

Assignment 8

Step 1: Networking Configuration

The screenshot captures the VPC creation summary page, displaying key details such as the VPC ID, name, and CIDR block. This confirms the successful creation of the VPC with the specified configuration.

The screenshot shows the AWS VPC dashboard for a VPC named "vpc-0983de8e5fd40357d / webapp-vpc". The main details pane shows the following configuration:

VPC ID	State	Block Public Access	DNS hostnames
vpc-0983de8e5fd40357d	Available	Off	Enabled
DNS resolution	Tenancy	DHCP option set	Main route table
Enabled	Default	dopt-03a2977c09a45226e	rtb-0558711f2968beda8
Main network ACL	Default VPC	IPv4 CIDR	IPv6 pool
acl-0512d161cc20f6214	No	10.0.0.0/16	-
IPv6 CIDR (Network border group)	Network Address Usage metrics	Route 53 Resolver DNS Firewall rule groups	Owner ID
-	Disabled	-	626635443515

Below the details, there are tabs for "Resource map", "CIDRs", "Flow logs", "Tags", and "Integrations". The "Resource map" tab is selected, showing three cards: "VPC", "Subnets (5)", and "Route tables (3)".

VPC (Show details): Your AWS virtual network
webapp-vpc

Subnets (5): Subnets within this VPC
us-east-1a
public-subnet-1
private-subnet-1

Route tables (3): Route network traffic to resources
rtb-0558711f2968beda8
rtb-0c0f9e7e6e3372cb0
rtb-0dc96d5c630b1559e

At the bottom, the footer includes links for "CloudShell", "Feedback", "© 2024, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

The screenshot captures the subnet configurations, showing details such as the CIDR blocks and the associated availability zones. This provides a clear overview of the network structure within the VPC.

Public Subnet 1

The screenshot shows the AWS VPC dashboard with the following details for Public Subnet 1:

Details	
Subnet ID subnet-0537dedfe51949d3c	Subnet ARN arn:aws:ec2:us-east-1:626635443515:subnet/subnet-0537dedfe51949d3c
IPv4 CIDR 10.0.1.0/24	State Available
Availability Zone us-east-1a	IPv6 CIDR -
Route table rtb-0dc96d5c630b1559e	Available IPv4 addresses 251
Auto-assign IPv6 address No	Network border group us-east-1
IPv4 CIDR reservations -	Default subnet No
Resource name DNS A record Disabled	Customer-owned IPv4 pool -
	IPv6-only No
	DNS64 Disabled
	Resource name DNS AAAA record Disabled

Other settings include:

- Block Public Access: Off
- IPv6 CIDR association ID: -
- VPC: vpc-0983de8e5fd40357d | webapp-vpc
- Auto-assign public IPv4 address: Yes
- Outpost ID: -
- Hostname type: IP name
- Owner: 626635443515

Public Subnet 2

The screenshot shows the AWS VPC dashboard with the following details for Public Subnet 2:

Details	
Subnet ID subnet-069c09c4fbdf2e0ed	Subnet ARN arn:aws:ec2:us-east-1:626635443515:subnet/subnet-069c09c4fbdf2e0ed
IPv4 CIDR 10.0.2.0/24	State Available
Availability Zone us-east-1b	IPv6 CIDR -
Route table rtb-0dc96d5c630b1559e	Available IPv4 addresses 250
Auto-assign IPv6 address No	Network border group us-east-1
IPv4 CIDR reservations -	Default subnet No
Resource name DNS A record Disabled	Customer-owned IPv4 pool -
	IPv6-only No
	DNS64 Disabled
	Resource name DNS AAAA record Disabled

Other settings include:

- Block Public Access: Off
- IPv6 CIDR association ID: -
- VPC: vpc-0983de8e5fd40357d | webapp-vpc
- Auto-assign public IPv4 address: Yes
- Outpost ID: -
- Hostname type: IP name
- Owner: 626635443515

