Cross Stack Resource Sharing between CDK Apps using SSM

The architecture is modularized into four independent CDK stacks, each responsible for a specific part of the infrastructure:

1. CDK1 – NetworkStack
    - Provisions a new VPC with CIDR 10.0.0.0/16 and a single public subnet.
    - This VPC will be used by other stacks via SSM.
2. CDK2 – SsmStack
    - Stores the manually fetched VPC ID (after CDK1 deployment) in SSM Parameter Store under the key /vpc/NetworkVPC.
    - Enables loose coupling by allowing other stacks to reference the VPC without tight dependencies.
3. CDK3 – WebserverStack
    - Fetches the VPC ID from SSM, uses ec2.Vpc.fromLookup() to import it.
    - Reads an IAM Role ARN from CDK4 via cdk.Fn.importValue.
    - Creates a Security Group with HTTP access on port 80.
    - Launches an EC2 instance in the imported VPC using the IAM role and security group.
    - Uses a custom userdata.sh script to bootstrap the EC2 instance (e.g., install web server).
    - Ensures the EC2 instance is launched in a public subnet with a public IP for accessibility.
4. CDK4 – SecurityStack
    - Creates an IAM Role assumable by EC2 (ec2.amazonaws.com) with AmazonS3FullAccess policy.
    - Exports the IAM Role ARN using CloudFormation CfnOutput, making it reusable in other stacks like CDK3.

Github link - https://github.com/likhith68/Resources-Sharing-bet-CDK-Apps/tree/main

Code -

network-stack.ts

```typescript
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as ec2 from 'aws-cdk-lib/aws-ec2'

export class NetworkStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    //Create a basic VPC with one public subnet and CIDR - 10.0.0.0/16
    const demoNetworkVpc = new ec2.Vpc(this, 'DemoVpcLID', {
      ipAddresses: ec2.IpAddresses.cidr('10.0.0.0/16'),
      maxAzs: 1, // Default is all AZs in the region
      vpcName : "NetworkVPC",
    });

  }
}
```

security-stack.ts

```typescript
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
// import * as sqs from 'aws-cdk-lib/aws-sqs';
import * as iam from 'aws-cdk-lib/aws-iam';

export class SecurityStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // Create an IAM Role that can be used by EC2 instances
    const demoRole = new iam.Role(this, 'DemoRole', {
      assumedBy: new cdk.aws_iam.ServicePrincipal('ec2.amazonaws.com'),
    });

    demoRole.addManagedPolicy(
      iam.ManagedPolicy.fromAwsManagedPolicyName('AmazonS3FullAccess')
    );

    //Use CfnOutput to make use of this role in the Webserver Stack
    const demoIam = new cdk.CfnOutput(this, 'DemoIamRole', {
      value: demoRole.roleArn,
      description: 'The IAM Role ARN for EC2 instances',
      exportName: 'DemoIamRoleArn',});
    }
}
```

Sharing Resources

EXPLORER

OPEN EDITORS

SHARING RESOURCES

- network
- security
- ssm
  - bin
  - cdk.out
  - lib
    - ssm-stack.ts M
  - node_modules
  - test
  - .gitignore
  - .npmignore
  - cdk.json
  - jest.config.js
  - package-lock.json
  - package.json
  - README.md
  - tsconfig.json
- webserver
- Project 5 - Cross-Stack Resource ... U

OUTLINE

TIMELINE

APPLICATION BUILDER

ssm > lib > ssm-stack.ts > ...

```typescript
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
// import * as sqs from 'aws-cdk-lib/aws-sqs';
import * as ssm from 'aws-cdk-lib/aws-ssm';

export class SsmStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // Create a basic SSM Parameter Store parameter
    const demoSSM = new ssm.StringParameter(this, 'demoSsmLID', {
      parameterName: '/vpc/NetworkVPC',
      stringValue: 'vpc-0c25bcf7b2e187421'
    });
  }
}
```

main* | 0 0 | AWS: profile:default | Amazon Q | Ln 17, Col 1 | Spaces: 2 | UTF-8 | LF | TypeScript

