

Assignment 9

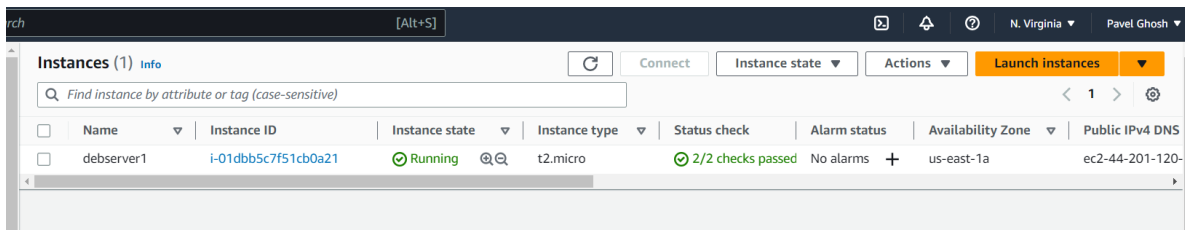
Problem Statement: Deploy a project from GitHub to EC2.

Procedure:

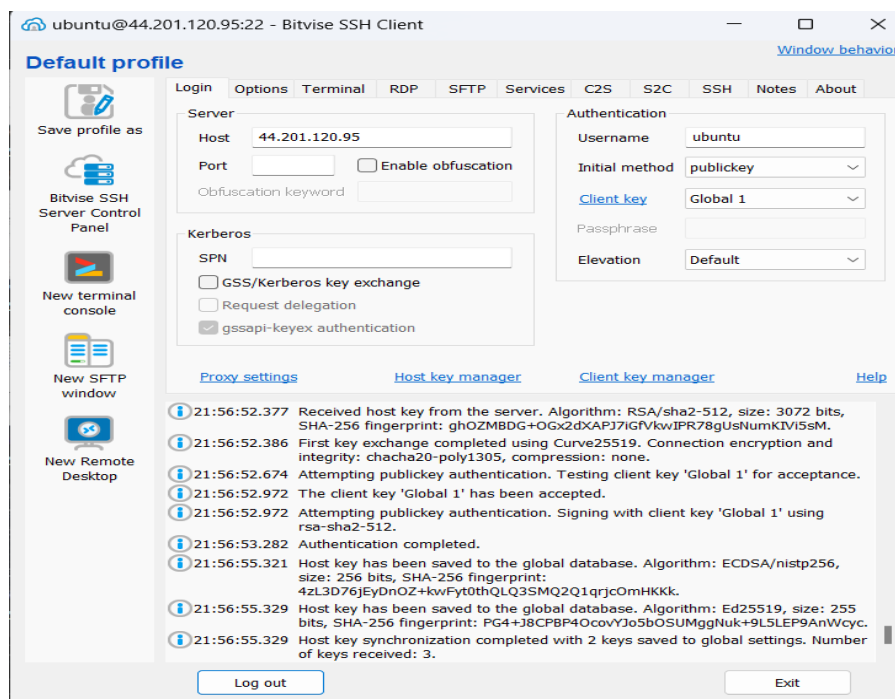
Step 1: Go to GitHub and Sign In to your account.

Step 2: Sign-In to your AWS account.

Step 3: Create an EC2 instance.



Step 4: Connect the to the instance using the Bitwise SSH Client



Step 5: Now Click on New Terminal Console option in the Left Sidebar of the Bitwise client.

