Testing in DevOps

DOu – Certified Tester in DevOps (CTD)
Exercise Solutions

Setup Code

Expense Manager, System Test, BDD Test, Feature Toggle Code

Step 1 - Getting Sample Code(Expense Manager App, System Test)

- Open the browser and go to https://github.com/login
- Login to your account
- Launch URL https://github.com/umangsaltuniv/verity-devops-ex
- Click Fork at right top section
- “verity-devops-ex” repository will be added on your GitHub account
- Launch URL https://github.com/umangsaltuniv/EMSystemTests-ex
- Click Fork at right top section
- “EMSystemsTests-ex” repository will be added on your GitHub account

Note: verity-devops-ex project has Expense Manager sample app & some unit tests those will run on the app to do unit testing of the app. EMSystemTests-ex project has system test that will run on the app to do system testing of the app.

Step 1 - Getting Sample Code(BDD Test, API Test)

- Launch URL https://github.com/umangsaltuniv/EMSystemTests_BDD
- Click Fork at right top section
- “EMSystemTests_BDD” repository will be added on your GitHub account
- Launch URL https://github.com/umangsaltuniv/EMAPITests-ex
- Click Fork at right top section
- “EMAPITests-ex” repository will be added on your GitHub account

Note: EMSystemTests_BDD project has bdd test that will run on the app to do user acceptance testing. EMAPITests-ex project has api test that will run on the app to do api testing.

Step 2 - Cloning Sample Code(Expense Manager App)

- After forking “verity-devops-ex” repository on GitHub Web, Click “Code” button
- Click “Open with GitHub Desktop”
- Click “Open GitHubDesktop”
- GitHub Desktop UI will be launched
- Browser **C:\DO-United\GitHubRepo** path under “Local path” section
- Click Clone
- GitHub Desktop UI will open the cloned repository and code will be downloaded on local machine under **C:\DO-United\GitHubRepo**

Step 3 - Cloning Sample Code(System Tests)

- After forking “EMSystemTests-ex” repository on GitHub Web, Click “Code” button
- Click “Open with GitHub Desktop”
- Click “Open GitHubDesktop”
- GitHub Desktop UI will be launched
- Browser **C:\DO-United\GitHubRepo** path under “Local path” section
- Click Clone
- GitHub Desktop UI will open the cloned repository and code will be downloaded on local machine under **C:\DO-United\GitHubRepo**

Step 4 - Cloning Sample Code(BDD Tests)

- After forking “EMSystemTests_BDD” repository on GitHub Web, Click “Code” button
- Click “Open with GitHub Desktop”
- Click “Open GitHubDesktop”
- GitHub Desktop UI will be launched
- Browser **C:\DO-United\GitHubRepo** path under “Local path” section
- Click Clone
- GitHub Desktop UI will open the cloned repository and code will be downloaded on local machine under **C:\DO-United\GitHubRepo**

Step 5 - Cloning Sample Code(API Test)

- After forking “EMAPITests-ex” repository on GitHub Web, Click “Code” button
- Click “Open with GitHub Desktop”
- Click “Open GitHubDesktop”
- GitHub Desktop UI will be launched
- Browser **C:\DO-United\GitHubRepo** path under “Local path” section
- Click Clone
- GitHub Desktop UI will open the cloned repository and code will be downloaded on local machine under **C:\DO-United\GitHubRepo**

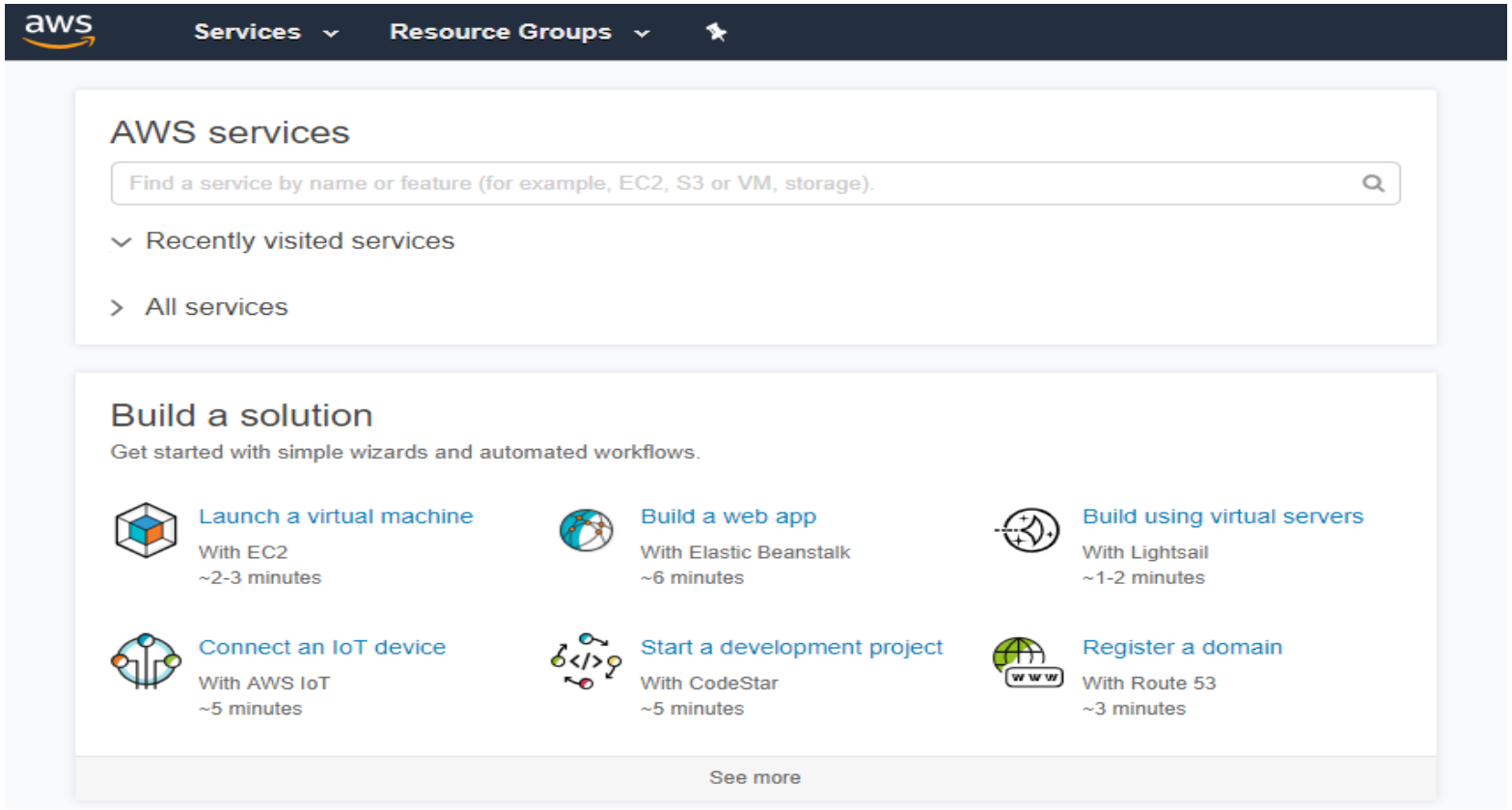
Setup Jenkins on AWS

Setup Jenkins on AWS

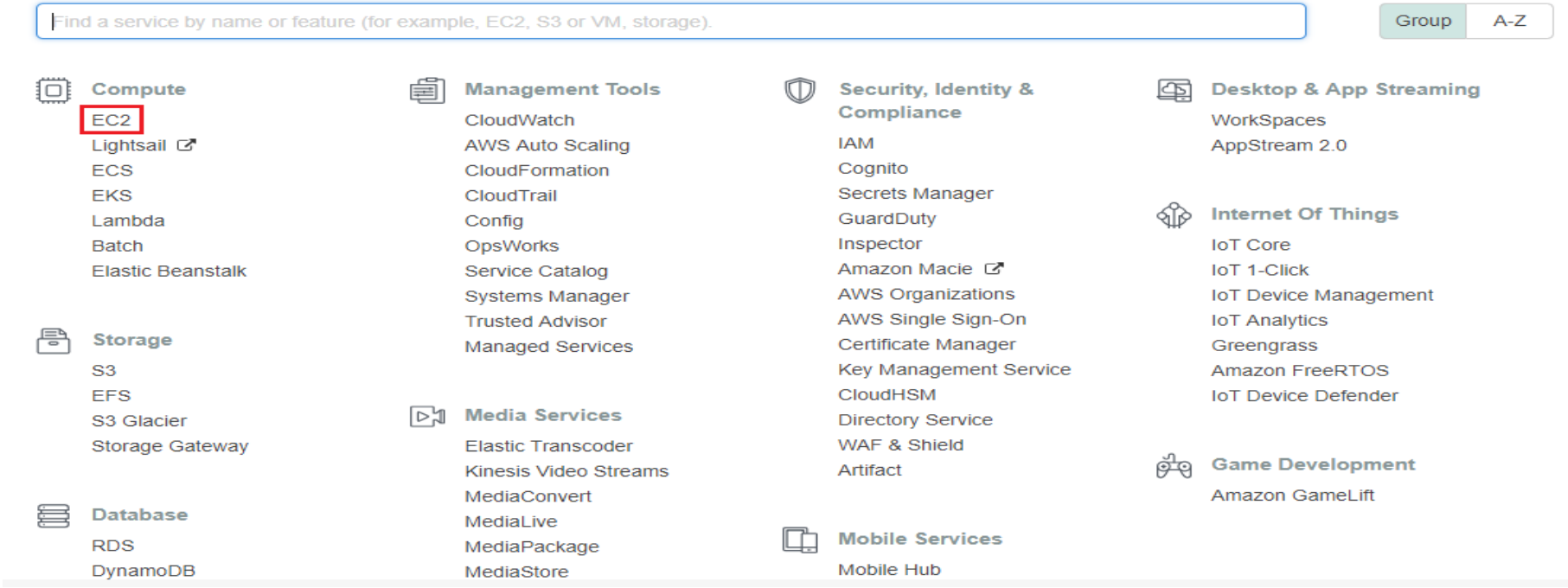
- Setup AWS Ubuntu machine
- Setup Docker
- Setup Jenkins
- Setup Jenkins project
- Setup Webhook in GitHub web & Jenkins project