Sharing Resources

EXPLORER

OPEN EDITORS

SHARING RESOURCES

- network
- security
- ssm
- webserver
  - bin
  - cdk.out
  - lib
    - userdata.sh
    - webserver-stack.ts M
  - node_modules
  - test
  - .gitignore
  - .npmignore
  - cdk.context.json M
  - cdk.json
  - jest.config.js
  - package-lock.json
  - package.json
  - README.md
  - tsconfig.json
- Project 5 - Cross-Stack Resource ... U

OUTLINE

TIMELINE

APPLICATION BUILDER

webserver > lib > webserver-stack.ts > WebserverStack > constructor

```typescript
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as ssm from 'aws-cdk-lib/aws-ssm';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as fs from 'fs';
import * as path from 'path';
import * as iam from 'aws-cdk-lib/aws-iam';

export class WebserverStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // Use the VPC ID from SSm Parameter Store
    const vpcId = ssm.StringParameter.valueFromLookup(this, '/vpc/NetworkVPC');
    const vpcFromSSM = ec2.Vpc.fromLookup(this, 'VPCPickedFromSSM', { vpcId:vpcId });


    // Import the IAM Role ARN from the Security Stack
    const importedRoleArn = iam.Role.fromRoleArn(this, 'ImportedRoleArn', cdk.Fn.importValue('DemoIamRoleArn'));

    //Create a security group
    const demoSecurityGroup = new ec2.SecurityGroup(this, 'DemoSecurityGroup', {
      vpc: vpcFromSSM,
      description: 'Demo Security Group',
      allowAllOutbound: true,
    });

    // Add an ingress rule to allow HTTP traffic
    demoSecurityGroup.addIngressRule(
      ec2.Peer.anyIpv4(),
      ec2.Port.tcp(80),
      'Allow HTTP traffic from anywhere'
    );

    // Create an EC2 instance in the VPC with the security group
    const demoInstance = new ec2.Instance(this, 'DemoInstanceLID', {
      vpc: vpcFromSSM,
      instanceType: new ec2.InstanceType('t2.micro'),
      role: importedRoleArn,
      machineImage: new ec2.AmazonLinuxImage({
        generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX_2,
        edition: ec2.AmazonLinuxEdition.STANDARD,
        virtualization: ec2.AmazonLinuxVirt.HVM,
        storage: ec2.AmazonLinuxStorage.GENERAL_PURPOSE,
      }),
```

main* | 0 0 | AWS: profile:default | Amazon Q | Likhith JSC (1 day ago) | Ln 12, Col 1 | Spaces: 2 | UTF-8 | LF | TypeScript

webserver > lib > webserver-stack.ts > WebserverStack > constructor

```typescript
export class WebserverStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {

    // Import the IAM Role ARN from the Security Stack
    const importedRoleArn = iam.Role.fromRoleArn(this, 'ImportedRoleArn', cdk.Fn.importValue('DemoIamRoleArn'));

    //Create a security group
    const demoSecurityGroup = new ec2.SecurityGroup(this, 'DemoSecurityGroup', {
      vpc: vpcFromSSM,
      description: 'Demo Security Group',
      allowAllOutbound: true,
    });

    // Add an ingress rule to allow HTTP traffic
    demoSecurityGroup.addIngressRule(
      ec2.Peer.anyIpv4(),
      ec2.Port.tcp(80),
      'Allow HTTP traffic from anywhere'
    );

    // Create an EC2 instance in the VPC with the security group
    const demoInstance = new ec2.Instance(this, 'DemoInstanceLID', {
      vpc: vpcFromSSM,
      instanceType: new ec2.InstanceType('t2.micro'),
      role: importedRoleArn,
      machineImage: new ec2.AmazonLinuxImage({
        generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX_2,
        edition: ec2.AmazonLinuxEdition.STANDARD,
        virtualization: ec2.AmazonLinuxVirt.HVM,
        storage: ec2.AmazonLinuxStorage.GENERAL_PURPOSE,
      }),
      securityGroup: demoSecurityGroup,
      associatePublicIpAddress: true,
      vpcSubnets: { subnetType: ec2.SubnetType.PUBLIC },
    });

  const userDataScript = fs.readFileSync(path.join(__dirname, 'userdata.sh'), 'utf8');
   userDataScript.split('\n').forEach(line => {
     demoInstance.userData.addCommands(line);
   });

   demoInstance.userData.addCommands(userDataScript);
   }
  }
```

