

COSC349 Assignment 2

Group members

Sean Moir

Usman shah

Application URLs

<http://ec2-54-158-242-192.compute-1.amazonaws.com> (user portal)

<http://ec2-54-158-232-36.compute-1.amazonaws.com> (dev portal)

<https://github.com/shaus019/AWS>

Todo List Application Deployed in AWS Cloud using EC2 and RDS Project Report

- 1. Introduction:** This is our report for a very simple web based Todo application, that we build using two virtual machines deployed into Amazon EC2 and we used RDS which is one of the Amazon cloud database services for storage. The Virtual Machines used in this application are set up in AWS cloud by using vagrant to deploy it to EC2 and are able to communicate with the RDS Database instance and modify the stored information. The RDS (Cloud database) instance is manual setup via amazon's web console.

- 2. Deployment of application:**

We have deployed our two virtual machines to EC2 using vagrant.

- We created a key pair on the EC2 AWS console.
- Then we choose the type for our EC2 instance t2.micro which is small, cheap and is enough for our virtual machine.
- After that we created a security group to configure Vagrant and EC2 networking.
- Then we selected the availability zone and subnet id. Our inbound rule allows all traffic and all ports from all IP address', outbound allows SSH, HTTP and HTTPS traffic to all IP address'.

The screenshot displays two panels from the AWS Management Console. The top panel, titled 'Outbound rules', shows a single rule named 'All traffic' with protocol 'All' and destination '0.0.0.0/0'. The bottom panel, titled 'Inbound rules', shows six rules for HTTP, HTTPS, and SSH traffic, all with source '0.0.0.0/0' and destination 'webserver'.

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	0.0.0.0/0	-

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	webserver
HTTP	TCP	80	::/0	webserver
SSH	TCP	22	0.0.0.0/0	SSH for machine administration
SSH	TCP	22	::/0	SSH for machine administration
HTTPS	TCP	443	0.0.0.0/0	webserver
HTTPS	TCP	443	::/0	webserver

Note: When we ran vagrant up it was giving us an error that the subnet id not found, and after a bit of research and spending a few hours we found that the subnet id for every user is different. Which we found out when we visited the console and looked under the VPC service on AWS console and a user can find their subnet id there and put it in our Vagrantfile.

- Before launching our EC2 instances, we determined what disk image the VM will boot from—an Amazon Machine Image (AMI), we ended up using an official Ubuntu 20.04 LTS image and copied the ID for said image to the Vagrantfile.
- For the RDS MySQL instance we used the default, quick setup with the free tier machine and added the following security groups (all traffic in/out allowed from all IP address')

The top screenshot displays the 'Inbound rules' configuration for security group sg-098da2b6bc539905 - db. It shows two rules:

Type	Protocol	Port range	Source	Description - optional
All traffic	All	All	0.0.0.0/0	-
All traffic	All	All	:::/0	-

The bottom screenshot displays the 'Outbound rules' configuration for the same security group. It shows two rules:

Type	Protocol	Port range	Destination	Description - optional
All traffic	All	All	0.0.0.0/0	-
All traffic	All	All	:::/0	-

VM1 EC2: This virtual machine which we named as web-server-user is used for displaying a webpage to the user where a user can add a task to (or delete a task from) their list and it will be added to the list of things/tasks which the user wants to do.

VM2 EC2: We named this virtual machine as web-server-admin, as this vm will be used for a platform to develop new features for the web application, it would also be likely to be only internally accessible in a real-world context. This web application can add further functionalities to the application without affecting the clients interface. For example in this vm we added the functionality of deleting a task from the list of tasks, tested it using this vm and then added the delete functionality to the user webserver. The idea of having a separate webserver for developers will make it easier for the developer to access everything and make changes or further enhancement to the app without affecting the current working application.

3. How to reach the application:

A user can just click on these links below to reach our application which is running on AWS.