Setup AWS Ubuntu machine

Login to AWS & Navigate to Home Page

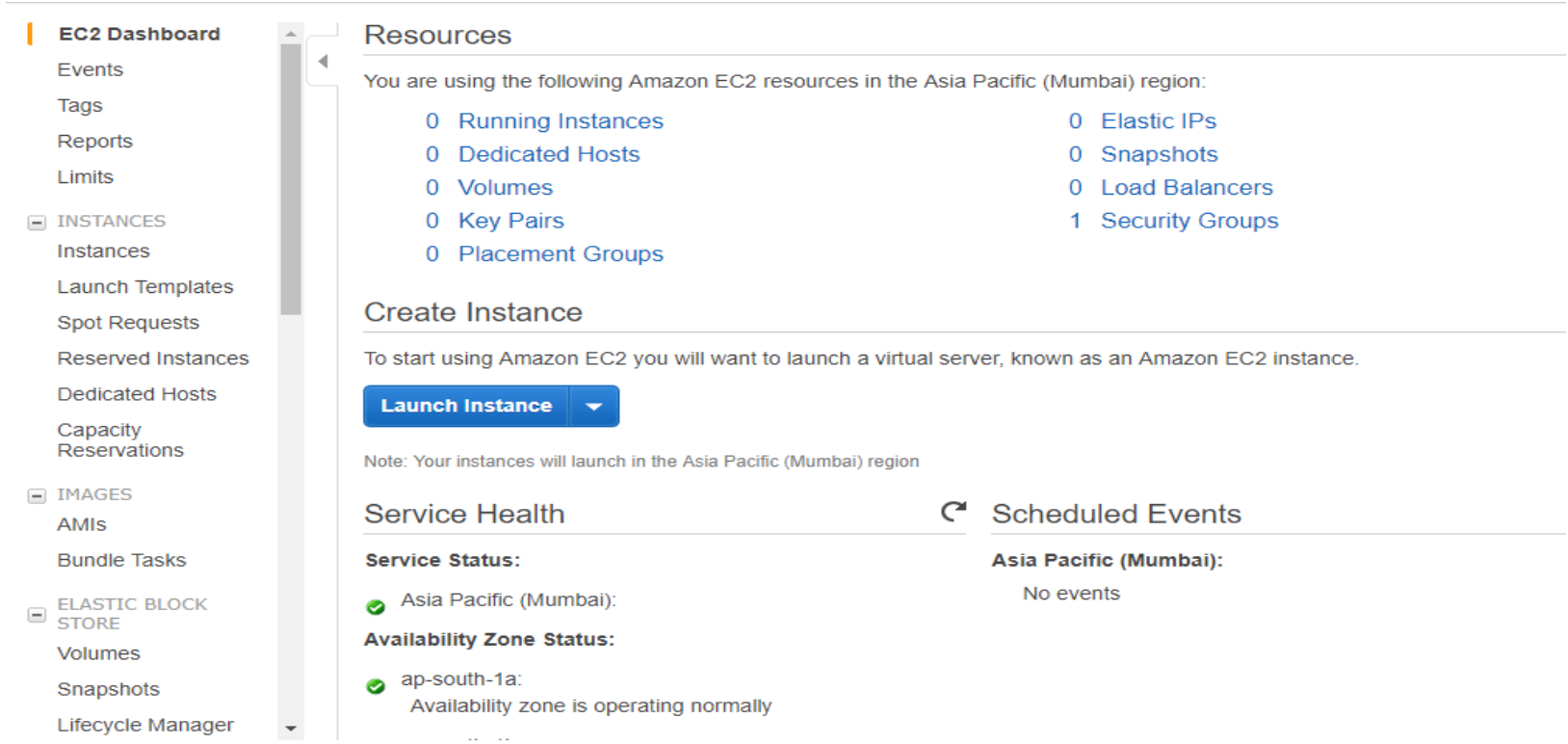


Invoking EC2

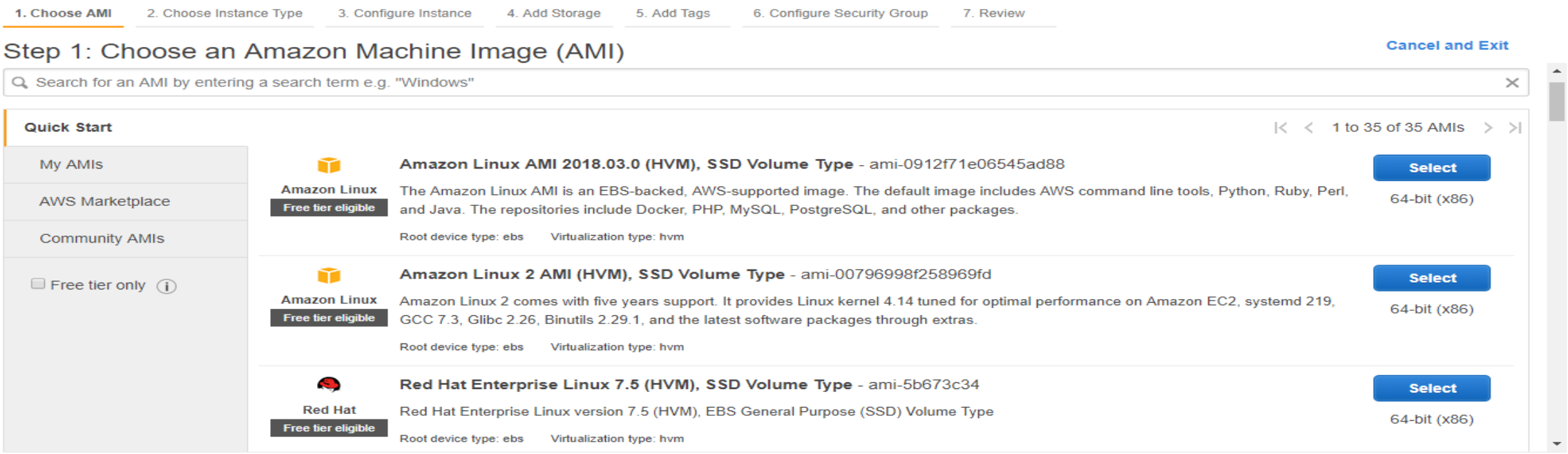


EC2 Dashboard

- Click Launch Instance



Choosing an AMI



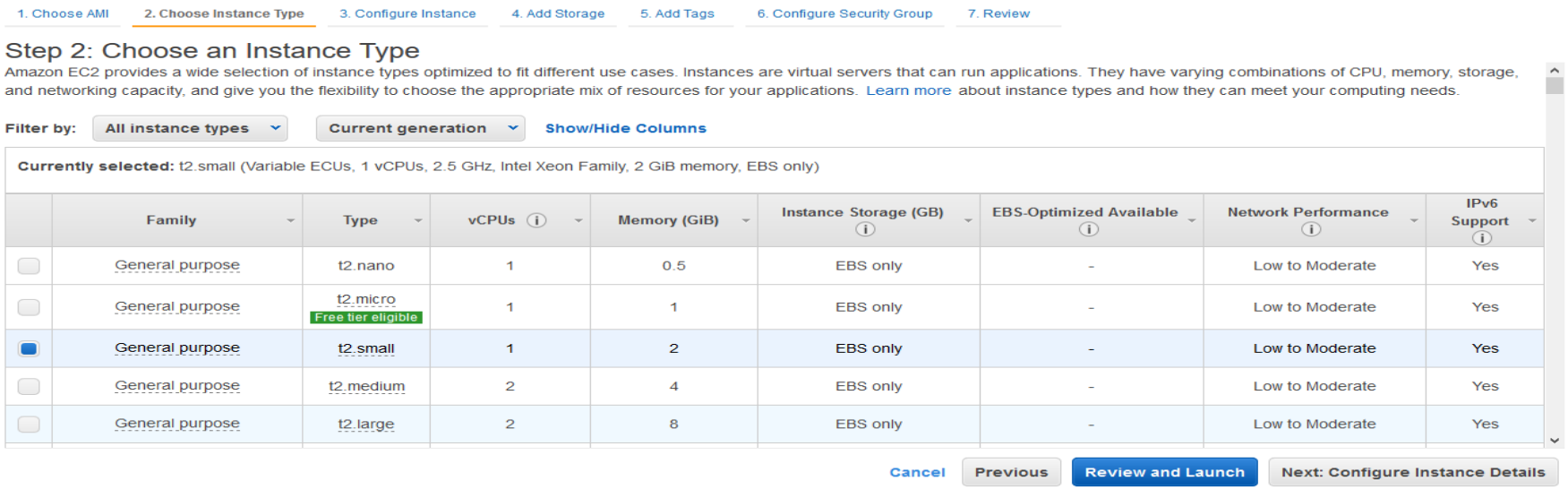
Choosing Ubuntu Image(Ubuntu Server 18.04)

Do not use any other image

- Click Select

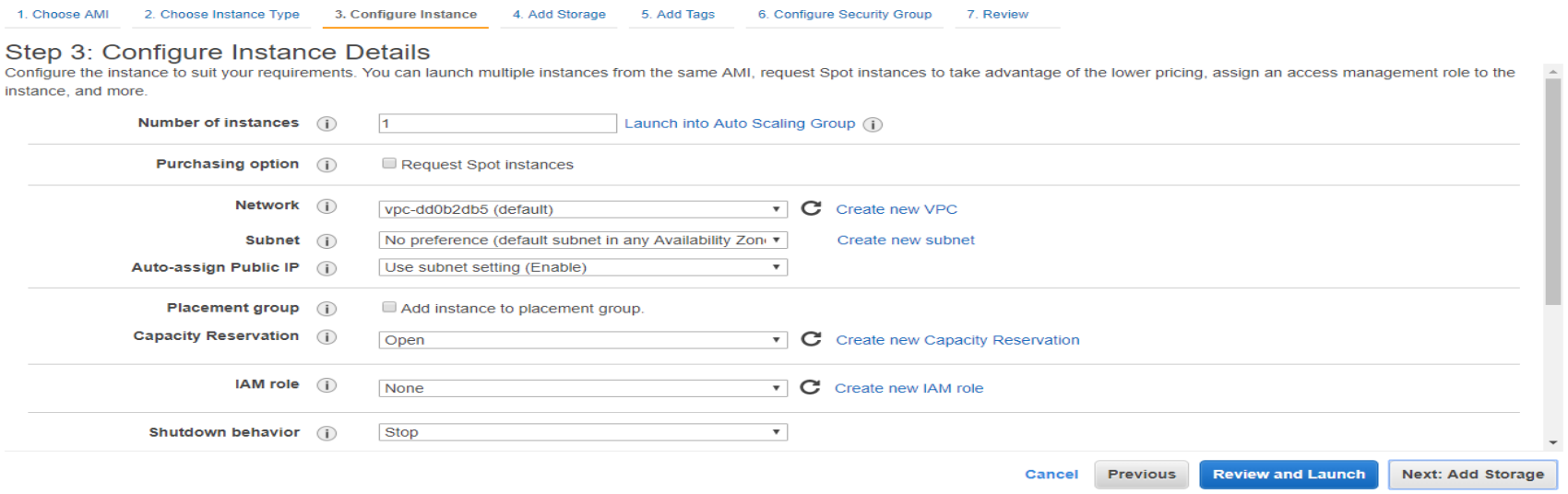
Choosing Instance Type(t2.small)