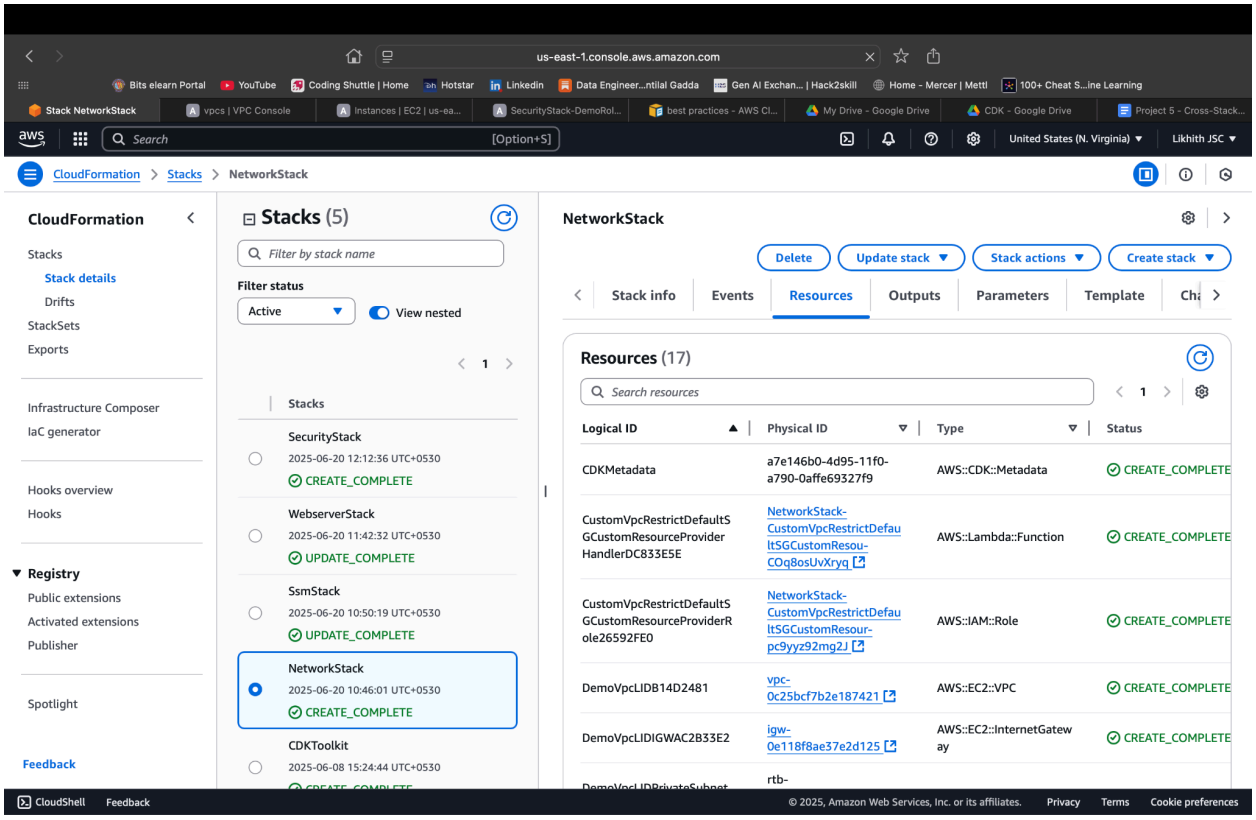
webserver > lib > $ userdata.sh

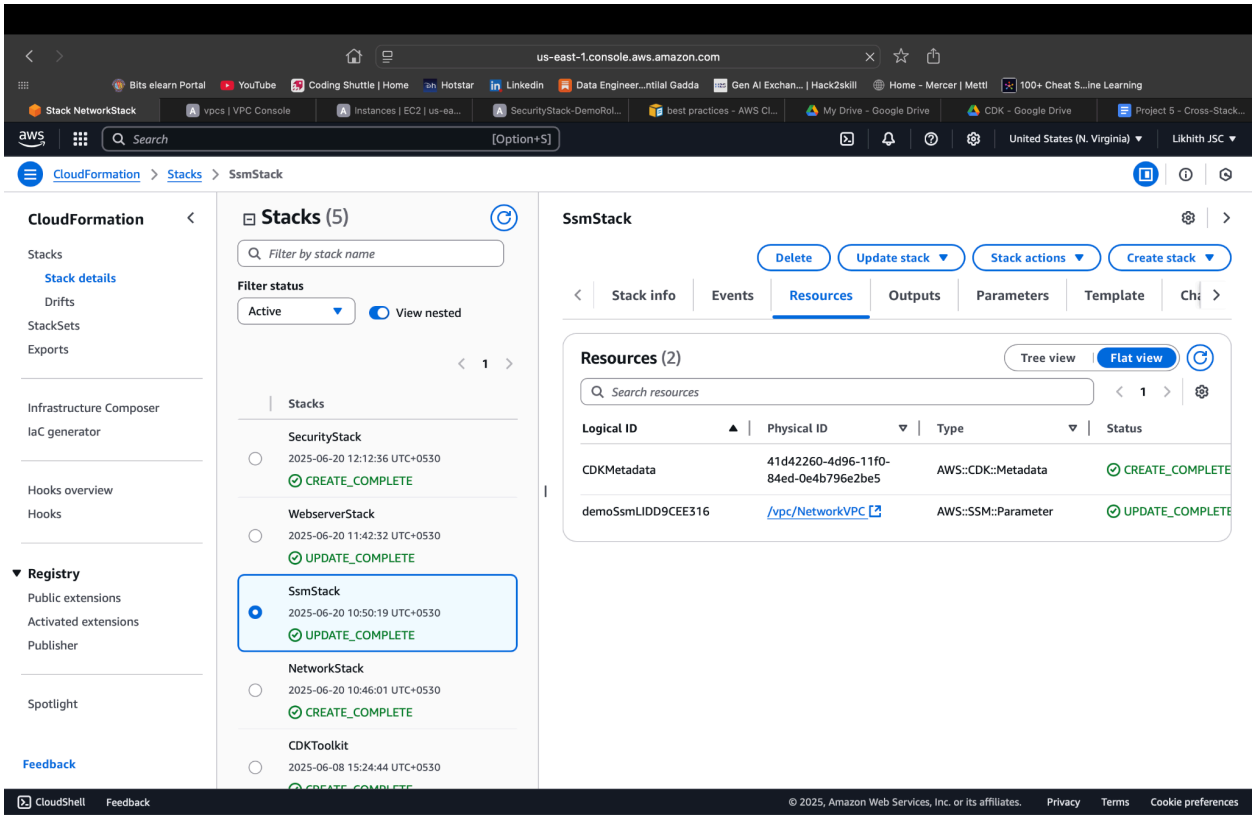
```bash
#!/bin/bash

sudo su
yum update -y
yum install -y httpd

systemctl start httpd
systemctl enable httpd
echo "<h1>Welcome to Likhith's Web Server</h1>" > /var/www/html/index.html
```

All 4 Stacks got created in Cloudformation -





us-east-1.console.aws.amazon.com

Bits elearn Portal | YouTube | Coding Shuttle | Home | Hotstar | Linkedin | Data Engineer...ntilal Gadda | Gen AI Exchan... | Hack2skill | Home - Mercer | Mettl | 100+ Cheat S...ine Learning

