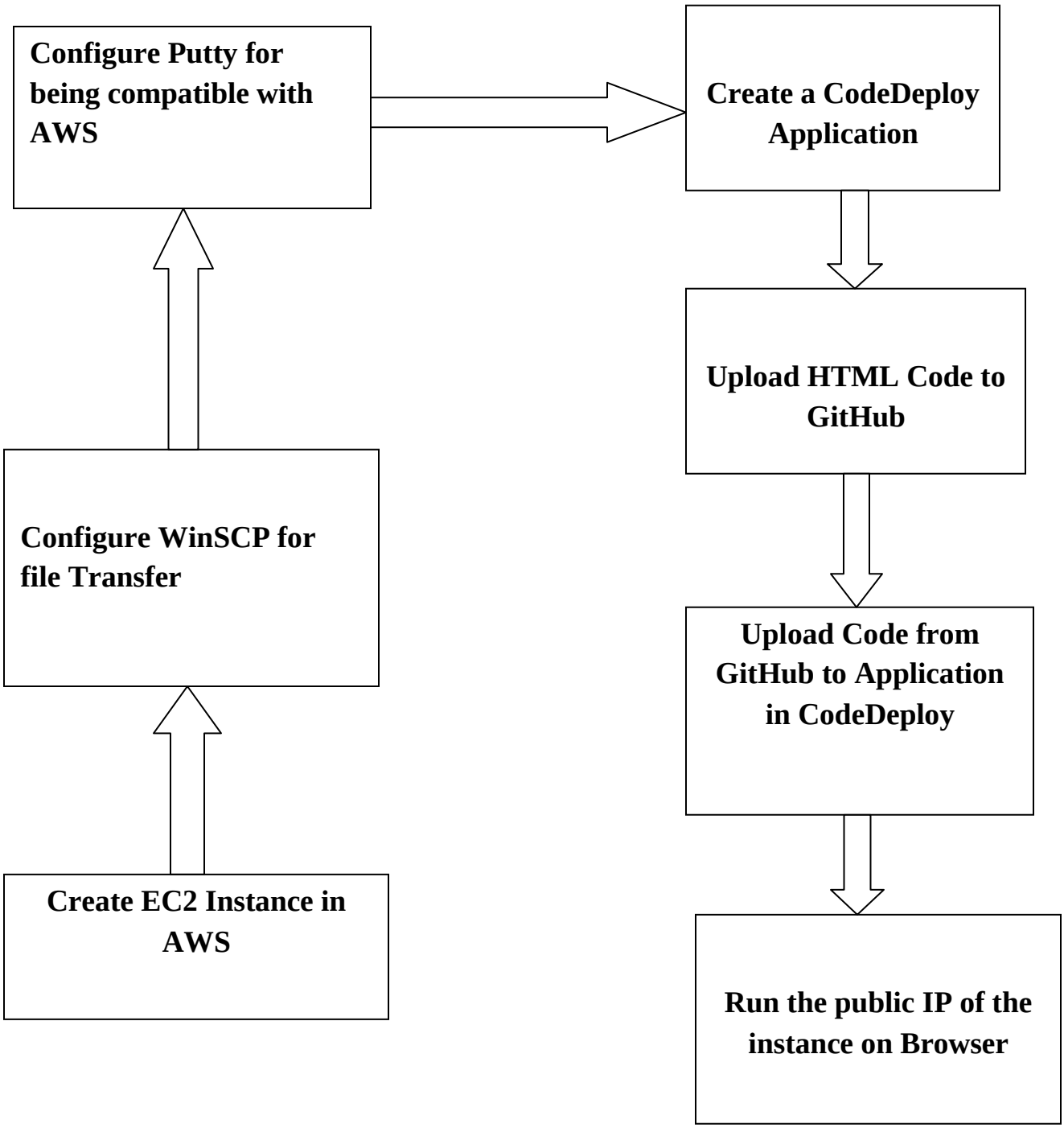
Hosting a Static Website using CodeDeploy

ABSTRACT

Amazon Web Services (AWS) provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis.An EC2 instance is nothing but a virtual server in Amazon Web services terminology. We create instances with the specifications as we require that is we configure memory, CPU cycles etc. Here we create instances in AWS Linux platform compatible for a 64 bit system. We create a key pair and attach it with the instances so that the instance can be accessed securely.When key pair is created a public key from Amazon is created and a private key is given to the host system to login with the instance. As the instance is in linux OS and we are operating on windows, to make file transfers between the differing operating systems we use WinSCP .We login to Putty using the key on the host system.Putty is used to configure the remote Linuxserver. After configuration the server is set up to run.

CodeDeploy is used to automate the deployment of applications to instances and to update the applications as required. There are two ways to upload code to the Applicaton one is through S3 bucket and other is through the GitHub. We here use GitHub and upload the code from GitHub to Applicaton and the application is deployed to selected instance. The instance public IP is used as the url to access the webpage.

WORKFLOW


Configure Putty for being compatible with AWS
Create a CodeDeploy Application
Upload HTML Code to GitHub
Upload Code from GitHub to Application in CodeDeploy
Run the public IP of the instance on Browser
Configure WinSCP for file Transfer
Create EC2 Instance in AWS

INTRODUCTION

Amazon Web Services (**AWS**)

AWS is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis with a free-tier option available for 12 months. The technology allows subscribers to have at their disposal a full-fledged virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers have most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, CRM, etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser. The browser acts as a window into the virtual computer, letting subscribers log-in, configure and use their virtual systems just as they would a real physical computer. They can choose to deploy their AWS systems to provide internet-based services for their own and their customers' benefit.

The AWS technology is implemented at server farms throughout the world, and maintained by the Amazon subsidiary. AWS was one of the first companies to introduce a pay-as-you-go cloud computing model that scales to provide users with compute, storage or throughput as needed. Fees are based on a combination of usage and the hardware/OS/software/networking features chosen by the subscriber, required availability, redundancy, security, and service options. Based on what the subscriber needs and pays for, they can reserve a single virtual AWS computer, a cluster of virtual computers, a physical (real) computer dedicated for their exclusive use, or even a cluster of dedicate physical computers.

In 2017, AWS comprised more than 90 services spanning a wide range including computing, storage, networking, database, analytics, applicationservices, deployment, management, mobile, developer tools, and tools for the Internet of Things. The most popular include Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service(S3). Most services are not exposed directly to end users, but instead offer functionality through APIs for

developers to use in their applications. Amazon Web Services' offerings are accessed over HTTP, using the REST architectural style and SOAP protocol.

AMAZON ELASTIC COMPUTE CLOUD (EC2)

Amazon Elastic Compute Cloud (**EC2**) forms a central part of Amazon.com's cloud-computing platform, Amazon Web Services(AWS), by allowing users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.

FEATURES OF AMAZON EC2

Amazon EC2 provides the following features:

- Virtual computing environments, known as *instances*
- Preconfigured templates for your instances, known as Amazon Machine Images (AMIs), that package the bits you need for your server (including the operating system and additional software)
- Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
- Secure login information for your instances using key pairs (AWS stores the public key, and you store the private key in a secure place)
- Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes
- Persistent storage volumes for your data using Amazon Elastic Block Store (Amazon EBS), known as Amazon EBS volumes

- Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as regions and Availability Zones
- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
- Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
- Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
- Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known as virtual private clouds (VPCs).

AMAZON EC2 INSTANCE

An **EC2 instance** is nothing but a virtual server in Amazon Web services terminology. It stands for Elastic Compute Cloud**.** It is a web service where an AWS subscriber can request and provision a compute server in AWS cloud.

An on-demand EC2 instance is an offering from AWS where the subscriber/user can rent the virtual server per hour and use it to deploy his/her own applications. The instance will be charged per hour with different rates based on the type of the instance chosen. AWS provides multiple instance types for the respective business needs of the user.

Thus, you can rent an instance based on your own CPU and memory requirements and use it as long as you want. You can terminate the instance when it's no more used and save on costs. This is the most striking advantage of an on-demand instance.

AWS CODEDEPLOY

AWS CodeDeploy is a deployment service that enables developers to automate the deployment of applications to instances and to update the applications as required.

We can deploy nearly unlimited variety of application content, such as code, serverless AWS Lambda functions, web and configuration files, executables, packages, scripts, multimedia files, and so on. AWS Code Deploy can deploy application content that runs on a server and is

stored in Amazon S3 buckets, GitHub repositories, or Bitbucket repositories. AWS Code Deploy can also deploy a serverless Lambda function. Changes are not made to existing code before AWS Code Deploy is used.

