Assessment Objective: The Application Engineering team has been diligently crafting a custom WooCommerce-based product as part of an ongoing initiative. To support the deployment of this product, we require a robust and scalable cloud-based infrastructure. As a member of the DevOps Engineering team, I propose to architect and implement the necessary cloud infrastructure to facilitate the deployment of the 3- Tier WooCommerce-based product.

Task Overview: Involves designing and implementing a reference architecture for a 3-tier application that will host the custom WooCommerce-based product. I have utilized modern IaC techniques CDK to provision the infrastructure, ensuring scalability, security, and performance.

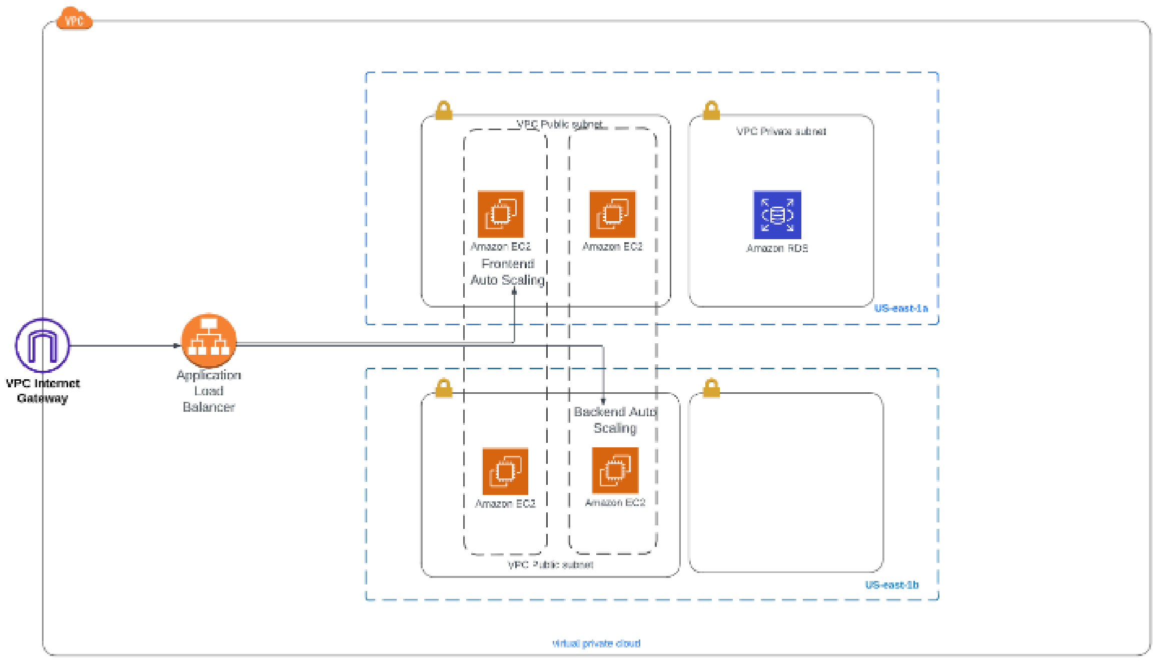
Deliverables:

- A well-documented reference architecture outlining the components and their interactions within the infrastructure.
- Implementation of the reference architecture using AWS CDK or similar IaC tools, with code hosted on GitHub for review.
- Designed and built AWS infrastructure following security, networking compliances.
- Integrated Autoscaling and ELB (Application Load balancer) to maintain high availability and scalability of application.

Architecture

Target technology stack

- AWS CDK
- VPC
- EC2
- RDS
- ELB
- Autoscaling



https://github.com/CloudRespoProject/BMOCDKapp/blob/main/3TierArchitecturaldiagram.pdf