- Click Next: Configure Instance Details



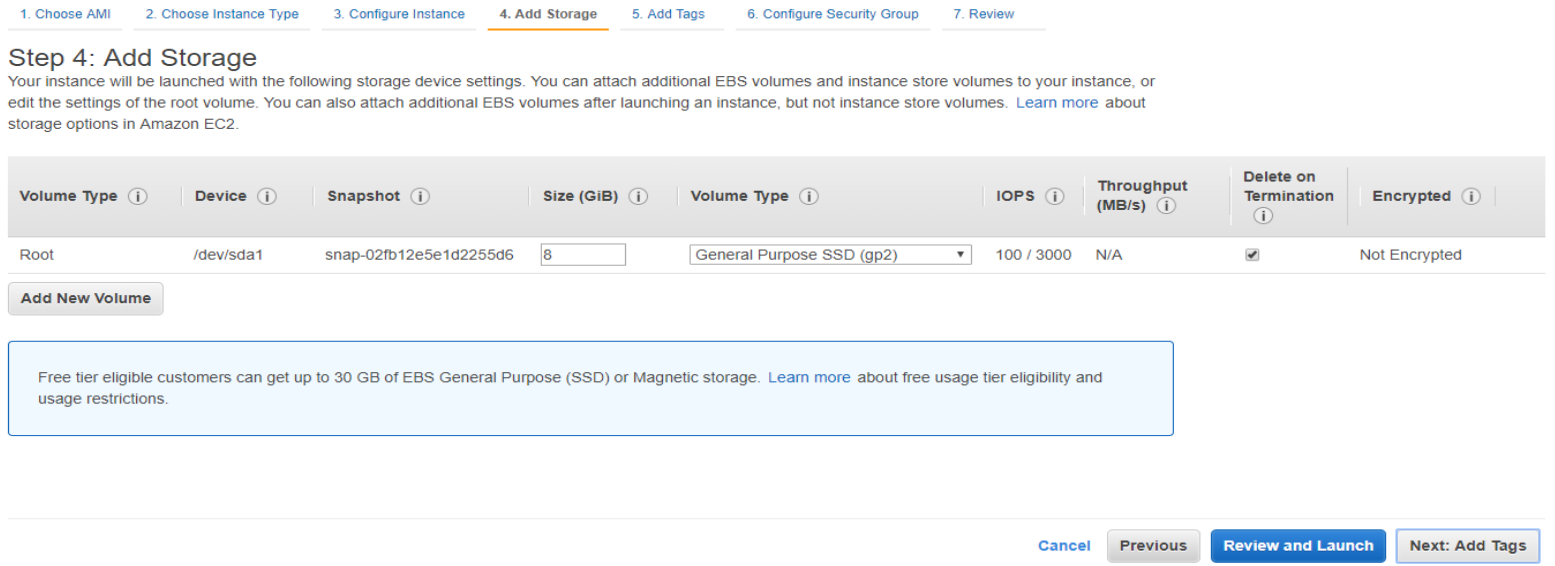
Configuring Instance Details(Keep as default)

- Click Next: Add Storage



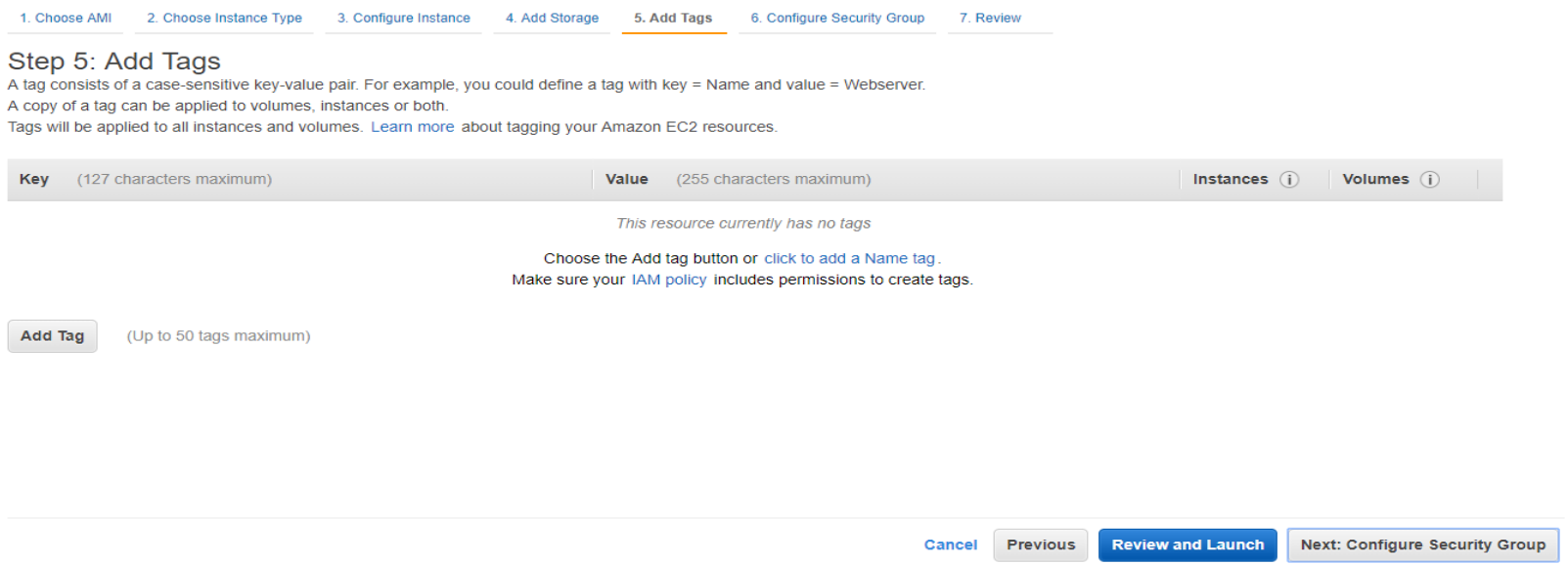
Adding Storage(Keep as default)

- Click Next: Add Tags



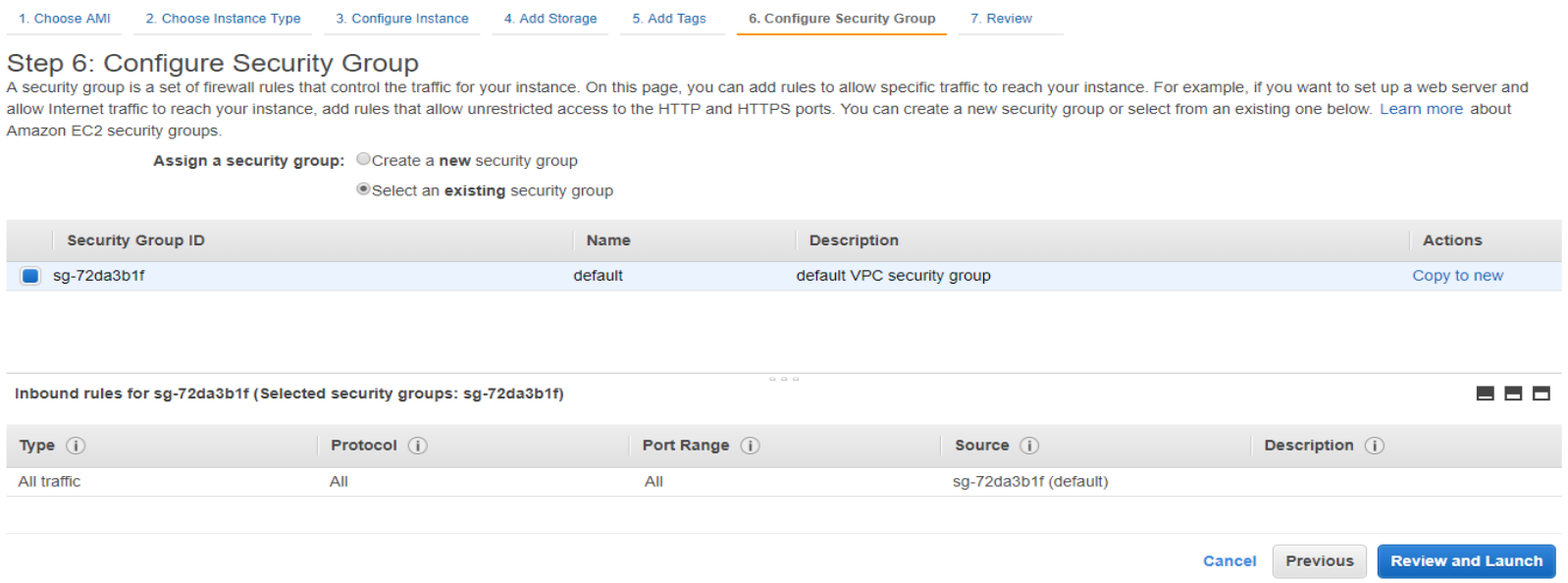
Adding Tags(Keep as default)

- Click Next: Configure Security Group



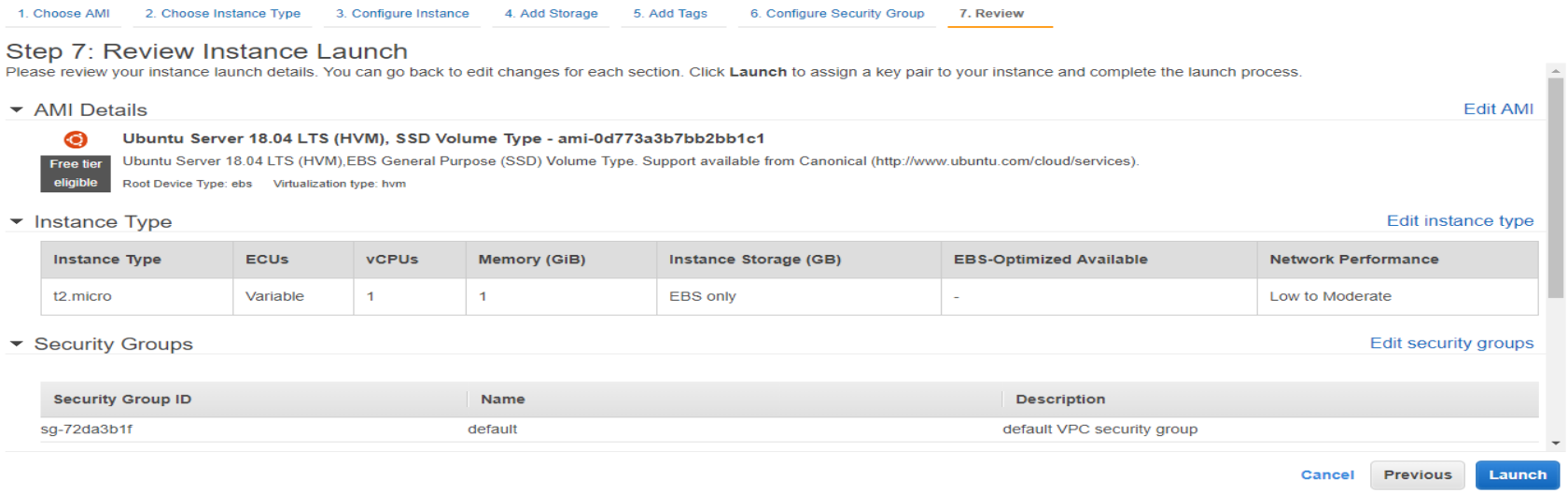
Configuring Security Group(Keep as default)

