**2. Create RDS DB**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/06-EKS-Storage-with-RDS-Database>

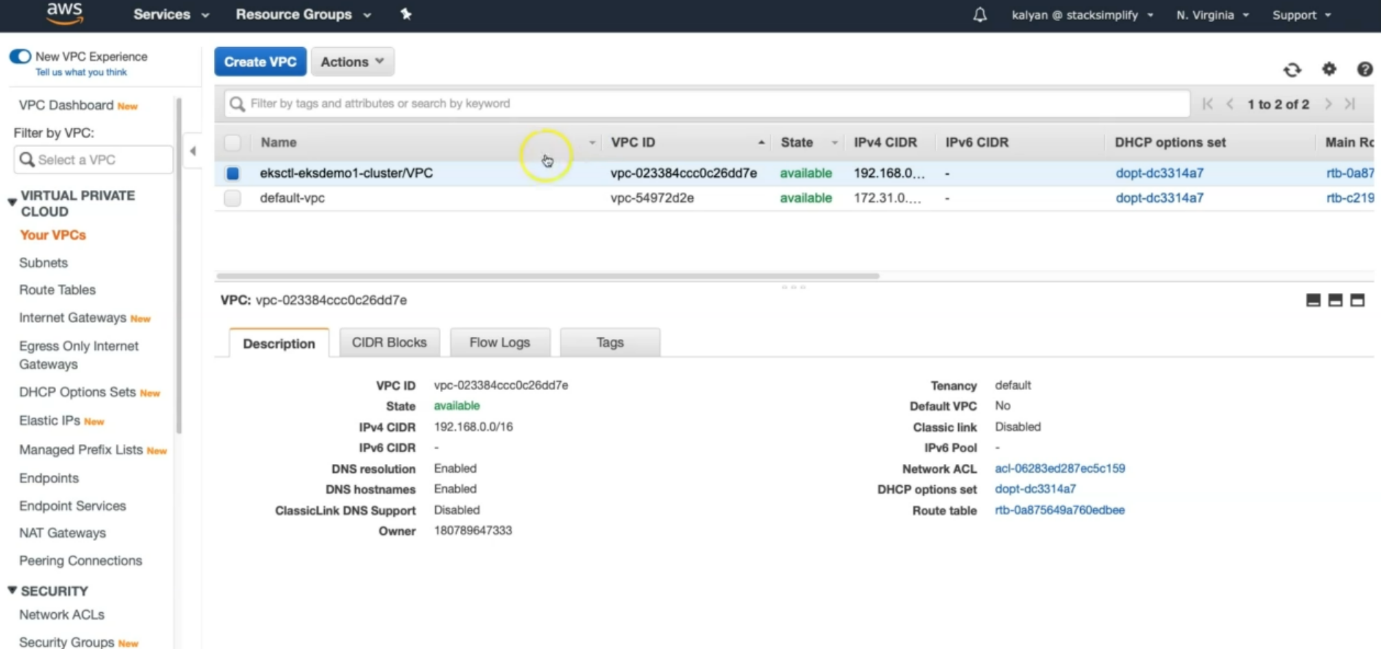
---

**Create RDS Database**

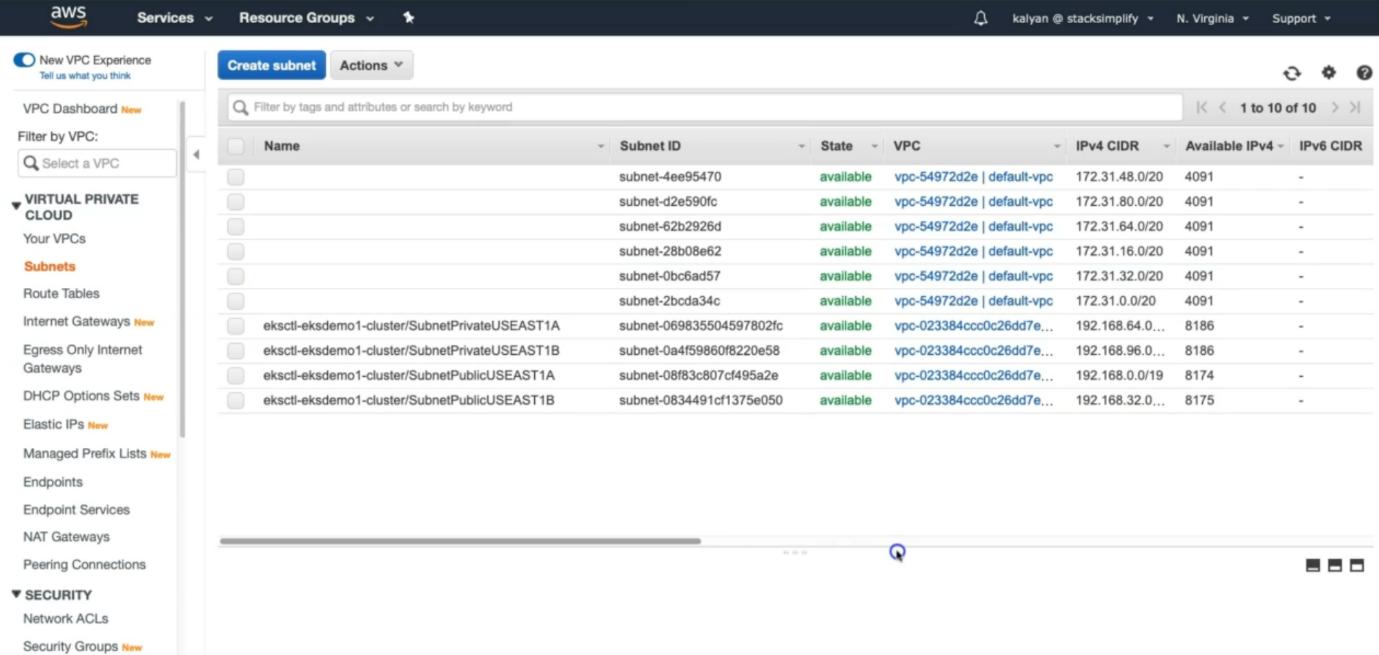
--- Review VPC of our EKS Cluster

--- Go to Services -> VPC

--- VPC Name: eksctl-eksdemo1-cluster/VPC



--- **note** – kubernetes created the vpc.