Stack NetworkStack | vpcs | VPC Console | Instances | EC2 | us-ea... | SecurityStack-DemoRol... | best practices - AWS Cl... | My Drive - Google Drive | CDK - Google Drive | Project 5 - Cross-Stack...

aws | Search [Option+S] | United States (N. Virginia) | Likhith JSC

CloudFormation > Stacks > WebserverStack

CloudFormation

- Stacks
  - Stack details
  - Drifts
- StackSets
- Exports
- Infrastructure Composer
- IaC generator
- Hooks overview
- Hooks
- **Registry**
  - Public extensions
  - Activated extensions
  - Publisher
- Spotlight

Feedback

Stacks (5)

Filter by stack name

Filter status: Active | View nested

Stacks	
SecurityStack	2025-06-20 12:12:36 UTC+0530 — CREATE_COMPLETE
WebserverStack	2025-06-20 11:42:32 UTC+0530 — UPDATE_COMPLETE
SsmStack	2025-06-20 10:50:19 UTC+0530 — UPDATE_COMPLETE
NetworkStack	2025-06-20 10:46:01 UTC+0530 — CREATE_COMPLETE
CDKToolkit	2025-06-08 15:24:44 UTC+0530

WebserverStack

Delete | Update stack | Stack actions | Create stack

Stack info | Events | Resources | Outputs | Parameters | Template | Cha

Resources (4)

Tree view | Flat view

Search resources

Logical ID	Physical ID	Type	Status
CDKMetadata	8d38cec0-4d9d-11f0-8bf9-0edfb9fb5273	AWS::CDK::Metadata	UPDATE_COMPLETE
DemoInstanceLID1DD7CB43	i-05fdf276425672bad	AWS::EC2::Instance	CREATE_COMPLETE
DemoInstanceLIDInstanceProfile5AD6EEA7	WebserverStack-DemoInstanceLIDInstanceProfile5AD6EEA7-zTQKRmn4a1JO	AWS::IAM::InstanceProfile	UPDATE_COMPLETE
DemoSecurityGroupA88C9F8E	sg-0841a0bb965189637	AWS::EC2::SecurityGroup	CREATE_COMPLETE

CloudShell | Feedback | © 2025, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences

us-east-1.console.aws.amazon.com

Bits elearn Portal | YouTube | Coding Shuttle | Home | Hotstar | Linkedin | Data Engineer...ntilal Gadda | Gen AI Exchan... | Hack2skill | Home - Mercer | Mettl | 100+ Cheat S...ine Learning

Stack NetworkStack | vpcs | VPC Console | Instances | EC2 | us-ea... | SecurityStack-DemoRol... | best practices - AWS Cl... | My Drive - Google Drive | CDK - Google Drive | Project 5 - Cross-Stack...

aws | Search [Option+S] | United States (N. Virginia) | Likhith JSC

CloudFormation > Stacks > SecurityStack

CloudFormation

- Stacks
  - Stack details
  - Drifts
- StackSets
- Exports
- Infrastructure Composer
- IaC generator
- Hooks overview
- Hooks
- **Registry**
  - Public extensions
  - Activated extensions
  - Publisher
- Spotlight

Feedback

Stacks (5)

Filter by stack name

Filter status: Active | View nested

Stacks	
SecurityStack	2025-06-20 12:12:36 UTC+0530 — CREATE_COMPLETE
WebserverStack	2025-06-20 11:42:32 UTC+0530 — UPDATE_COMPLETE
SsmStack	2025-06-20 10:50:19 UTC+0530 — UPDATE_COMPLETE
NetworkStack	2025-06-20 10:46:01 UTC+0530 — CREATE_COMPLETE
CDKToolkit	2025-06-08 15:24:44 UTC+0530

SecurityStack

Delete | Update stack | Stack actions | Create stack

Stack info | Events | Resources | Outputs | Parameters | Template | Cha

Resources (2)