AWS CodeDeploy is easier to:

- Rapidly release new features.
- Update AWS Lambda function versions.
- Avoid downtime during application deployment.
- Handle the complexity of updating your applications, without many of the risks associated with error-prone manual deployments.
- The service scales with our infrastructure so we can easily deploy to one instance or thousands.

IAM Roles

An IAM role is an IAM entity that defines a set of permissions for making AWS service requests. IAM roles are not associated with a specific user or group. Instead, trusted entities assume roles, such as IAM users, applications, or AWS services such as EC2. IAM roles allow delegating access with defined permissions to trusted entities without having to share long-term access keys. We use IAM roles to delegate access to IAM users managed within your account, to IAM users under a different AWS account, or to an AWS service such as EC2.

With IAM roles for Amazon ECS tasks, you can specify an IAM role that can be used by the containers in a task. Applications must sign their AWS API requests with AWS credentials, and this feature provides a strategy for managing credentials for your applications to use, similar to the way that Amazon EC2 instance profiles provide credentials to EC2 instances. Instead of creating and distributing your AWS credentials to the containers or using the EC2 instance's role, you can associate an IAM role with an ECS task definition or RunTask API operation. The applications in the task's containers can then use the AWS SDK or CLI to make API requests to authorized AWS services.

GIT-HUB

GitHub is a Web-based Git version control repository hosting service. It is mostly used for computer code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features.

It provides access controland several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

GitHub offers both plans for private and free repositories on the same account which are commonly used to host open-sourcesoftware projects. As of April 2017, GitHub reports having almost 20 million users and 57 million repositories, making it the largest host of source code in the world.

WinSCP

WinSCP (Windows Secure Copy) is a free and open source SFTP, WebDAV and SCP client for Microsoft Windows. Its main function is secure file transfer between a local and a remote computer. Beyond this, WinSCP offers basic file manager and file synchronization functionality. For secure transfers, it uses Secure Shell (SSH) and supports the SCP protocol in addition to SFTP.

WinSCP is based on the implementation of the SSH protocol from PuTTY and FTP protocol from FileZilla. It is also available as a plugin for Altap Salamander file manager, and there exists a third-party plugin for the FAR file manager.

PuTTY

PuTTY is a free and open-source terminal emulator,serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port. The name "PuTTY" has no definitive meaning.

PuTTY was originally written for Microsoft Windows, but it has been ported to various other operating systems. Official ports are available for some Unix-like platforms, with work-in-

progress ports and unofficial ports.PuTTY was written and is maintained primarily by Simon Tatham.

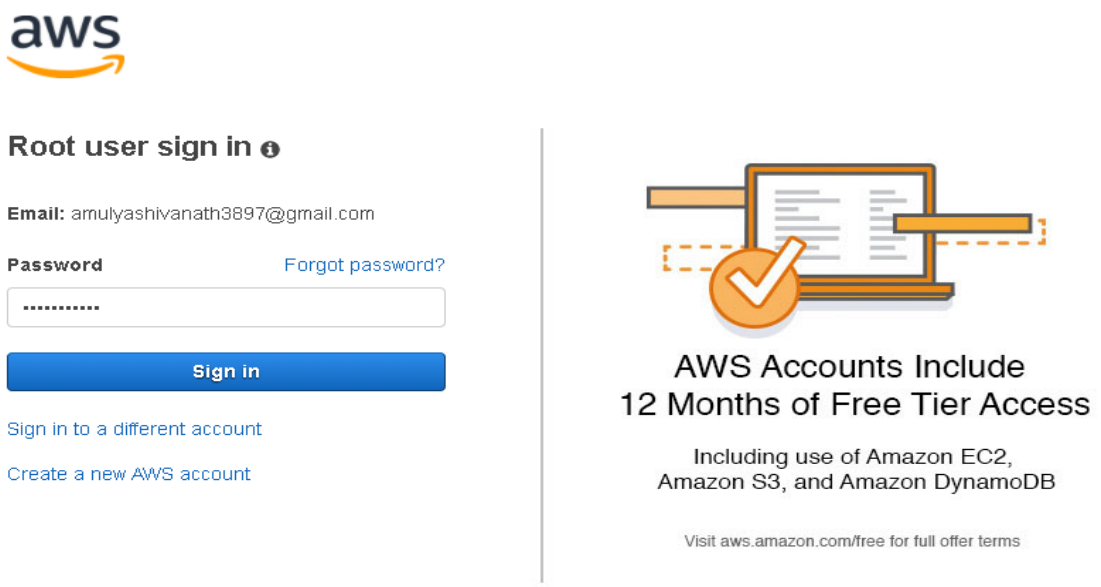
PuTTY supports many variations on the secure remote terminal, and provides user control over the SSH encryption key and protocol version, alternate ciphers such as 3DES, Arcfour, Blowfish, DES, and Public-key authentication.

PuTTY comes bundled with command-line SCP and SFTP clients, called "pscp" and "psftp" respectively, and plink, a command-line connection tool, which is used for non-interactive sessions.

WORKING

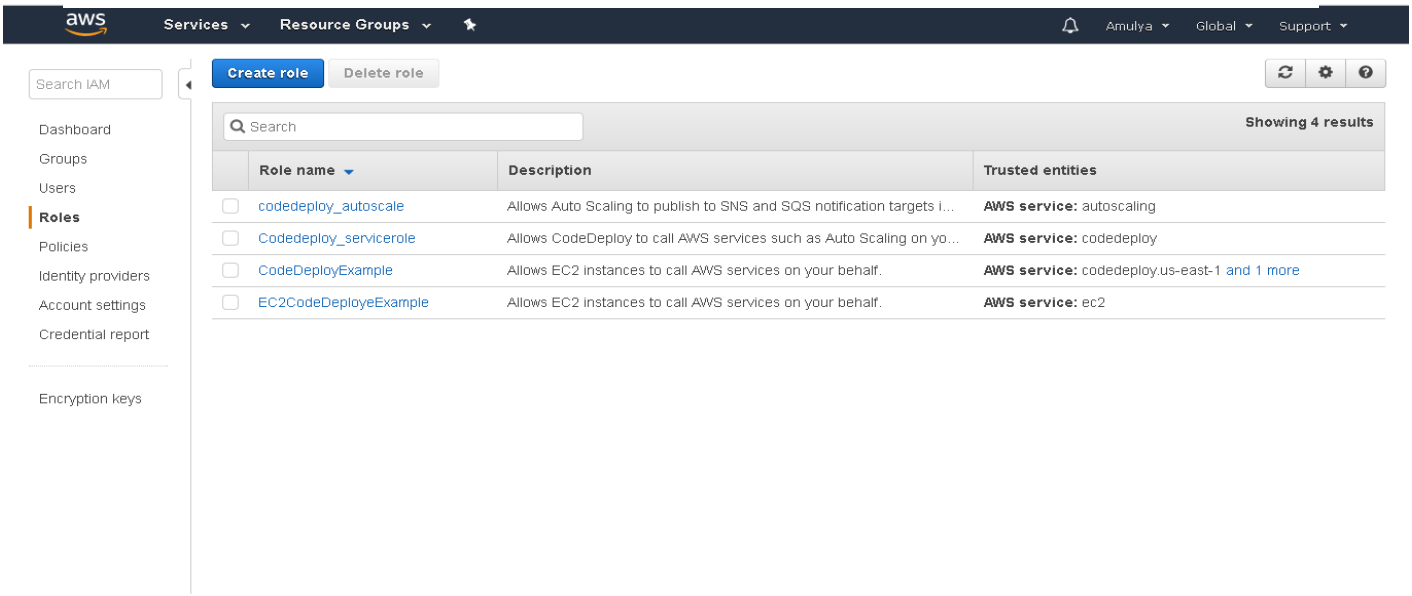
LOGIN AND ACCESS TO AWS SERVICES:

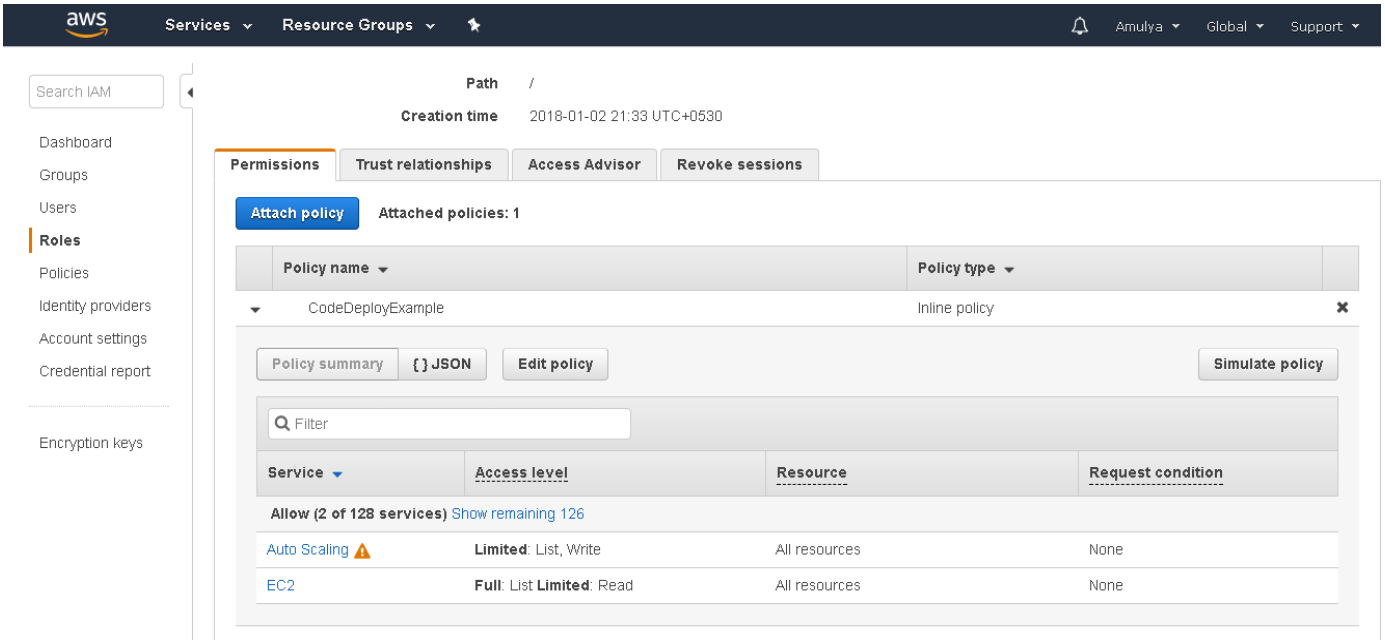
STEP1. Login to your AWS account



STEP 2 : Create IAM roles

- Go to Security Credentials in the account name displayed on right of the task bar.
- Select Roles on the left corner of the page
- Create new Role
- Create two roles
  - ✓ One for EC2 by including AmazonEC2 policy for AWS
  - ✓ Another for CodeDeploy by including AmazonEC2Policy for CodeDeploy
- Here two roles as stated above arecreated and EC2 role is inlined inside CodeDeployExample.


aws
Services
Resource Groups
Amulya
Global
Support
Search IAM
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Encryption keys
Create role
Delete role
Search
Showing 4 results
Role name
Description
Trusted entities
codedeploy_autoscale
Allows Auto Scaling to publish to SNS and SQS notification targets i...
AWS service: autoscaling
Codedeploy_servicerole
Allows CodeDeploy to call AWS services such as Auto Scaling on yo...
AWS service: codedeploy
CodeDeployExample
Allows EC2 instances to call AWS services on your behalf.
AWS service: codedeploy.us-east-1 and 1 more
EC2CodeDeployeExample
Allows EC2 instances to call AWS services on your behalf.
AWS service: ec2


aws
Services
Resource Groups
Amulya
Global
Support
Search IAM
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Encryption keys
Path /
Creation time 2018-01-02 21:33 UTC+0530
Permissions
Trust relationships
Access Advisor
Revoke sessions
Attach policy
Attached policies: 1
Policy name
Policy type
CodeDeployExample
Inline policy
Policy summary
{ } JSON
Edit policy
Simulate policy
Filter
Service
Access level
Resource
Request condition
Allow (2 of 128 services) Show remaining 126
Auto Scaling
Limited: List, Write
All resources
None
EC2
Full: List Limited: Read
All resources
None

STEP 3:

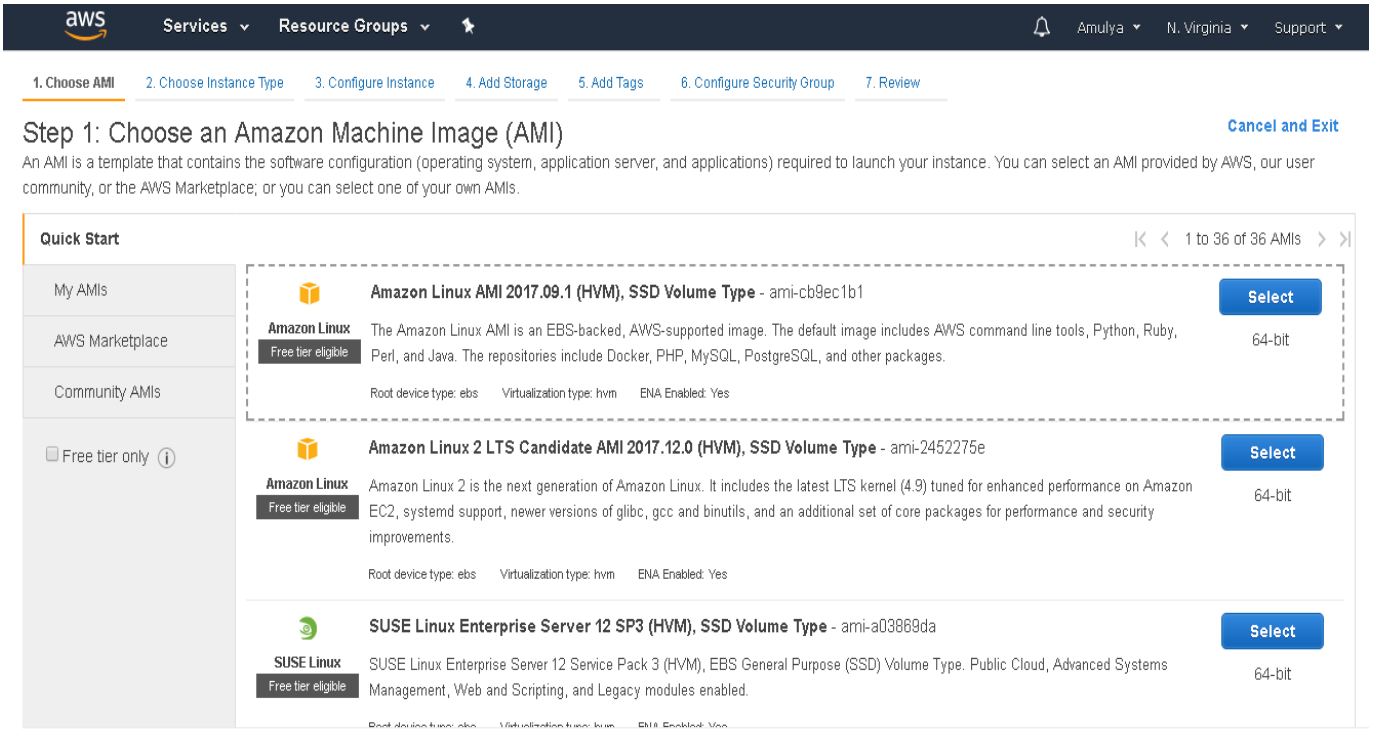
- Go to the AWS Services tab at the top left corner.
- Here, you will see all of the AWS Services categorized as per their area viz. Compute, Storage, Database, etc. For creating an EC2 instance, we have to choose Computeà EC2 as in the next step.
- Open all the services and click on EC2 under Compute services. This will launch the dashboard of EC2.
- Here is the EC2 dashboard. Here you will get all the information in gist about the AWS EC2 resources running.
- On the top right corner of the EC2 dashboard, choose the AWS Region in which you want to provision the EC2 server.
- Once your desired Region is selected, come back to the EC2 Dashboard.
- Click on 'Launch Instance' button in the section of Create Instance.
- Instance creation wizard page will open as soon as you click 'Launch Instance'.

a.CHOOSE AMI

- You will be asked to choose an AMI of your choice. (An AMI is an Amazon Machine Image. It is a template basically of an Operating System platform which you can use as a base to create your instance).

  Once you launch an EC2 instance from your preferred AMI, the instance will automatically be booted with the desired OS.

- Here we are choosing the default Amazon Linux (64 bit) AMI.