--- it is also created the subnets. We are going to create Amazon RDS in private subnet.

--- **note** – as part of the kubernetes cluster, it created 2 public subnet and 2 private subnets.

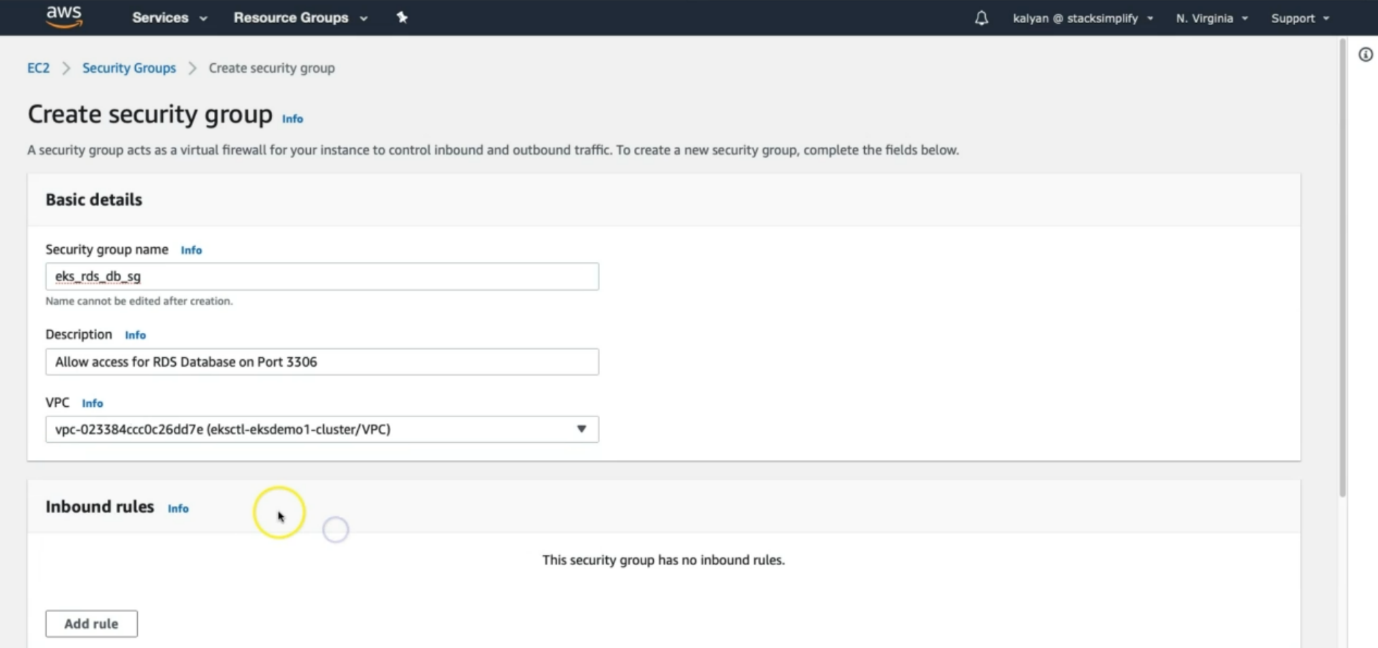
**Create DB Security Group**

--- Create security group to allow access for RDS Database on port 3306

--- Security group name: eks\_rds\_db\_sg

--- Description: Allow access for RDS Database on Port 3306

--- VPC: eksctl-eksdemo1-cluster/VPC



--- **Inbound Rules**

Type: MySQL/Aurora

Protocol: TPC

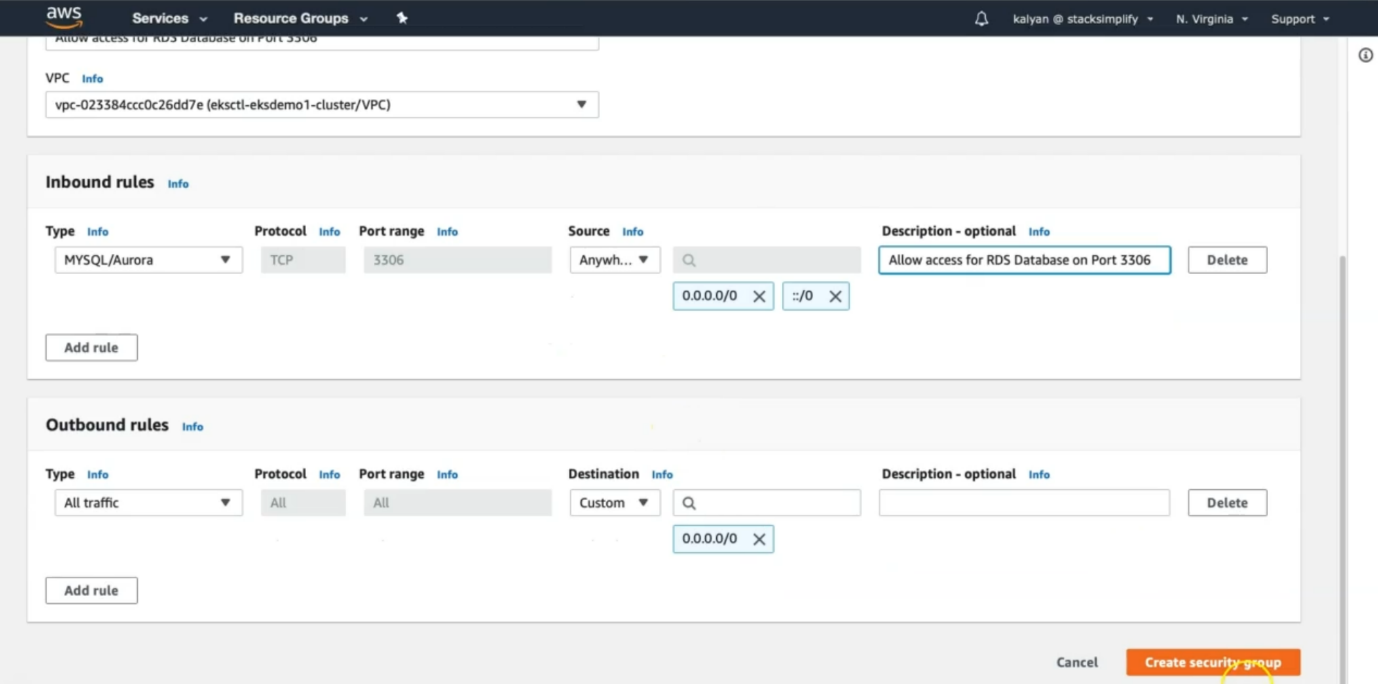
Port: 3306

Source: Anywhere (0.0.0.0/0)