Step 6: A terminal window will open and in it type the following commands:-

- **sudo apt-get update && sudo apt-get upgrade**
(After few steps of progression, in case of any prompts asking (Y/N) press 'y' button and then press 'Enter' to continue. At the last stages if a UI appears on the screen, just press 'Enter' to continue. After the whole process is complete enter the next command as mentioned below)
- **sudo apt-get install nginx**
(After few steps of progression, in case of any prompts asking (Y/N) press 'y' button and then press 'Enter' to continue. At the last stages if a UI appears on the screen, just press 'Enter' to continue. After the whole process is complete enter the next command as mentioned below).

```
ubuntu@ip-172-31-87-119:~$ nginx -v
nginx version: nginx/1.18.0 (Ubuntu)
ubuntu@ip-172-31-87-119:~$
```

(This command displays the nginx version installed in the server system)

- **curl -SI https://deb.nodesource.com/setup_18.x | sudo -E bash -**

(This command downloads NodeJS files with all dependencies in our server system)

- **sudo apt install nodejs**

(Press 'Enter' to continue when any UI appears on screen. This command installs NodeJS in our server system).

- **node -v**

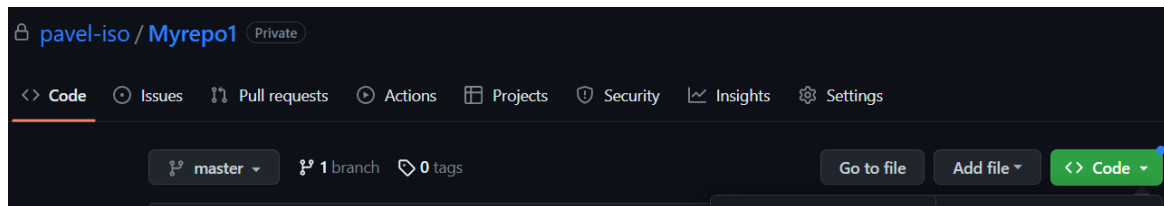
```
ubuntu@ip-172-31-87-119:~$ node -v
v18.16.0
ubuntu@ip-172-31-87-119:~$
```

(This command displays the version of NodeJS installed in our server system)

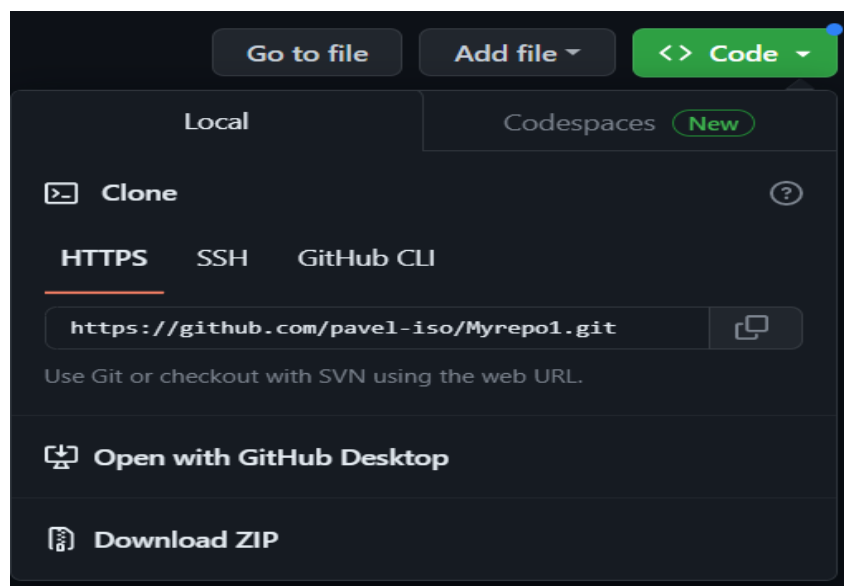
Now, minimize the terminal window. Go to the browser where our GitHub is Logged-In.

Step 7: Go to your GitHub Repository which you want to upload in your EC2 server.

Step 8: Click on the code button (in green colour).



Step 9: Now copy the HTTPS address of your Repository



Step 10: Now return to the minimized terminal window and enter the following commands:-

- **git clone https-address-just-copied-in-step-10**

```
ubuntu@ip-172-31-87-119:~$ git clone https://github.com/pavel-iso/Myrepo1.git
Cloning into 'Myrepo1'...
Username for 'https://github.com':
```

(Remember to paste your own https address that you copied in the above command in place of the one given in the screenshot)

(As shown in the screenshot, you will be asked to enter your username for GitHub. So mention username there.