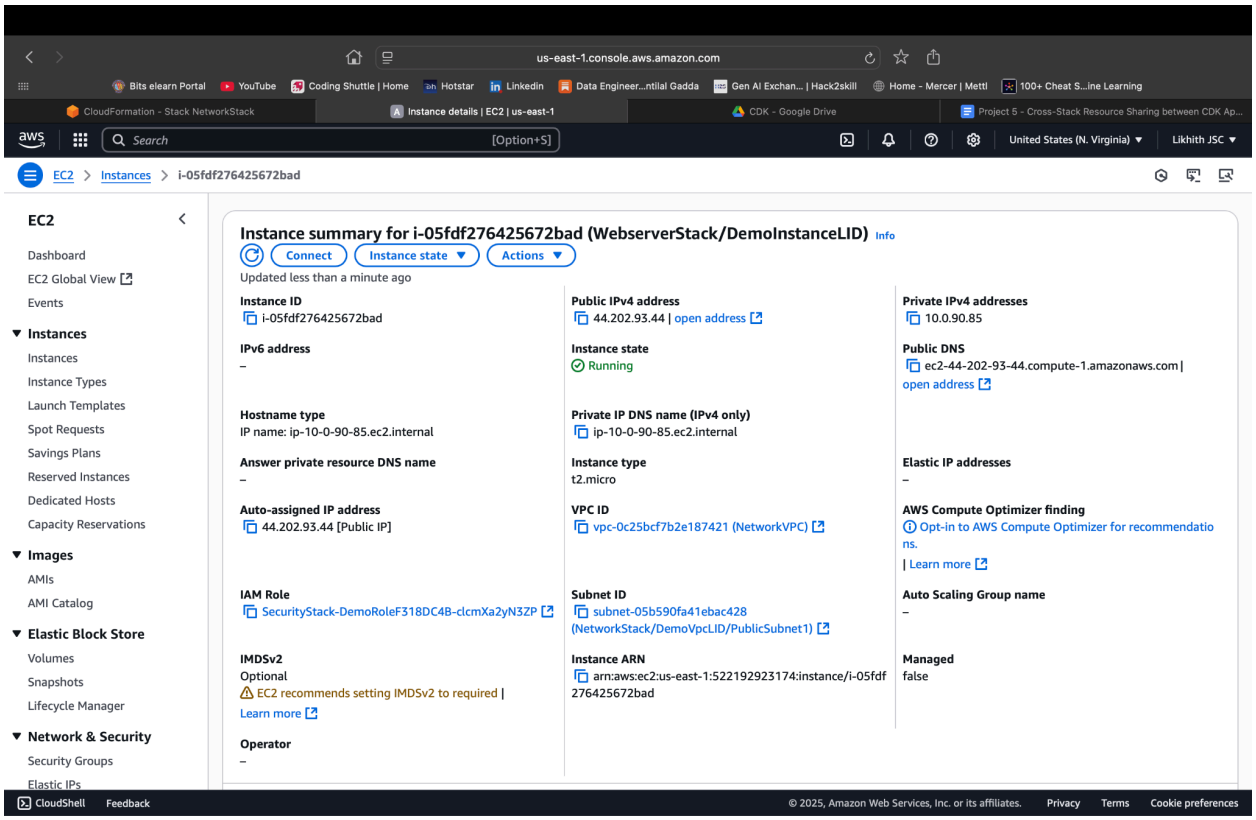
Tree view | Flat view

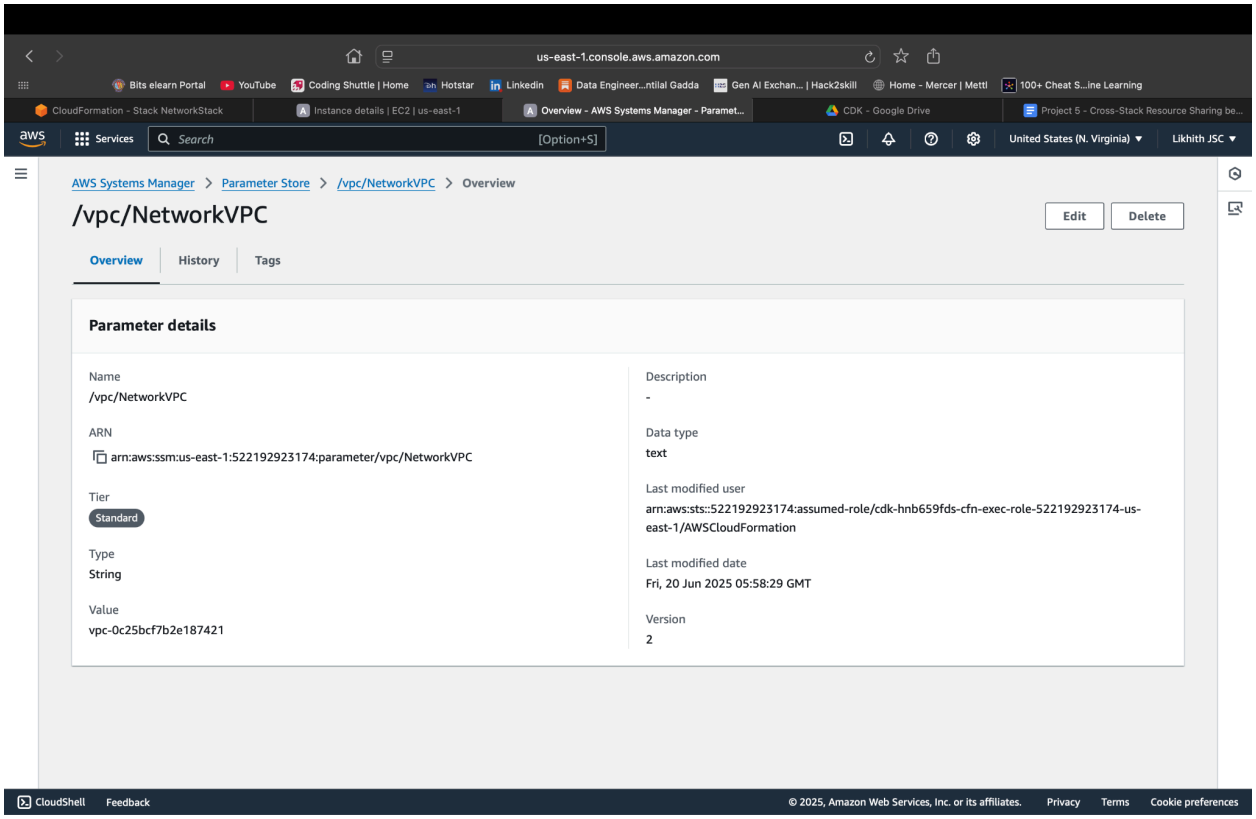
Search resources

Logical ID	Physical ID	Type	Status
CDKMetadata	c08a7c20-4da1-11f0-8909-0eeb6b44d95f	AWS::CDK::Metadata	CREATE_COMPLETE
DemoRoleF318DC4B	SecurityStack-DemoRoleF318DC4B-clcmXa2yN3ZP	AWS::IAM::Role	CREATE_COMPLETE

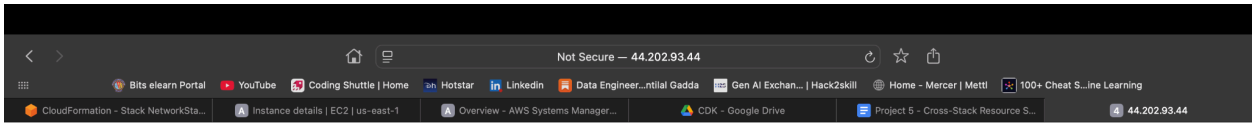
CloudShell | Feedback | © 2025, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences

EC2 Instance from the WebserverStack is created inside the VPC from the Network Stack using the VPC ID picked from the SMM.





WebServer is running successfully -



Welcome to Likhith's Web Server