Description: Allow access for RDS Database on Port 3306

**Outbound Rules**

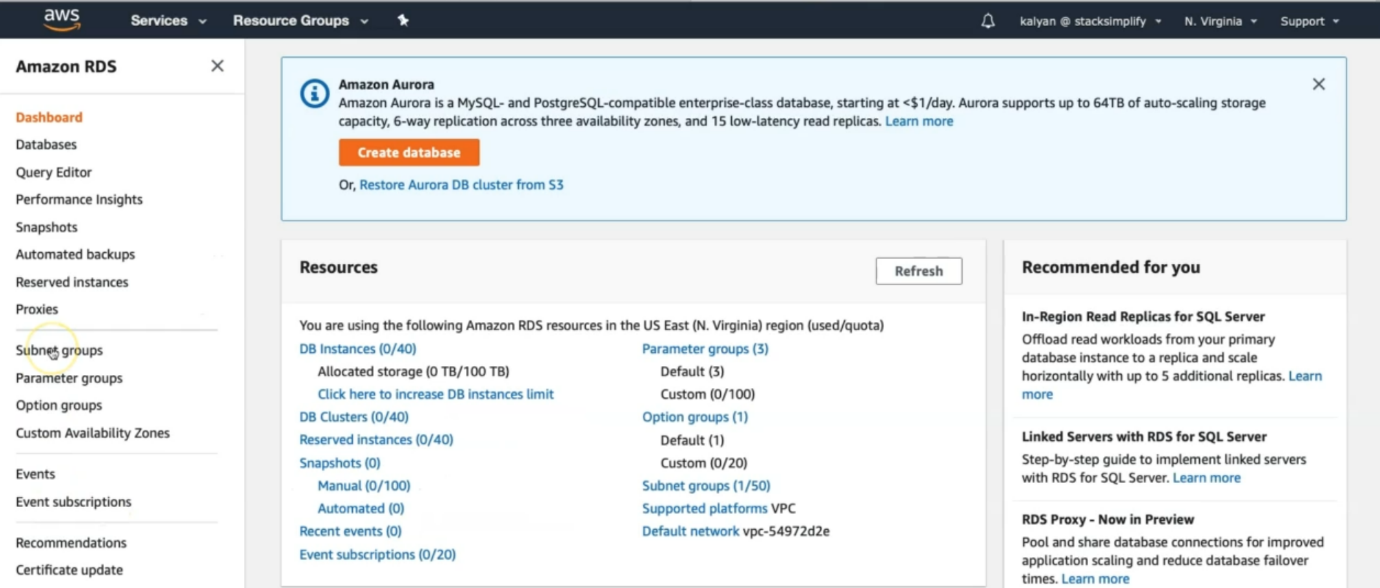
Leave to defaults

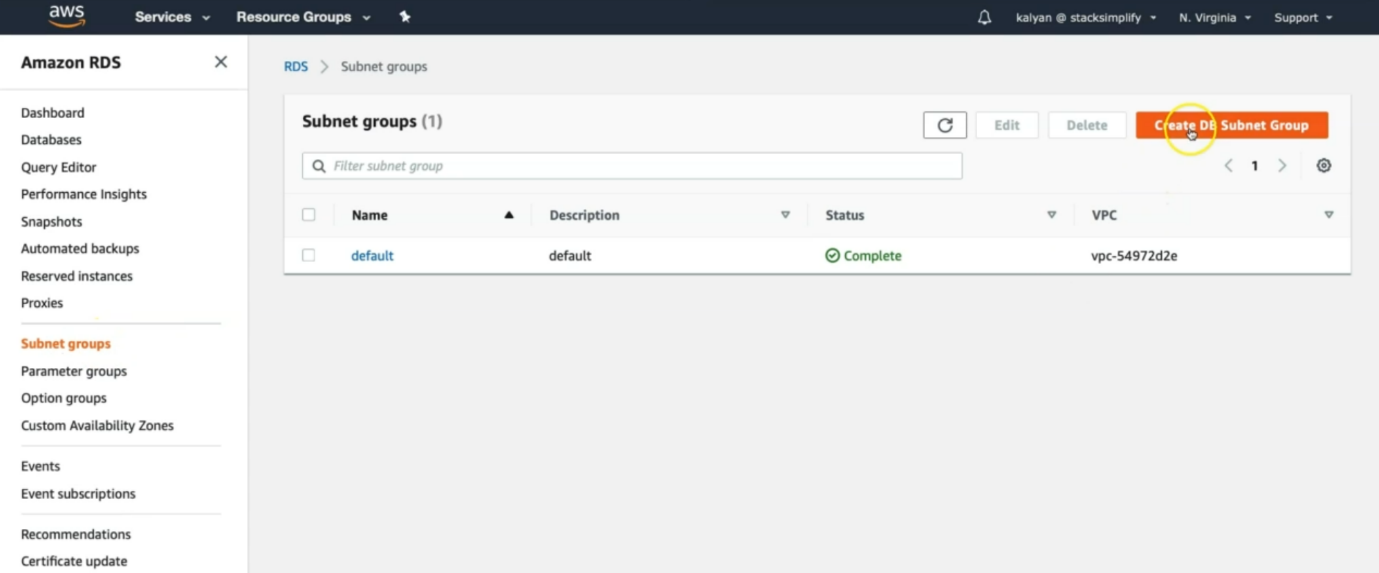


**Create DB Subnet Group in RDS**

--- Go to RDS -> Subnet Groups

--- Click on Create DB Subnet Group



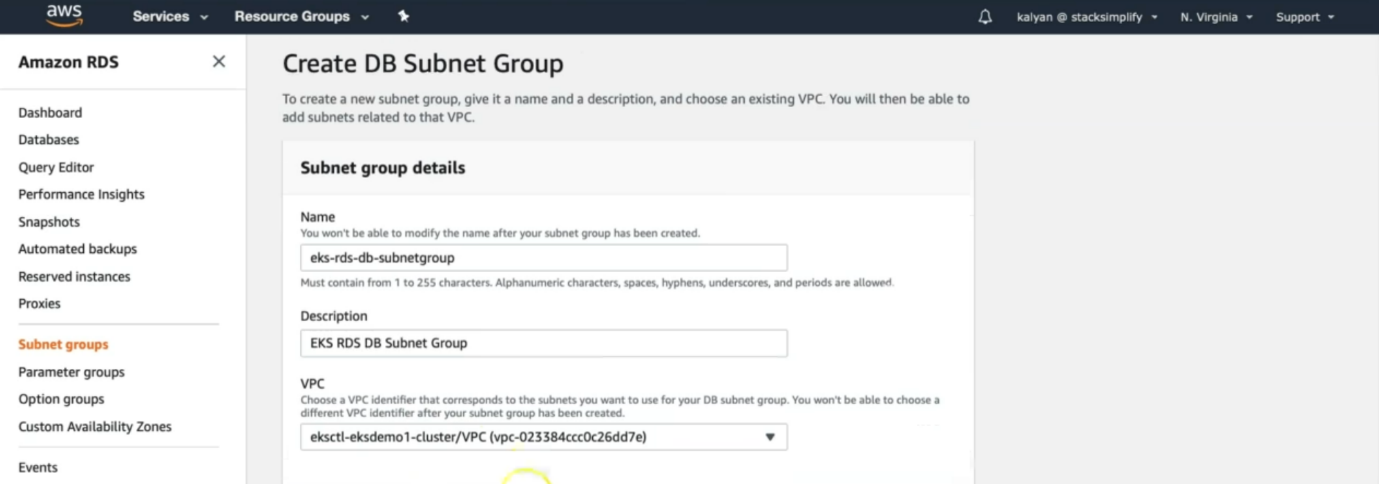