- Click Review and Launch



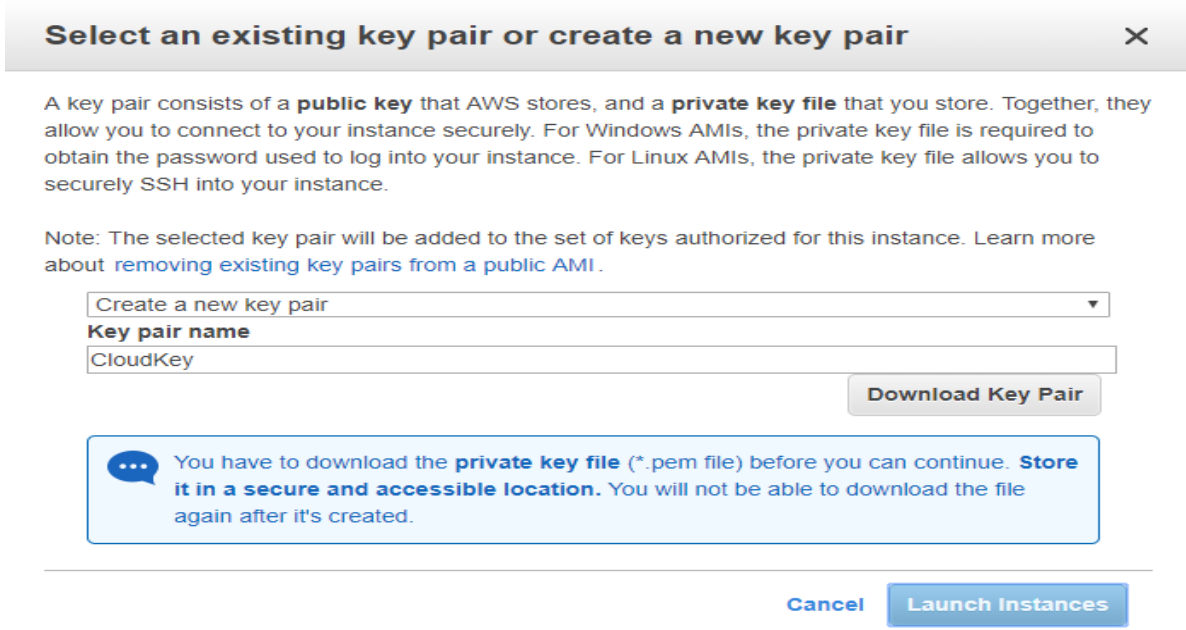
Reviewing

- Click Launch



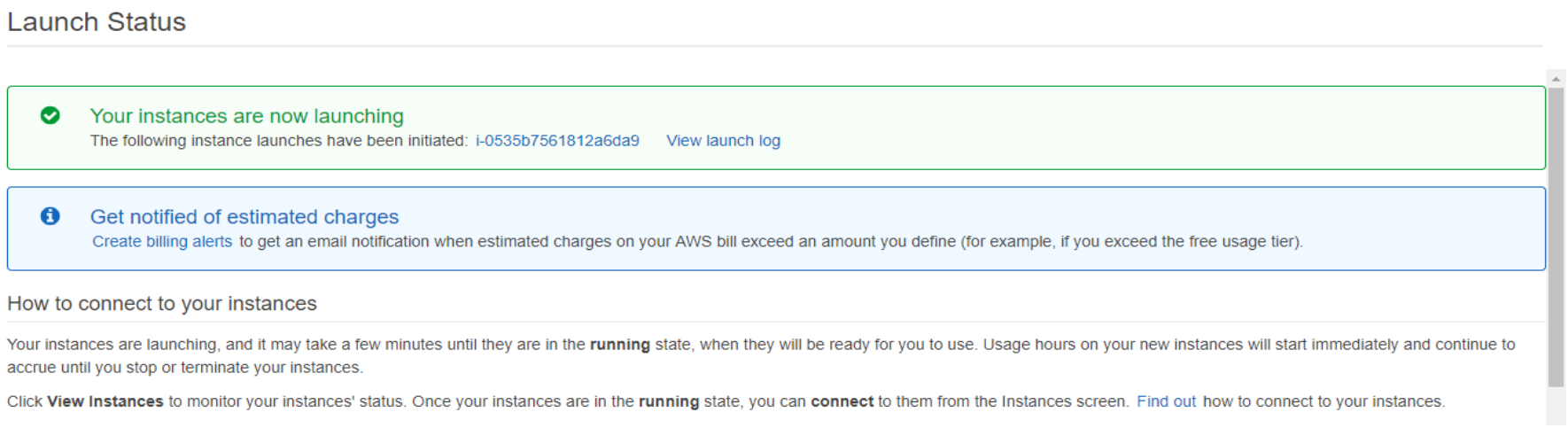
Key-Pair Creation

- Select **Create a new key pair**
- Enter key pair name(e.g. CloudKey)
- Click Download Key Pair
- Key (.pem) will be downloaded



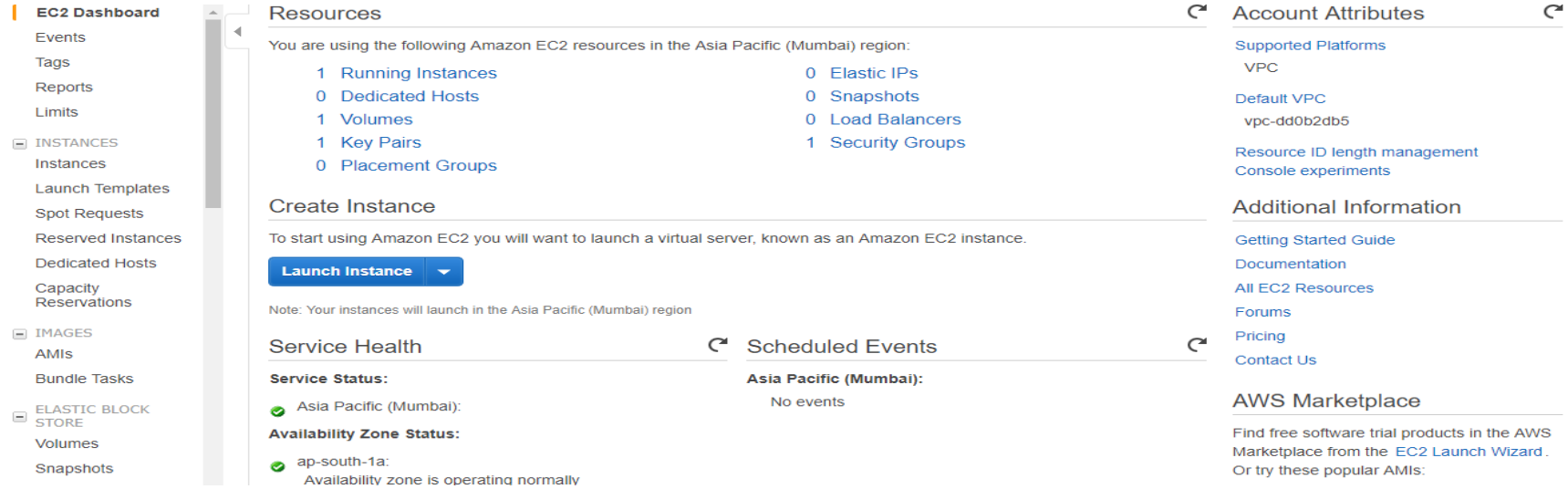
Launch Status

- Click instance id(e.g. i-0535b7…)



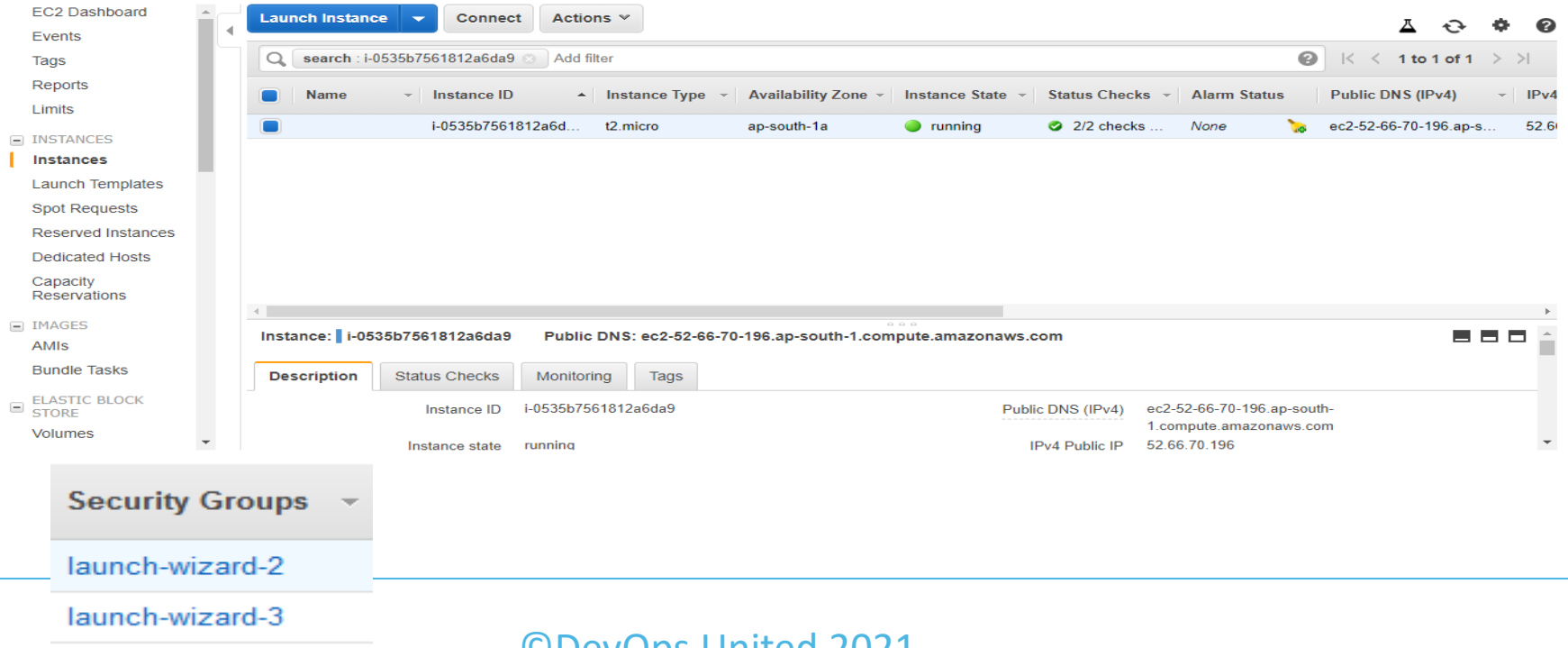
EC2 Dashboard

- Click Running Instances link



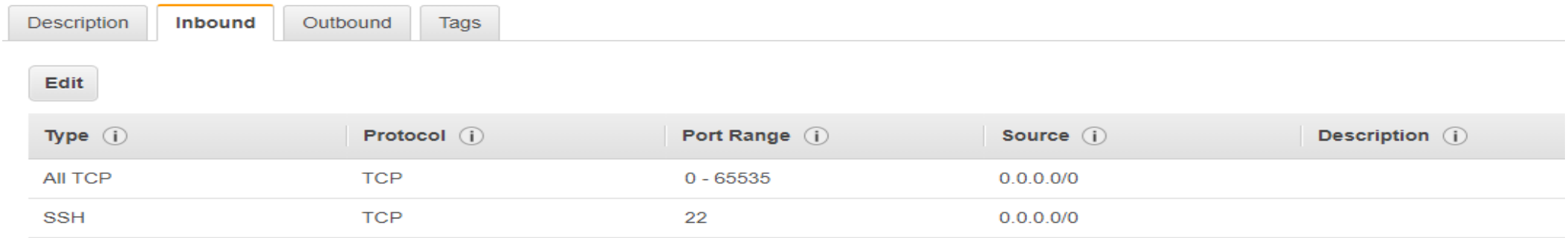
Running Instance

- Drag the page from left to right, you will find Security Groups section
- Click Security Groups link(e.g., launch-wizard-2)
- You will navigate to Inbound section(next slide)