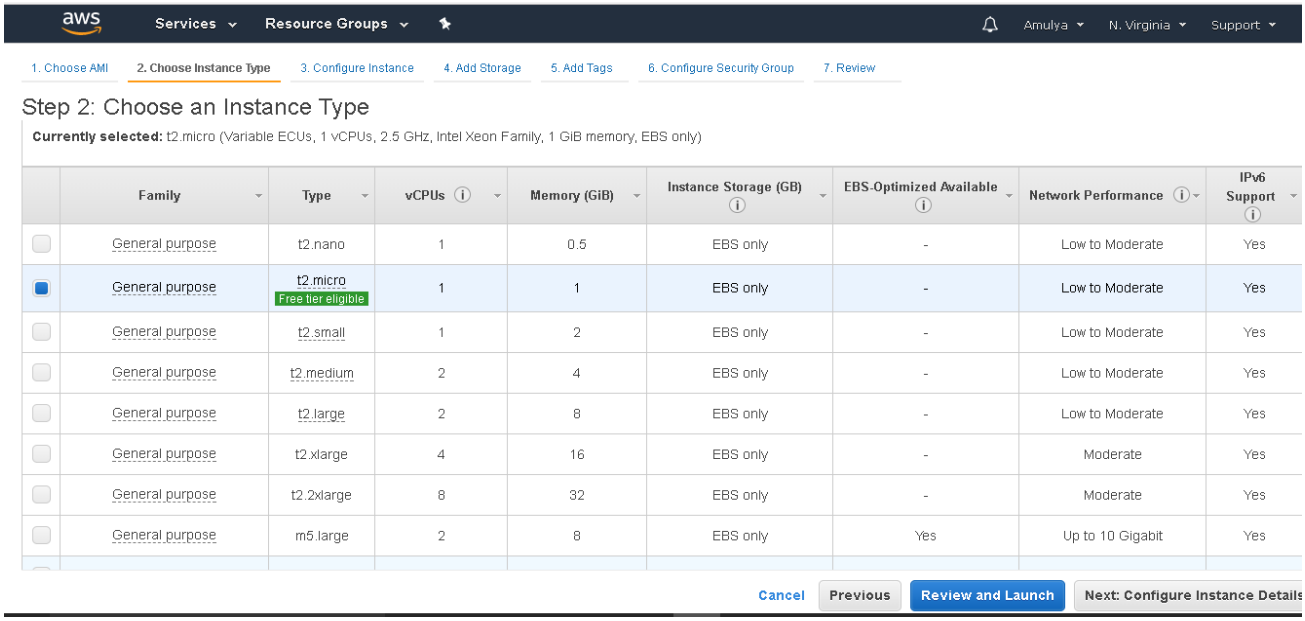


b. CHOOSE INSTANCE TYPES

In the next step, we have to choose the type of instance required based on the business needs.

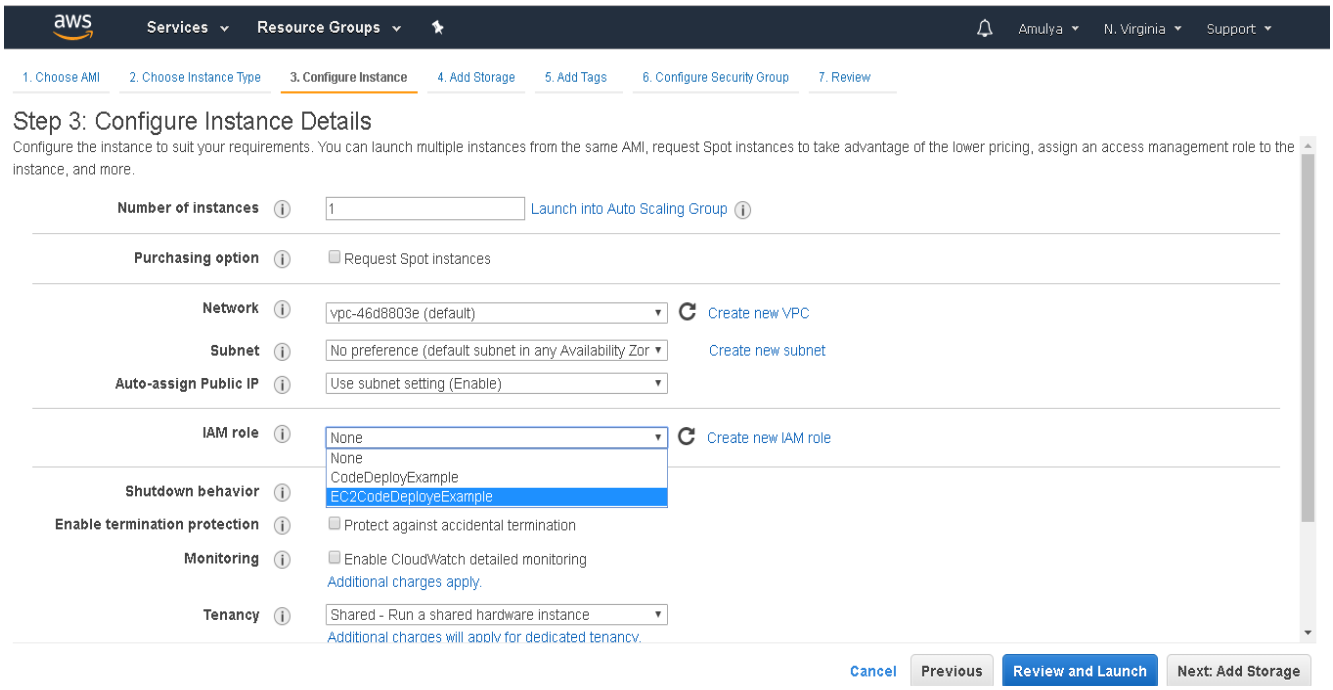
We will choose t2.micro instance type, which is a 1vCPU and 1GB memory server offered by AWS.

1. Click on "Configure Instance Details" for further configurations
2. In the next step of the wizard, enter details like no. of instances you want to launch at a time.
3. Here we are launching one instance.



c. CONFIGURE INSTANCE

- No. of instances- you can provision up to 20 instances at a time. Here we are launching one instance.
- Under Subnets, you can choose the subnet where you want to place your instance.
- I have chosen an already existing public subnet.
- Once your instance is launched in a public subnet, AWS will assign a dynamic public IP



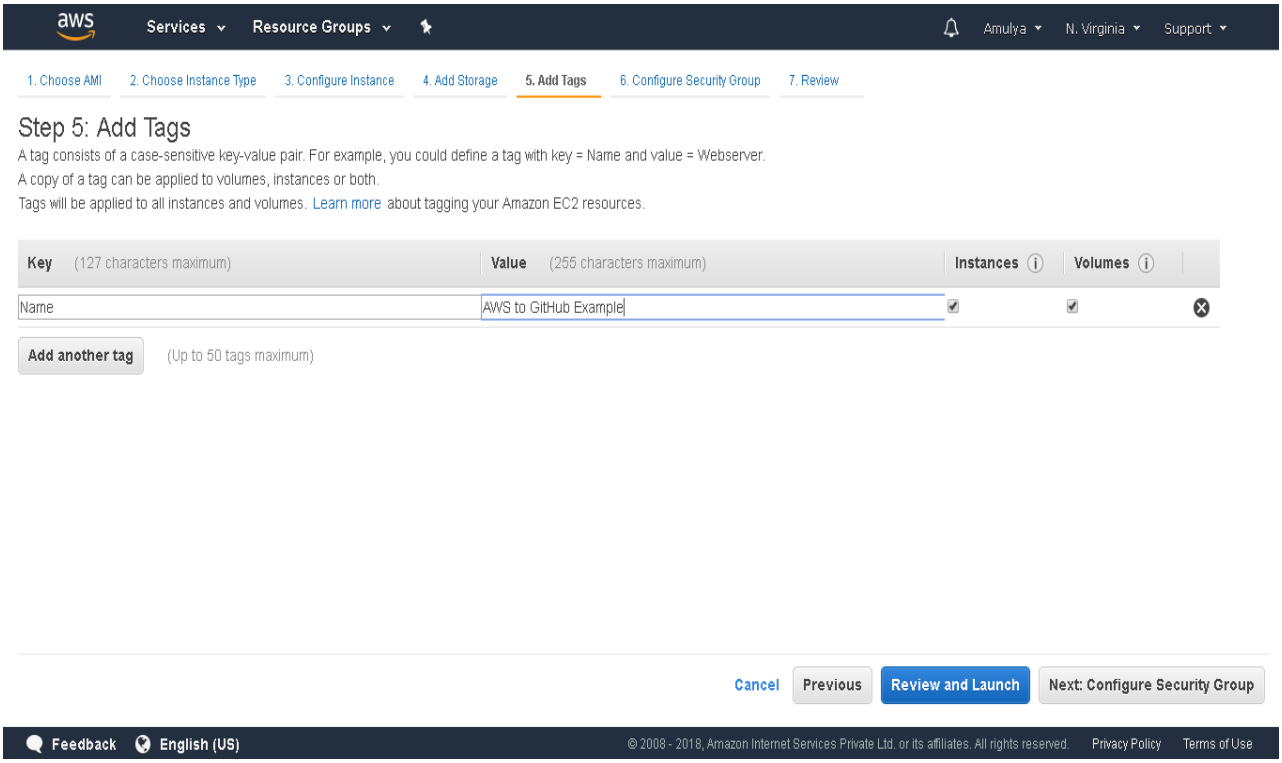
to it from their pool of IPs.

d. ADD STORAGE

- In the Add Storage step, instance has been automatically provisioned a General Purpose SSD root volume of 8GB. (Maximum volume size we can give to a General Purpose volume is 16GB)
- We can change your volume size, add new volumes, change the volume type, etc.
- AWS provides 3 types of EBS volumes- Magnetic, General Purpose SSD, Provisioned IOPs. You can choose a volume type based on your application's IOPs needs.