--- **note** – click on crate DB subnet group.

--- Name: **eks-rds-db-subnetgroup**

--- Description: EKS RDS DB Subnet Group

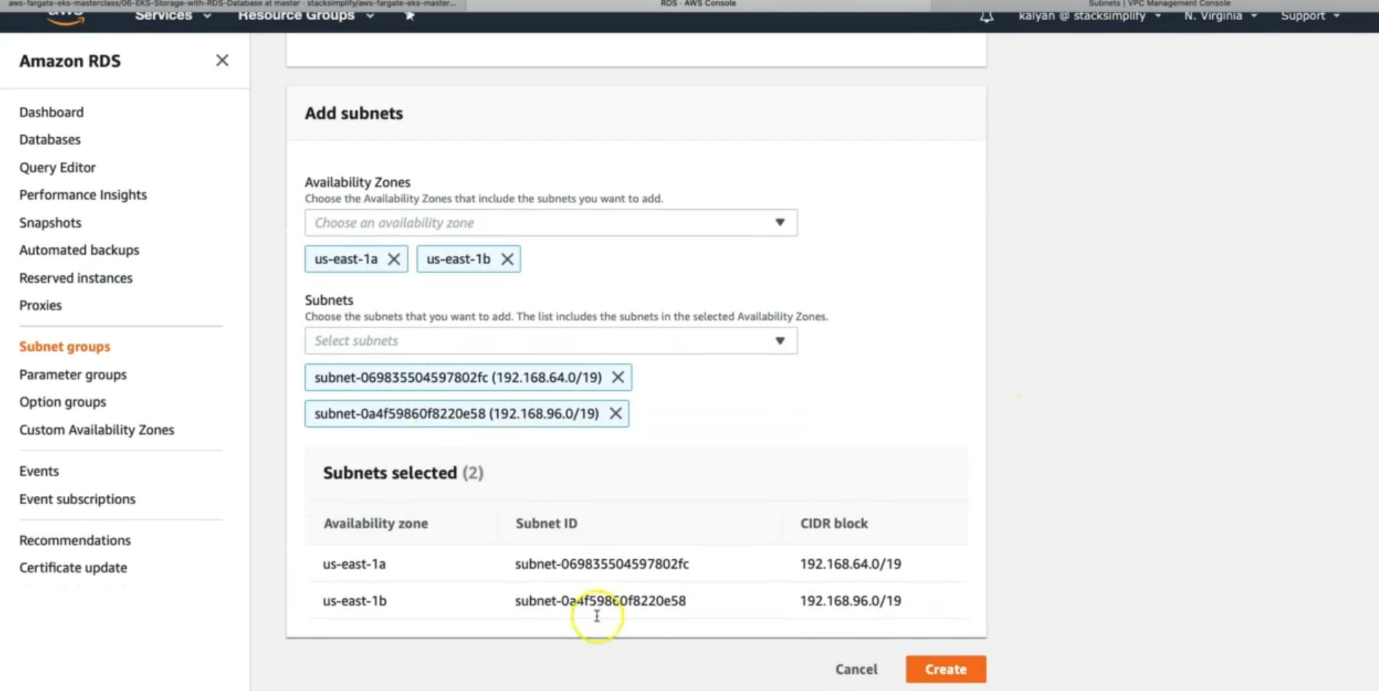
--- VPC: eksctl-eksdemo1-cluster/VPC



--- Availability Zones: us-east-1a, us-east-1b

--- Subnets: 2 subnets in 2 AZs

Click on Create



--- **note** – identify and select private subnets only.

