

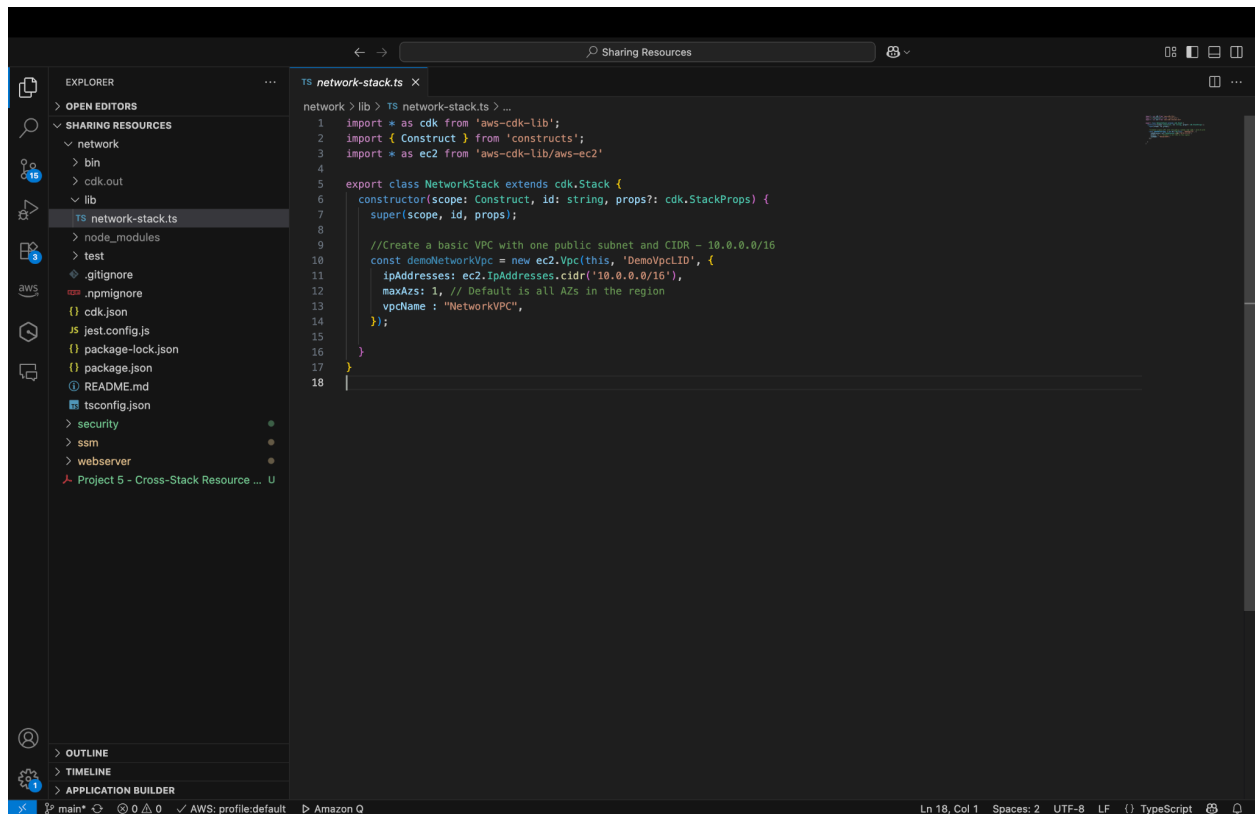
Cross Stack Resource Sharing between CDK Apps using SSM

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.

