CREATE EC2-INSTANCE WITH MFA BY USING ANSIBLE

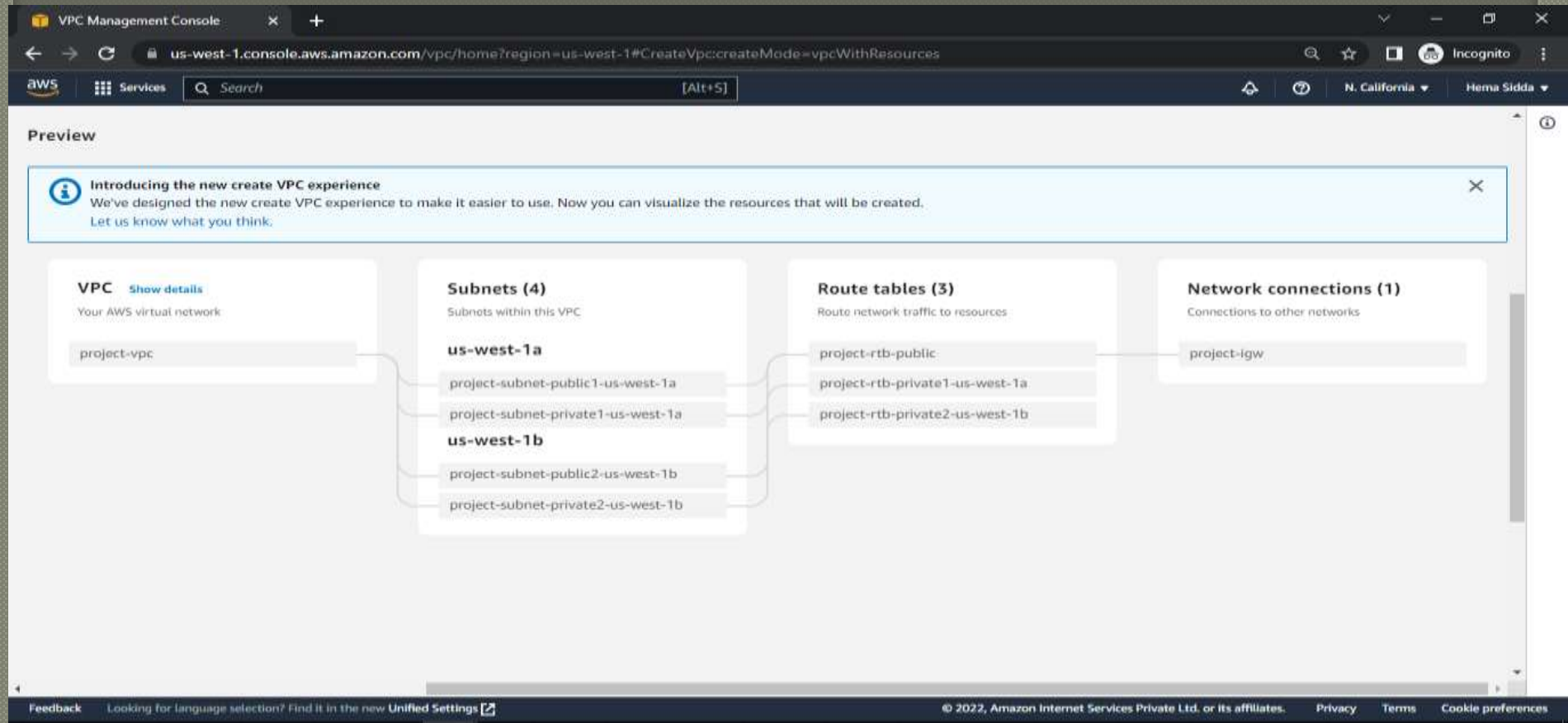
SHAIK GOUSERABBANI

PART-1

CREATE VPC

- Create a VPC with CIDR range 20.0.0.0/16
- Create internet gate way for VPC and attach to the VPC
- Create one public with CIDR range(20.0.0.0/24) and one private subnets with CIDR range (20.0.1.0/24).
- Create one more public subnet with CIDR range (20.0.2.0/24) and one more private subnet with CIDR range (20.0.3.0/24).
- Create route table two public route table and two private route table