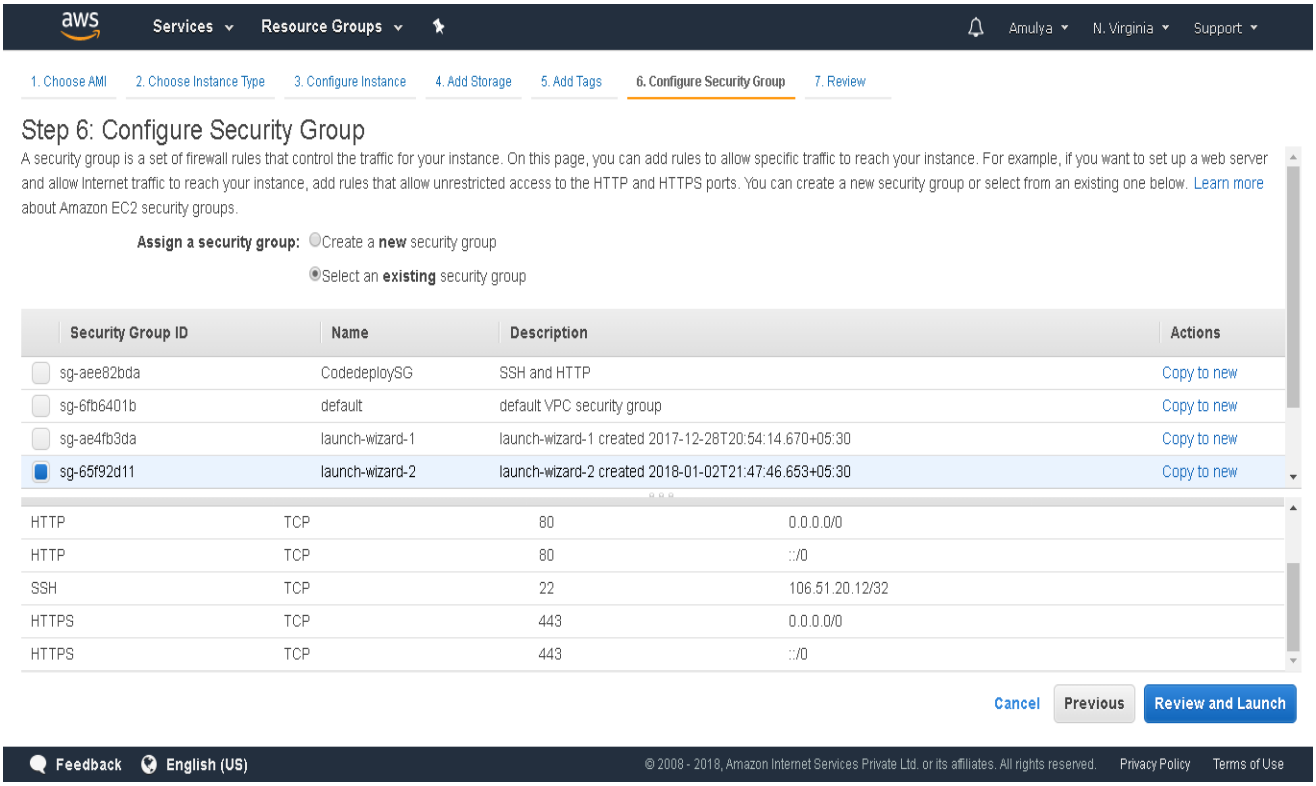
e. TAG INSTANCE:

- You can tag your instance with a key-value pair. This gives visibility to the AWS account administrator when there are lot numbers of instances.
- The instances should be tagged based on their department, environment like Dev/SIT/Prod. Etc. this gives a clear view of the costing on the instances under one common tag.



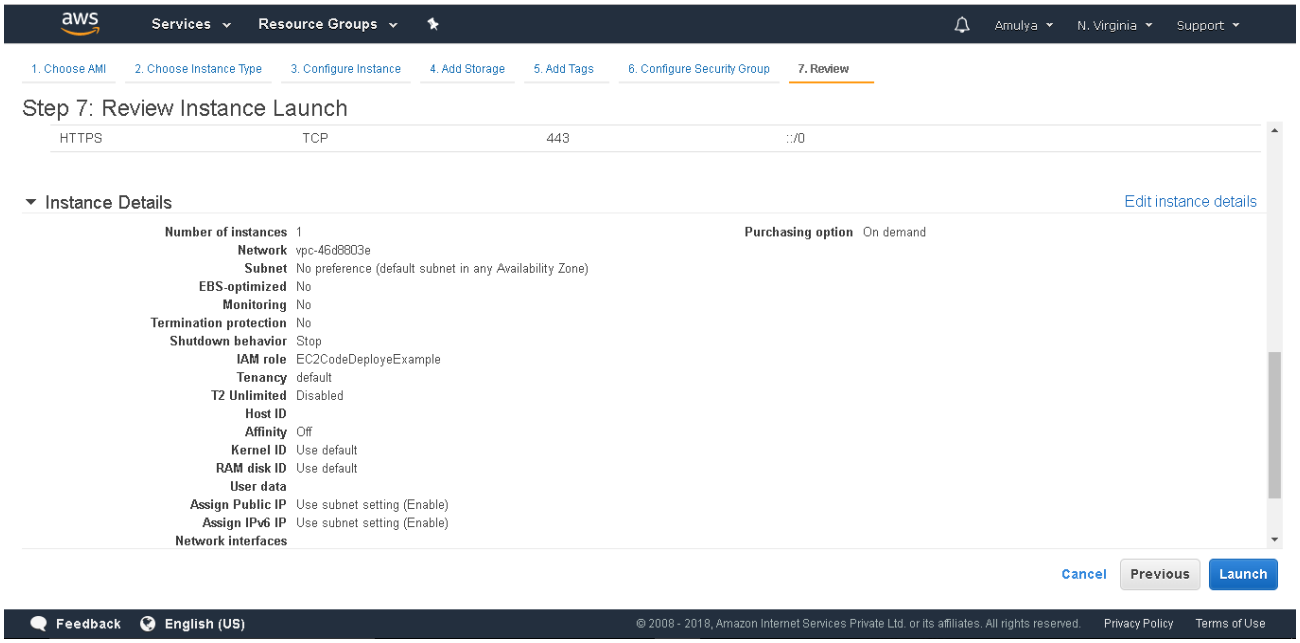
f.CONFIGURING SECURITY GROUPS

- Here you can restrict traffic on your instance ports. This is an added firewall mechanism provided by AWS apart from your instance's OS firewall.
- You can define open ports and IPs.
- Defining protocols which we want enabled on my instance
- Assigning IPs which are allowed to access our instance on the said protocols
- Once, the firewall rules are set- Review and launch