Public Subnet 3

The screenshot shows the AWS VPC Subnets Details page for a public subnet. The subnet ID is subnet-01836bd362b685651. Key details include:

- Subnet ID:** subnet-01836bd362b685651
- Subnet ARN:** arn:aws:ec2:us-east-1:626635443515:subnet/subnet-01836bd362b685651
- State:** Available
- IPv4 CIDR:** 10.0.3.0/24
- IPv6 CIDR:** -
- Available IPv4 addresses:** 249
- Network border group:** us-east-1
- Default subnet:** No
- Customer-owned IPv4 pool:** -
- Auto-assign IPv6 address:** No
- Auto-assign customer-owned IPv4 address:** No
- Resource name DNS A record:** Disabled
- IPv6 CIDR reservations:** -
- DNS64:** Enabled
- Resource name DNS AAAA record:** Disabled

Other settings include:

- Block Public Access:** Off
- IPv6 CIDR association ID:** -
- VPC:** vpc-0983de8e5fd40357d | webapp-vpc
- Auto-assign public IPv4 address:** Yes
- Outpost ID:** -
- Hostname type:** IP name
- Owner:** 626635443515

At the bottom, there are tabs for Flow logs, Route table, Network ACL, CIDR reservations, Sharing, and Tags. A "Create flow log" button is also present.

Private Subnet 1

The screenshot shows the AWS VPC Subnets Details page for a private subnet. The subnet ID is subnet-0ad2f5349372e3a7f. Key details include:

- Subnet ID:** subnet-0ad2f5349372e3a7f
- Subnet ARN:** arn:aws:ec2:us-east-1:626635443515:subnet/subnet-0ad2f5349372e3a7f
- State:** Available
- IPv4 CIDR:** 10.0.4.0/24
- IPv6 CIDR:** -
- Available IPv4 addresses:** 250
- Network border group:** us-east-1
- Default subnet:** No
- Customer-owned IPv4 pool:** -
- Auto-assign IPv6 address:** No
- Auto-assign customer-owned IPv4 address:** No
- Resource name DNS A record:** Disabled
- IPv6 CIDR reservations:** -
- DNS64:** Enabled
- Resource name DNS AAAA record:** Disabled

Other settings include:

- Block Public Access:** Off
- IPv6 CIDR association ID:** -
- VPC:** vpc-0983de8e5fd40357d | webapp-vpc
- Auto-assign public IPv4 address:** No
- Outpost ID:** -
- Hostname type:** IP name
- Owner:** 626635443515

At the bottom, there are tabs for Flow logs, Route table, Network ACL, CIDR reservations, Sharing, and Tags. A "Create flow log" button is also present.

Private Subnet 2

The screenshot shows the AWS VPC dashboard with the following details for a private subnet:

Setting	Value
Subnet ID	subnet-0723da3b86c83bda4
IPv4 CIDR	10.0.5.0/24
Availability Zone	us-east-1b
Route table	rtb-0c0f9e7e6e3372cb0
Auto-assign IPv6 address	No
IPv4 CIDR reservations	-
Resource name DNS A record	Disabled
Subnet ARN	arn:aws:ec2:us-east-1:626635443515:subnet/subnet-0723da3b86c83bda4
State	Available
IPv6 CIDR	-
Network border group	us-east-1
Default subnet	No
Customer-owned IPv4 pool	-
IPv6-only	No
DNS64	Disabled
Resource name DNS AAAA record	Disabled
Block Public Access	Off
IPv6 CIDR association ID	-
VPC	vpc-0983de8e5fd40357d webapp-vpc
Auto-assign public IPv4 address	No
Outpost ID	-
Hostname type	IP name
Owner	626635443515

Below the details, there are tabs for Flow logs, Route table, Network ACL, CIDR reservations, Sharing, and Tags. The Flow logs tab is selected.

The screenshot shows the Internet Gateway (IGW) attached to the VPC. It displays details such as the IGW ID, its name, and the associated VPC ID, confirming that the IGW is successfully connected to enable internet access for resources within the VPC.

