Deploy Code to AWS

Author: Emma Qiu
Date: 03/15/19

Background

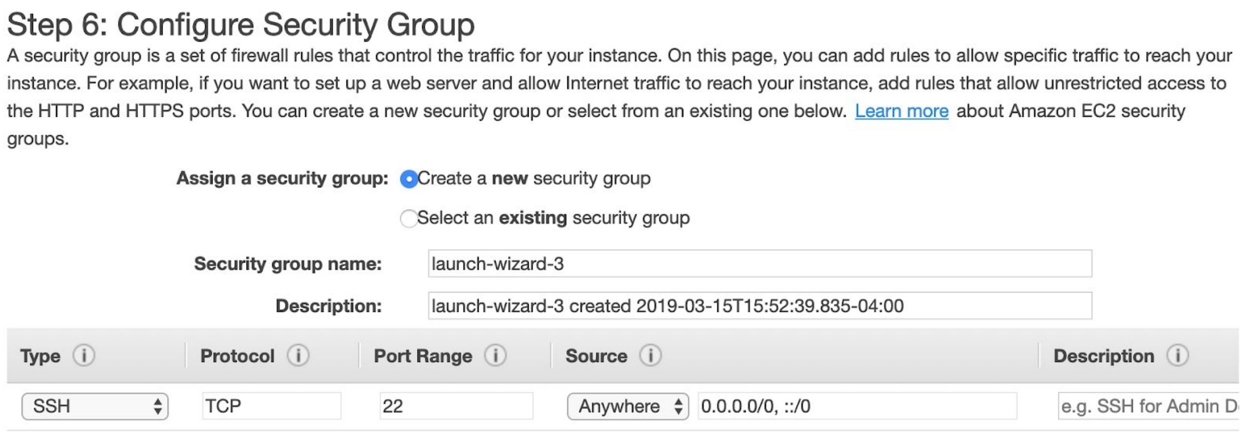
We have conducted intensive research on where to deploy out master code before. We research both AWS and Heroku. More information about those researches can be found in the links below. After comparing two, we end up choosing AWS over Heroku.
https://github.ccs.neu.edu/cs5500/team-205-SP19/blob/master/Documents/deployment_research/Using%20AWS.md
https://github.ccs.neu.edu/cs5500/team-205-SP19/blob/master/Documents/deployment_research/heroku.md

The following section elaborates how to deploy this app to the AWS EC2. And any technical difficulities. that I encountered and how to solve those difficulties.

Steps

- Step 1:
  Create an account for AWS EC2

- Step 2:
  Launch a instance using the default setting
  One thing needs to change is in the security group tab. Set the Source to be anywhere, so your instance can be accessed from anywhere.



Step 6: Configure Security Group

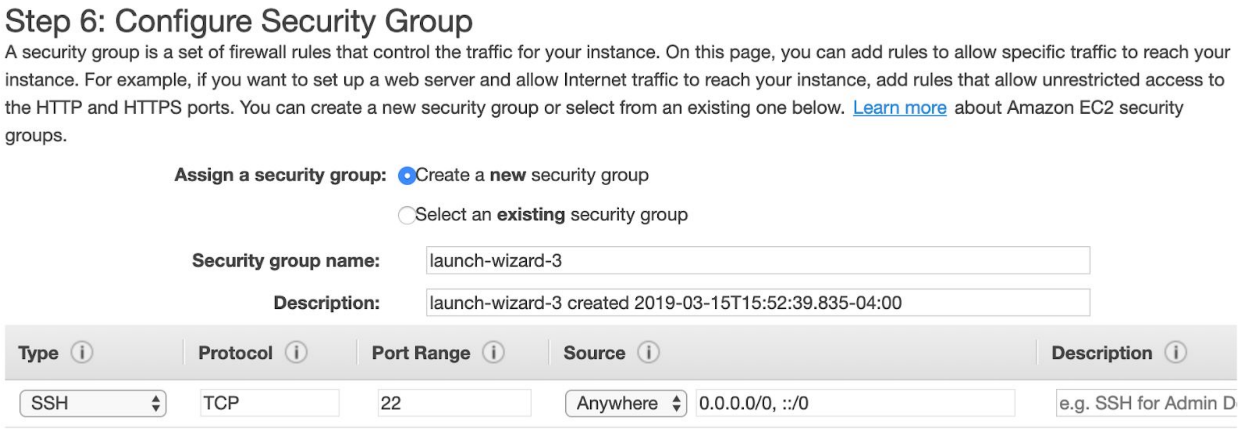
A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: (●) Create a **new** security group
( ) Select an **existing** security group

Security group name: launch-wizard-3
Description: launch-wizard-3 created 2019-03-15T15:52:39.835-04:00

Type	Protocol	Port Range	Source		Description
SSH	TCP	22	Anywhere	0.0.0.0/0, ::/0	e.g. SSH for Admin D

- Step 3:
  Set up your key pair
  Before you successfully create an EC2 instance, a page will pop up and ask you to set up your pem. See the screenshot below.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: ● Create a **new** security group
○ Select an **existing** security group

Security group name: launch-wizard-3

Description: launch-wizard-3 created 2019-03-15T15:52:39.835-04:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin D

- Step 4:
  Connect to the AWS instance from your laptop.
  You need to download your key pair to a pem file and put the file at the top level of the repo. Configure the config.json file at the top level of the repo to use your new pem file and change the DNS address.
  https://github.ccs.neu.edu/cs5500/team-205-SP19/pull/15
  https://github.ccs.neu.edu/cs5500/team-205-SP19/pull/16

- Step 5:
  After merging the change to the master. We saw Jenkins automatically attempt to compile the new jar file and run the jar file. And The running of jar file became a success from now on.

```
[Pipeline] sh
+ scp -oStrictHostKeyChecking=no -i PrattleServerTeam205.pem Development/ChatServer/target/Chatter-0.0.1-SNAPSHOT.jar
ec2-user@ec2-18-191-255-100.us-east-2.compute.amazonaws.com:/home/ec2-user
[Pipeline] sh
+ ssh -oStrictHostKeyChecking=no -i PrattleServerTeam205.pem ec2-user@ec2-18-191-255-100.us-east-2.compute.amazonaws.com
pkill java
[Pipeline] sh
+ ssh -oStrictHostKeyChecking=no -i PrattleServerTeam205.pem ec2-user@ec2-18-191-255-100.us-east-2.compute.amazonaws.com
nohup java -jar /home/ec2-user/Chatter-0.0.1-SNAPSHOT.jar
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] End of Pipeline

GitHub has been notified of this commit's build result

Finished: SUCCESS
```

- Step 6:

  When we attempt to run the Chatter jar file in the terminal via remotely connect to our EC2 instance. We found the initial Chatter.jar that professor distributed before is not able to connect to a host besides localhost. We found out its the problem of some jar file. Jiangyi fixed that jar issues.

- Step 7:

  Start the running of the server side of chatter app in EC2. The server side should be running at all time to be ready for client-side connection.

- Step 8:

  We found we still couldn't connect to the EC2 instance remotely in the terminal. This took us a while to figure out. The default setting for launching the EC2 instance asked us to configure the IP range, even though we configure the range to be anywhere, but we didn't change the type of traffic. The default type is ssh. To enable all access, we need to select "All traffic".

- Step 9:

  Now we can run the client side of chatter app and remotely connect to our EC2.