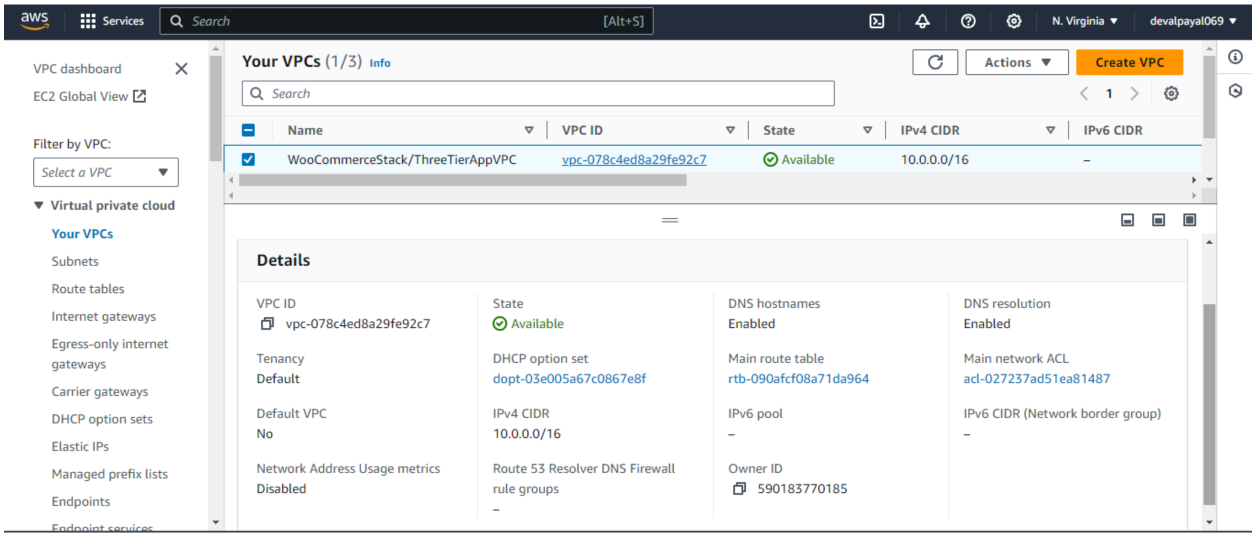
Infrastructure Implementation

Woocommerce 3-Tier application is deployed leveraging following services:

- **Region** : Woocommerce 3 tier website is deployed in the North.Virginia region.

- **AZ**: To maintain availability of application it is deployed in two different subnets.
  1. Us-east-1a
  2. Us-east-2b

- **VPC (Virtual Private Cloud):** Create a VPC to isolate your resources and provide network security ensure VPC has both public and private subnets.



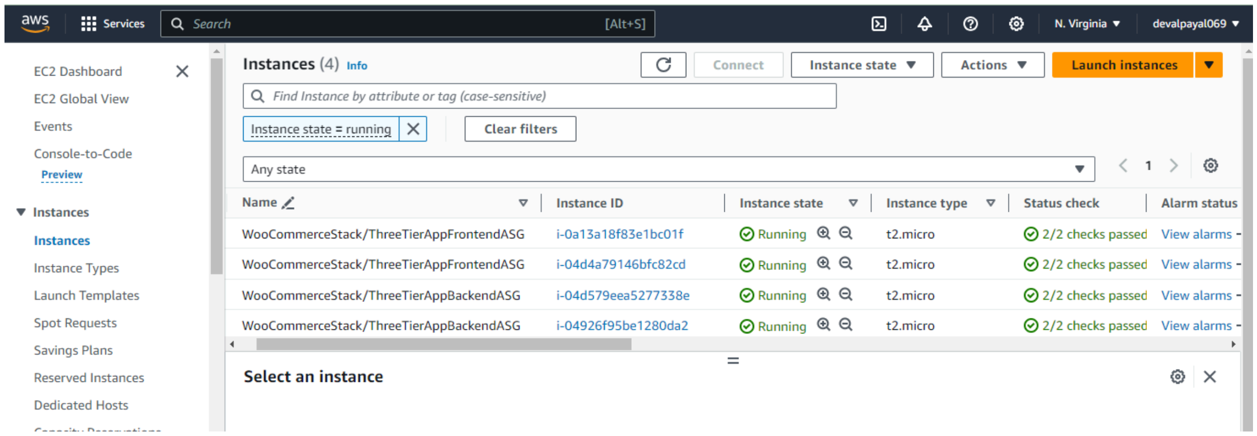
Your VPCs (1/3) Info

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
WooCommerceStack/ThreeTierAppVPC	vpc-078c4ed8a29fe92c7	Available	10.0.0.0/16	–

Details

VPC ID: vpc-078c4ed8a29fe92c7	State: Available	DNS hostnames: Enabled	DNS resolution: Enabled
Tenancy: Default	DHCP option set: dopt-03e005a67c0867e8f	Main route table: rtb-090afcf08a71da964	Main network ACL: acl-027237ad51ea81487
Default VPC: No	IPv4 CIDR: 10.0.0.0/16	IPv6 pool: –	IPv6 CIDR (Network border group): –
Network Address Usage metrics: Disabled	Route 53 Resolver DNS Firewall rule groups: –	Owner ID: 590183770185	

- **EC2 Instances**: Use t2.micro EC2 instances to host the frontend and backend of the application. You can use different instance types based on your application's requirements and configure them to launch in the private subnets.