**Create RDS Database**

--- Go to Services -> RDS

--- Click on Create Database

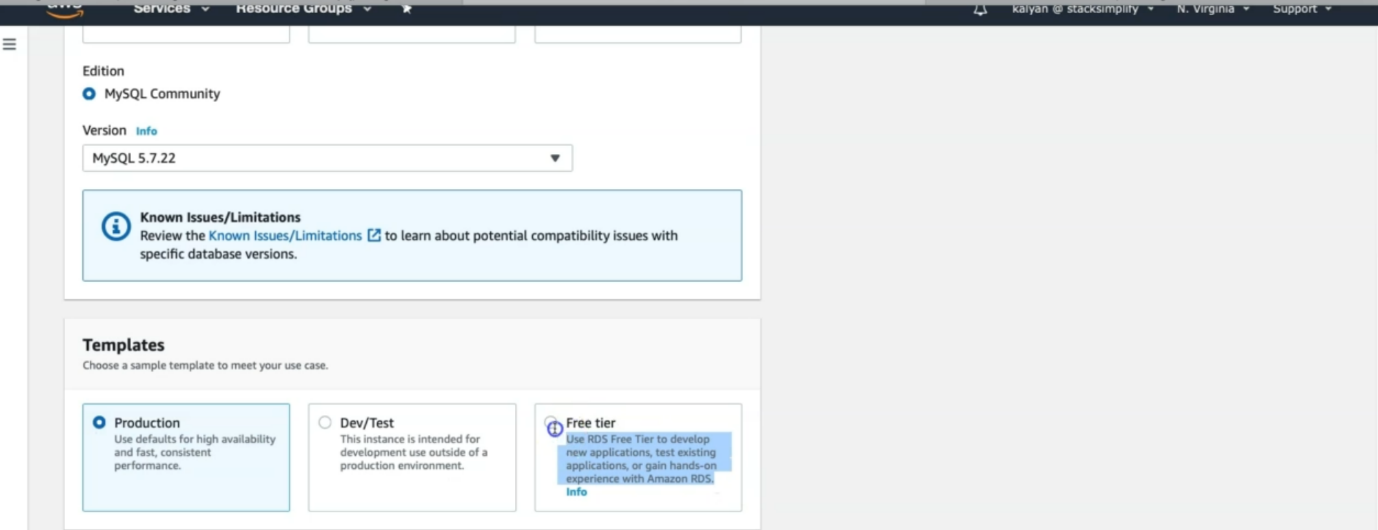
--- Choose a Database Creation Method: Standard Create

--- Engine Options: MySQL

--- Edition: MySQL Community

--- Version: 5.7.22 (default populated)

