FA16: CMPE-272 Sec 98 - Enterprise Software Platforms
Assignment #1

Requirement:

1. Configure Ansible to deploy webserver, and bring it up a port 80 with a web page that is publically accessible that displays the message: “Hello World”.
2. Include in the Ansible playbook, plays to deploy and un-deploy the resources.

Team Name: Shield

STUDENT NAME	STUDENT ID	GITHUB REPO
Anushri Srinath Aithal	012506897	https://github.com/shriaithal/Shield_dev.git
Anuradha Rajashekar	012409956	https://github.com/AnuradhaIyer/Ansible-Play-book
Nidhi Jamar	010070593	https://github.com/nidhijamar/Shield.git
Ashwini Shankar Narayan	012506910	https://github.com/Ashwinisnv/Shield_Ashwini.git

Team Github Repository: https://github.com/AnuradhaIyer/Shield.git

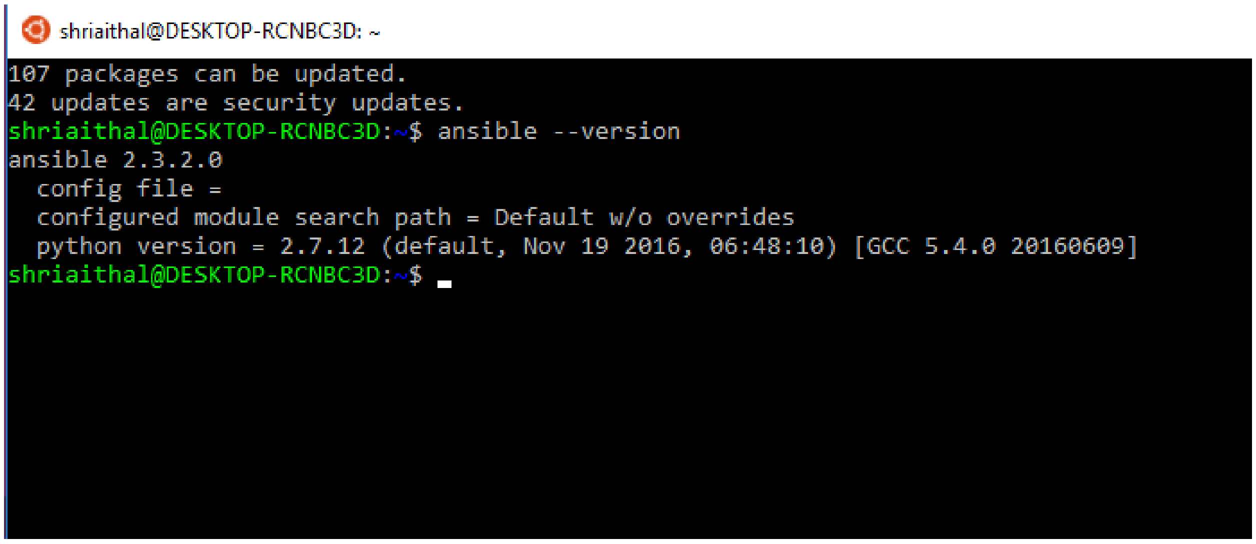
This is a help guide to
1. Install Ansible
2. Deploy a web server on EC2
3. Un-deploy the webserver from EC2

Install Ansible:

Ansible is an automation platform that helps in configuration management, application deployment and automating tasks.
To install Ansible on Ubuntu machine
1. Open terminal
2. Run the command apt-get install ansible

After Ansible installation is completed you can verify if the ansible folder is created under /etc

To verify if ansible is installed run the below command
ansible --version



```
shriaithal@DESKTOP-RCNBC3D: ~
107 packages can be updated.
42 updates are security updates.
shriaithal@DESKTOP-RCNBC3D:~$ ansible --version
ansible 2.3.2.0
  config file =
  configured module search path = Default w/o overrides
  python version = 2.7.12 (default, Nov 19 2016, 06:48:10) [GCC 5.4.0 20160609]
shriaithal@DESKTOP-RCNBC3D:~$
```