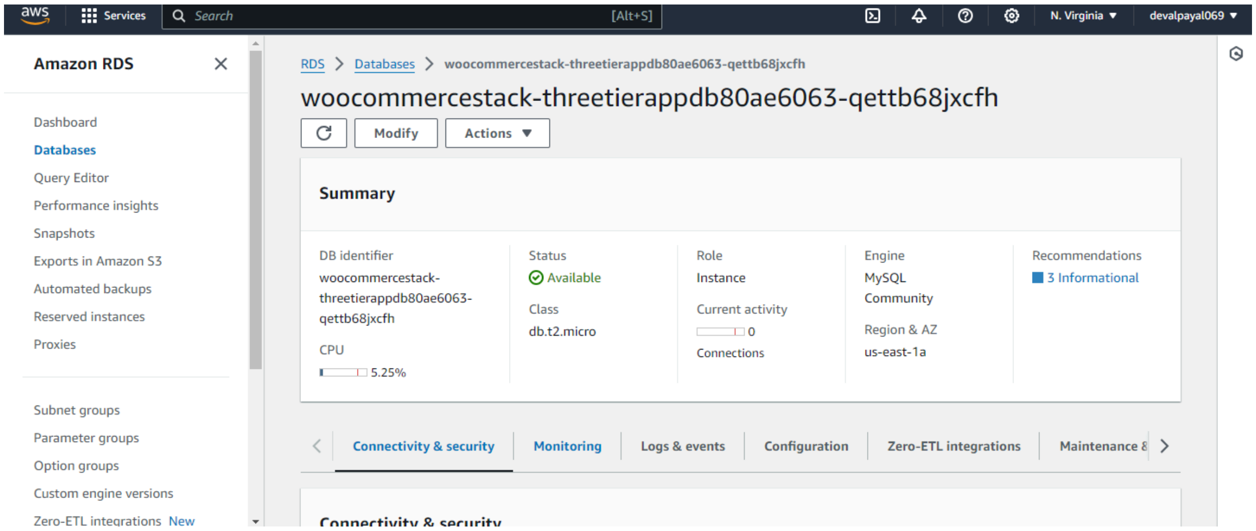


Instances (4) Info

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
WooCommerceStack/ThreeTierAppFrontendASG	i-0a13a18f83e1bc01f	Running	t2.micro	2/2 checks passed	View alarms
WooCommerceStack/ThreeTierAppFrontendASG	i-04d4a79146bfc82cd	Running	t2.micro	2/2 checks passed	View alarms
WooCommerceStack/ThreeTierAppBackendASG	i-04d579eea5277338e	Running	t2.micro	2/2 checks passed	View alarms
WooCommerceStack/ThreeTierAppBackendASG	i-04926f95be1280da2	Running	t2.micro	2/2 checks passed	View alarms

- **RDS (Relational Database Service):** db.t2.micro MYSQL RDS to host databases in the private of primary AZ.
  [Can also deploy read-replicas to multi AZ to reduce latency and maintain high availability.]



aws Services Search [Alt+S] N. Virginia devalpayal069

Amazon RDS

Dashboard
Databases
Query Editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies

Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New

RDS > Databases > woocommercestack-threetierappdb80ae6063-qettb68jxcfh

woocommercestack-threetierappdb80ae6063-qettb68jxcfh

Modify Actions

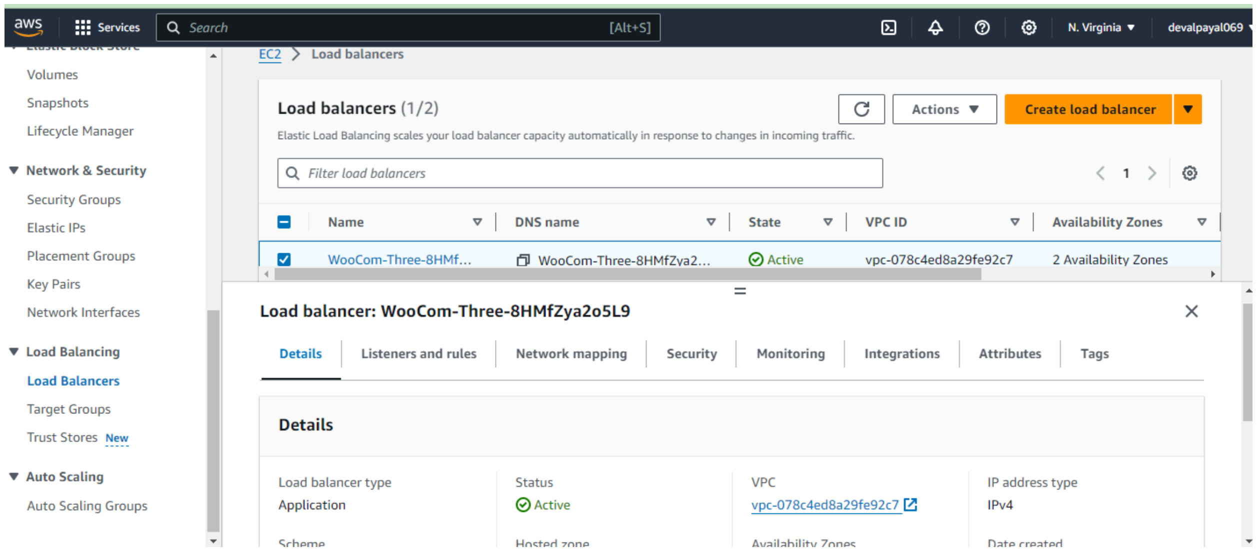
Summary

DB identifier	Status	Role	Engine	Recommendations
woocommercestack-threetierappdb80ae6063-qettb68jxcfh	Available	Instance	MySQL Community	3 Informational
CPU	Class	Current activity	Region & AZ	
5.25%	db.t2.micro	0 Connections	us-east-1a	

Connectivity & security | Monitoring | Logs & events | Configuration | Zero-ETL integrations | Maintenance &

Connectivity & security

- **ELB (Elastic Load Balancer):** Use an Application Load Balancer (ALB) to distribute traffic to your EC2 instances. This helps in achieving high availability and scalability. Ensure that your ALB is placed in the public subnet.



aws Services Search [Alt+S] N. Virginia devalpayal069

Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups

EC2 > Load balancers

Load balancers (1/2)

Actions Create load balancer

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

Name	DNS name	State	VPC ID	Availability Zones
WooCom-Three-8HMf…	WooCom-Three-8HMfZya2…	Active	vpc-078c4ed8a29fe92c7	2 Availability Zones

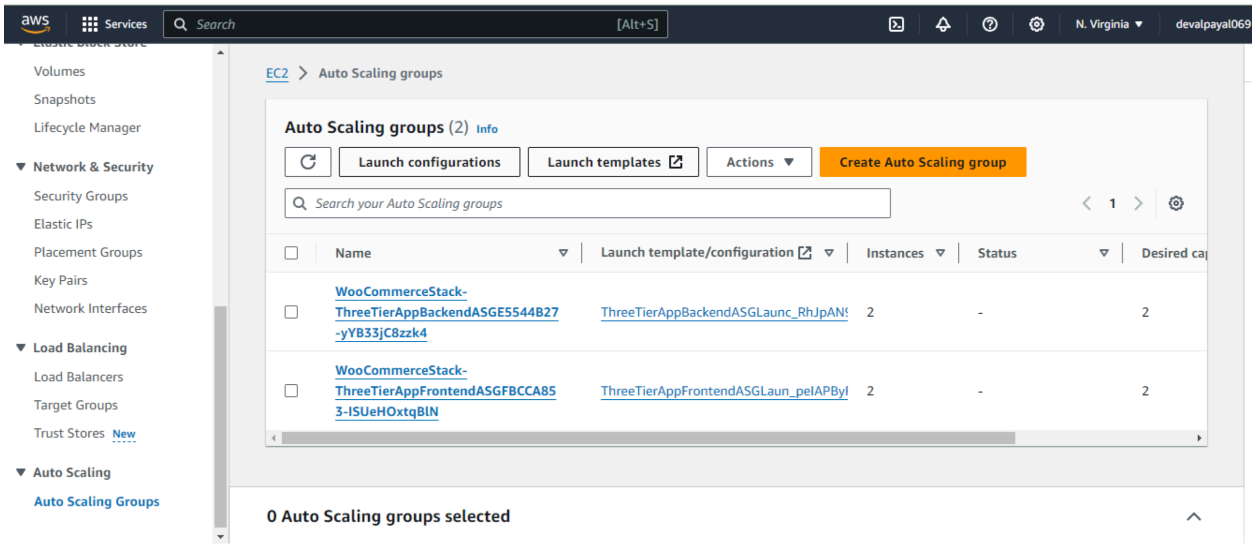
Load balancer: WooCom-Three-8HMfZya2o5L9

Details | Listeners and rules | Network mapping | Security | Monitoring | Integrations | Attributes | Tags

Details

Load balancer type	Status	VPC	IP address type
Application	Active	vpc-078c4ed8a29fe92c7	IPv4
Scheme	Hosted zone	Availability Zones	Date created

- **Autoscaling:** Implement auto scaling for your EC2 instances to automatically adjust the number of instances based on traffic. This helps in maintaining performance and availability during traffic spikes. Here max is 4 and Min number of instances are 2.