g.REVIEW INSTANCES

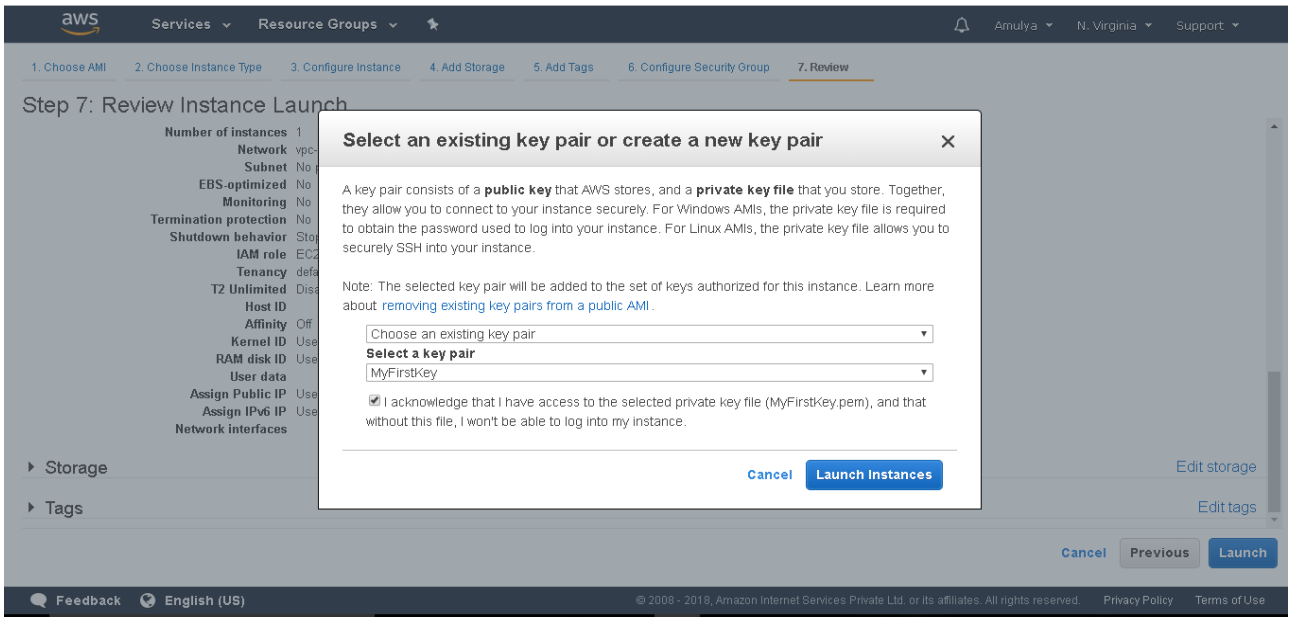
- In this step, we will review all our choices and parameters and go ahead to launch our instance.



In the next step you will be asked to create a key pair to login to you an instance. A key pair is a set of public-private keys.

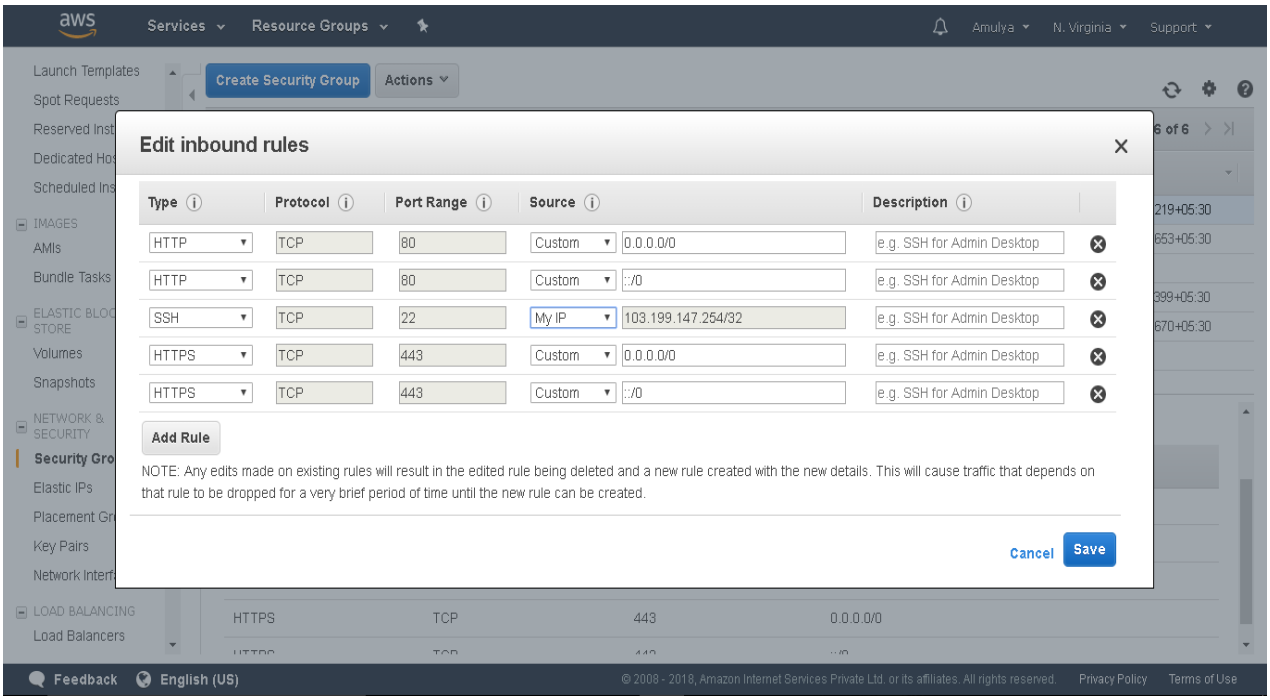
AWS stores the private key in the instance, and you are asked to download the public key. Make sure you download the key and keep it safe and secured; if it is lost you cannot download it again.

1. Create a new key pair
2. Give a name to your key
3. Download and save it in your secured folder
4. When you download your key, you can open and have a look at your RSA private key.
5. Once you are done downloading and saving your key, launch your instance.



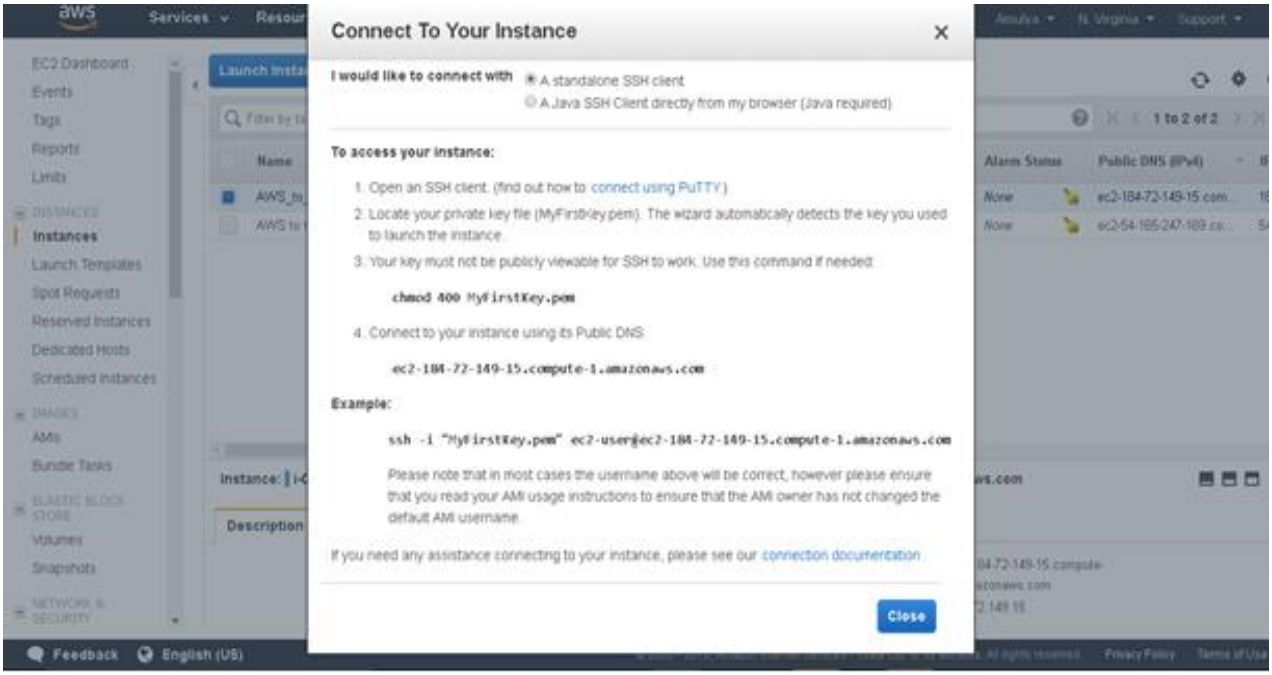
STEP 4:

- Go to Security Group
- Choose your instance
- Go to Actions and edit
- Then change the IP to the current system IP as shown in the figure