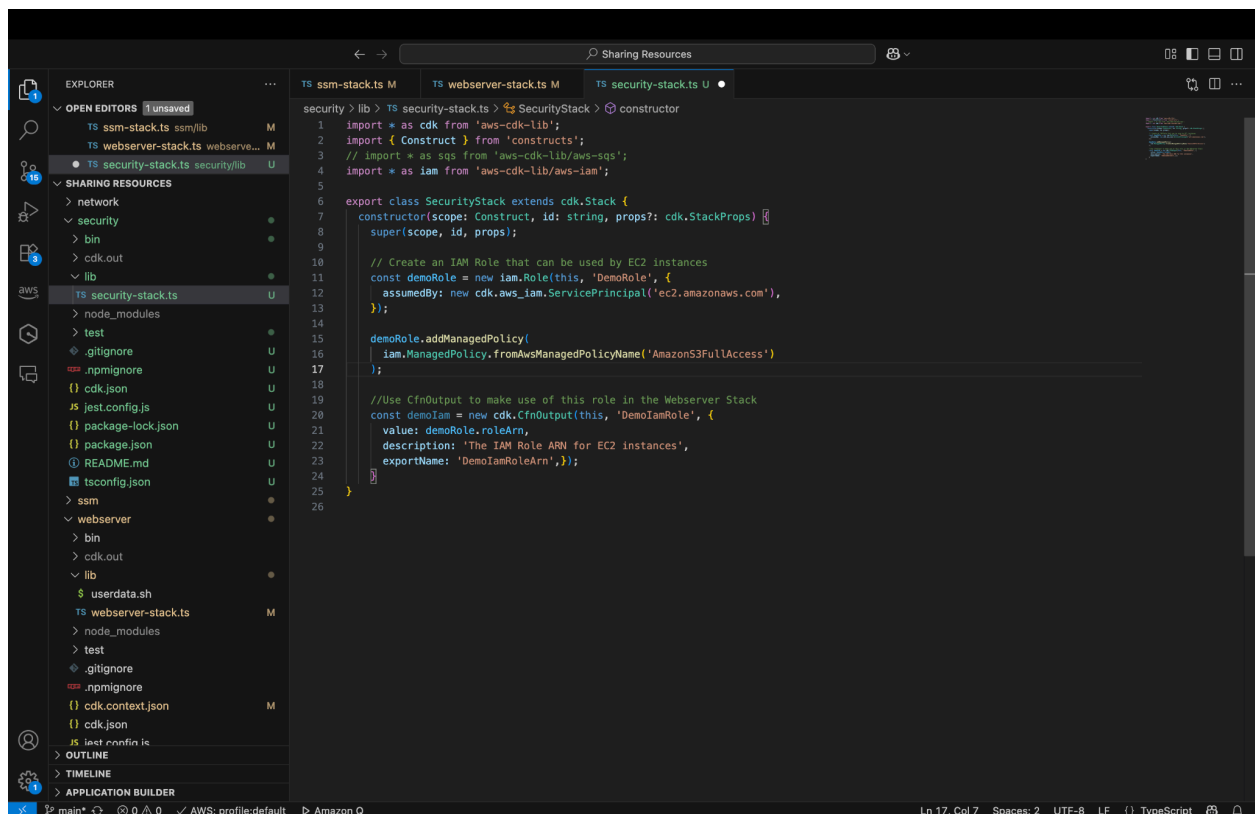
Github link - <https://github.com/likhith68/Resources-Sharing-bet-CDK-Apps/tree/main>

Code -



The screenshot shows the VS Code editor interface with the Explorer sidebar on the left. The Explorer sidebar is expanded to show the 'network' folder, which contains subfolders 'bin', 'cdk.out', and 'lib'. The 'lib' folder is selected, showing the 'network-stack.ts' file. The main editor area displays the code for 'network-stack.ts'. The code defines a 'NetworkStack' class that extends 'cdk.Stack'. The constructor takes 'scope', 'id', and 'props' as arguments. The class includes a comment about creating a basic VPC with one public subnet and a CIDR of 10.0.0.0/16. It defines a 'demoNetworkVpc' as a new 'ec2.Vpc' with the 'demoVpcID' and 'ipAddresses' set to 'ec2.IpAddresses.cidr('10.0.0.0/16')'. It also sets 'maxAzs' to 1 and 'vpcName' to 'NetworkVPC'.

```
network > lib > TS network-stack.ts > ...
1 import * as cdk from 'aws-cdk-lib';
2 import { Construct } from 'constructs';
3 import * as ec2 from 'aws-cdk-lib/aws-ec2';
4
5 export class NetworkStack extends cdk.Stack {
6   constructor(scope: Construct, id: string, props?: cdk.StackProps) {
7     super(scope, id, props);
8
9     //Create a basic VPC with one public subnet and CIDR - 10.0.0.0/16
10    const demoNetworkVpc = new ec2.Vpc(this, 'DemoVpcID', {
11      ipAddresses: ec2.IpAddresses.cidr('10.0.0.0/16'),
12      maxAzs: 1, // Default is all AZs in the region
13      vpcName: "NetworkVPC",
14    });
15  }
16 }
17
18
```



