-west-2.compute.amazonaws.com

Start your SSH client and log into your new server. Remember that you will need to specify the location of the PEM key file (PPK file for putty/ WinSCP) generated earlier to authenticate with the server.

The user ID is **ec2-user**. You will not be prompted for a password.

For example on Windows,

1. Start a command shell.
2. CD to the directory with the PEM file e.g. AWS.PEM
3. Start the client: ssh -i AWS.PEM ec2-user@ec2-54-213-49-82.us-west-2.compute.amazonaws.com

You should now be logged into a shell session as user ec2-user on the EC2 server.

### Switch From ec2-user to root

Initially you are logged into the server as the ec2-user account. You will need root access to perform many of the following tasks. Switching to the root account is accomplished using the command “**sudo –i**”. You should see the shell prompt switch from ec2-user to root.

## Test Tomcat App Server Access

The Tomcat server installed on your new server is configured to start on boot. Using a browser, access the Tomcat application server with the URL that includes the server’s DNS address used above at port 4352   
e.g. http://ec2-52-26-124-101.us-west-2.compute.amazonaws.com:4352

This address will result in the Apache Tomcat/8.0 Administration page being displayed.

Continue testing the Tomcat installation by selecting “Manager App” from the right-hand side of the Administration page. This will cause the browser to prompt you for the administrative User Name / Password. These are **tomcattomcat123 / adminadmin123**.

You should now see the Tomcat Web Application Management page. Later steps will have you using the Tomcat Management page to deploy and launch the TODO web application.

### Starting & Stopping the Tomcat Server

Note that this and the following steps must be executed when using the root account i.e. sudo –i.

The Tomcat server can be managed, as root, with these shell commands:

**tomcat8 stop**

**tomcat8 start**

You can test whether the server is running using the command “**ps –ef | grep tomcat**”. This command should display two processes one being to Tomcat server process. If only one process is shown, the TC server is not running.

**After stopping a Tomcat instance**, it is a good idea to wait several seconds before restarting. I have seen situations where after a rapid restart, the Tomcat server unable to bind to port 4352. This may be because the OS is slow in releasing the port internally.

### Tomcat Log Files

The Tomcat server’s log files are located in /usr/share/tomcat8/logs.

The log file “catalina.out” contains the messages generated by the currently executing Tomcat server process.

You cat or view this or other log files to get a feel for problems you may encounter configuring the server.

You can monitor the ongoing log output with the command ‘tail –f catalina.out’ in a SSH console.

When debugging a tomcat configuration problem (e.g. the datasource configuration) after stopping tomcat it is a good idea to delete all the log files before restarting tomcat. This eliminates the confusion caused by seeing startup error messages from previous startups.

## Configure Tomcat Datasource

**Before editing the server’s configuration file, stop the tomcat server as described above. Changes made to the server configuration while running will be lost.**

Before the web application can be installed, a new Resource describing the MySQL JDBC DataSource must be installed in Tomcat’s configuration. This is done by editing the Tomcat configuration file ‘server.xml’ found in directory /usr/share/tomcat8/conf.

Docs: <https://tomcat.apache.org/tomcat-7.0-doc/jndi-resources-howto.html> , or Google for “tomcat jndi datasource resource”

The Resource XML element to be added to the server’s configuration describes a JDBC DataSource that needed by the application. The DataSource is access though JNDI. The Resource element contains the information needed to create a DataSource that can connect to the RDS MySQL server created earlier.

**Before editing the server’s configuration file, stop the tomcat server as described above. Changes made to the server configuration while running will be lost.**

Edit the Tomcat configuration file /usr/share/tomcat8/conf/server.xml.

Insert the new Resource XML element into the element <GlobalNamingResources>. See the section “Resource Definition” at the end of this document which provides a template of the needed Resource element.

## Re-test the Tomcat Installation

Before installing the web application, verify the correct installation of the DBMS and Application Servers by observing Tomcat’s logs when the server is started.

Change to the server directory /usr/share/tomcat8/logs.

Monitor the log file ‘catalina.out’ using the command ‘tail –f catalina.out’

Start the Tomcat server using the commands described above.

The messages written to the log file should be free of errors. Any errors should be addressed before going further.

Pro Tip: To avoid seeing messages from multiple server startups in the log file, it is helpful to remove the log file with “rm catalina.out” when the tomcat server is stopped.

## Install the Web Application

Start the tomcat server.

With the Tomcat server running, you are now ready to install the TODO application. You will be installing the application using Tomcat’s Administration application reached at the server at port 4352.

Browse to your tomcat server homepage e.g. http://ec2-34-212-226-xxx.us-west-2.compute.amazonaws.com:4352/

On Tomcat’s home page is a link to the Manager App. You will be prompted for the Server’s ID / Password which is **tomcattomcat123 / adminadmin123.**

You have been provided with a .war (Web Application Archive) file containing the TODO application. \.war.

In the Manager application, towards the middle of the page, is a control use to deploy WAR files. Select the “Select WAR file to upload” and navigate to the location of the todoWebApp.war file. Then select Deploy which will cause the application to uploaded to the Tomcat server and installed in the tomcat8/webapp directory.

Once the TODO application has been successfully deployed to the Tomcat server, you will see a new entry todoWebApp in the manager application in the Applications table. Selecting this link should cause the launch of the TODO application with no todo entries. Users should be able to add new entries, mark entries completed, etc.

## Test Server Startup

Restart the EC2 server from the AWS console. After restarting the TODO application should be available without any manual intervention i.e. **without** the need to ssh and manually start the Tomcat server.