The screenshot shows the AWS VPC dashboard with the following details for an Internet Gateway:

Setting	Value
Internet gateway ID	igw-0428ecb554d4afe70
State	Attached
VPC ID	vpc-0983de8e5fd40357d webapp-vpc
Owner	626635443515

Below the details, there is a section for Tags with a search bar and a manage tags button. The tag listed is Name: webapp-igw.

The screenshot shows the public route table associated with the VPC, highlighting the route `0.0.0.0/0` that directs traffic to the Internet Gateway (IGW). This confirms that the route table is configured to allow internet access for resources in the public subnet.

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0763154348852f106	Active	No
10.0.0.0/16	local	Active	No

The screenshot shows the public route table associated with the VPC, including the subnet associations. It displays the subnets linked to the route table, confirming that these subnets are configured to use the public route table for internet access through the Internet Gateway (IGW).

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0763154348852f106	Active	No
10.0.0.0/16	local	Active	No

The screenshot shows the private route table associated with the VPC, including its subnet associations. It displays the subnets linked to the private route table, confirming that these subnets are configured for internal communication without direct internet access.

The screenshot captures the AWS VPC Route Tables interface. On the left, a navigation sidebar lists various VPC-related services like EC2 Global View, Filter by VPC, Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections), Security (Network ACLs, Security groups), PrivateLink and Lattice (Getting started, Endpoints), and CloudShell/Feedback. The main content area shows a route table named "rtb-01b91e90036ffb024 / webapp-private-rt". The "Details" tab is selected, displaying information such as the Route table ID (rtb-01b91e90036ffb024), Main (No), Owner ID (vpc-08c1429f0d06f7ee4 | webapp-vpc), Explicit subnet associations (2 subnets), and Edge associations (None). Below this, the "Routes" tab is active, showing one route entry: Destination 10.0.0.0/16, Target local, Status Active, and Propagated No. The bottom right corner includes copyright information (© 2024, Amazon Web Services, Inc. or its affiliates.) and links for Privacy, Terms, and Cookie preferences.

Step 2: Security Groups

The screenshot displays the RDS security group rules, showing inbound configurations to control database access and outgoing traffic.

The screenshot shows the AWS VPC Security Groups console. On the left, a navigation sidebar lists various VPC-related services and features. The main content area shows the details for a specific security group named "sg-0139298af581f4bdd".

Details:

Security group name	sg-0139298af581f4bdd	Description	VPC ID
Owner	626635443515	Inbound rules count 1 Permission entry	Outbound rules count 0 Permission entries

Inbound rules (1):

Name	Security group rule...	Type	Protocol	Port
-	sgr-0609667ee4239d...	-	PostgreSQL	TCP 5432

At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information and links to Privacy, Terms, and Cookie preferences.

Step 3: Provision RDS Instance for the Web Application

The screenshot shows the RDS instance configuration summary, including details such as the instance class, database engine, and network settings. This confirms the setup of the RDS instance with the specified resources and connectivity configurations.

The screenshot displays the Amazon RDS web console. On the left, a sidebar lists various RDS management options like Dashboard, Databases, and Performance insights. The main area shows the configuration for the 'webapp-db' instance. At the top, there's a summary card with details such as DB Identifier (webapp-db), Status (Available), Role (Instance), Engine (PostgreSQL), and Region & AZ (us-east-1a). Below this, a tab bar includes 'Connectivity & security' (which is selected), Monitoring, Logs & events, Configuration, Maintenance & backups, and Data migrations - new. The 'Connectivity & security' section contains three tabs: Endpoint & port, Networking, and Security. Under Endpoint & port, it shows the Endpoint (webapp-db.c10e4e004g8b.us-east-1.rds.amazonaws.com) and Port (5432). Under Networking, it shows the Availability Zone (us-east-1a), VPC (webapp-vpc (vpc-0983de8e5fd40357d)), Subnet group (webapp-db-subnet-group), and Subnets (subnet-0723da3b86c83bda4, subnet-0ad2f5349372e3a7f). Under Security, it shows VPC security groups (webapp-rds-sg (sg-0139298af581f4bdd)), Publicly accessible (No), Certificate authority (Info rds-ca-rsa2048-g1), and Certificate authority date (May 25, 2021, 16:34 (UTC-07:00)).