Services | Search [Alt+S] | N. Virginia | devalpayal069

Volumes
Snapshots
Lifecycle Manager
▼ Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
▼ Load Balancing
Load Balancers
Target Groups
Trust Stores New
▼ Auto Scaling
Auto Scaling Groups

EC2 > Auto Scaling groups

Auto Scaling groups (2) Info

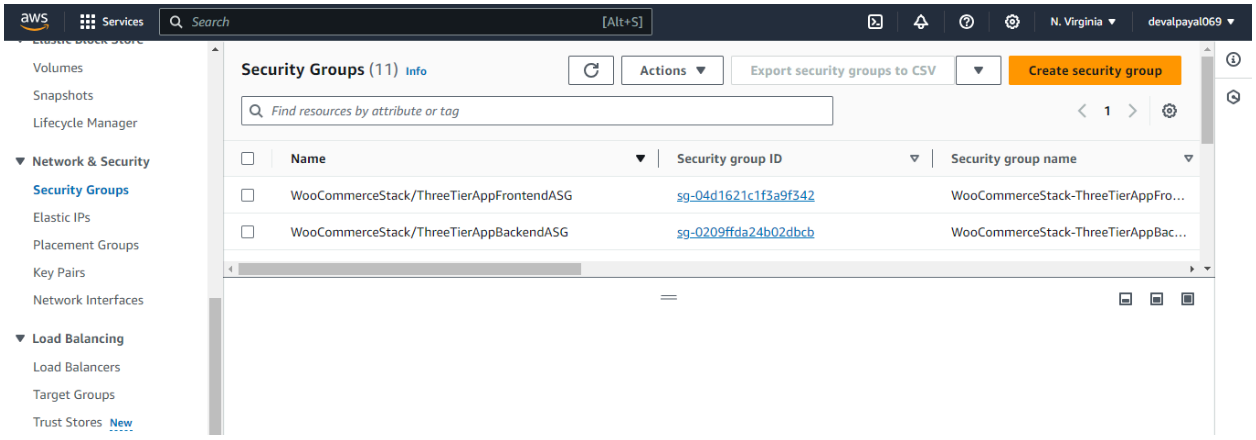
Launch configurations | Launch templates | Actions | Create Auto Scaling group

Search your Auto Scaling groups

	Name	Launch template/configuration	Instances	Status	Desired ca
☐	WooCommerceStack-ThreeTierAppBackendASGE5544B27-yYB33jC8zzk4	ThreeTierAppBackendASGLaunc_RhJpAN	2	-	2
☐	WooCommerceStack-ThreeTierAppFrontendASGFBCCA853-ISUeHOxtqBlN	ThreeTierAppFrontendASGLaun_pelAPByl	2	-	2

0 Auto Scaling groups selected

- **Security Groups and Network ACLs:** Configured security groups and network ACLs to control inbound and outbound traffic to your resources. Follow the principle of least privilege.



Services | Search [Alt+S] | N. Virginia | devalpayal069

Volumes
Snapshots
Lifecycle Manager
▼ Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
▼ Load Balancing
Load Balancers
Target Groups
Trust Stores New

