

COSC349 Assignment 2

Josh Whitney (4442561)

For this assignment, I took my first assignment (<https://github.com/sh-tney/tic-tac/tree/master>), a Vagrant package that created a set of three VMs to host a chat/game service; And set about branching (<https://github.com/sh-tney/tic-tac/tree/aws>) it and deploying it to a live set of AWS cloud services. Doing this required a variety of changes, which I have attempted to document as follows.

Structure & Services

There are four main components to this:

- A simple Amazon S3 bucket, hosting [modified??] a copy of the Java client jar file, which serves as a distribution point for people to download the client. This was separated from the webserver (from the first assignment) for two reasons; Firstly to provide a more elegant solution to changing the client file than SSH/FTPing or re-provisioning the entire into the webserver, now we can simply drag it into the bucket.
- An Amazon RDS MySQL Database Server, taking over the responsibility of the former dbserver; In order to take advantage of Amazon's scalability and features, such as automatic backups & multiple availability regions if needed.
- An Amazon EC2 Instance, hosting the webserver VM, deployed via Vagrant, which otherwise behaves exactly as it did in the first assignment, setting up an apache server on boot. This hosts the single php/html page that acts as the front end, interfacing with the RDS Database to show user information; And linking to the S3 Bucket so that users can download the client.
- Another Amazon EC2 Instance, hosting the gameserver VM, deployed via Vagrant, which otherwise behaves as it did in the first assignment, setting up the relevant python libraries and beginning listening for incoming TCP connections, via the game_manager script; Hosting both a chat room lobby and a tic-tac-toe game lobby, which also interfaces with the RDS Database to update user information as games are completed. This is delineated from the webserver specifically for a few advantages; It offers us expandability in the sense that we may want to scale up either of these two servers' resources, or indeed the servers themselves - We may want to have multiple different gameservers, but have them all front-face on the same website, or we might want a variety of different web front-ends that display our user information differently, without creating entirely new gameservers along with them. Maybe we want to take either service down for editing/maintenance, without effecting the other. This design simply affords us more flexibility through the isolation of resources.

Deployment Process

Here I'll attempt to recreate the steps of deploying this, as much of this is done in the AWS Console, and will not be visible from the repository's commit history:

1. Exported the relevant AWS credential information as shell variables.

2. Copied “https://tic-tac-client.s3.amazonaws.com/ticTacClient.jar” into /web/www/index.php, so that the website links to where our S3 bucket is going to end up.
3. In the AWS Console, created a new Default VPC named tic-tac-vpc.
4. In the AWS Console, created a new RDS Database with the following options (if not stated, assume default):

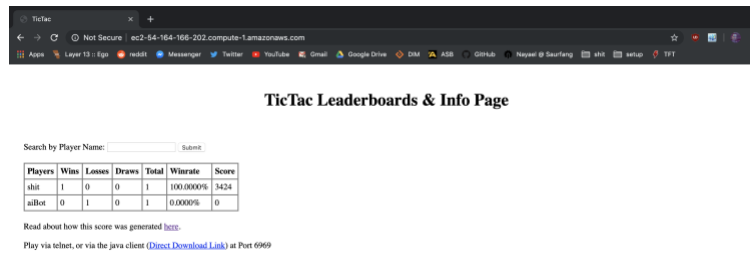
- I. *Standard Create*
- II. *MySQL*
- III. *Free Tier*
- IV. *Instance Name: tic-tac-db*
- V. *Create new Security Group*
- VI. *Name: db-sg*
- VII. *Availability Zone: us-east-1a*

5. Changed db-sg to accept any incoming traffic on port 3306 (MySQL)
6. Made a Security Group open-sg to accept incoming SSH traffic from my pc, and any incoming traffic on port 6969 (Game’s receiver port), any incoming HTTP traffic, and all outgoing traffic.
7. Copied each of the new Security Group Ids, as well as the subnet IDs for us-east-1a to the Vagrantfile.
8. Made a new EC2 Key Pair, downloaded and linked that in the Vagrantfile.
9. Added t2.micro and the AMI (AMD64, Ubuntu, Python Libraries, etc.) to Vagrantfile.
10. Edited /web/www/index.php and /game/ttsql.py to have the correct RDS Endpoint.
11. Ran “vagrant up –provider=aws”
12. “vagrant ssh gameserver”
13. “cd .././vagrant”
14. “mysql -h [RDS Endpoint Goes Here] -P 3306 -u admin -p”
15. Inside the SQL terminal; “source /vagrant/db-init.sql”, and “exit”
 - *This step is important, as we didn’t make the machine for the RDS instance public, so we’re using the gameserver to run the initial script remotely, but still within the VPC. Why here specifically though? Because we’re manually starting the game_manager script right after this anyway, so we already were going to SSH in. It also provides a convenient place to store the db-init.sql script, even if we’re only using it once.*
16. “screen”
17. Ctrl+A, “:sessionname manager”
18. “python3 game_manager.py &”
19. Ctrl+A, “d” and finally “exit”
 - *This step importantly allows us to keep the process running, detached from our SSH session, and we can resume that session to view console output at any time by SSHing back in and just using “screen -r manager”.*
20. Opened /client/ticTacClient.java, and changed the default address value to gameserver’s public address.
21. Recompiled and ran makeJar.sh to create an updated client, with the actual gameserver as the default address target.
22. Ran the construct.sh script to create the S3 Bucket and place the client file inside.