The screenshot shows the VS Code editor interface with the Explorer sidebar on the left. The Explorer sidebar is expanded to show the 'security' folder, which contains subfolders 'bin', 'cdk.out', and 'lib'. The 'lib' folder is selected, showing the 'security-stack.ts' file. The main editor area displays the code for 'security-stack.ts'. The code defines a 'SecurityStack' class that extends 'cdk.Stack'. The constructor takes 'scope', 'id', and 'props' as arguments. The class includes a comment about creating an IAM Role that can be used by EC2 instances. It defines a 'demoRole' as a new 'iam.Role' with the 'demoRole' and 'assumedBy' set to 'cdk.aws_iam.ServicePrincipal('ec2.amazonaws.com')'. It also adds a managed policy 'AmazonS3FullAccess' to the role. The class includes a comment about using 'CfnOutput' to make use of this role in the 'Webserver Stack'. It defines a 'demoIam' as a new 'cdk.CfnOutput' with the 'demoIamRole' and 'value' set to 'demoRole.roleArn'. It also sets the 'description' to 'The IAM Role ARN for EC2 instances' and the 'exportName' to 'DemoIamRoleArn'.

```
security > lib > TS security-stack.ts > SecurityStack > constructor
1 import * as cdk from 'aws-cdk-lib';
2 import { Construct } from 'constructs';
3 // import * as sqs from 'aws-cdk-lib/aws-sqs';
4 import * as iam from 'aws-cdk-lib/aws-iam';
5
6 export class SecurityStack extends cdk.Stack {
7   constructor(scope: Construct, id: string, props?: cdk.StackProps) {
8     super(scope, id, props);
9
10    // Create an IAM Role that can be used by EC2 instances
11    const demoRole = new iam.Role(this, 'DemoRole', {
12      assumedBy: new cdk.aws_iam.ServicePrincipal('ec2.amazonaws.com'),
13    });
14
15    demoRole.addManagedPolicy(
16      iam.ManagedPolicy.fromAwsManagedPolicyName('AmazonS3FullAccess')
17    );
18
19    //Use CfnOutput to make use of this role in the Webserver Stack
20    const demoIam = new cdk.CfnOutput(this, 'DemoIamRole', {
21      value: demoRole.roleArn,
22      description: 'The IAM Role ARN for EC2 instances',
23      exportName: 'DemoIamRoleArn',
24    });
25  }
26 }
27
```

The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer sidebar is expanded to show the 'ssh' folder, which contains the 'ssm-stack.ts' file. The 'ssm-stack.ts' file is selected and its content is displayed in the main editor area. The code defines a CDK stack named 'SsmStack' that extends 'cdk.Stack'. It imports necessary modules from 'aws-cdk-lib' and 'constructs'. The stack's constructor takes a scope, ID, and optional props. It creates a basic SSM Parameter Store parameter named 'demoSsm' with the name '/vpc/NetworkVPC' and a string value 'vpc-0c29bcf7b2e187421'. The stack is exported as 'SsmStack'.

```
1 import * as cdk from 'aws-cdk-lib';
2 import { Construct } from 'constructs';
3 // import * as sqs from 'aws-cdk-lib/aws-sqs';
4 import * as ssm from 'aws-cdk-lib/aws-ssm';
5
6 export class SsmStack extends cdk.Stack {
7   constructor(scope: Construct, id: string, props?: cdk.StackProps) {
8     super(scope, id, props);
9
10    // Create a basic SSM Parameter Store parameter
11    const demoSsm = new ssm.StringParameter(this, 'demoSsmID', {
12      parameterName: '/vpc/NetworkVPC',
13      stringValue: 'vpc-0c29bcf7b2e187421'
14    });
15  }
16 }
17
```