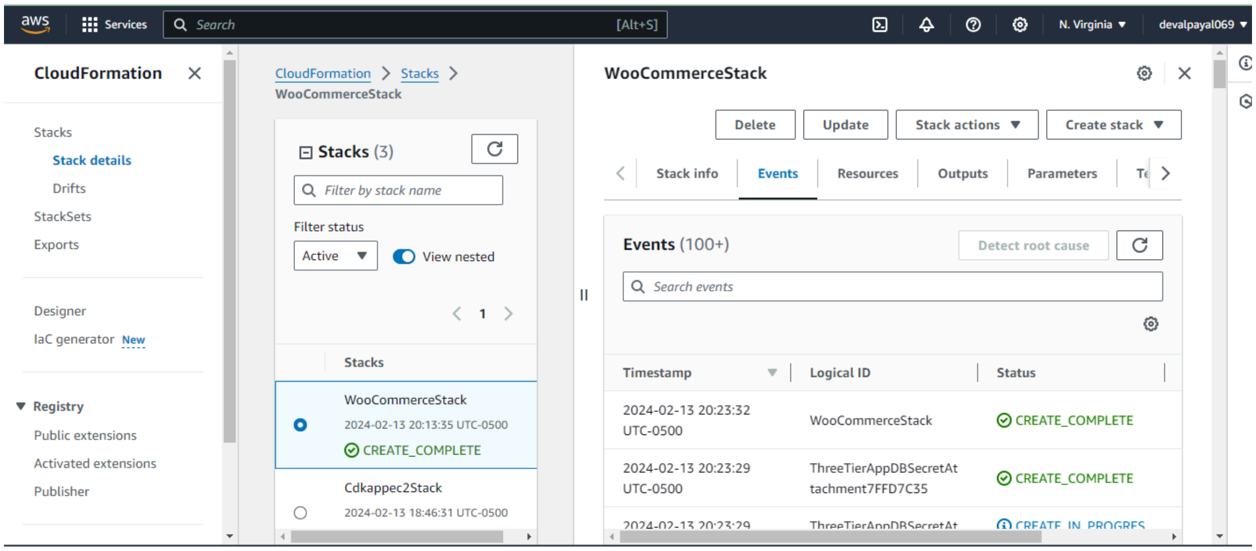
Security Groups (11) Info

Actions | Export security groups to CSV | Create security group

Find resources by attribute or tag

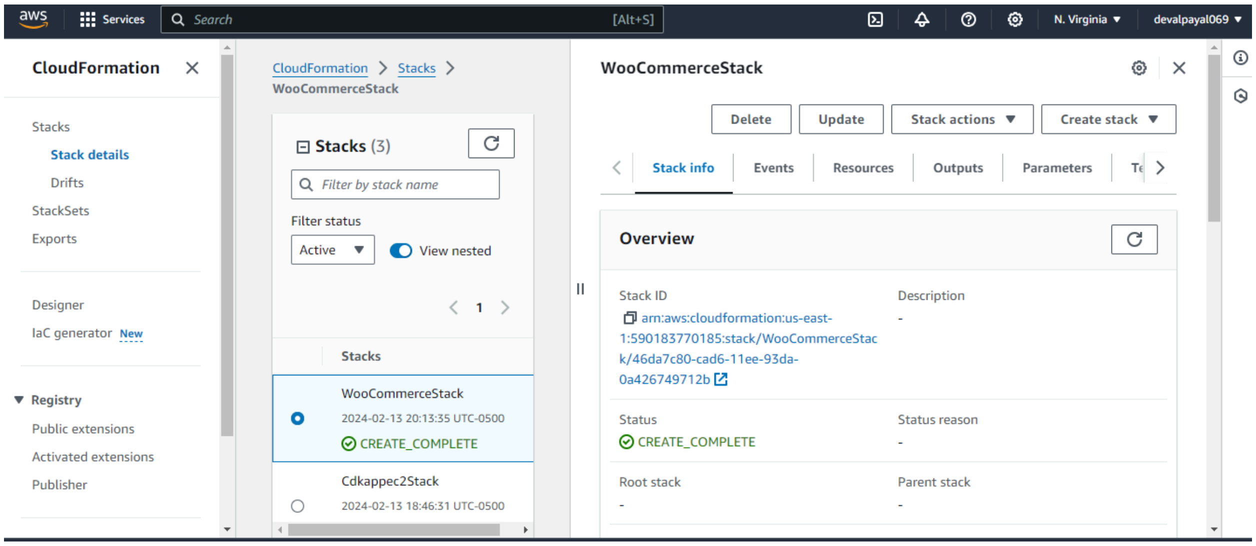
	Name	Security group ID	Security group name
☐	WooCommerceStack/ThreeTierAppFrontendASG	sg-04d1621c1f3a9f342	WooCommerceStack-ThreeTierAppFro…
☐	WooCommerceStack/ThreeTierAppBackendASG	sg-0209ffda24b02dbcb	WooCommerceStack-ThreeTierAppBac…

CDK Stack



Stacks
WooCommerceStack 2024-02-13 20:13:35 UTC-0500 CREATE_COMPLETE
Cdkappec2Stack 2024-02-13 18:46:31 UTC-0500

Timestamp	Logical ID	Status
2024-02-13 20:23:32 UTC-0500	WooCommerceStack	CREATE_COMPLETE
2024-02-13 20:23:29 UTC-0500	ThreeTierAppDBSecretAttachment7FFD7C35	CREATE_COMPLETE
2024-02-13 20:23:29	ThreeTierAppDBSecretAt	CREATE_IN_PROGRES



