Use the CDK environment created from the Must Do section of Part 1 and create a sample CDK Project.

```
lib > TS part2_cdk-stack.ts > Part2CdkStack > constructor
                                                          Find  Aa  ab  .*  No results
export class Part2CdkStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    //  s3 bucket with lifecycle transitions and KMS encryption enabled
    const bucket = new s3.Bucket(this, 'MyEncryptedBucket', {
      encryption: s3.BucketEncryption.KMS,
      versioned:true,
      bucketName:"part2asigncdklbhhgbgboo007",
      lifecycleRules: [
      {
      transitions: [
      {
      storageClass: s3.StorageClass.INFREQUENT_ACCESS,
      transitionAfter: cdk.Duration.days(30),
      },
      {
      storageClass: s3.StorageClass.GLACIER,
      transitionAfter: cdk.Duration.days(90),
      },
      ],
      },
      ],
      });
```

Step 2: Create a CDK code to deploy an EC2 instance using Amazon Linux v2 AMI.

->First created VPC where we will launch instance-

```
    const vpc = new ec2.Vpc(this, 'my-cdk-vpc', {
    cidr: '10.0.0.0/16',
    natGateways: 0,
    subnetConfiguration: [
      {name: 'public', cidrMask: 24, subnetType: ec2.SubnetType.PUBLIC},
    ],
  });
```

-> creating security groups

```
    const webserverSG = new ec2.SecurityGroup(this, 'webserver-sg', {
      vpc,
      allowAllOutbound: true,
    });

    webserverSG.addIngressRule(
      ec2.Peer.anyIpv4(),
      ec2.Port.tcp(22),
      'allow SSH access from anywhere',
    );

    webserverSG.addIngressRule(
      ec2.Peer.anyIpv4(),
      ec2.Port.tcp(80),
      'allow HTTP traffic from anywhere',
    );

    webserverSG.addIngressRule(
      ec2.Peer.anyIpv4(),
      ec2.Port.tcp(443),
      'allow HTTPS traffic from anywhere',
    );
```

-> role for EC2 instance:

```
    const webserverRole = new iam.Role(this, 'webserver-role', {
      assumedBy: new iam.ServicePrincipal('ec2.amazonaws.com'),
      managedPolicies: [

iam.ManagedPolicy.fromAwsManagedPolicyName('AmazonS3ReadOnlyAccess'),
      ],
    });
```

->EC2 instance with using Amazon linux

```
    const ec2Instance = new ec2.Instance(this, 'ec2-instance', {
      vpc,
      vpcSubnets: {
        subnetType: ec2.SubnetType.PUBLIC,
      },
      role: webserverRole,
      securityGroup: webserverSG,
      instanceType: ec2.InstanceType.of(
        ec2.InstanceClass.BURSTABLE2,
        ec2.InstanceSize.MICRO,
      ),
      machineImage: new ec2.AmazonLinuxImage({
        generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX_2,
      }),
      keyName: 'assignment',
    });
```

Step 3: Using CDK code create a custom config set and have a config to install nginx server on the Instance and also added config to update the host name of the Instance.

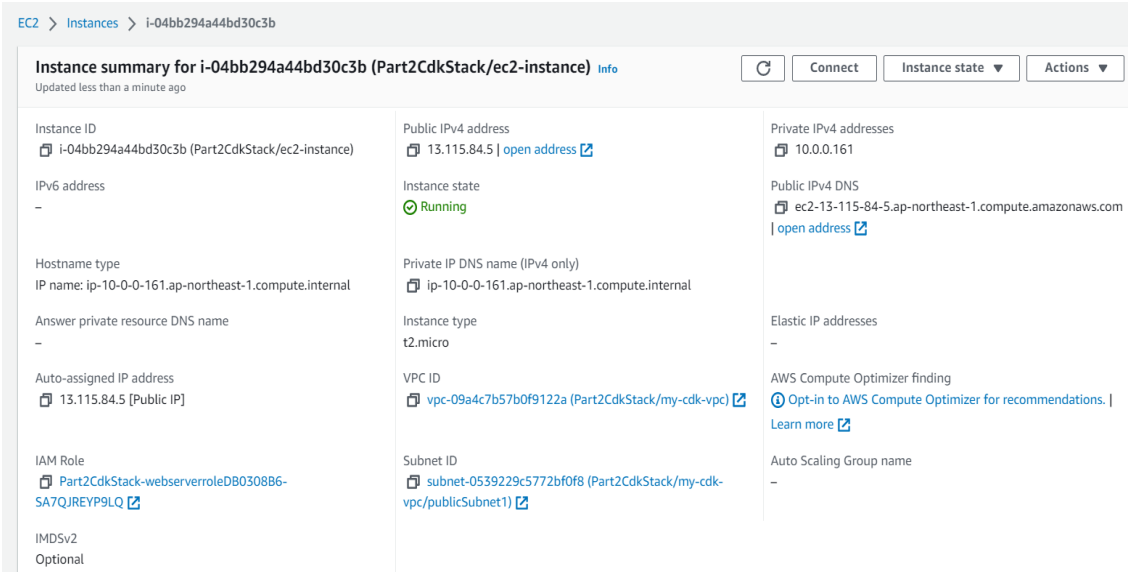
```
#!/bin/bash
yum update -y
sudo su

amazon-linux-extras install -y nginx1
systemctl start nginx
systemctl enable nginx
sudo hostnamectl set-hostname part2assign4
sudo systemctl restart systemd-hostnamed
```

Step 4: Use the s3 Bucket from the Must Do Section of CDK Part 1 and set up an optional Integration by granting read-write permissions to the Bucket (Once we log into the Instance, we should be able to access the objects, and also able to upload the objects to the Bucket)

```
    bucket.grantReadWrite(ec2Instance);
```

->Instance launched



-> ngnix server

```
[ec2-user@part2assign4 ~]$ sudo systemctl statis
Unknown operation 'statis'.
[ec2-user@part2assign4 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2023-03-31 20:30:05 UTC; 2 days ago
 Main PID: 3256 (nginx)
   CGroup: /system.slice/nginx.service
           ├─3256 nginx: master process /usr/sbin/nginx
           └─3258 nginx: worker process

Mar 31 20:30:04 part2assign4 systemd[1]: Starting The nginx HTTP and reverse proxy server...
Mar 31 20:30:05 part2assign4 systemd[1]: Started The nginx HTTP and reverse proxy server.
Mar 31 20:30:07 part2assign4 nginx[3248]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Mar 31 20:30:07 part2assign4 nginx[3248]: nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@part2assign4 ~]$
```

i-04bb294a44bd30c3b (Part2CdkStack/ec2-instance)

PublicIPs: 13.115.84.5 PrivateIPs: 10.0.0.161

-> granted read-write permissions to the Bucket, we can view the objects in the bucket

```
Last login: Fri Mar 31 20:32:49 2023 from ec2-3-112-23-3.ap-northeast-1.compute.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@part2assign4 ~]$ aws s3 ls s3://part2asigncdklbhhgbgboo007
[ec2-user@part2assign4 ~]$ aws s3 ls s3://part2asigncdklbhhgbgboo007
2023-03-31 20:58:33     574838 JIO UPLOAD.pdf
2023-03-31 20:58:34      46392 generic-food.csv
[ec2-user@part2assign4 ~]$
```

i-04bb294a44bd30c3b (Part2CdkStack/ec2-instance)

PublicIPs: 13.115.84.5 PrivateIPs: 10.0.0.161