All the servers that need to be managed with Ansible need to be present in /etc/ansible/hosts with the below format
[Name]
Ansible_SSH_Server_IP

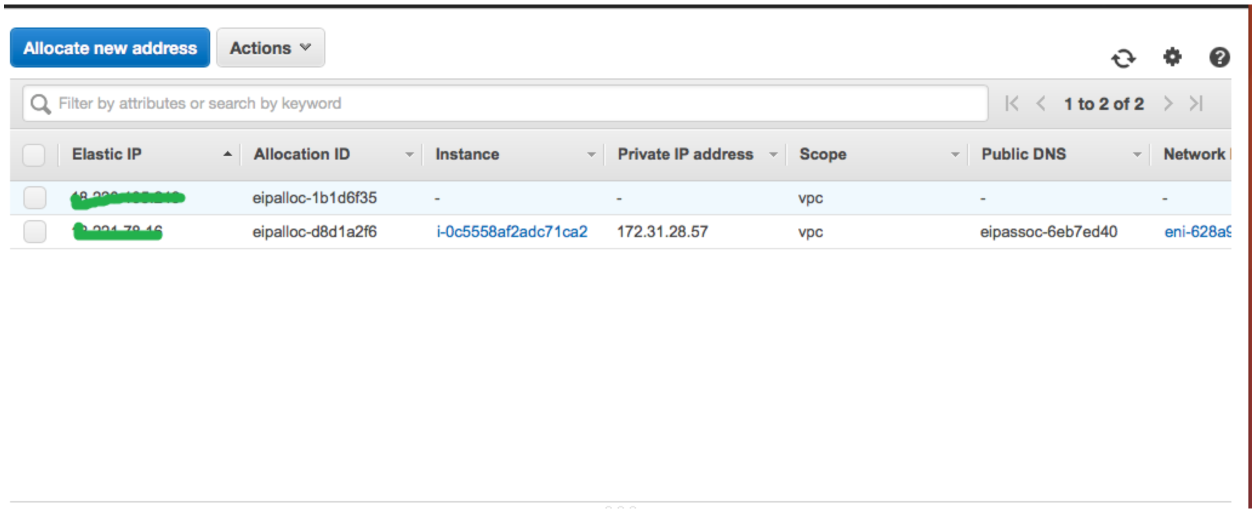
<u>Create an EC2 Instance and enable port 80 for HTTP requests:</u>

EC 2 installation steps:

1. In order to make the web page accessible publicly, we need to host the web server on a cloud instance. Here we are deploying Apache2 on Amazon EC2 instance.
2. Go to https://aws.amazon.com/ec2/ create free tier account with your username and password.
3. In the dropdown menu, we can find all the services provided by AWS. Click on EC2 and launch an instance.
4. Choose the appropriate Operating System to boot EC2 instance with. We installed Ubuntu 16.04 on our EC2 instance.
5. Select the free tier version available for Ubuntu and review the instance and click launch.
6. As the EC2 instance IP is generated using DHCP, each time we stop the instance the IP address is returned to the pool of addresses and can be allocated to another resource. Hence, it changes each time we reboot the EC2 instance. To avoid this, create an Elastic IP(static IP) address and associate this with your EC2 instance. To create a static IP