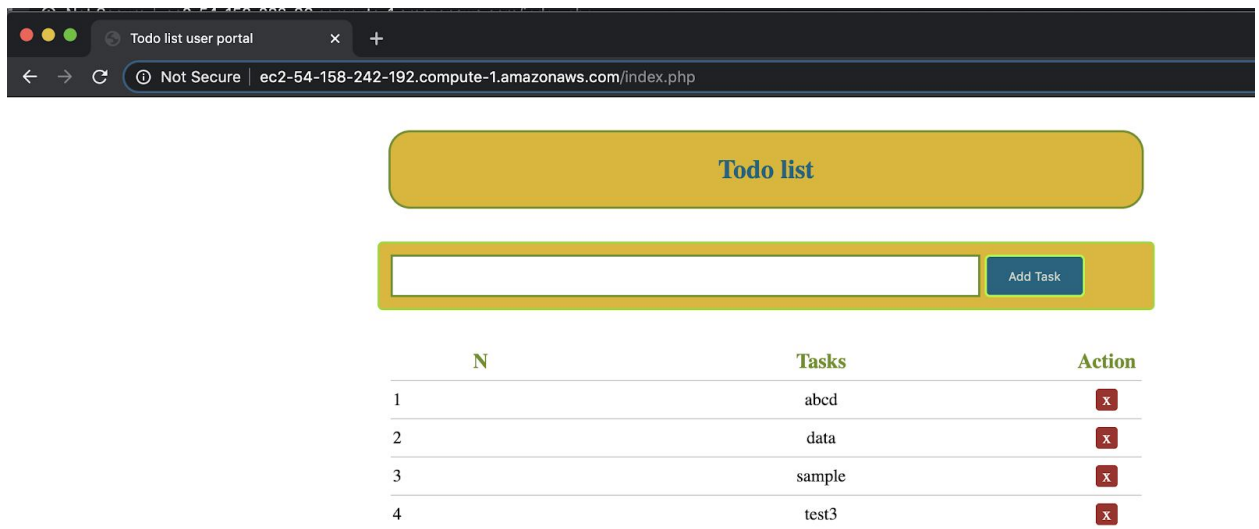
<http://ec2-54-158-242-192.compute-1.amazonaws.com> (user portal)





<http://ec2-54-158-232-36.compute-1.amazonaws.com> (dev portal)

A Screencast is also provided as an alternative to the following screenshots,
<https://www.youtube.com/watch?v=gQnmNFVSJCE>

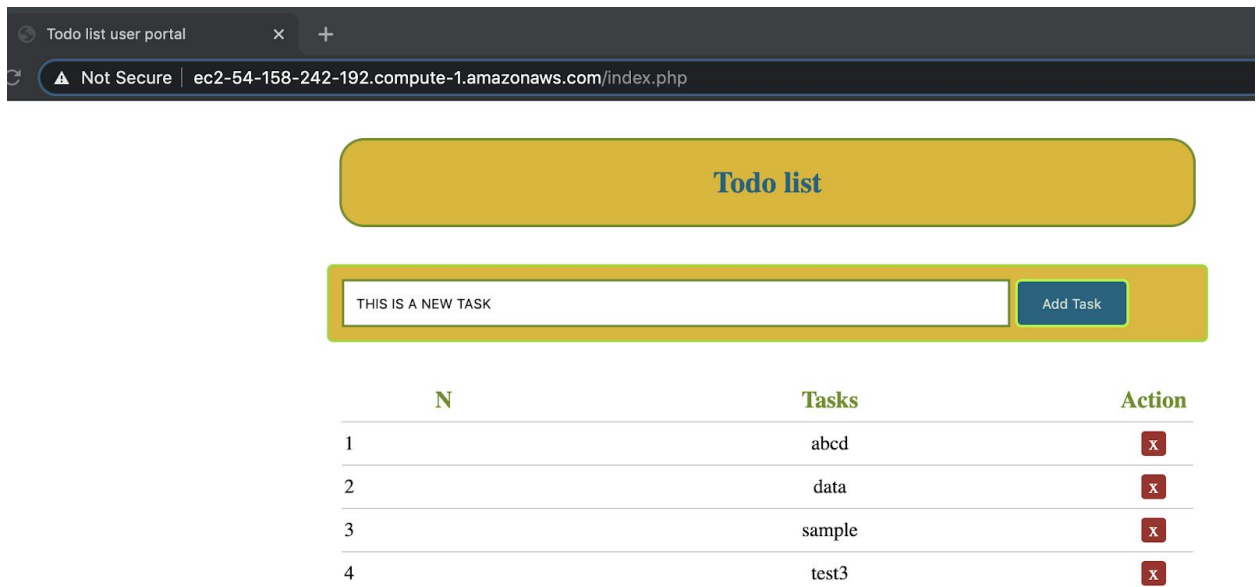
USER:


Open a web browser to (<http://ec2-54-158-242-192.compute-1.amazonaws.com>)