Stacks
WooCommerceStack 2024-02-13 20:13:35 UTC-0500 CREATE_COMPLETE
Cdkappec2Stack 2024-02-13 18:46:31 UTC-0500

Overview

Stack ID
arn:aws:cloudformation:us-east-1:590183770185:stack/WooCommerceStack/46da7c80-cad6-11ee-93da-0a426749712b

Description
-

Status
CREATE_COMPLETE

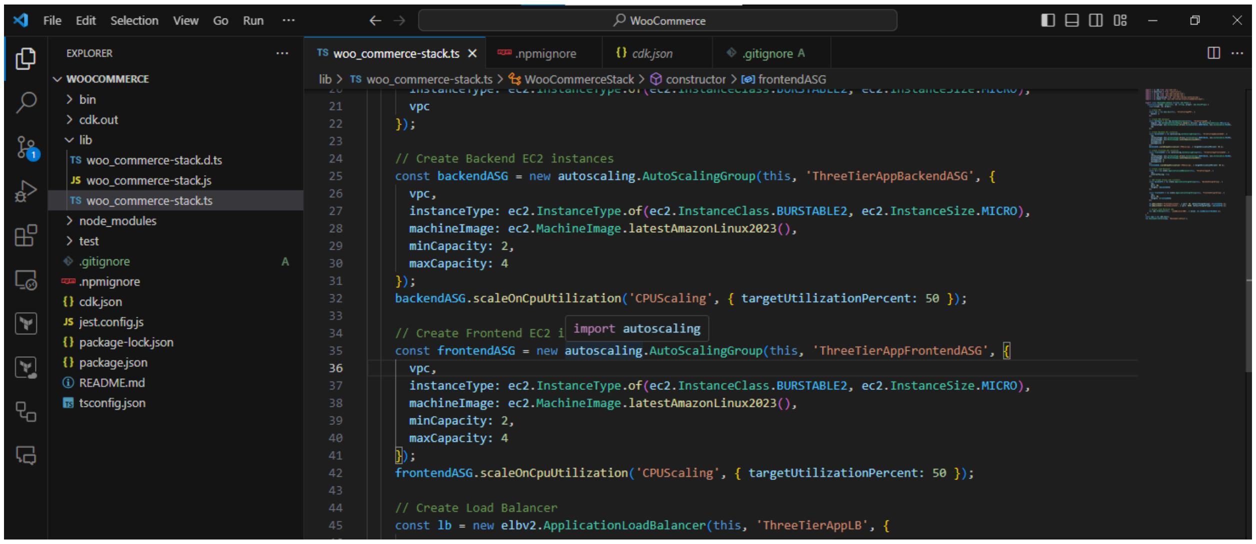
Status reason
-

Root stack
-

Parent stack
-

Implementation of CDK stack:

To create AWS CDK stack utilize AWS CLI, npm and AWS CDK with typescript language to install necessary packages and dependencies.