The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer sidebar is expanded to show the 'webserver' folder, which contains the 'webserver-stack.ts' file. The 'webserver-stack.ts' file is selected and its content is displayed in the main editor area. The code defines a CDK stack named 'WebserverStack' that extends 'cdk.Stack'. It imports necessary modules from 'aws-cdk-lib' and 'constructs'. The stack's constructor takes a scope, ID, and optional props. It uses the VPC ID from the SSM Parameter Store to create a VPC. It imports the IAM Role ARN from the Security Stack. It creates a security group named 'demoSecurityGroup' and adds an ingress rule to allow HTTP traffic. It creates an EC2 instance named 'demoInstance' with the security group, VPC, and IAM role. The stack is exported as 'WebserverStack'.

```
1 import * as cdk from 'aws-cdk-lib';
2 import { Construct } from 'constructs';
3 import * as ssm from 'aws-cdk-lib/aws-ssm';
4 import * as ec2 from 'aws-cdk-lib/aws-ec2';
5 import * as fs from 'fs';
6 import * as path from 'path';
7 import * as iam from 'aws-cdk-lib/aws-iam';
8
9 export class WebserverStack extends cdk.Stack {
10   constructor(scope: Construct, id: string, props?: cdk.StackProps) {
11     super(scope, id, props);
12
13    // Use the VPC ID from SSM Parameter Store
14    const vpcId = ssm.StringParameter.valueFromLookup(this, '/vpc/NetworkVPC');
15    const vpcFromSSM = ec2.Vpc.fromLookup(this, 'VPC Picked From SSM', { vpcId: vpcId });
16
17    // Import the IAM Role ARN from the Security Stack
18    const importedRoleArn = iam.Role.fromRoleArn(this, 'ImportedRoleArn', cdk.Fn.importValue('DemoIamRoleArn'));
19
20    // Create a security group
21    const demoSecurityGroup = new ec2.SecurityGroup(this, 'DemoSecurityGroup', {
22      vpc: vpcFromSSM,
23      description: 'Demo Security Group',
24      allowAllOutbound: true,
25    });
26
27    // Add an ingress rule to allow HTTP traffic
28    demoSecurityGroup.addIngressRule(
29      ec2.Peer.anyIpv4(),
30      ec2.Port.tcp(80),
31      'Allow HTTP traffic from anywhere'
32    );
33
34    // Create an EC2 instance in the VPC with the security group
35    const demoInstance = new ec2.Instance(this, 'DemoInstanceID', {
36      vpc: vpcFromSSM,
37      instanceType: new ec2.InstanceType('t2.micro'),
38      role: importedRoleArn,
39      machineImage: new ec2.AmazonLinuxImage({
40        generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX_2,
41        edition: ec2.AmazonLinuxEdition.STANDARD,
42        virtualization: ec2.AmazonLinuxVirt.HVM,
43        storage: ec2.AmazonLinuxStorage.GENERAL_PURPOSE,
44      }),
45    });
46  }
47 }
48
```

```
webserver > lib > TS webserver-stack.ts > WebserverStack > constructor
9 export class WebserverStack extends cdk.Stack {
10   constructor(scope: Construct, id: string, props?: cdk.StackProps) {
11     // Import the IAM Role ARN from the Security Stack
12     const importedRoleArn = iam.Role.fromRoleArn(this, 'ImportedRoleArn', cdk.Fn.importValue('DemoIamRoleArn'));
13
14     // Create a security group
15     const demoSecurityGroup = new ec2.SecurityGroup(this, 'DemoSecurityGroup', {
16       vpc: vpcFromSM,
17       description: 'Demo Security Group',
18       allowAllOutbound: true,
19     });
20
21     // Add an ingress rule to allow HTTP traffic
22     demoSecurityGroup.addIngressRule(
23       ec2.Peer.anyIpv4(),
24       ec2.Port.tcp(80),
25       'Allow HTTP traffic from anywhere'
26     );
27
28     // Create an EC2 instance in the VPC with the security group
29     const demoInstance = new ec2.Instance(this, 'DemoInstanceID', {
30       vpc: vpcFromSM,
31       instanceType: new ec2.InstanceType('t2.micro'),
32       role: importedRoleArn,
33       machineImage: new ec2.AmazonLinuxImage({
34         generation: ec2.AmazonLinuxGeneration.AMAZON_LINUX_2,
35         edition: ec2.AmazonLinuxEdition.STANDARD,
36         virtualization: ec2.AmazonLinuxVirt.HVM,
37         storage: ec2.AmazonLinuxStorage.GENERAL_PURPOSE,
38       }),
39       securityGroup: demoSecurityGroup,
40       associatePublicIpAddress: true,
41       vpcSubnets: { subnetType: ec2.SubnetType.PUBLIC },
42     });
43
44     const userDataScript = fs.readFileSync(path.join(__dirname, 'userdata.sh'), 'utf8');
45     userDataScript.split('\n').forEach(line => {
46       demoInstance.userData.addCommands(line);
47     });
48
49     demoInstance.userData.addCommands(userDataScript);
50   }
51 }
52
53
54
55
56
57
58
59
```

```
webserver > lib > $ userdata.sh
1 #!/bin/bash
2
3 sudo su
4 yum update -y
5 yum install -y httpd
6
7 systemctl start httpd
8 systemctl enable httpd
9 echo -e"<h1>Welcome to Likhith's Web Server</h1>" > /var/www/html/index.html
```