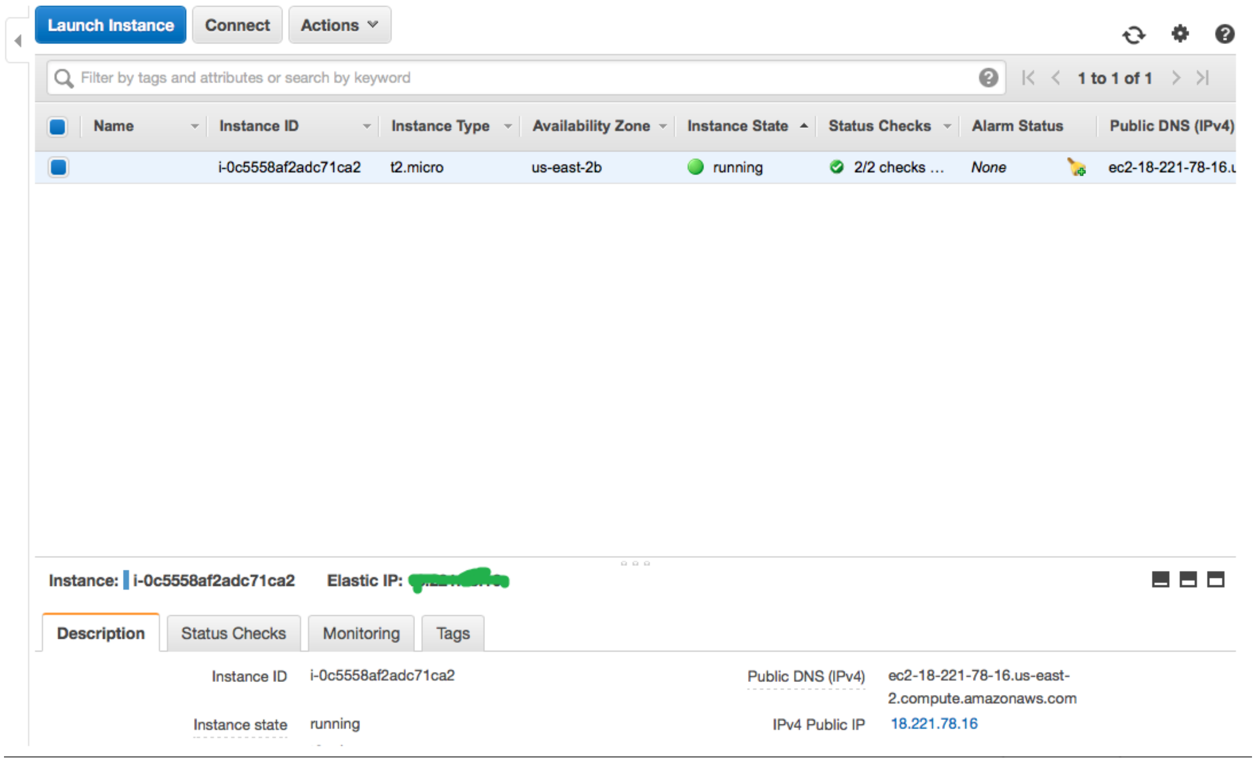
   a. Under Network and Security go to Elastic IPs
   b. Click on Allocate New address
   c. Associate your EC2 instance with this static IP

   Below is the screenshot once you create a new static IP



Elastic IP	Allocation ID	Instance	Private IP address	Scope	Public DNS	Network
[redacted]	eipalloc-1b1d6f35	-	-	vpc	-	-
[redacted]	eipalloc-d8d1a2f6	i-0c5558af2adc71ca2	172.31.28.57	vpc	eipassoc-6eb7ed40	eni-628a9