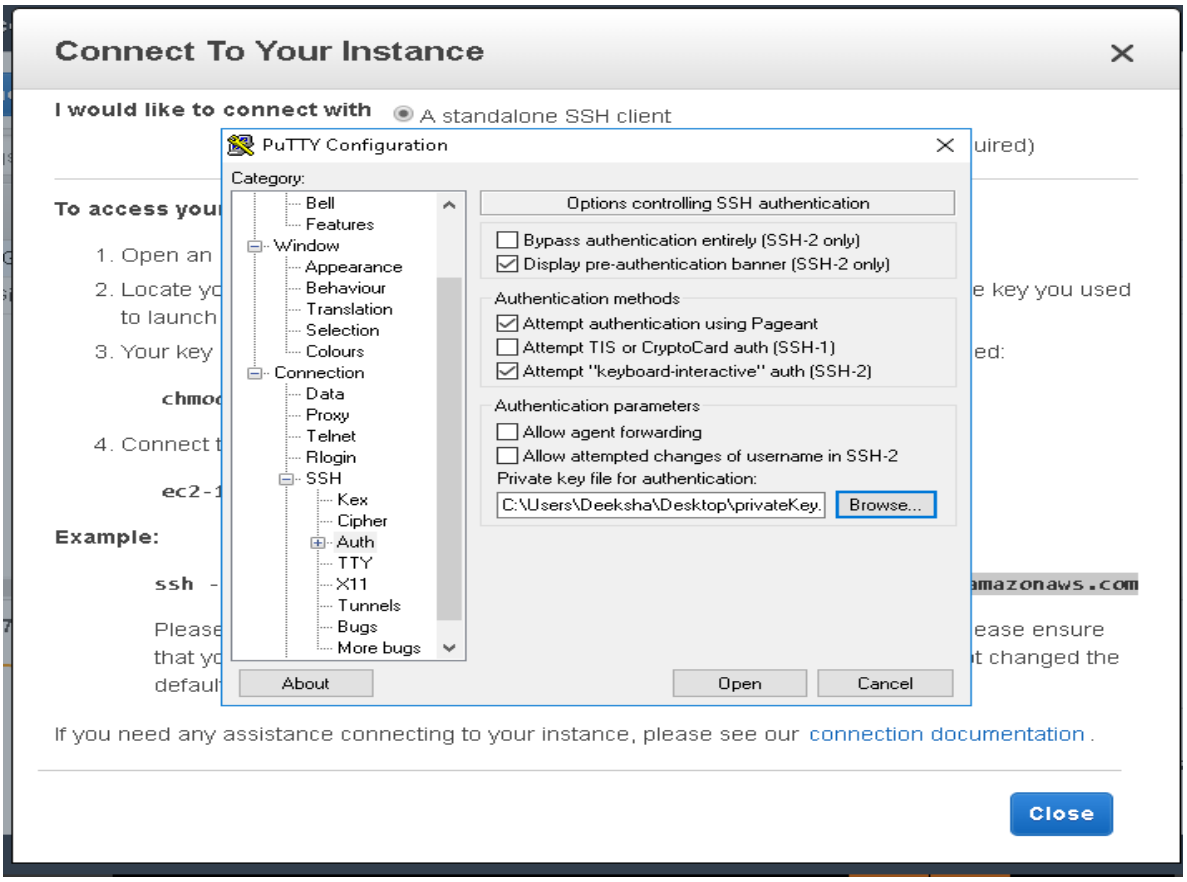
STEP 5:

- Go back to EC2 Dashboard and display your running instances
- Choose the instance in which you want to upload the code
- Click connect and the pop up as shown below will be displayed.



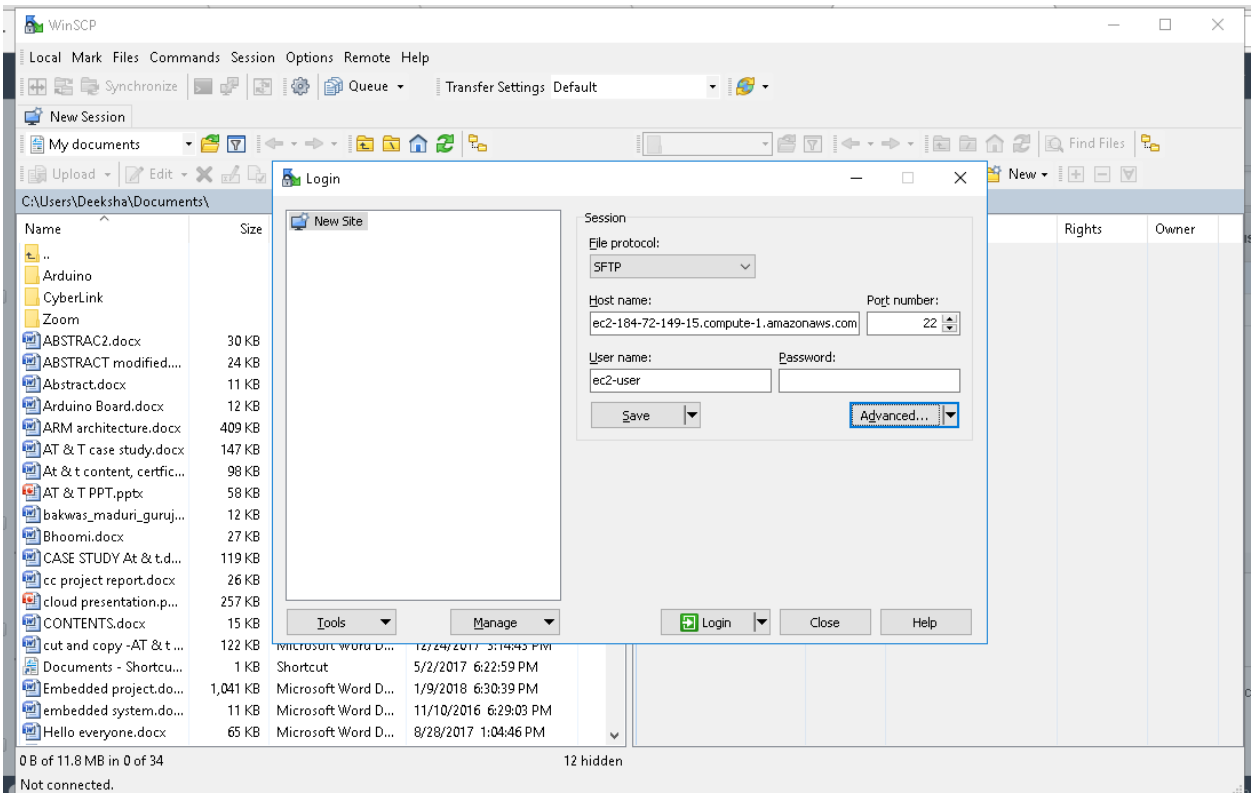
STEP 6:

- Open Putty and copy the public DNS address of the instance
- Paste the address in the session name in Putty. It will automatically detect Name
- Go to SSH and Auth in the left corner of Putty and Browse the for the Private key file that is in pem extension.
- Pem file generated during Key Pair generation is to be stored in the system that needs to run the code.



STEP 7:

- After logging in to Putty open the WinSCP which is used to transfer files between windows and linux.
- Login to WinSCP using the same public key as used in Putty.
- Upload the pem file that acts as private key
- Transfer the pem file from windows to linux system