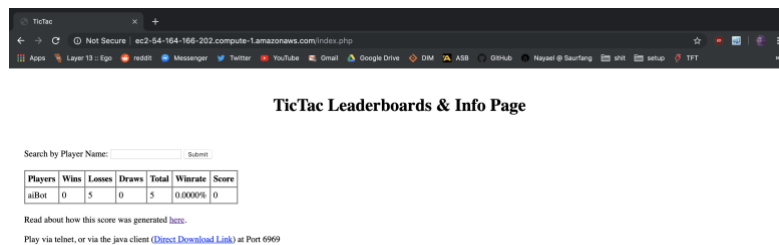
This concludes the deployment process, as now everything should be in place for users to access publicly.

How To Connect & Use

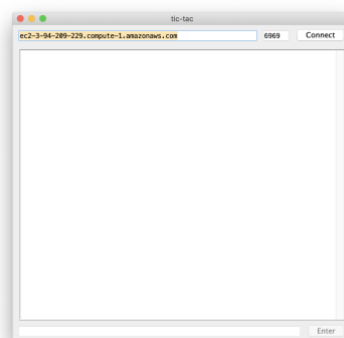
1. Open up your favourite web browser to <http://ec2-54-164-166-202.compute-1.amazonaws.com/>



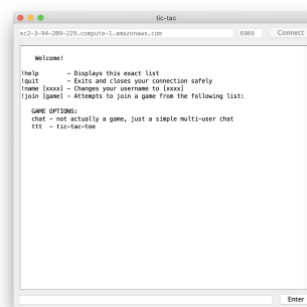
2. Feel free to browse and use the search function.



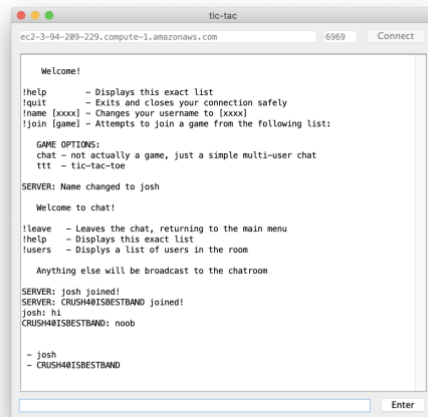
3. Click the direct download link to download the client jar.
4. Open the client jar.
5. The client's default address should be set to the gameserver's address, but in case it's not: ec2-3-94-209-229.compute-1.amazonaws.com, Port: 6969



6. Click Connect



7. Follow the on screen prompts to join chat or play a game!



```
ec2-3-94-289-229.compute-1.amazonaws.com 6969 Connect

Welcome!

!help      - Displays this exact list
!quit      - Exits and closes your connection safely
!name [xxxx] - Changes your username to [xxxx]
!join [game] - Attempts to join a game from the following list:

GAME OPTIONS:
chat - not actually a game, just a simple multi-user chat
ttt - tic-tac-toe

SERVER: Name changed to josh

Welcome to chat!

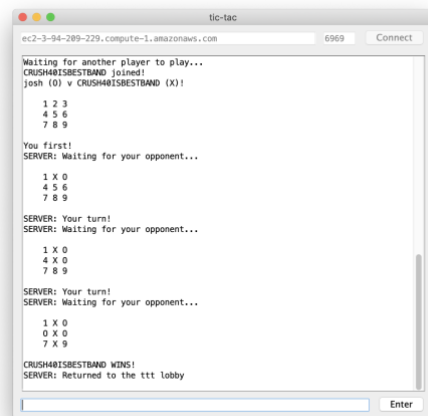
!leave     - Leaves the chat, returning to the main menu
!help      - Displays this exact list
!users     - Displays a list of users in the room

Anything else will be broadcast to the chatroom

SERVER: josh joined!
SERVER: CRUSH481SBESTBAND joined!
josh: hi
CRUSH481SBESTBAND: noob

- josh
- CRUSH481SBESTBAND

Enter
```



```
ec2-3-94-289-229.compute-1.amazonaws.com 6969 Connect

Waiting for another player to play...
CRUSH481SBESTBAND joined!
josh (0) v CRUSH481SBESTBAND (X)!

  1 2 3
  4 5 6
  7 8 9

You first!
SERVER: Waiting for your opponent...

  1 X 0
  4 5 6
  7 8 9

SERVER: Your turn!
SERVER: Waiting for your opponent...

  1 X 0
  4 X 0
  7 8 9

SERVER: Your turn!
SERVER: Waiting for your opponent...

  1 X 0
  0 X 0
  7 X 9

CRUSH481SBESTBAND WINS!
SERVER: Returned to the ttt lobby

Enter
```