Below is the screenshot of EC2 dashboard where an EC2 instance is running

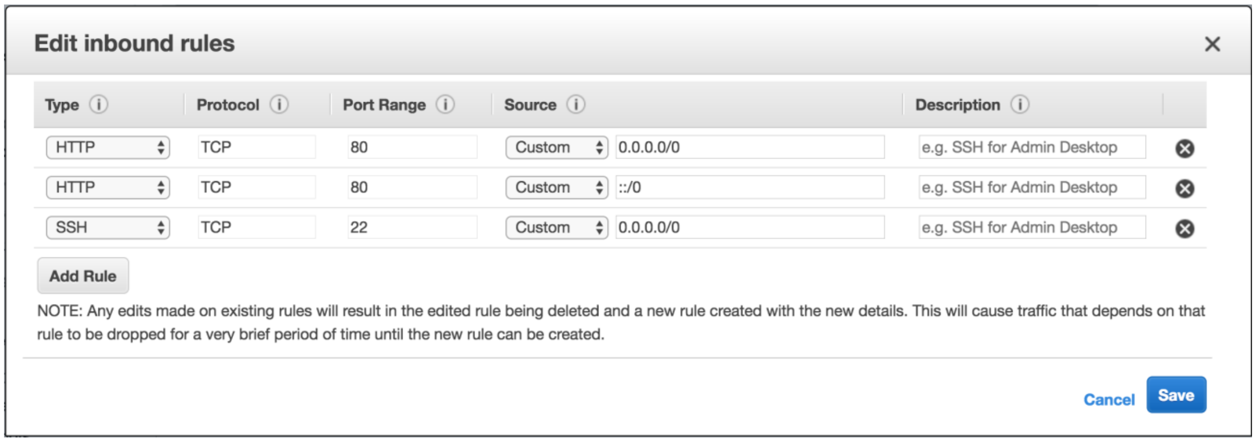


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
	i-0c5558af2adc71ca2	t2.micro	us-east-2b	● running	✓ 2/2 checks ...	None	ec2-18-221-78-16.u

Instance: i-0c5558af2adc71ca2 Elastic IP: [redacted]

Instance ID	i-0c5558af2adc71ca2	Public DNS (IPv4)	ec2-18-221-78-16.us-east-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	18.221.78.16

7. To Enable Port 80 for HTTP requests, below steps are to be followed.On the EC2 instance, go to Security Groups and Edit the Inbound rules to enable port 80 for HTTP requests.


Edit inbound rules

Type	Protocol	Port Range	Source		Description
HTTP	TCP	80	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	::/0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

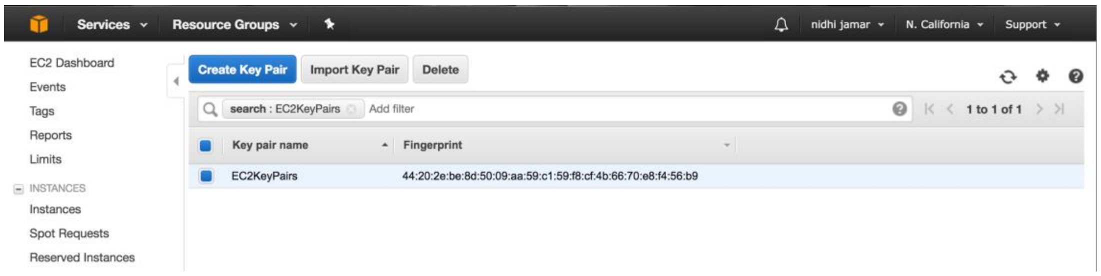
SSH Key Generation:

It is necessary to set up passwordless ssh communication between the EC2 instance and our machine. To login without any key, we need to generate ssh key on our machine which will be shared with EC2 instance. Below are the steps to be followed.

1. Run the below command to verify if we are able to establish connection and login to EC2 via ssh from our respective machine using the EC2 key.
   ssh -i /Users/user/.ssh/EC2KeyPairs.pem ubuntu@ip_address_EC2_instance

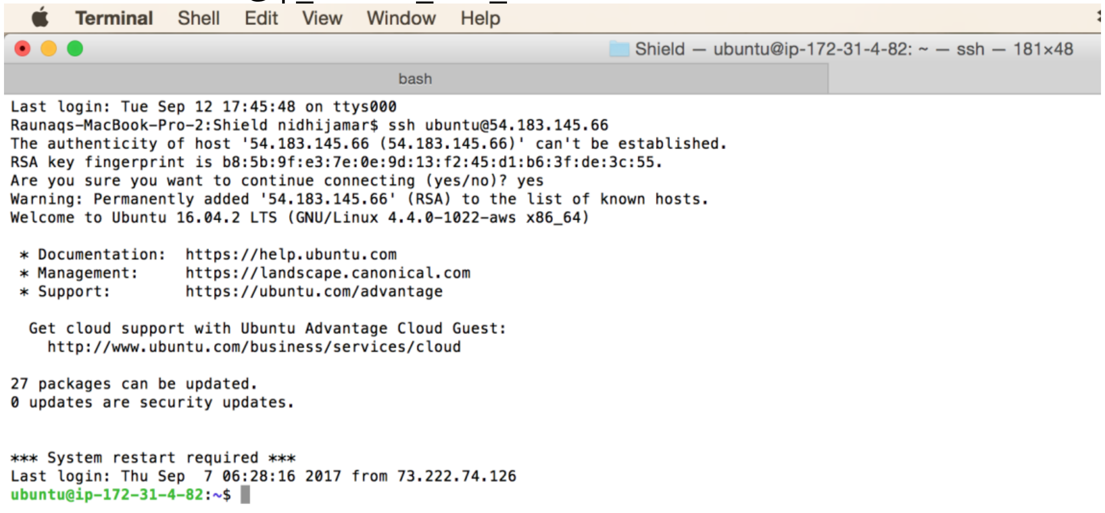
2. On the terminal, run the below command to create an ssh key.
   ssh-keygen
   Following inputs will be required
   a. Enter the file in which the key has to be saved: /Users/user/.ssh/id_rsa.
   b. Enter Passphrase: Leave it blank and press Enter.
   c. Enter same passphrase again: Leave it blank and press Enter.