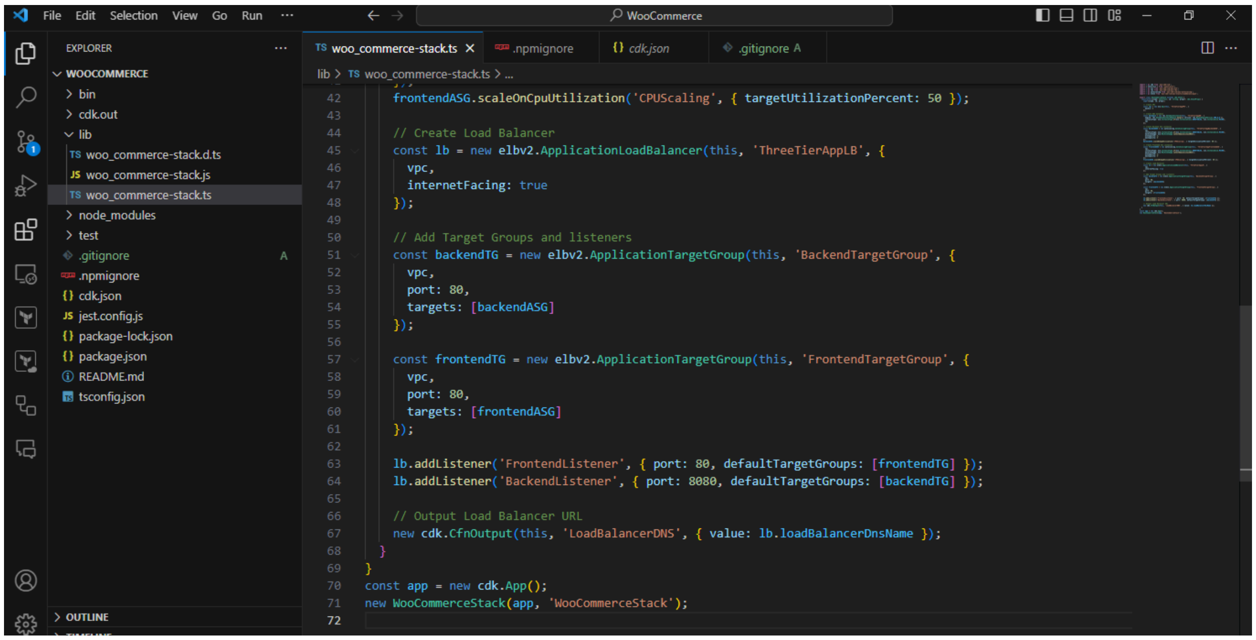


```
      vpc
    });

    // Create Backend EC2 instances
    const backendASG = new autoscaling.AutoScalingGroup(this, 'ThreeTierAppBackendASG', {
      vpc,
      instanceType: ec2.InstanceType.of(ec2.InstanceClass.BURSTABLE2, ec2.InstanceSize.MICRO),
      machineImage: ec2.MachineImage.latestAmazonLinux2023(),
      minCapacity: 2,
      maxCapacity: 4
    });
    backendASG.scaleOnCpuUtilization('CPUScaling', { targetUtilizationPercent: 50 });

    // Create Frontend EC2 i  import autoscaling
    const frontendASG = new autoscaling.AutoScalingGroup(this, 'ThreeTierAppFrontendASG', {
      vpc,
      instanceType: ec2.InstanceType.of(ec2.InstanceClass.BURSTABLE2, ec2.InstanceSize.MICRO),
      machineImage: ec2.MachineImage.latestAmazonLinux2023(),
      minCapacity: 2,
      maxCapacity: 4
    });
    frontendASG.scaleOnCpuUtilization('CPUScaling', { targetUtilizationPercent: 50 });

    // Create Load Balancer
    const lb = new elbv2.ApplicationLoadBalancer(this, 'ThreeTierAppLB', {
```



```
    frontendASG.scaleOnCpuUtilization('CPUScaling', { targetUtilizationPercent: 50 });

    // Create Load Balancer
    const lb = new elbv2.ApplicationLoadBalancer(this, 'ThreeTierAppLB', {
      vpc,
      internetFacing: true
    });

    // Add Target Groups and listeners
    const backendTG = new elbv2.ApplicationTargetGroup(this, 'BackendTargetGroup', {
      vpc,
      port: 80,
      targets: [backendASG]
    });

    const frontendTG = new elbv2.ApplicationTargetGroup(this, 'FrontendTargetGroup', {
      vpc,
      port: 80,
      targets: [frontendASG]
    });

    lb.addListener('FrontendListener', { port: 80, defaultTargetGroups: [frontendTG] });
    lb.addListener('BackendListener', { port: 8080, defaultTargetGroups: [backendTG] });

    // Output Load Balancer URL
    new cdk.CfnOutput(this, 'LoadBalancerDNS', { value: lb.loadBalancerDnsName });
  }
}
const app = new cdk.App();
new WooCommerceStack(app, 'WooCommerceStack');
```

In conclusion, the assignment focused on designing and implementing a scalable cloud-based infrastructure for deploying a custom WooCommerce-based product. The goal was to ensure high availability, security, and performance of the application.

I have proposed and implemented a reference architecture using modern Infrastructure as Code (IaC) techniques, specifically AWS CDK. The architecture includes the use of Amazon EC2 instances for hosting frontend and backend services, Amazon RDS for database hosting, and an Application Load Balancer for traffic distribution. Autoscaling was implemented to manage the number of EC2 instances based on traffic demand, ensuring optimal performance.

The deliverables include a well-documented reference architecture outlining the components and their interactions, as well as the implementation of the infrastructure using AWS CDK with the code hosted on GitHub for review. The architecture complies with security and networking best practices to ensure a robust and reliable deployment.

Overall, the assignment provided valuable hands-on experience in designing, implementing, and managing cloud-based infrastructure for modern applications, which aligns with my interests and future ambitions in the field of DevOps Engineering.