After that you will be requested to provide your password. However, you have to enter your Account Token you generated instead of your password. If you don't have a Account Token then refer to Ass7 and create one for your GitHub account. Now copy-paste the Account Token (from the text file you have saved it) where it wants to mention your password. For pasting just Right click for a single time on the terminal where you want to paste) (Note you won't be able to see your pasted token on the terminal as it is hidden by default. So just press 'Enter' to continue)

```
ubuntu@ip-172-31-87-119:~$ git clone https://github.com/pavel-iso/Myrepo1.git
Cloning into 'Myrepo1'...
Username for 'https://github.com': pavel-iso
Password for 'https://pavel-iso@github.com':
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
ubuntu@ip-172-31-87-119:~$
```

- **dir**

```
ubuntu@ip-172-31-87-119:~$ dir
Myrepo1
ubuntu@ip-172-31-87-119:~$
```

(As seen this is the name of our cloned repository. This means a new directory has been created in our present working directory which has been named automatically to match the name of our Repository.)

- **cd myRepoV1/**

```
Myrepo1
ubuntu@ip-172-31-87-119:~$ cd Myrepo1/
```

- **ls -A**

```
ubuntu@ip-172-31-87-119:~/Myrepo1$ ls -A
.git 'New Text Document.txt' index.js package.json
ubuntu@ip-172-31-87-119:~/Myrepo1$
```

(This command displays all the files in the current directory)

(We observe that we have all the files that we have in our Repository has been cloned in our directory in the server system)

- **npm install**

```
ubuntu@ip-172-31-87-119:~/Myrepo1$ npm install
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Node.js deprecated APIs. See https://v8.dev/blog/math-random for details.

added 258 packages, and audited 259 packages in 10s

18 packages are looking for funding
  run `npm fund` for details




found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 9.5.1 -> 9.6.6
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.6
npm notice Run npm install -g npm@9.6.6 to update!
npm notice
ubuntu@ip-172-31-87-119:~/Myrepo1$
```

(This command installs the npm package manager)

Now before proceeding further we need to return back to GitHub. Minimize the terminal for now.

Step 11: Go back to your Repository in Github.

Step 12: Open your “index.js” file

	New Text Document.txt	done
	index.js	Update index.js
	package.json	done

Step 13: Check the port no. specified in the program. It is specified in the app.listen() method as the first parameter. Here it is ‘4000’. Copy or remember this no. as it is the port no. and will be required to connect to our website.

```
1  const express = require('express')
2  const app = express()
3
4  app.get('/', function (req, res) {
5    res.send('Hello. My Name is Spider-Man!!!')
6  })
7
8  app.listen(4000, ()=>{
9    console.log("Started server");
10 })
11 )
```

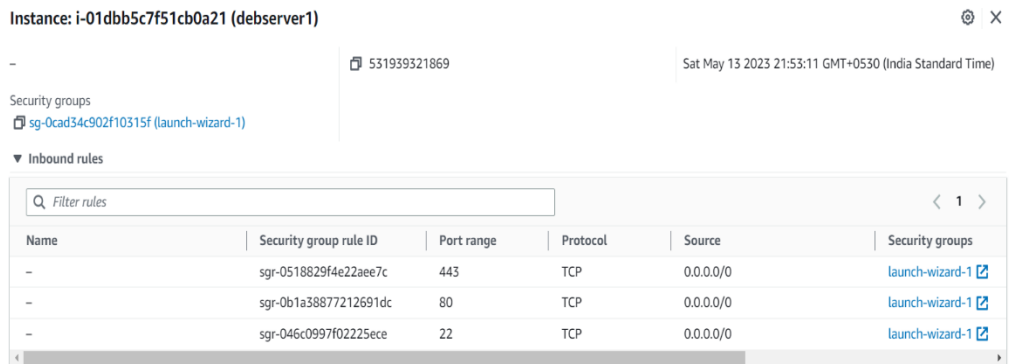
We have to add this port no. to our EC2 instance security group rule otherwise we won't be able to access the website from anywhere.