All 4 Stacks got created in Cloudformation -

The screenshot shows the AWS CloudFormation console for the **NetworkStack**. The left sidebar displays the navigation menu with options like Stacks, Stack details, Drifts, StackSets, Exports, and Registry. The main content area is divided into two panels. The left panel, titled **Stacks (5)**, lists five stacks: SecurityStack, WebserverStack, SsmStack, NetworkStack, and CDKToolkit. The **NetworkStack** is selected and highlighted with a blue border. The right panel, titled **NetworkStack**, shows the **Resources (17)** tab. It contains a table with 17 resources, all of which are in the **CREATE_COMPLETE** state. The resources include CDKMetadata, CustomVpcRestrictDefaultSGCustomResourceProviderHandlerDCB33E5E, CustomVpcRestrictDefaultSGCustomResourceProviderole26592FE0, DemoVpcLDB14D2481, DemoVpcLIDIGWAC2B33E2, and DemoVpcLIDPrivateSubnet.

Logical ID	Physical ID	Type	Status
CDKMetadata	a7e146b0-4d95-11f0-a790-0affe69327f9	AWS::CDK::Metadata	CREATE_COMPLETE
CustomVpcRestrictDefaultSGCustomResourceProviderHandlerDCB33E5E	NetworkStack-CustomVpcRestrictDefaultSGCustomResourceProviderole26592FE0	AWS::Lambda::Function	CREATE_COMPLETE
CustomVpcRestrictDefaultSGCustomResourceProviderole26592FE0	NetworkStack-CustomVpcRestrictDefaultSGCustomResourceProviderole26592FE0	AWS::IAM::Role	CREATE_COMPLETE
DemoVpcLDB14D2481	vpc-0c25bcf7b2e187421	AWS::EC2::VPC	CREATE_COMPLETE
DemoVpcLIDIGWAC2B33E2	igw-0e118f8ae37e2d125	AWS::EC2::InternetGateway	CREATE_COMPLETE
DemoVpcLIDPrivateSubnet	rtb-		

The screenshot shows the AWS CloudFormation console for the **SsmStack**. The left sidebar displays the navigation menu with options like Stacks, Stack details, Drifts, StackSets, Exports, and Registry. The main content area is divided into two panels. The left panel, titled **Stacks (5)**, lists five stacks: SecurityStack, WebserverStack, SsmStack, NetworkStack, and CDKToolkit. The **SsmStack** is selected and highlighted with a blue border. The right panel, titled **SsmStack**, shows the **Resources (2)** tab. It contains a table with 2 resources, both of which are in the **CREATE_COMPLETE** state. The resources include CDKMetadata and demoSsmLIDD9CEE316.