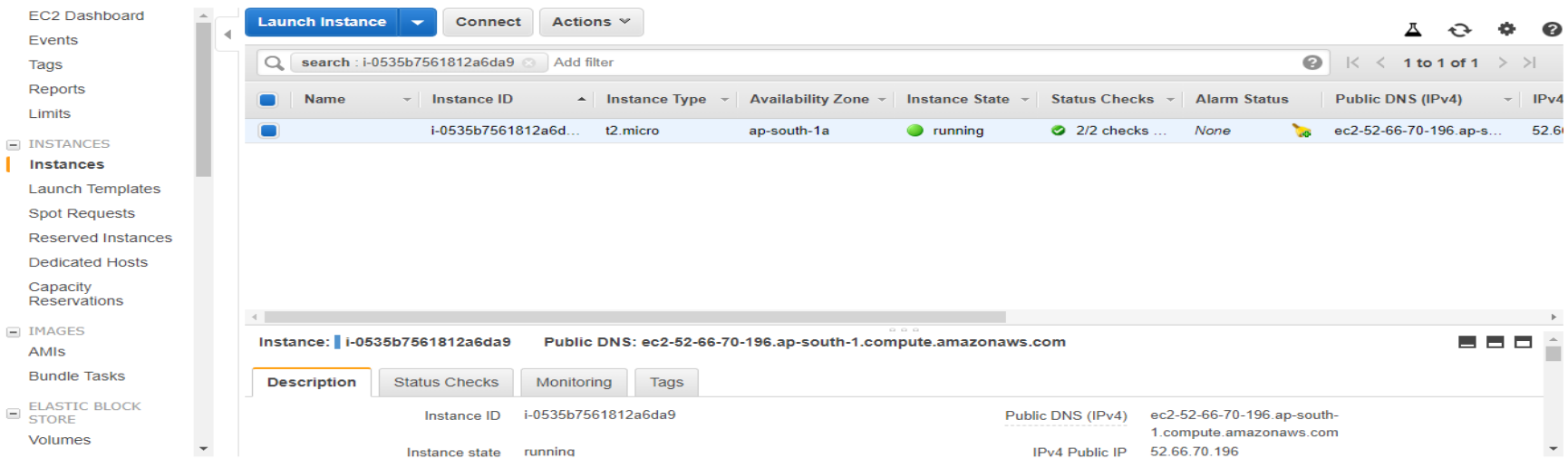
Create an In-bound rule on AWS

- Click Edit
- Add 2 rules(All TCP)
- Note: SSH rule will be shown in the list default
- Save



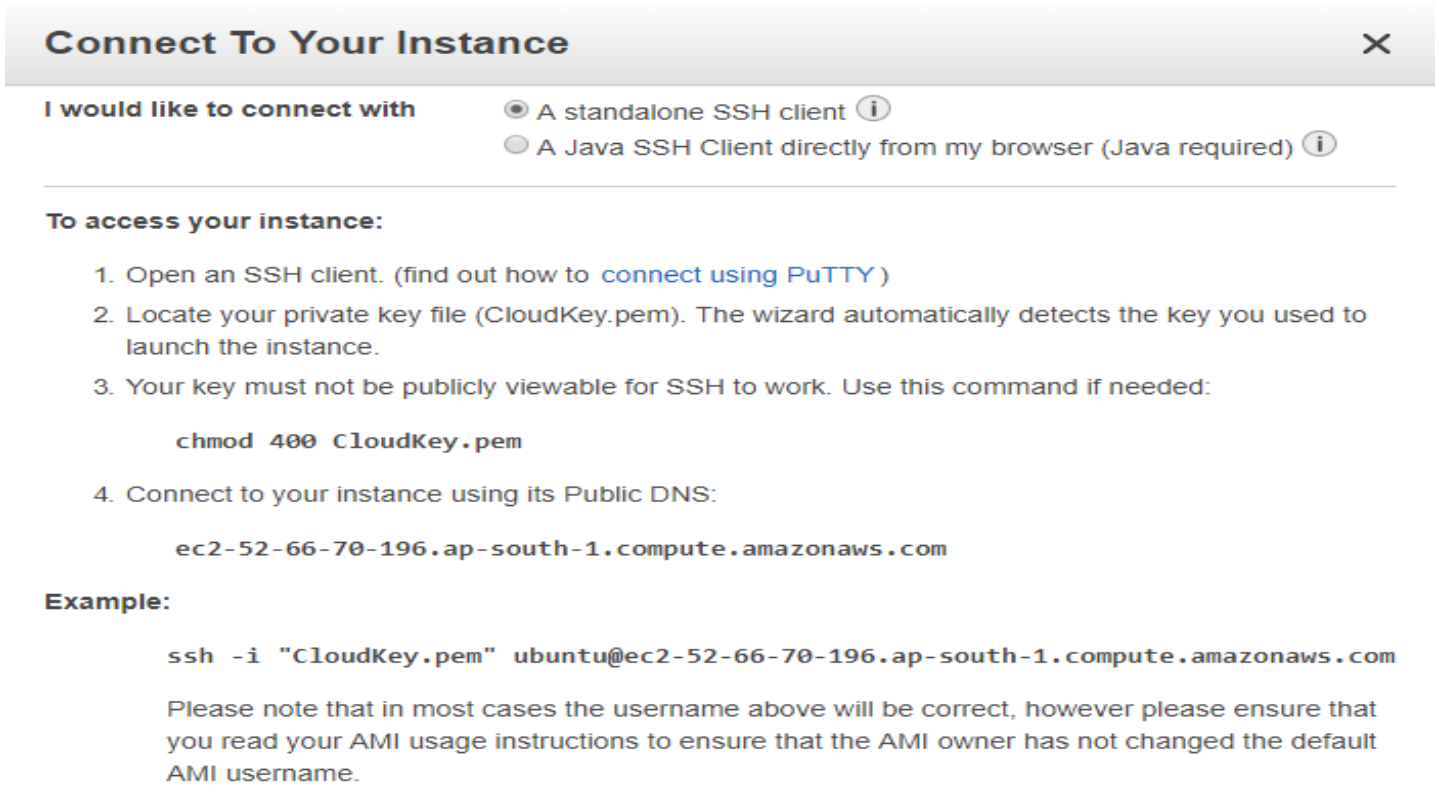
Running Instance

- Come back to running instance page
- Click Connect



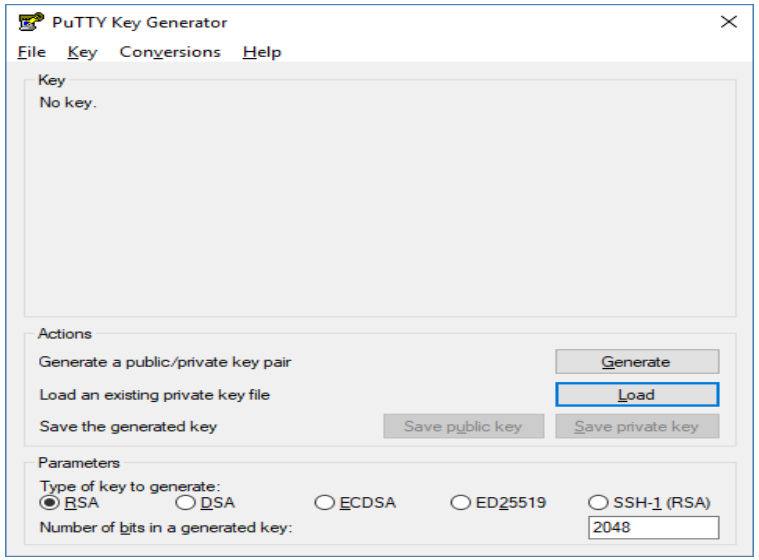
Connecting To Instance

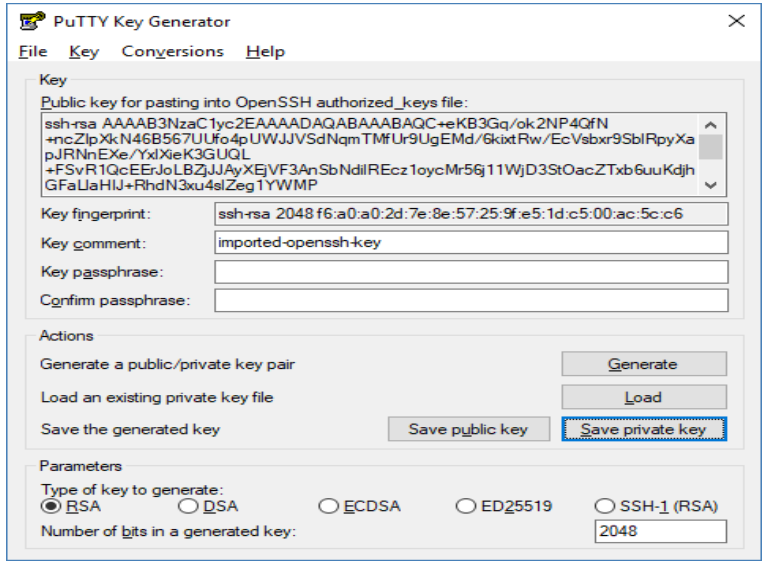
- We will use PuTTy to access AWS instance