--- Template Size: Free Tier

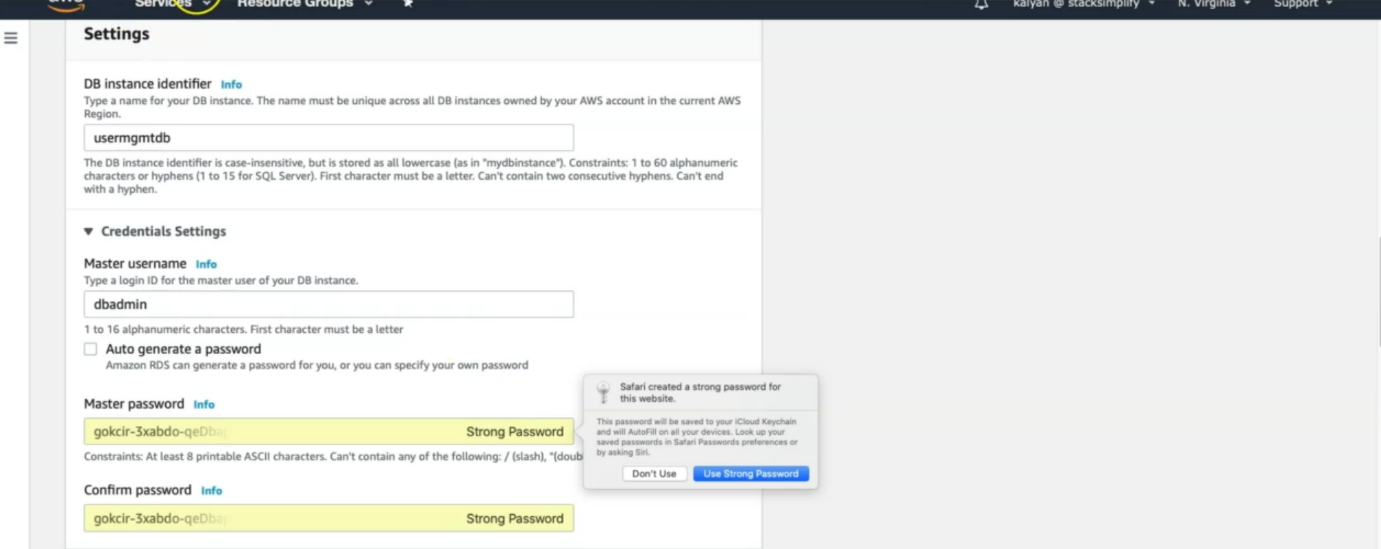


--- DB instance identifier: usermgmtdb

--- Master Username: dbadmin

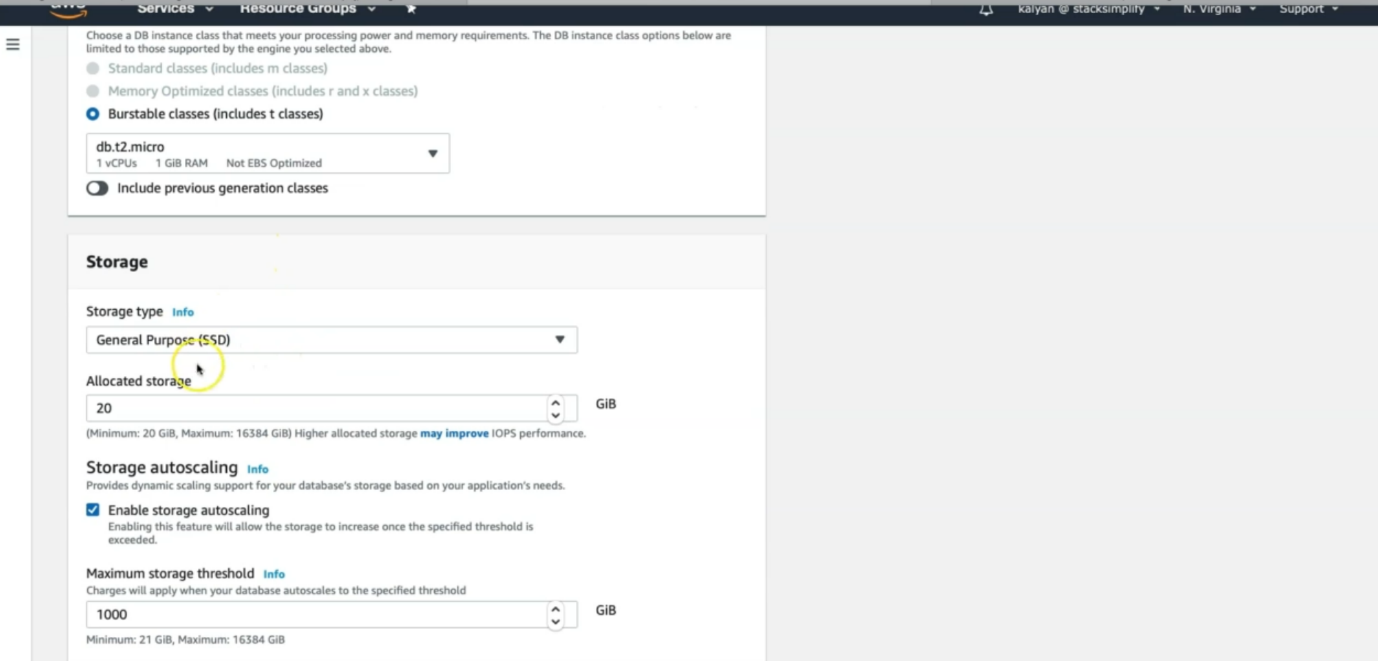
--- Master Password: dbpassword11

--- Confirm Password: dbpassword11



--- DB Instance Size: leave to defaults

--- Storage: leave to defaults

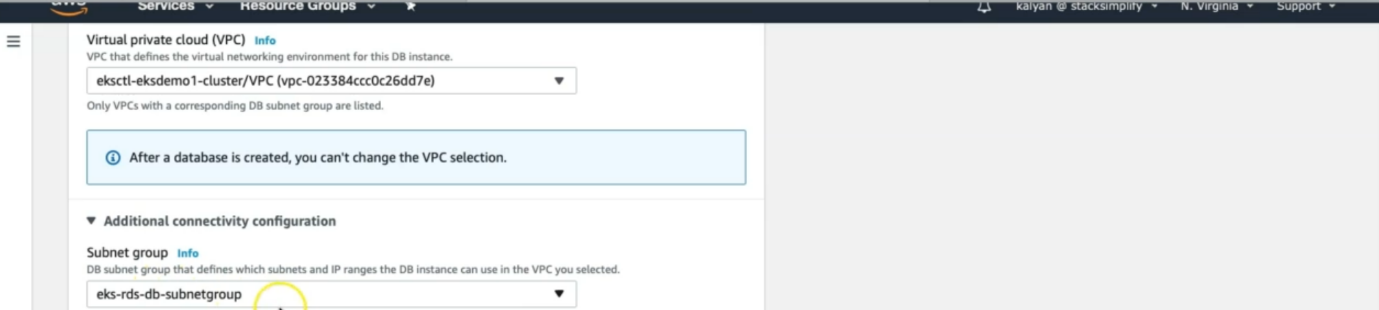


--- Connectivity

--- VPC: eksctl-eksdemo1-cluster/VPC

--- Additional Connectivity Configuration

--- Subnet Group: eks-rds-db-subnetgroup



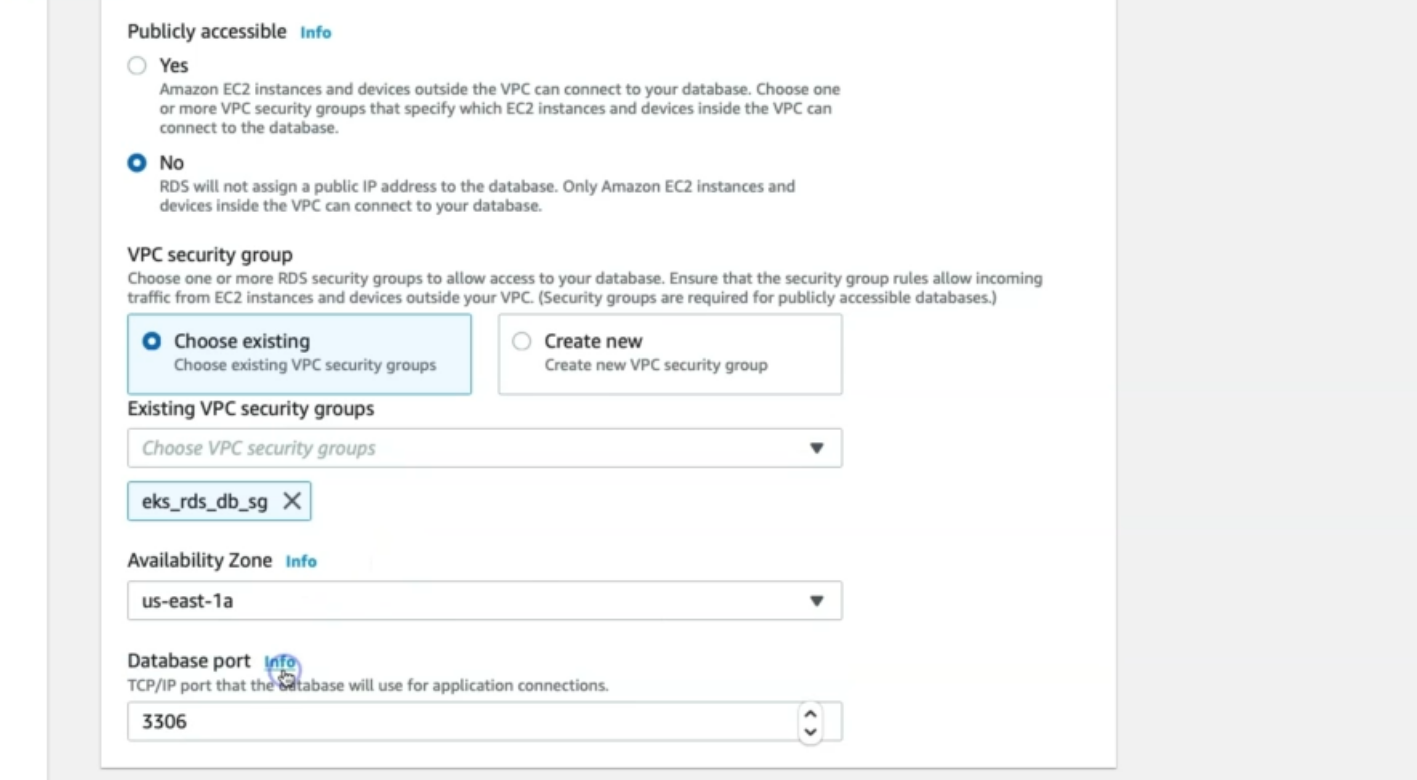
--- Publicly accessible: YES (for our learning and troubleshooting - if required)