Step 14: Now go back to your AWS EC2 instances page.

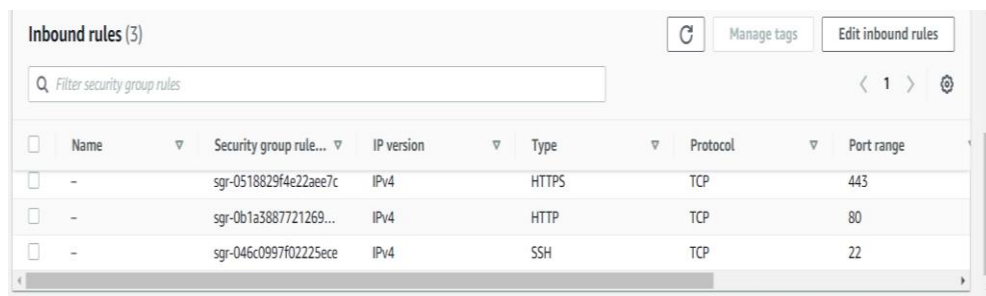
Step 15: Click on the instance that is being used.

Step 16: Scroll down until we find a section bar where by default the details option is selected. Select the Security option. It is beside the Details option.

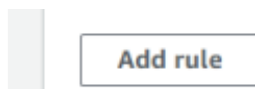
Step 17: Then click on the security groups link under security groups.



Step 18: Then click on Edit Inbound Rules button.

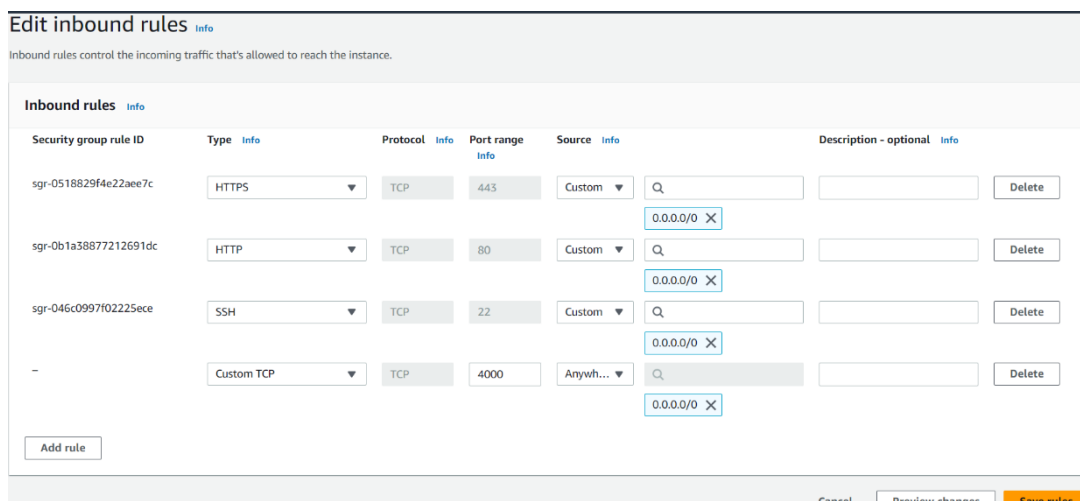


Step 19: Click on the Add Rule button.



Step 20: A new Row will be generated. Let the type remain Custom TCP. Under Port Range write your Port no. you want to open. In this case we have 4000 port no. as we found out earlier in our index.js code. Next in source click on the search box and the first option with value 0.0.0.0/0 should be selected.

Step 21: Now click on the save rules button.



We have successfully added the Port No. to our Inbound rules. Now we can access the our website. But first we need to start our server.