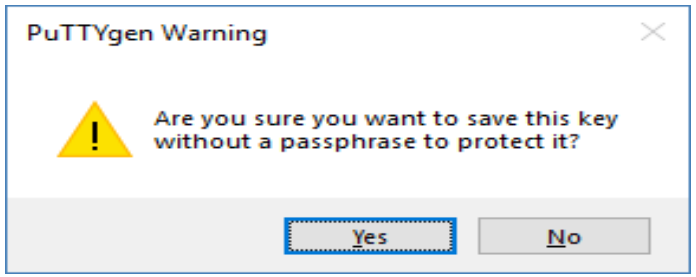


Using Puttygen to Import Key

- Launch puttygen.exe
- Click **Load**
- Browse key(.pem) from machine
- Click **Save private key**
- Click **Yes** on pop-up
- Browse the location
- (e.g. C:\DO-United\Tools\) where you want to save new key(.ppk) & Save it
- Close Putty Key Generator window

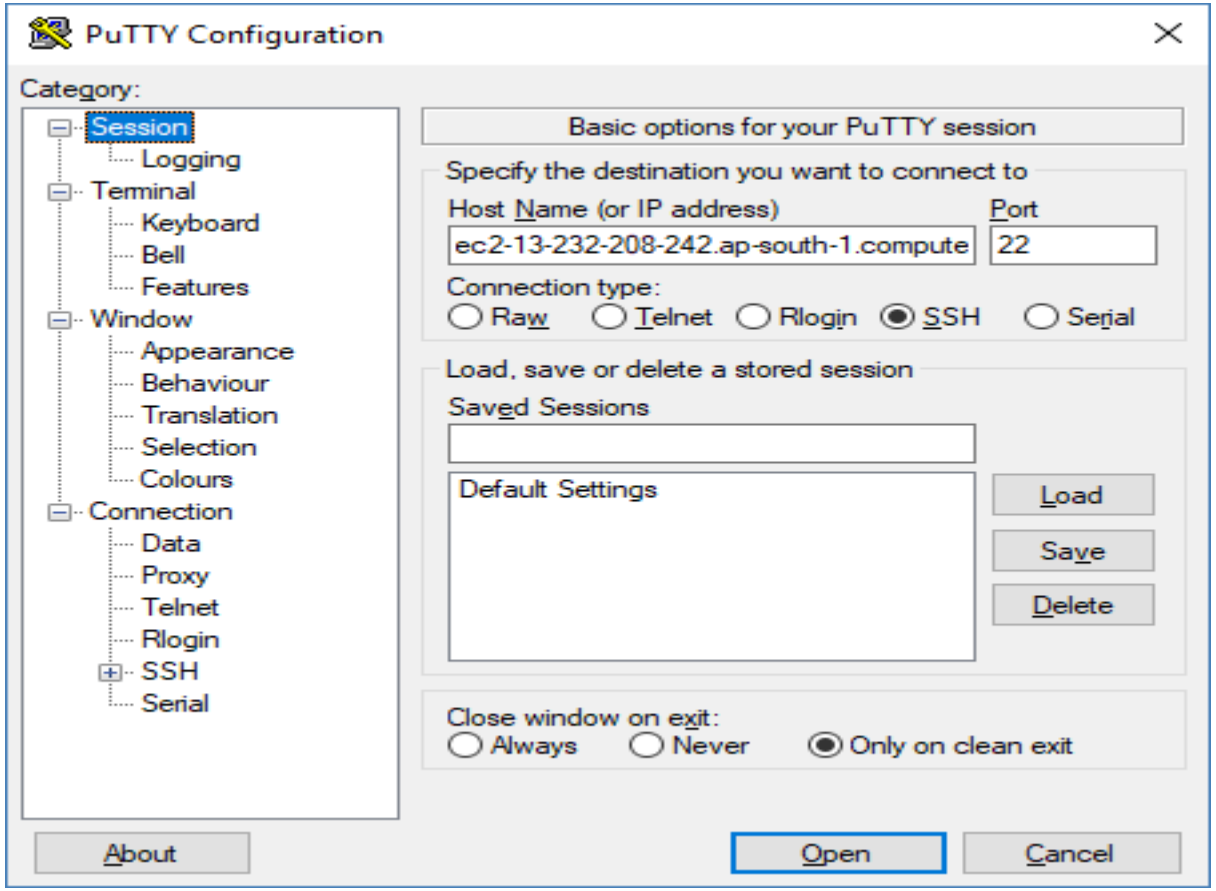






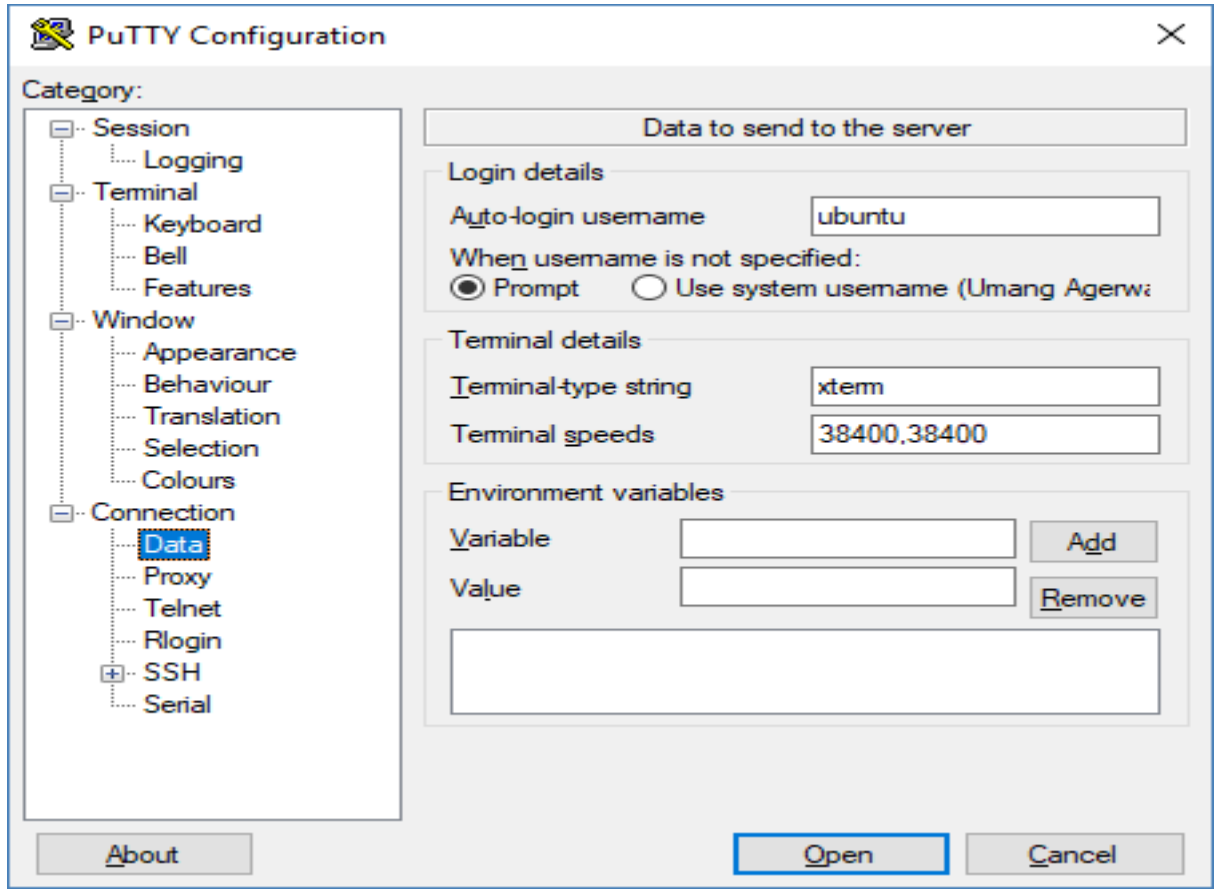
Using Putty to Connect

- Launch putty.exe
- Go to **Session**
- Enter AWS Host Name or IP address(e.g. Ec2-13-233…)
- Note: AWS Host name / IP address will be available on your AWS web account



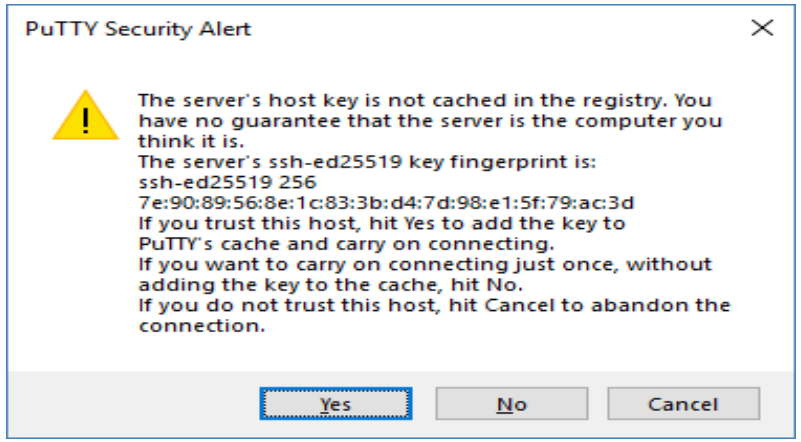
User → ubuntu

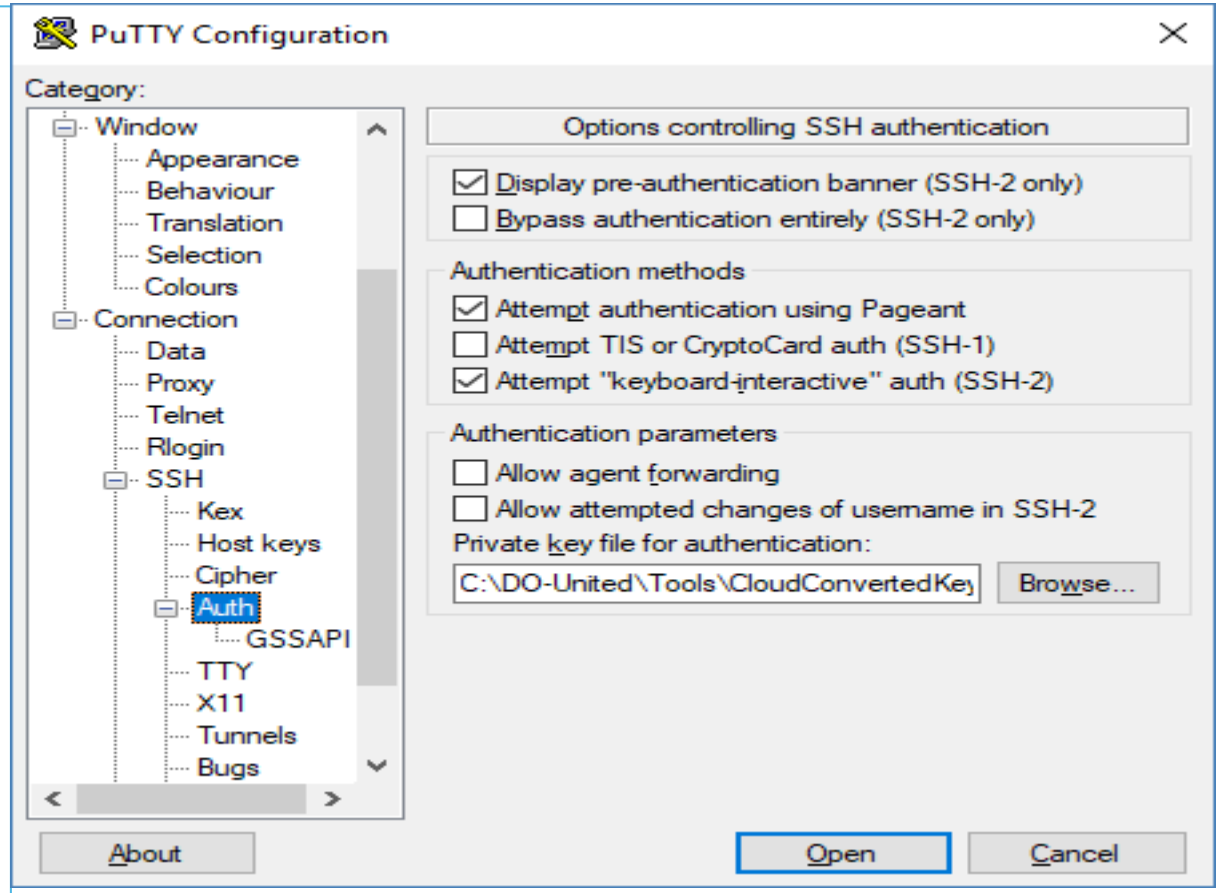
- Go to **Connection**
- Go to Data
- Enter username **ubuntu** in **Auto-login username** section
- Note: For linux machine, enter username **ec2-user**