Logical ID	Physical ID	Type	Status
CDKMetadata	41d42260-4d96-11f0-84ed-0e4b796e2be5	AWS::CDK::Metadata	CREATE_COMPLETE
demoSsmLIDD9CEE316	/vpc/NetworkVPC	AWS::SSM::Parameter	UPDATE_COMPLETE

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

Stack NetworkStack ypcs | VPC Console Instances | EC2 | us-ea... SecurityStack-DemoRol... test practices - AWS CL... My Drive - Google Drive CDK - Google Drive Project 5 - Cross-Stack...

CloudFormation > Stacks > WebserverStack

CloudFormation

- Stacks
 - Stack details
 - Drifts
 - StackSets
 - Exports
- Infrastructure Composer
- laC 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
 - CREATE_COMPLETE

WebserverStack

Stack info Events Resources Outputs Parameters Template Ch...

Resources (4)

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
DemoInstanceLIDInstanceProfileSAD6EEA7	WebserverStack-DemoInstanceLIDInstanceProfileSAD6EEA7-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

Stack NetworkStack ypcs | VPC Console Instances | EC2 | us-ea... SecurityStack-DemoRol... test practices - AWS CL... My Drive - Google Drive CDK - Google Drive Project 5 - Cross-Stack...

CloudFormation > Stacks > SecurityStack

CloudFormation

- Stacks
 - Stack details
 - Drifts
 - StackSets
 - Exports
- Infrastructure Composer
- laC 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
 - CREATE_COMPLETE

SecurityStack

Stack info Events Resources Outputs Parameters Template Ch...

Resources (2)

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

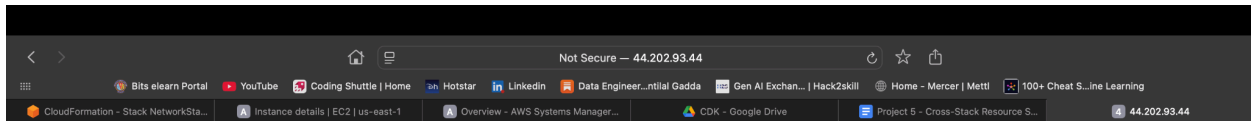
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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

The screenshot shows the AWS Management Console for the us-east-1 region. The left sidebar displays the navigation menu with categories like EC2, Images, Elastic Block Store, and Network & Security. The main content area shows the 'Instance summary for i-05fdf276425672bad (WebserverStack/DemoInstanceLID)'. The instance is in a 'Running' state. Key details include: Instance ID: i-05fdf276425672bad, Public IPv4 address: 44.202.93.44, Private IPv4 address: 10.0.90.85, Public DNS: ec2-44-202-93-44.compute-1.amazonaws.com, Instance type: t2.micro, VPC ID: vpc-0c25bcf7b2e187421, Subnet ID: subnet-05b590fa41ebac428, and Instance ARN: arn:aws:ec2:us-east-1:522192923174:instance/i-05fdf276425672bad. The bottom of the console shows the footer with copyright information and links to Privacy, Terms, and Cookie preferences.

The screenshot shows the AWS Systems Manager Parameter Store in the us-east-1 region. The breadcrumb navigation indicates the path: AWS Systems Manager > Parameter Store > /vpc/NetworkVPC > Overview. The parameter details for '/vpc/NetworkVPC' are displayed, including its Name, ARN (arn:aws:ssm:us-east-1:522192923174:parameter/vpc/NetworkVPC), Tier (Standard), Type (String), and Value (vpc-0c25bcf7b2e187421). The Description field is empty. The Data type is text. The Last modified user is arn:aws:sts::522192923174:assumed-role/cdk-hnb659fds-cfn-exec-role-522192923174-us-east-1/AWSCloudFormation. The Last modified date is Fri, 20 Jun 2025 05:58:29 GMT. The Version is 2. The bottom of the console shows the footer with copyright information and links to Privacy, Terms, and Cookie preferences.

WebServer is running successfully -



Welcome to Likhith's Web Server