--- VPC Security Group: Create New

--- Name: eks-rds-db-securitygroup

--- Availability Zone: No Preference

--- Database Port: 3306



--- **note** – we are creating it in private subnet, so we don’t need to worry about publicly accessible.

--- **note** – we have already created vpc security group, select that security group here and I am creating it here in us-east-1a.

--- Rest all leave to defaults

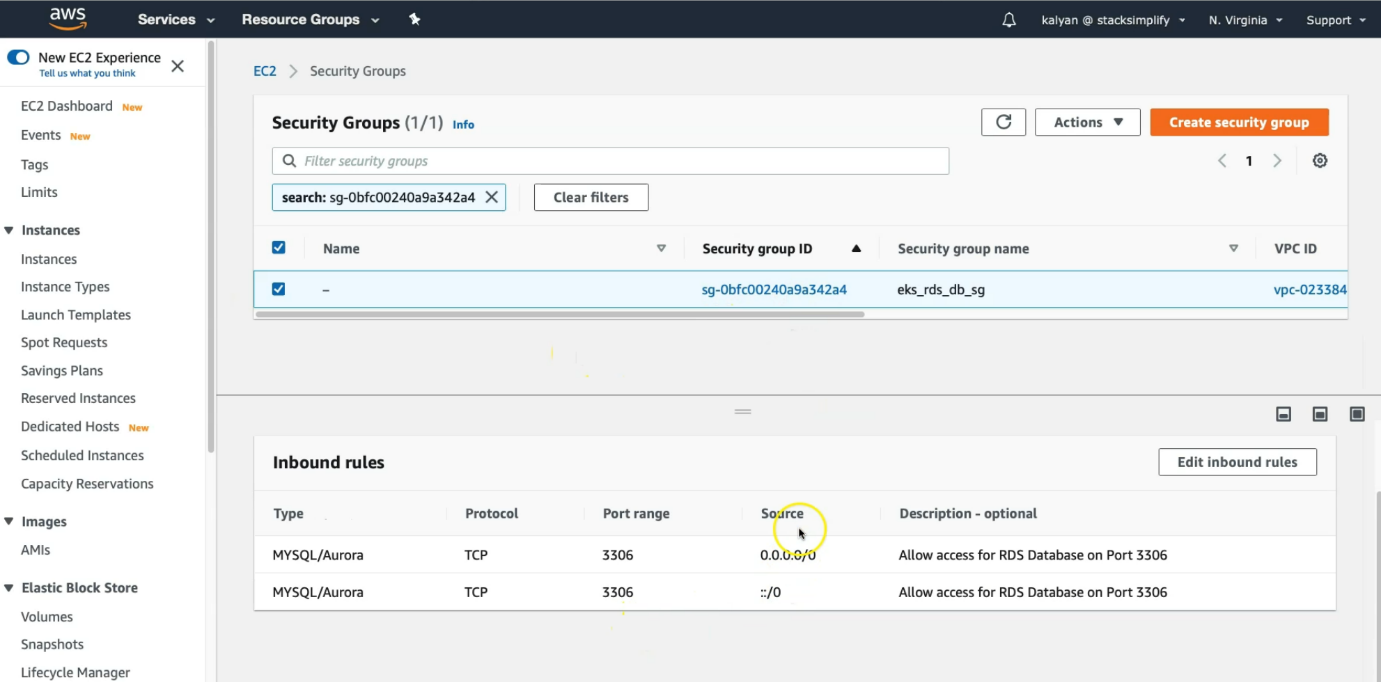
--- Click on Create Database

**Edit Database Security to Allow Access from 0.0.0.0/0**

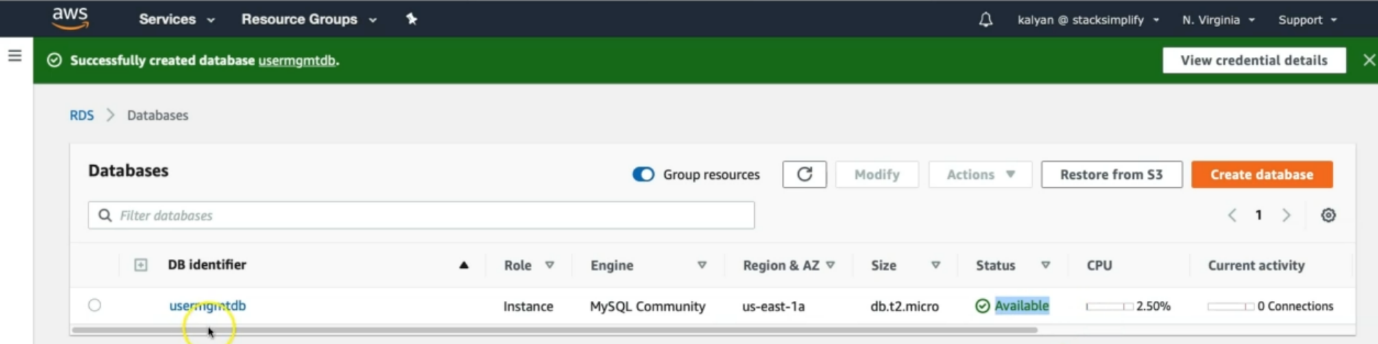
--- Go to EC2 -> Security Groups -> eks-rds-db-securitygroup