3. The public key will be saved in the location /Users/user/.ssh/id_rsa.pub.
4. During EC2 instance creation, we need to download and save the key provided by AWS (EC2KeyPairs.pem) which would later be used to access the EC2 instance from our machine.
5. We will have to reduce the permission assigned to the ".pem" file as it should not be accessible by anyone except privileged users. Run the below command
   chmod 400 /Users/user/.ssh/EC2KeyPairs.pem
6. Next open the authorized_keys file on Ubuntu server using the command
   vi /home/ubuntu/.ssh/authorized_keys
   Paste the public key contents from /Users/user/.ssh/id_rsa.pub into this and save it. Alternatively, you can also navigate to AWS and import the public key by following these simple steps
   a. Login to AWS, Navigate to EC2
   b. Navigate to Network and Security, click on Key Pairs
   c. Click on Import Key Pair. Browse to the location /Users/user/.ssh/id_rsa.pub
   d. The contents of the public key are pasted in the textbox. Now click on import.
   e. Key Pair is shown below.



Key pair name	Fingerprint
EC2KeyPairs	44:20:2e:be:8d:50:09:aa:59:c1:59:f8:cf:4b:66:70:e8:f4:56:b9

7. Run the below command to verify if we are able to establish connection and login to EC2 via ssh from our respective machine without any key.
   ssh ubuntu@ip_address_EC2_instance



```
Last login: Tue Sep 12 17:45:48 on ttys000
Raunaqs-MacBook-Pro-2:Shield nidhijamar$ ssh ubuntu@54.183.145.66
The authenticity of host '54.183.145.66 (54.183.145.66)' can't be established.
RSA key fingerprint is b8:5b:9f:e3:7e:0e:9d:13:f2:45:d1:b6:3f:de:3c:55.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.183.145.66' (RSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

27 packages can be updated.
0 updates are security updates.


*** System restart required ***
Last login: Thu Sep  7 06:28:16 2017 from 73.222.74.126
ubuntu@ip-172-31-4-82:~$ 
```

Create Playbook to deploy Apache Webserver and bring up requested web page

Writing Hosts file:

1. After installing Ansible, we need to create a “hosts” file to tell Ansible which server/hosts to talk to. In order to do this,
   a. Navigate to the directory /etc/ansible: cd /etc/ansible
   b. Then open the hosts file to add the hosts IP addresses: vi hosts
2. Follow the below steps
   a. Add the following lines into hosts file.
      [apache]
      localhost ansible_connection=local
      secondary_server_ip ansible_ssh_user=username
      ansible_ssh_private_key_file=EC2_private_key
   b. In the above code, replace the following,
      secondary_server_ip: Public IP address of EC2 instance

username: Ubuntu(if Ubuntu is installed on EC2. Or any default user as specified by AWS)
EC2_private_key: Key pair generated.

c. In the above script snippet, “apache” specifies a host group. This host group contains one or many IP addresses, “username” to be replaced with SSH username and then the “EC2_private_key” to be replaced with the key pair that you generated in AWS. We have added localhost(127.0.0.1) also into the host group.