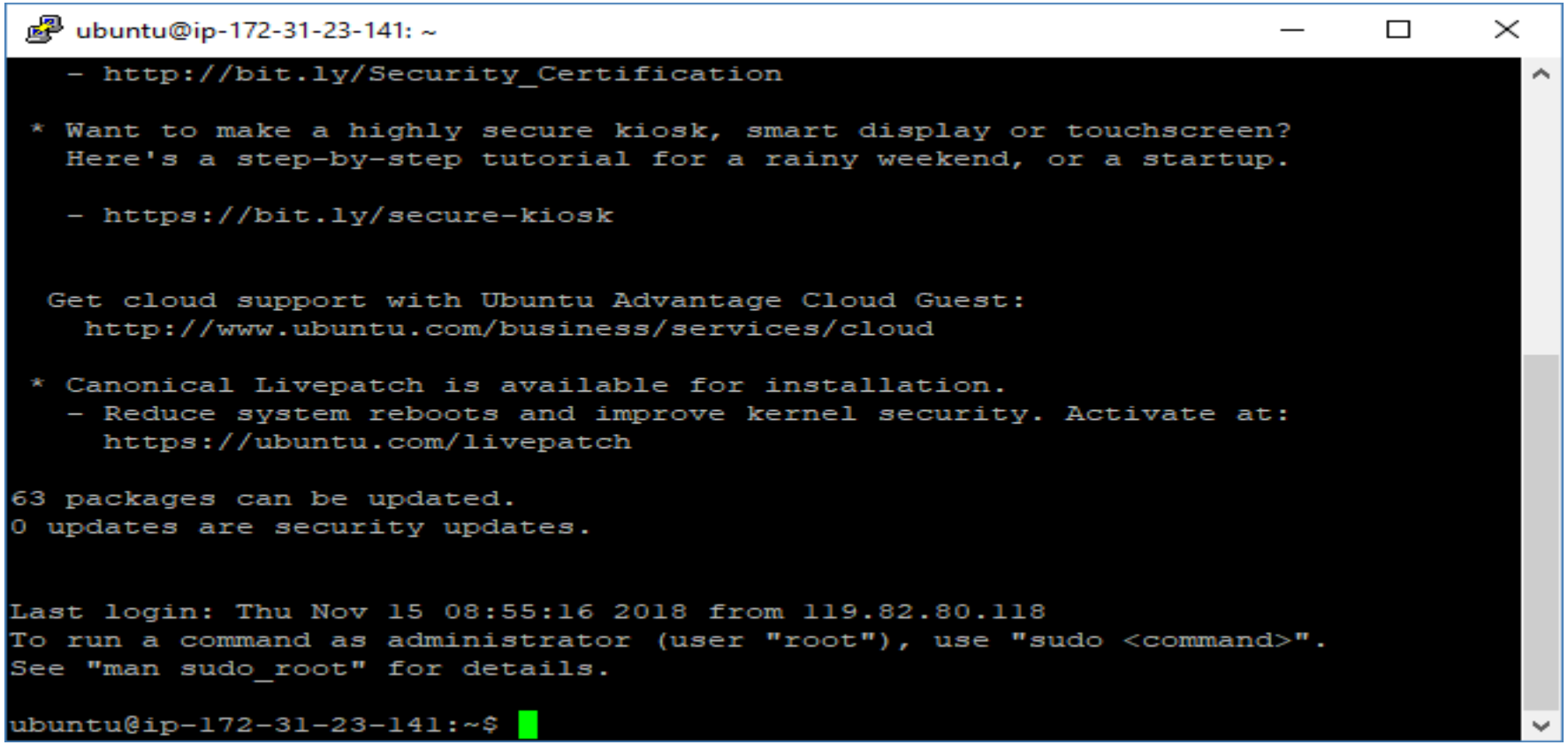
Selecting the Private Key

- Go to **SSH**
- Go to **Auth**
- Browse key(.ppk) from machine
- Click Open
- Click Yes





Connected to Putty



DOu DevOps united

Setup Docker on AWS Ubuntu machine

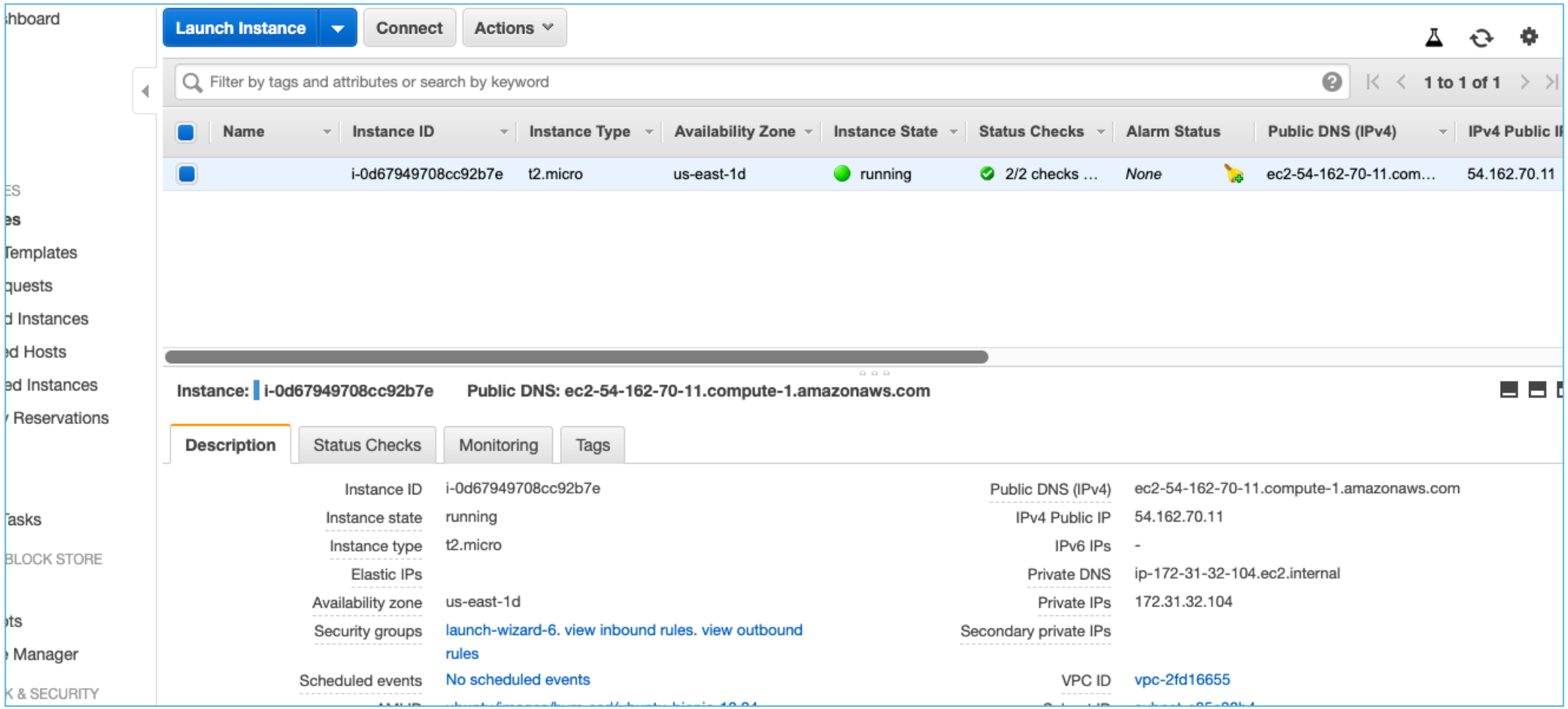
Install Docker on AWS server

- If you create an Ubuntu instance on AWS, run following commands in terminal
  - sudo apt-get update
  - sudo apt-get install docker.io
  - docker --version

Run the container on AWS server

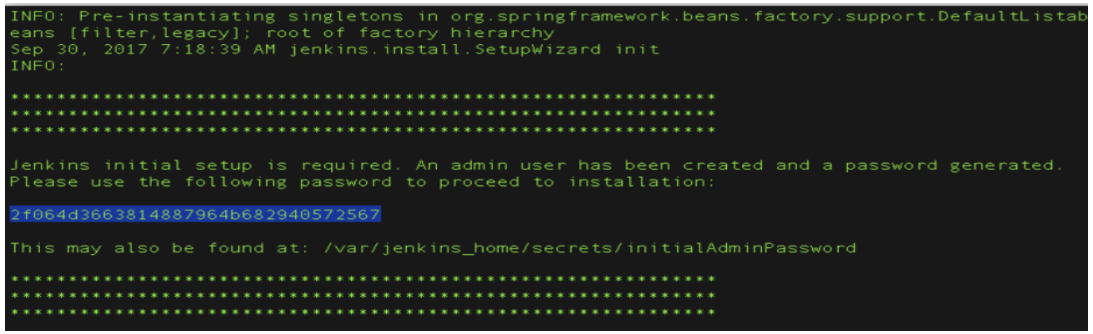
- sudo docker run -u root -p 8089:8089 -p 8080:8080 -v jenkins-data:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v $HOME:/home umangsaltuniv/verity-devops-2-222-3