--- Edit Inboud Rules

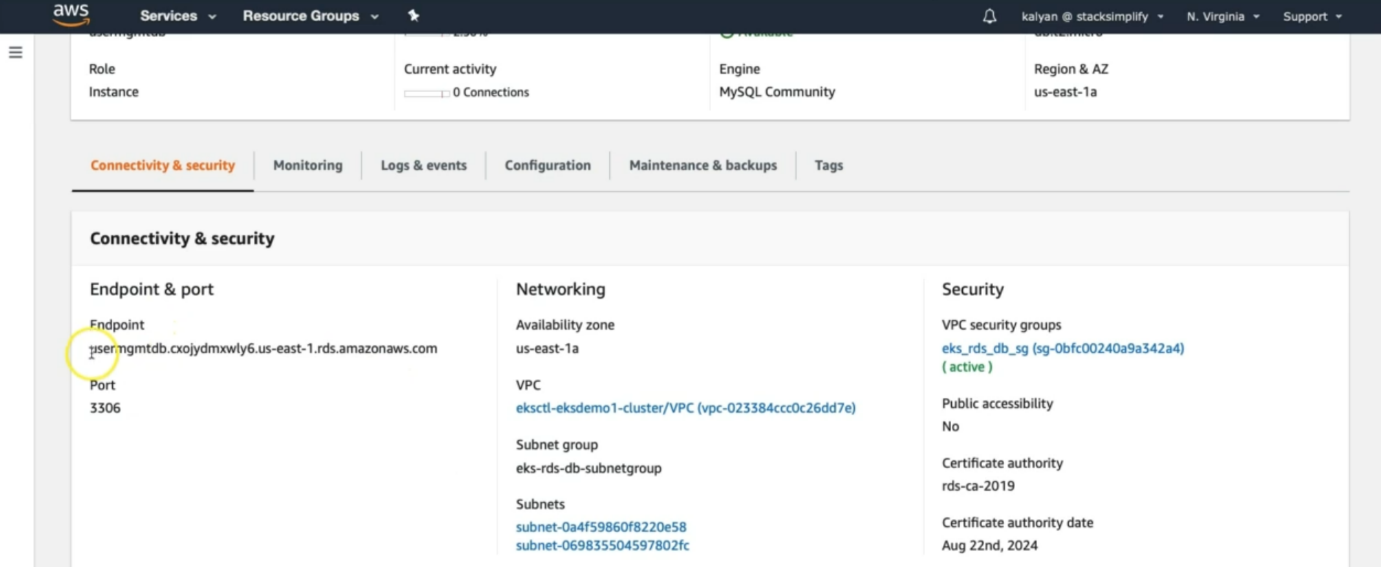
--- Source: Anywhere (0.0.0.0/0) (Allow access from everywhere for now)



**Create Kubernetes externalName service Manifest and Deploy**



--- **note** – our db is available now and click on the db.



--- **note** – you will find the endpoint our db under connectivity and security. This is the endpoint, we will configure in external service.

--- Create mysql externalName Service

--- **01-MySQL-externalName-Service.yml**

apiVersion: v1

kind: Service

metadata:

  name: mysql

spec:

  type: ExternalName

  externalName: usermgmtdb.c7hldelt9xfp.us-east-1.rds.amazonaws.com

**Deploy Manifest**

--- **kubectl apply -f kube-manifests/01-MySQL-externalName-Service.yml**

**Connect to RDS Database using kubectl and create usermgmt schema/db**

--- **kubectl run -it --rm --image=mysql:5.7.22 --restart=Never mysql-client -- mysql -h usermgmtdb.c7hldelt9xfp.us-east-1.rds.amazonaws.com -u dbadmin -pdbpassword11**

--- **note** – we created the db in the private subnet so we cannot connect the db over the internet. We can connect the db using above command from master node.

--- **mysql> show schemas;**

--- **mysql> create database usermgmt;**

--- **mysql> show schemas;**

--- **mysql> exit**

**In User Management Microservice deployment file change username from root to dbadmin**

--- **02-UserManagementMicroservice-Deployment-Service.yml**

# Change From

          - name: DB\_USERNAME

            value: "root"

# Change To

          - name: DB\_USERNAME

            value: "dbadmin"

--- **02-UserManagementMicroservice-Deployment-Service.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: usermgmt-microservice

  labels:

    app: usermgmt-restapp

spec:

  replicas: 1

  selector:

    matchLabels:

      app: usermgmt-restapp

  template:

    metadata:

      labels:

        app: usermgmt-restapp

    spec:

      initContainers:

        - name: init-db

          image: busybox:1.31

          command: ['sh', '-c', 'echo -e "Checking for the availability of MySQL Server deployment"; while ! nc -z mysql 3306; do sleep 1; printf "-"; done; echo -e "  >> MySQL DB Server has started";']

      containers:

        - name: usermgmt-restapp

          image: stacksimplify/kube-usermanagement-microservice:1.0.0

          ports:

            - containerPort: 8095

          env:

            - name: DB\_HOSTNAME

              value: "mysql"

            - name: DB\_PORT

              value: "3306"

            - name: DB\_NAME

              value: "usermgmt"

            - name: DB\_USERNAME

              value: "dbadmin"        # RDS DB Username is dbadmin

            - name: DB\_PASSWORD

              valueFrom:

                secretKeyRef:

                  name: mysql-db-password

                  key: db-password

          livenessProbe:

            exec:

              command:

                - /bin/sh

                - -c

                - nc -z localhost 8095

            initialDelaySeconds: 60

            periodSeconds: 10

          readinessProbe:

            httpGet:

              path: /usermgmt/health-status

              port: 8095

            initialDelaySeconds: 60

            periodSeconds: 10

**Deploy User Management Microservice and Test**

**# Deploy all Manifests**

--- **kubectl apply -f kube-manifests/**

**# List Pods**

--- **kubectl get pods**

**# Stream pod logs to verify DB Connection is successful from SpringBoot Application**

--- **kubectl logs -f <pod-name>**

**Access Application**

**# Capture Worker Node External IP or Public IP**

--- **kubectl get nodes -o wide**

**# Access Application**

--- **http://<Worker-Node-Public-Ip>:31231/usermgmt/health-status**

**Clean Up**

**# Delete all Objects created**

--- **kubectl delete -f kube-manifests/**

**# Verify current Kubernetes Objects**

--- **kubectl get all**