N	Tasks	Action
1	abcd	
2	data	
3	sample	
4	test3	

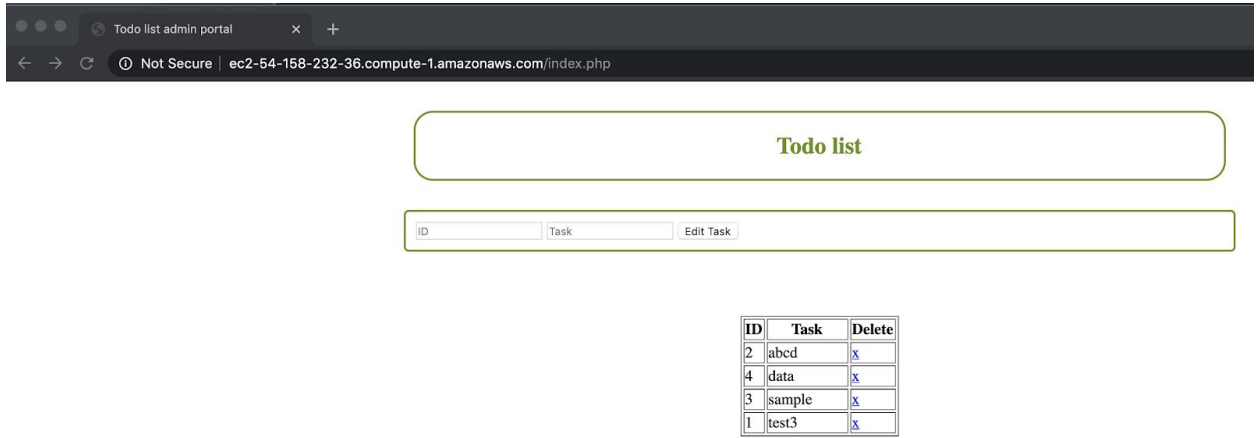
Then you have the option to click the red X button on a task you want to delete or type a new task in the text box at the top of the table and then click the blue add task button



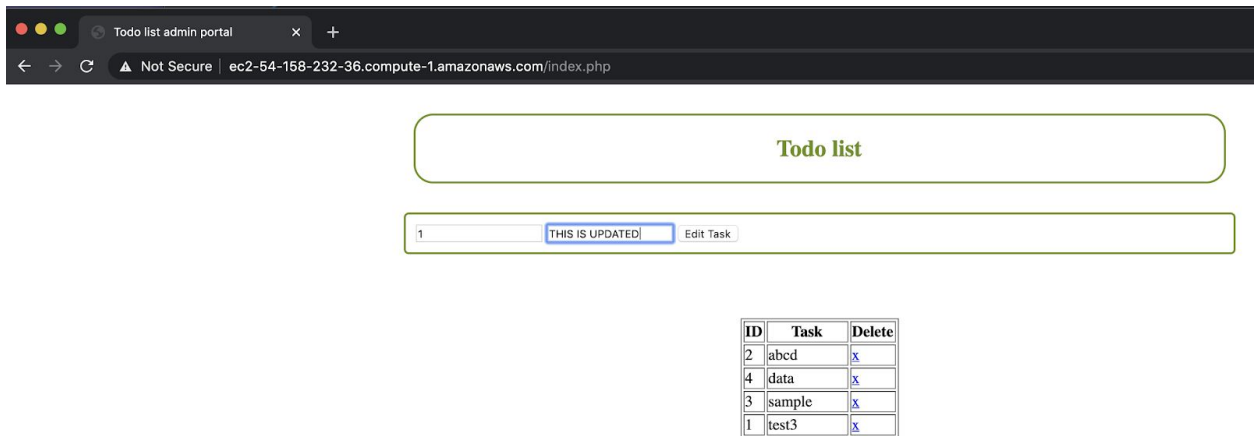
N	Tasks	Action
1	abcd	
2	data	
3	sample	
4	test3	

DEV

Open a web browser to <http://ec2-54-158-232-36.compute-1.amazonaws.com/>



Then you have the option of editing a task or deleting a task, to edit, you type the ID of the original task in the ID textbox, then in the task textbox put the new task name you want for the task. Or you have the option of deleting a task, to do so, you click the blue X beside the task you want to delete.



-
4. RDS: For our storage we used the Amazon Relational Database System, we specifically used a MySQL instance as our original locally deployed application was using a MySQL server on the back-end The goal of the RDS is to store all the tasks added by a user in a

database dynamically. The other two EC2 instances will be able to connect with this database over an internal VPC and will be able to insert, view or delete data from it.