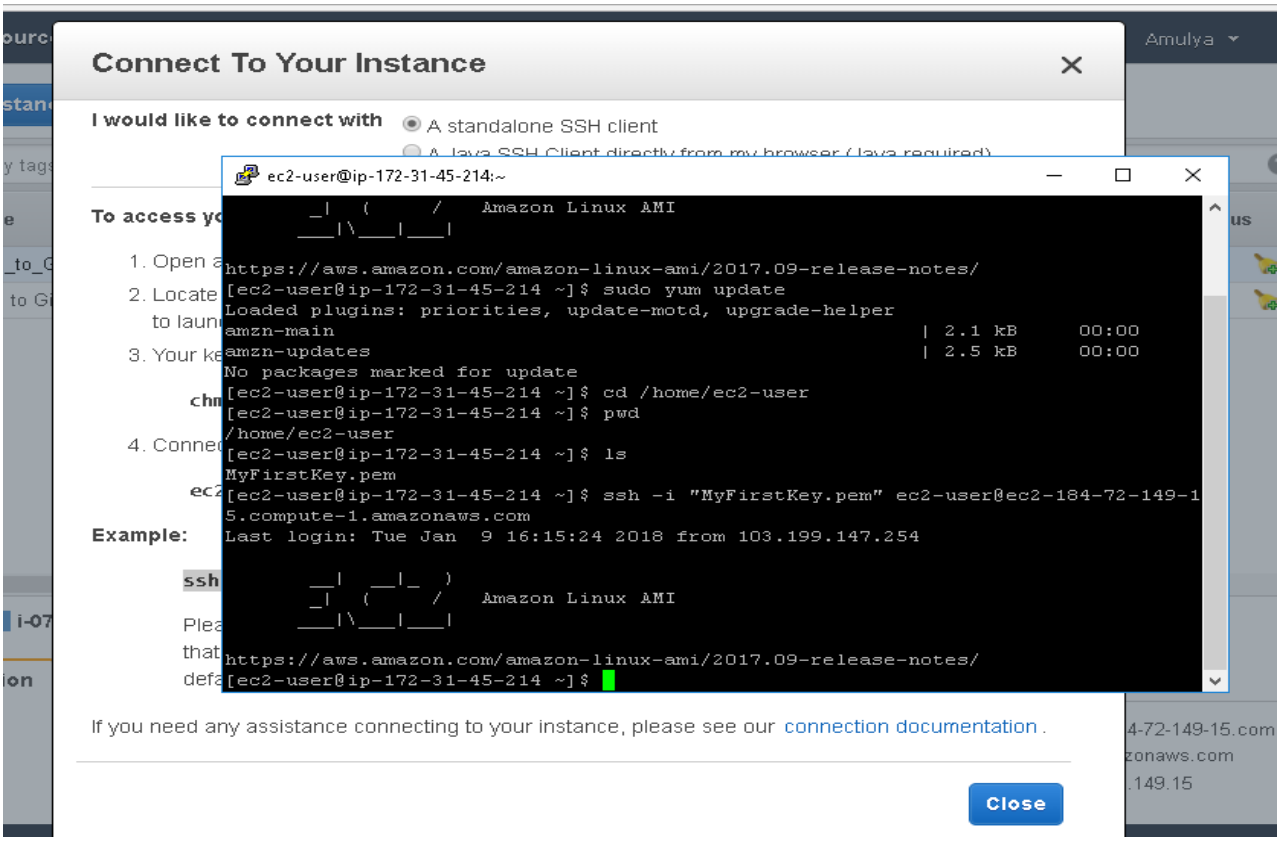


STEP 8: Upload the following commands to Putty

- cd/home/ec2-user
- ssh -i "MyFirstKey.pem" ec2-user@ec2-54-165-247-169.compute-1.amazonaws.com
- When server is booted
    - sudo su
    - yum -y update
    - yum install -y aws-cli
    - cd /home/ec2-user
- Here you will setup your AWS access, secret, and region.
    - aws configure
    - aws s3 cp s3://aws-codedeploy-us-east-1/latest/install . --region us-east-1 (if in east AWS)
    - aws s3 cp s3://aws-codedeploy-us-west-2/latest/install . --region us-west-2 (if in west AWS)

    - chmod +x ./install

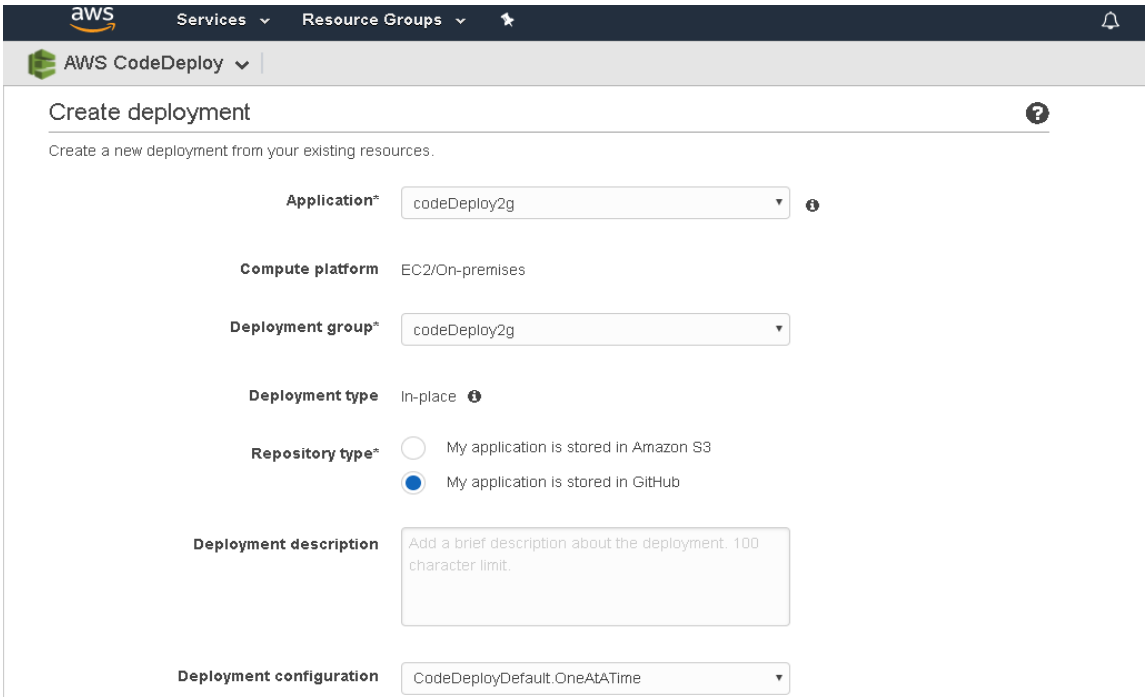
- This is simply a quick hack to get the agent running faster.
    - sed -i "s/sleep(.*)/sleep(10)/" install
    - ./install auto
- Verify it is running.
    - service codedeploy-agent status
    - You will recive a PID



.

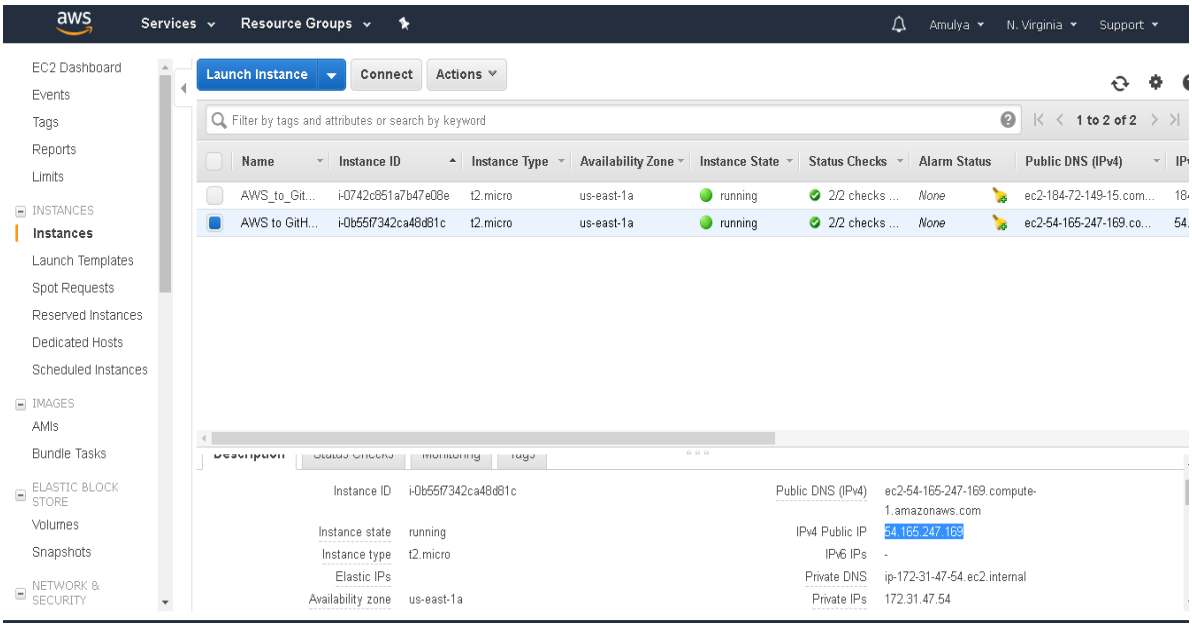
STEP 9:

- Now we need to Create an application in codeDeploy available in the service list of AWS
- Select the created Application and select the Action button and choose revisions to upload the code from Git-Hub.
- Select the Repository and the commit code and click deploy.



STEP 10:

- Go back to EC2 dashboard instances and choose the instance to which you have uploaded the code.



- Copy the public IP and run it on any browser.