Step 4: EC2 Instances Running the Web Application

The screenshot displays the Launch Template configuration page, showing the selected instance type and the user data script. This confirms the configuration of the template for launching EC2 instances with predefined settings and initialization scripts.

The screenshot shows the AWS EC2 Launch Templates configuration page. The left sidebar lists various EC2 management options. The main area is titled 'webapp-launch-template (lt-09586479340ffff22)'. It shows 'Launch template details' with fields for Launch template ID (lt-09586479340ffff22), Launch template name (webapp-launch-template), Default version (1), and Owner (arn:aws:iam::626635443515:root). Below this, a 'Launch template version details' table is shown, listing a single version (1 (Default)) with an AMI ID (ami-012867b8de898a1e6), Instance type (t3.micro), Availability Zone (-), and Key pair name (-). The table also includes tabs for Instance details, Storage, Resource tags, Network interfaces, and Advanced details.

Step 5: Application Load Balancer (ALB) Configuration

The screenshot shows the target group settings, including the health check configurations. It displays details such as the protocol, port, health check path, and thresholds for healthy and unhealthy targets, confirming the setup to monitor the health of instances in the target group.

The screenshot captures the AWS EC2 Target Groups interface. On the left, a navigation sidebar lists various services like Dashboard, EC2 Global View, Events, Instances, Images, and more. The main content area is titled "webapp-target-group".
Details: Shows the ARN of the target group, its protocol (HTTP), port (8080), and VPC (vpc-0983de8e5fd40357d). It also indicates the load balancer is "webapp-alb".
Targets: A summary table shows 1 total target, with 1 healthy target (green), 0 unhealthy targets (red), 0 unused targets, 0 initial targets, and 0 draining targets. Below this, a note says "Select values in this table to see corresponding filters applied to the Registered targets table below."
Health checks: This tab is currently selected. It displays the following settings:

Protocol	Path	Port	Healthy threshold
HTTP	/healthz	Traffic port	2 consecutive health check successes
Unhealthy threshold	Timeout	Interval	Success codes
2 consecutive health check failures	5 seconds	30 seconds	200

At the bottom right of the page, there are links for "Edit", "Targets", "Monitoring", "Attributes", "Tags", and copyright information: "© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences".

The screenshot shows the configuration of the webapp-alb, an internet-facing Application Load Balancer in the webapp-vpc.

The screenshot displays the AWS EC2 Load Balancers console. On the left, a navigation sidebar lists various services like Images, Elastic Block Store, Network & Security, Load Balancing, Auto Scaling, and more. The main panel is titled "webapp-alb" and shows the following details:

- Load balancer type:** Application
- Status:** Active
- VPC:** [vpc-0983de8e5fd40357d](#)
- Hosted zone:** Z555XDOTRQ7X7K
- Availability Zones:**
 - [subnet-01836bd362b685651](#) us-east-1c (use1-a2)
 - [subnet-0537ddfe51949d3c](#) us-east-1a (use1-a2)
 - [subnet-069c09c4fbdf2e0ed](#) us-east-1b (use1-a2)
- Load balancer ARN:** arn:aws:elasticloadbalancing:us-east-1:626635443515:loadbalancer/app/webapp-alb/044c8d43f8545651
- DNS name:** [webapp-alb-1348500897.us-east-1.elb.amazonaws.com](#) (A Record)

Below the details, there are tabs for "Listeners and rules", "Network mapping", "Resource map - new", "Security", "Monitoring", "Integrations", "Attributes", and "Logs". The "Listeners and rules" tab is selected, showing one rule: "Protocol:Port" (HTTP:80) with a "Default action" pointing to the "Forward to target group" section. The "Forward to target group" section lists "Default actions" for "Forward to target group" with two items: "webapp-target-group" (1: 100%) and "Target group stickiness: Off".