3. In order to test if we can successfully connect to servers using ansible, we run the below command to ping:
```
ansible -m ping host_group_name
```
The output looks like below:
```
127.0.0.1 | SUCCESS => {
     "changed": false,
     "ping": "pong"
   }
  secondary_server_ip | SUCCESS => {
     "changed": false,
     "ping": "pong"
          }
```

4. Now that the setup is done, when we try connecting to EC2 we get an error for missing python on the server.Follow the below steps to install python on EC2:
    a. Connect to EC2 server using ssh: ssh ubuntu@EC2_instance_IP_address
    b. apt-get update
       sudo apt-get install build-essential checkinstall
       cd /usr/src
       /usr/src$sudo
       wget https://www.python.org/ftp/python/2.7.13/Python-2.7.13.tgz
    c. sudo tar xzf Python-2.7.13.tgz
       cd Python-2.7.13
       usr/src/Python-2.7.13$ sudo ./configure
       /usr/src/Python-2.7.13$ sudo make altinstall
    d. check version installed
       usr/src/Python-2.7.13$ python2.7 -V
    e. Alternatively you can also install Python using apt
       apt-get install python
    f. Close connection to EC2 machine: /usr/src/Python-2.7.13$ exit

Creating a Playbook:
A playbook is an “yaml” file that specifies the host group and a set of tasks that has to run on the hosts.

1. In the directory /etc/ansible/ create a yaml file(playbook) “apache.yml”. Use the below command to create yaml file:
```
nano apache.yml
```
2. Next

a. Add the following code into the file.
```
---
- hosts: apache
  sudo: yes
```

```
tasks:
        - name: install apache2
          apt: name=apache2 update_cache=yes state=latest

        - name: replace default index.html file
          copy: src=/home/Ashwini/index.html dest=/var/www/html/index.html
```

b. In the above code snippet,

- The "hosts: apache" indicates Ansible that we are making use of apache hosts group.
- "sudo: yes" is needed to make sure we have right user privileges.
- Then follows the list of tasks to be performed.
- Task 1 is to install Apache2 webserver and this is done using the module "apt" which installs the apache2 package. The "update_cache=yes" makes sure that the cache is updated and the "state" is set to latest so that it always installs the lates version of Apache2.
- Task 2 is to replace the contents of index.html file in /var/www/html/ directory with an index.html that has "Hello World" written in it.
- Create an index.html file on local machine and write the below HTML code to display "Hello World" in it.

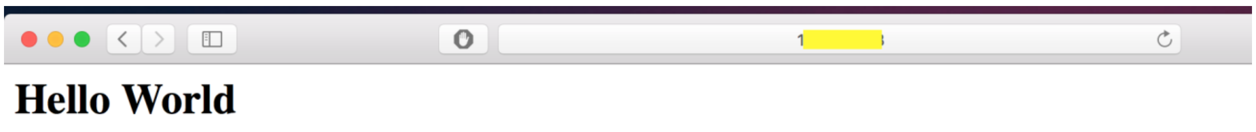
```
<html>
<h1>
<!--- To display Hello World on screen -->
  Hello World
</h1>
</html>
```

3. Now run the playbook with below command to see if the connection to host(server mentioned in the ansible hosts file) is successful.

```
        ansible –playbook  apache.yml –ask-sudo-pass
```

4. Then, enter the IP address of the host in any browser to see if a web page showing 'Hello world' is being displayed on screen.

Below is the screenshot when we try to access the webpage of the webserver post deployment of the resources. The web page is opened on port 80.