- ✓ In route table edit the routes. In route give internet gateway connection to the route table and associate with subnet.
- ✓ For private route table associate with private subnet
- ✓ Create NAT gateway



Create EC2instance:

- Create Public EC2 instance with name ansible-master
  - Ami = amazon Linux
  - Instance type = t2. Micro
  - Key pair = Pem file
  - Subnet = Public subnet
  - Security group = SSH(22),HTTP(80)
- Create private EC2 instance with name worker node
  - Ami = amazon Linux
  - Instance type = t2. Micro
  - Key pair = Pem file
  - Subnet = Private subnet
  - Security group = SSH(22), HTTP(80)

- Connect Public instance into terminal by using
  - ***ssh -i .\pem ec2-user@Public IP***
- Change host name by using
  - ***Sudo hostname master***
  - ***Sudo su – ec2-user***
- Copy ‘pem’ file into terminal from another tab of terminal
  - ***scp -i .\pem .\pem ec2-user@Public IP***
- Give read permission for ‘pem’ file
  - ***sudo chmod 400 workshop.pem***
- Install EPEL Repo in instance
  - EPEL → EXTRA PACKAGE ENTERPRICE LINUX
  - Google Authenticator is part of the EPEL repo and you should install the EPEL repo in your EC2 instance.
  - ***sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm***

- Install Google Authenticator
- The following command would install the Google Authenticator. The package name may vary if your Operating system is different so you can perform a quick search using yum search google to find the right package name
  - ***sudo yum install google-authenticator.x86_64 -y***
- Configure EC2 SSH to use Google Authentication module
- In this step, we are going to perform some file modifications and editing to install google authentication and to setup MFA in this EC2 instance.
- Update the sshd and PAM

- Add the following to the bottom of the file to use Google Authenticator. If there are service accounts or users who should be able to log in without MFA, add nullok at the end of the following statement. This will mean that users who don't run Google Authenticator initialization won't be asked for a second authentication.
  - ***sudo vi /etc/pam.d/sshd***
- Comment out the password requirement as we want to use only the key-based authentication.
  - ***#auth subnets password-auth***
- update the sshd configuration
- In this step we are going to tell sshd that we have one more level of multifactor authentication for the user to login along with the Keybased auth.

- This step is to make sshd daemon to prompt the user for the Verification Code.
  - ***Sudo vi /etc/ssh/sshd_config***
- Comment out the line which says ChallengeResponseAuthentication ‘no’ and uncomment the line which says ‘yes’.
  - ***ChallengeResponseAuthentication yes***
  - ***#ChallengeResponseAuthentication no***
- we need to let sshd daemon know that it should ask the user for an SSH key and a verification code
  - ***AuthenticationMethods publickey,keyboard-interactive***
- install Google Authenticator in Mobile from Play Store
- run the following command for getting QR code
  - ***google-authenticator***

Your new secret key is: BKOUXEKE5RMXIAS3R2VKFF6RN4

Your verification code is 270704

Your emergency scratch codes are:

42239025

96058307

50909854

40529374

63496374

- Do you want me to update your "/home/ec2-user/.google_authenticator" file? (y/n) y

- Do you want to disallow multiple uses of the same authentication token? This restricts you to one login about every 30s, but it increases your chances to notice or even prevent man-in-the-middle attacks (y/n) y

- By default, a new token is generated every 30 seconds by the mobile app. In order to compensate for possible time-skew between the client and the server, we allow an extra token before and after the current time. This allows for a

- time skew of up to 30 seconds between authentication server and client. If you experience problems with poor time synchronization, you can increase the windowfrom its default size of 3 permitted codes (one previous code, the current code, the next code) to 17 permitted codes (the 8 previous codes, the current code, and the 8 next codes). This will permit for a time skew of up to 4 minutes between client and server.
  Do you want to do so? (y/n) n

- If the computer that you are logging into isn't hardened against brute-force login attempts, you can enable rate-limiting for the authentication module. By default, this limits attackers to no more than 3 login attempts every 30s.
  Do you want to enable rate-limiting? (y/n) y

- Connect private instance into public terminal
  - ***ssh -i .\pem ec2-user@private IP***
  - ***ssh -i "workshop.pem" ec2-user@ip address***
  - Verification code:

THANK YOU