The screenshot shows the Load Balancer listener configuration, where a listener on port 80 is set up to forward HTTP traffic to the designated target group. This setup ensures that incoming web traffic is directed appropriately to the resources in the target group.

The screenshot shows the configuration of the "HTTP:80" listener for the "webapp-alb" load balancer. The left sidebar is identical to the previous screenshot. The main panel is titled "HTTP:80 Listener details".

Details: A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Protocol:Port: HTTP:80

Load balancer: [webapp-alb](#)

Default actions:

- Forward to target group
 - [webapp-target-group](#): 1 (100%)
 - Target group stickiness: Off

Listener ARN: arn:aws:elasticloadbalancing:us-east-1:626635443515:listener/app/webapp-alb/044c8d43f8545651/f6c41e014ef3910f

Rules: Listener rules (1)

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Name tag	Priority	Conditions (If)	Actions (Then)	ARN
Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> webapp-target-group: 1 (100%) Target group stickiness: Off 	ARN

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Step 6: Implement Auto Scaling

The screenshot displays the Auto Scaling Group (ASG) settings, including the configured instance counts. It shows details such as the desired, minimum, and maximum number of instances, confirming the ASG's capacity management to maintain optimal resource levels based on demand.

The screenshot shows the AWS EC2 Auto Scaling Groups console. On the left, a navigation sidebar lists various services like Instances, Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main content area is titled "webapp-asg" and contains two tabs: "Capacity overview" and "Launch template".

Capacity overview:

Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
1	1 - 3	Units (number of instances)	-

Launch template:

Launch template	AMI ID	Instance type	Owner
lt-012cf2e7a4b1b965 webapp-lt- 20241207201524792500000003	ami-012867b8de898a1e6	t3.micro	arn:aws:iam::626635443515:user/Assig n1

Below the launch template table, there are sections for Version (Latest), Security groups (-), Description (-), Storage (volumes) (-), Security group IDs (sg-0ed56ae2c441acc9), Key pair name (-), Create time (Sat Dec 07 2024 12:15:25 GMT-0800 (Pacific Standard Time)), and Request Spot Instances (No).

The screenshot shows the CloudWatch alarm configurations used for scaling policies. It details the metrics monitored, such as CPU utilization, and specifies the thresholds that trigger scaling actions, ensuring that resources are adjusted automatically to maintain performance and efficiency.

The screenshot displays the AWS CloudWatch Auto Scaling console for the 'webapp-asg' group. On the left, a navigation sidebar lists various AWS services like EC2, Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The 'Auto Scaling Groups' section is currently selected. The main content area shows two 'Dynamic scaling policies' (Info) for the 'webapp-asg' group:

- webapp-scale-down**:
 - Policy type: Simple scaling
 - Enabled or disabled: Enabled
 - Execute policy when: **cpu-utilization-low**: breaches the alarm threshold: CPUUtilization < 30 for 2 consecutive periods of 300 seconds for the metric dimensions: AutoScalingGroupName = webapp-asg
 - Take the action: Remove 1 capacity units
 - And then wait: 300 seconds before allowing another scaling activity
- webapp-scale-up**:
 - Policy type: Simple scaling
 - Enabled or disabled: Enabled
 - Execute policy when: **cpu-utilization-high**: breaches the alarm threshold: CPUUtilization > 70 for 2 consecutive periods of 300 seconds for the metric dimensions: AutoScalingGroupName = webapp-asg
 - Take the action: Add 1 capacity units
 - And then wait: 300 seconds before allowing another scaling activity

Below these, there is a section for 'Predictive scaling policies (0)' with an 'Info' link and a 'Create predictive scaling policy' button. The evaluation period is set to 'Evaluation based on 2 days'. The bottom of the page includes standard AWS footer links: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Step 7: Testing and Validation