Step 22: We return back to the terminal and type:-

- **node index.js**

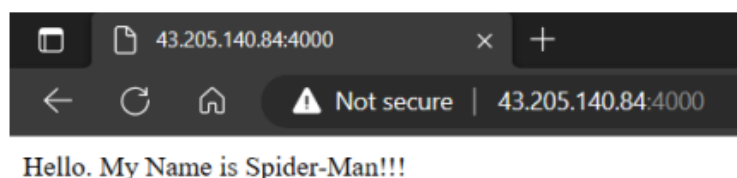
```
ubuntu@ip-172-31-87-119:~/Myrepo1$ node index.js
Started server
```

Our server has started and it is also reflected by the terminal prompt. Now to check we need to open another browser and type in the IPv4 address of our EC2 server to access our website.

Step 23: Now copy the IPv4 address of your EC2 server and paste it in another browser. But before pressing Enter add a colon (:) and then mention the Port No. mentioned in the index.js file. For our case it is 4000. In our case the full address resulted in the one in the below screenshot.

43.205.140.84:4000

Step 24: Now press Enter to load the website.



Step 25: We have successfully deployed our project from GitHub to our EC2 server. Now if we want to change something in our project or file or code we will follow these general steps:

Step 26: Suppose we want to modify the displayed message. So open the index.js file in GitHub.

Step 27: After opening we click on the pen icon on the right corner side of the code viewer.

```
1  const express = require('express')
2  const app = express()
3
4  app.get('/', function (req, res) {
5    res.send('Hello. My Name is Spider-Man!!!')
6  })
7
8  app.listen(4000, ()=>{
9    console.log("Started server");
10 })
11 )
```

Step 28: We then modify the string passed through the res.send() method.

```

1  const express = require('express')
2  const app = express()
3
4  app.get('/', function (req, res) {
5      res.send('Hello. My Name is Spider-Man!!! Nice to meet You!!!')
6  })
7
8  app.listen(4000, ()=>{
9      console.log("Started server");
10 })
11 )

```

Step 29: Now after editing, scroll-down and click the Commit changes button.

Step 30: Our changes have finally been committed in our Repository in GitHub.

However our changes have not been reflected in our Remote Server. For that we have to 'pull' the new updated files into the repository directory in our Remote Server.

Step 31: We have to Close our already running Server. For this, go to the Terminal which we have been working with. Then press <CTRL + C> shortcut to close the server.

```

Started server
^C
ubuntu@ip-172-31-38-127:~/myRepoV1$

```

Our server has been stopped.

Step 32: Now type:-

- **git pull**
 (Enter the username when asked)
 (Enter your account Token as your Password when asked for password)
 (Right click once to paste, then press Enter)

```

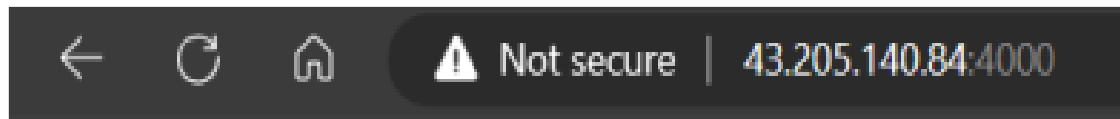
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 685 bytes | 228.00 KiB/s, done.
From https://github.com/DebrupPramanik/myRepoV1
   c367e05..188da63  master    -> origin/master
Updating c367e05..188da63
Fast-forward
 index.js | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
ubuntu@ip-172-31-38-127:~/myRepoV1$

```

Now we have to restart the server.

- **node index.js**
 (We restarted the server)

Step 33: We now have to Refresh our browser where we have our website open.



Hello. My Name is Spider-Man!!! Nice to meet You!!!

The changes have been successfully reflected. This is how we have to edit and update our project if required.

We have successfully completed our task of Deploying our project from GitHub to our EC2 server.