Hello World

Un-deploy Apache2 from Ansible

Below are the steps to un-deploy Apache2 from EC2 instance

1. **Host File**

The ansible host file must contain the group of hosts from which we need to un-deploy Apache2. The host file must contain the public IP of the EC2 instance from which we are removing Apache2 instance in the below mentioned format

```
[apache]
secondary_server_ip ansible_ssh_user=username
ansible_ssh_private_key_file=EC2_private_key
```

2. **Playbook**
   a. Create a playbook called apache_uninstall.yml using the command

      ```
      sudo vim apache_uninstall.yml
      ```

   b. Declare the hosts at top to reference the EC2 mentioned in the host file

      ```
      -hosts: apache
      ```

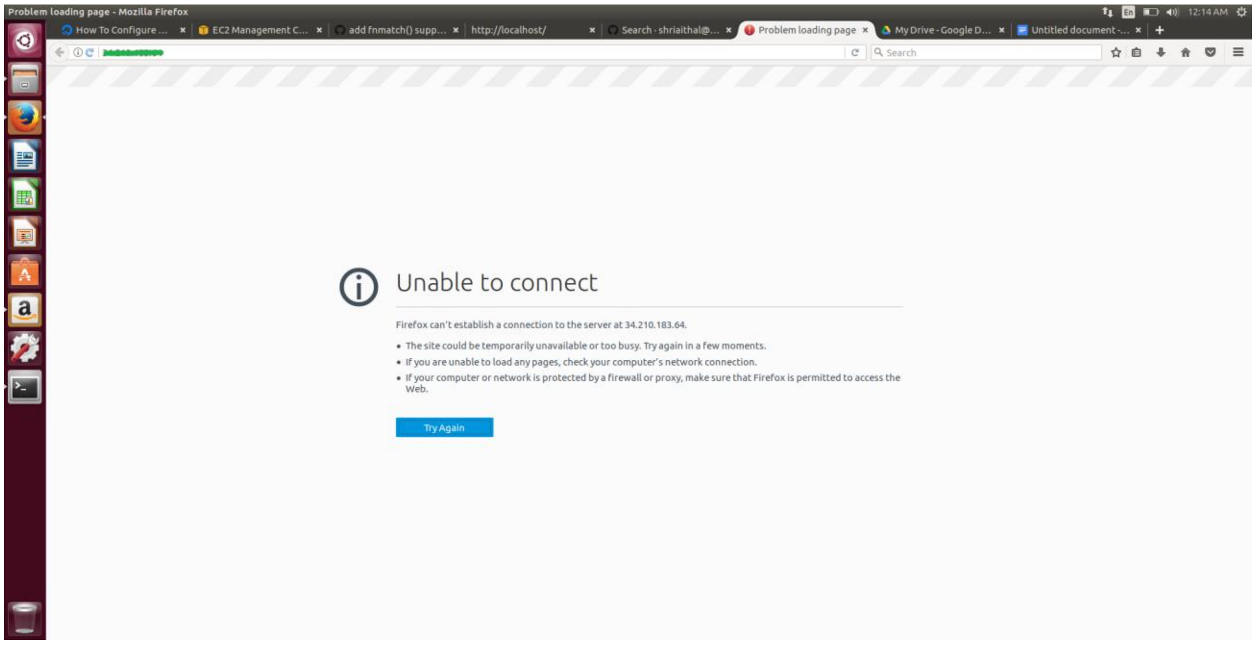
   c. Define the tasks as below to un-deploy apache2

      ```
      - name: uninstall apache2 packages
          apt: >
          pkg={{ item }}
          state=absent
          purge=yes
          force=yes
          autoremove=yes
          with_items:
          - apache2*
          sudo: yes
      ```

      The pkg module referred to all the packages on which the instructions are to be performed. {{item}} here refers to the package name. Here, we need to un-deploy all the apache2 resources from EC2 hence, we mention the items to be apache2* under the with_items module. By setting state=absent we ensure that Apache2 modules are not active. The parameters force=yes, autoremove=yes and purge=yes will force purging of all the configuration files as state=absent. Sudo will be required to ensure that the non-root users have the right privileges.
   d. Run the playbook using the command

      ```
      ansible -playbook apache_uninstall.yml
      ```

Below is the screenshot when we try to access the webserver post un-deployment of the resources



Unable to connect

Firefox can't establish a connection to the server at 34.210.183.64.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.