Capture the Public DNS/IP Address of the AWS Instance



Unlocking Jenkis

- After the 2 sets of asterisks appear in the terminal/command prompt window, browse to http://<<IP Address>>:8080 and wait until the Unlock Jenkins page appears
  - This is the same IP which we captured in the previous slide
- From the terminal/command prompt window, copy the automatically-generated alphanumeric password (between the 2 sets of asterisks)
- On Customize Jenkins page click **Install suggested plugins**
- Click **Continue**
- **Some plugins may be failed so do not worry about that**
- **IF YOU ARE RUNNING ALL THIS SECOND TIME then password will not be available on the screen. See next slide for recovering the password if you did not change the password from Jenkins UI**

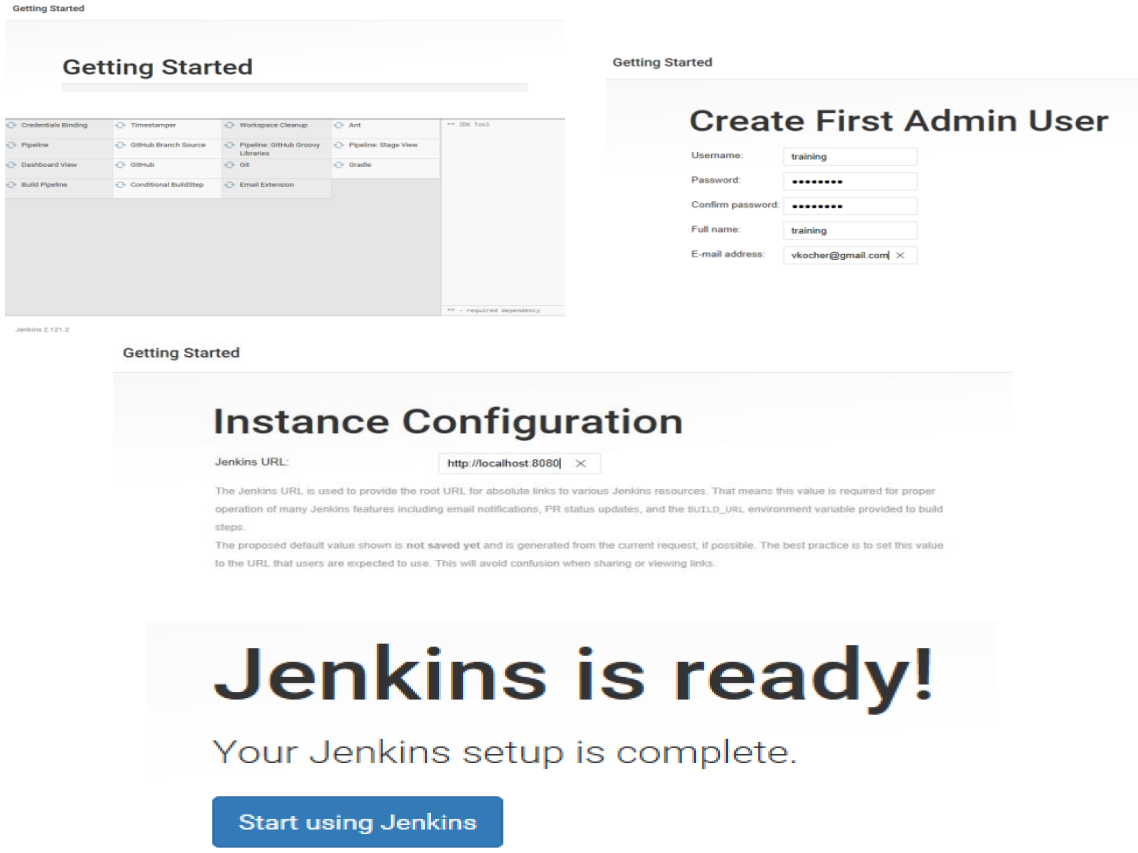


Getting Jenkins Password

- Launch below commands on terminal:
  - sudo docker container ls
  - sudo docker exec -it <container id> /bin/bash
  - cat /var/jenkins_home/secrets/initialAdminPassword
- Copy the password

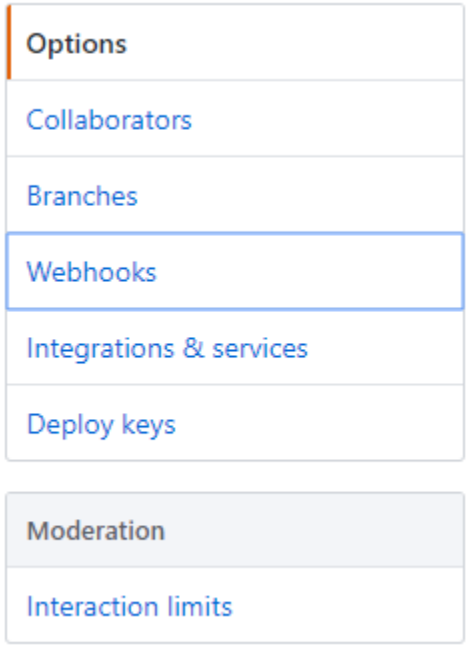
Getting Started

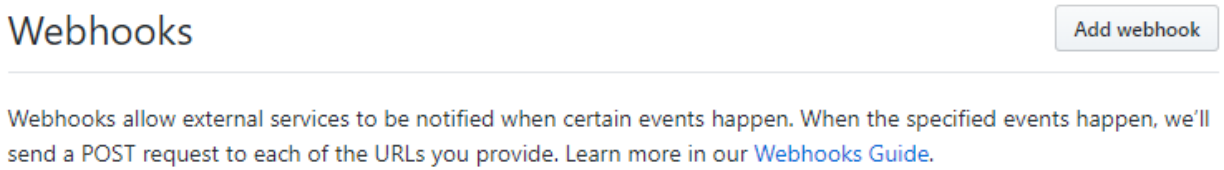
- When the Create First Admin User page appears, specify your details in the respective fields and click **Save and Continue**.(Here you can change the password)
- Click **Save and Finish**
- When the Jenkins is ready page appears, click **Start using Jenkins** OR click **Restart**, whatever is appeared there.
- Note: If the page doesn't automatically refresh after a minute, use your web browser to refresh the page manually.
- Log in to Jenkins with the credentials of the user you just created



Setup Webhook on GitHub

- Go to GitHub Web
- Go to “verity-devops-ex” repository
- Navigate to the "Settings" tab
- Select the "Webhooks" option on the left menu
- Click "Add webhook"





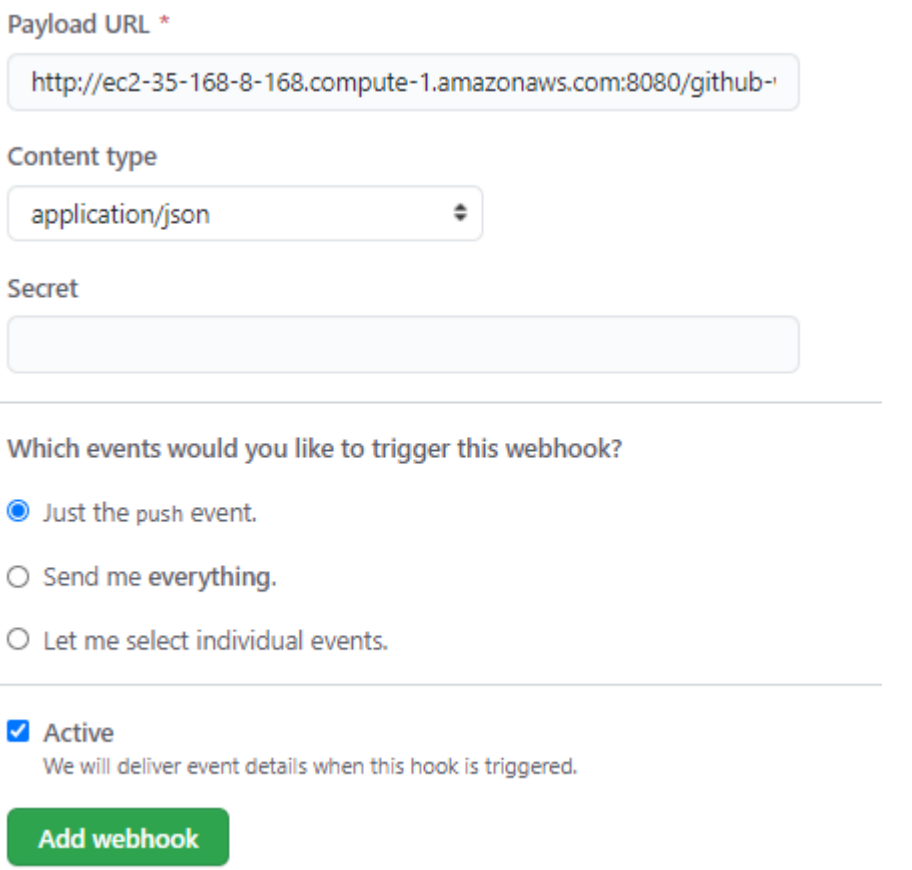
GitHub Settings

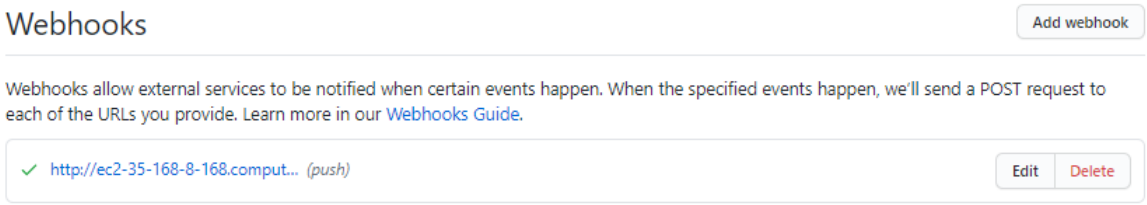
For "Payload URL":

- Use the public dns or ip address for the aws machine instance

  e.g. http://ec2-35-168-8-168.compute-1.amazonaws.com:8080
- Add **/github-webhook/** to the end of it

  e.g. **http://ec2-35-168-8-168.compute-1.amazonaws.com:8080/github-webhook/**
- Select "application/json" as the encoding type
- Leave "Secret" blank (unless a secret has been created and configured in the Jenkins "Configure System -> GitHub plugin" section)
- Select “Just the push event"
- Make sure "Active" is checked
- Click "Add Webhook"





Create Pipeline Project

- Go to Jenkins Dashboard
- Click “New Item” at top left section
- Enter item name(e.g. verity-devops-ex) > Select Pipeline > Click OK
- Select ”GitHub hook trigger for GITScm polling” checkbox under Build Triggers section
- Go to Pipeline section
- Choose the ”Pipeline script from SCM” option from the ”Definition” field
- Choose the ”Git” option from the ”SCM” field
- Enter your Repository URL(e.g.) https://github.com/<Your username>/verity-devops-ex.git
- Enter “Jenkinsfile.txt” under “Script Path” section
- Click Apply
- Click Save