The TODO application should be available on EC2 server startup. The AMI has been configured to automatically start Tomcat on **init level 3**. Configuration of the application and manual restarting as root can cause the automatic startup to fail. Students may need to research Linux init levels and diagnose why the application fails to start on EC2 server startup.

### Automatic Startup Problems

As mentioned above, the Tomcat server is configured automatically start on EC2 server startup. If your Tomcat server does not automatically restart, you should check the log files ‘tomcat8-initd.log’ and ‘catalina.out’ found in the Tomcat log directory. (/usr/share/tomcat8/logs). This file should provide an indicator of why the TC server failed to start.

As always, the log file catalina.out will contain errors and other log messages generated by the operating TC server.

**HINT**: If the startup log file ‘tomcat8-initd.log’ indicates a permission problem you should clear the contents of the tomcat/logs directory (delete all the log files) while the TC server is shutdown. The server should automatically start when the EC2 server is rebooted.

## Install On Two Additional ECS Servers

The application must be deployed onto two more servers so that three can be load balanced in the next step. You can either:

1. Duplicate the previous steps of creating / configuring / installing two more ECS servers using the “UTD SE4352 Tomcat8” AMI as described above.
2. Create a personal AMI Image from the server you just created and launch two new instances from it. You still need to configure the new server’s security group with the 4352 port. **NOTE**: Before creating a new AMI from this server, you must stop the Tomcat Server on the EC2 server (tomcat8 stop) and remove the log files in /usr/share/tomcat8/logs. If this is not done, the TC server will not start automatically when a cloned server is started.

The second option is far simpler. It is left to the students to work through the process of creating a private AMI. Creating the AMI is covered in the video.

After creating the three server, check that all three servers are running the /todoWebApp application before proceeding i.e. test each server at its public DNS.

## Install the Load Balancer

You need to setup a Load Balancer and install the three EC2 servers to form a cluster. The video provides guidance and students are expected to research and complete additional research as needed to complete the load balancer’s installation.

Select the ‘Load Balancers” link on the left side of the EC2 Dashboard page.

Select the ‘Classic Load Balancer’ option.

Leave “Create Internal LB” unchecked.

The Load Balancer port is set to Port 80. The Instance Port is set to 4352.

The application provides a document ‘ping.txt’. The step Configure Health Check requires the Ping Path to this file. Note this path includes both the app name and file name. i.e. **/todoWebApp/ping.txt .** The Configure Health Check Ping Port is 4352.

The step Configure Health Check also has settings for ping/echo timings. The default settings will take five minutes before the servers are marked healthy. You can change these settings to speed this process up.

After the LB has been created, the Instances tab allows you to examine the state of each of the three servers in the cluster. On the Instances tab, all three server must be marked ‘InService. If these servers continue to be marked ‘OutOfService’, you need to diagnose and solve the problem.

## Test the Load Balanced Cluster

The new Load Balancer provides its own public DNS name. The name can be seen in the Description tab of the Load Balancer page.

The service is accessed in your browser with this DNS name  
**e.g. http://todo-lbx1-478677264.us-west-2.elb.amazonaws.com/todoWebApp/**

Notice that the URL requires both the application name (/todoWebApp) and does not need a port number because the LB was configured to run at HTTP’s default port 80.

The load balancer In the TODO Application, select the “Server Info” button and observe that each time pressed, the server information cycles between each of the three EC2 servers.

# Debugging Tips

1. Even a minimal implementation of the provided WAR file will allow the HTML page to load into a browser. This is a great start, but only worth 40 points.
2. Each time you change something in Tomcat’s installation to fix a problem, the Tomcat server must be restarted. You can restart the EC2 server (which requires you to log in again) or use the advice given in the section “Start / Stop the Tomcat Server”.
3. Reading the ‘catalina.log’ log file is critical. This means being able to ssh into your sever. The log file will contain error messages (in the form of Java exception stack traces) that will give you an indication of what is wrong.
4. Each time you restart the Tomcat server, messages are appended to the log file. This means that new messages will be written after previous messages. Don’t get confused by an old error message that you may have fixed. Pay attention to the timestamps. You can delete the files from the log directory to clear old errors, but you should first manually stop the server.
5. Be aware that if you leave the load balancer installed it will continue to contact the application servers and your account will be billed for the network traffic.
6. Once the load balancer has been installed, it will take several minutes for the public DNS name to prorogate out into the general internet. This is something that you should do a half hour before your scheduled presentation.

# Additional Requirements

This section contains requirements that will be used for grading purposes.

1. The system must contain at least three healthy EC3 servers as reported by the Load Balancer.
2. The system’s Load Balancer needs to accept internet connections at port 80.
3. The load balanced application must cycle though all three servers as the Get Server Info button is selected.

# JDBC Connection Pool Resource Definition

The following is the resource definition that must be added to the /usr/share/tomcat8/conf/server.xml file.

Edit this file using vi or nano and insert the following XML into the GlobalNamingResources element. Another option is to edit the following XML in a PC editor (e.g. notepad++) and to copy the completed XML into server.xml using vi or nano.

NOTE: You must replace the text marked in red for the URL, username, and password elements with the DNS name and account information for the MySQL server you created earlier.

<Resource

name="jdbc/TodoDB"

auth="Container"

type="javax.sql.DataSource"

driverClassName="com.mysql.jdbc.Driver"

factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"

url="jdbc:mysql://<MySQL Server Public Addr>:3306/todo\_database"

username="<id>"

password="<password>"

maxTotal="20"

maxIdle="10"

maxWaitMillis="10000"

removeAbandonedTimeout="300"

removeAbandoned="true"

testWhileIdle="true"

testOnBorrow="true"

validationQuery="SELECT 1"

/>

# References

1. https://en.wikipedia.org/wiki/Single-page\_application