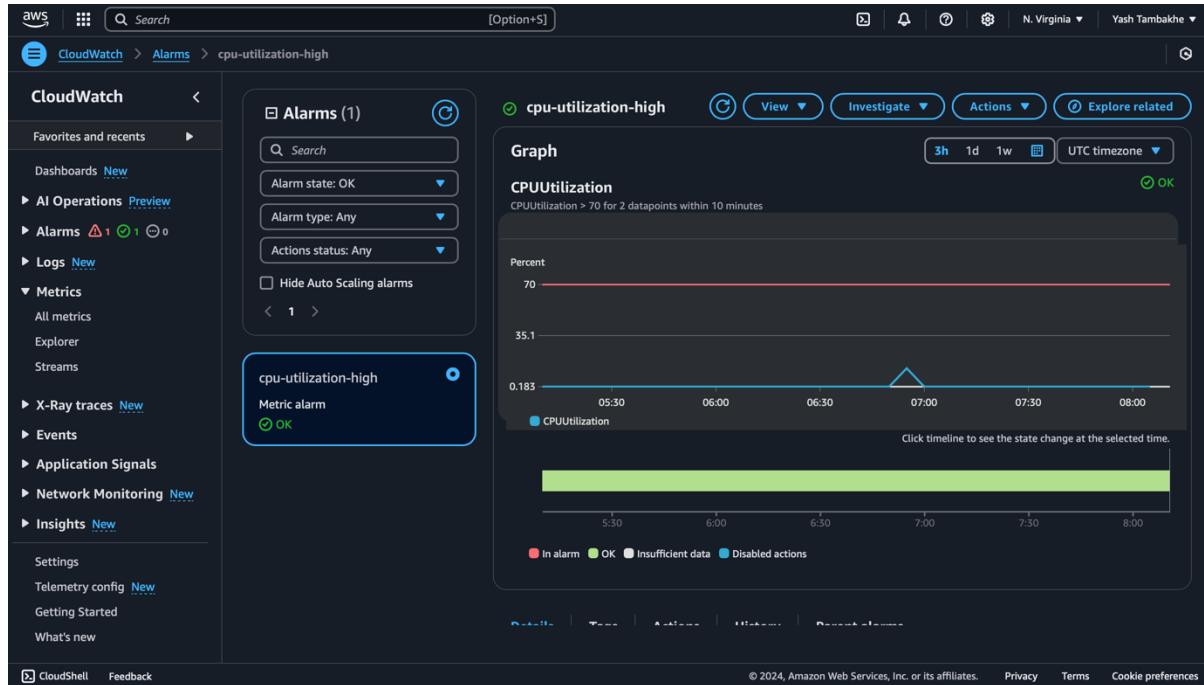
The screenshot captures the scaling activity of instances in the Auto Scaling Group (ASG) during load testing. It shows instances being scaled up to handle increased demand and scaled down when the load decreases, demonstrating the dynamic resource adjustment based on the defined scaling policies.

This screenshot shows the AWS EC2 Auto Scaling Groups interface. The left sidebar is collapsed, and the main navigation bar includes 'Search' and 'Option+S'. The top navigation bar shows 'N. Virginia' and 'Yash Tambakhe'. The main content area has tabs: 'Details', 'Integrations - new', 'Automatic scaling', 'Instance management', 'Instance refresh', 'Activity' (which is selected), and 'Monitoring'. The 'Activity' section is titled 'Activity notifications (0)' and contains a search bar for 'Filter notifications' and a dropdown for 'Send to'. It also includes a button for 'Create notification'. Below this is the 'Activity history' section, which shows one entry: 'Successful' status for launching a new EC2 instance (ID: 043678f125f701b46) at 2024-12-07T20:15:32Z. The entry details the user request changing the desired capacity from 0 to 1, and an instance starting in response to the difference between desired and actual capacity, increasing the capacity from 0 to 1. The timestamp is 2024 Dec 07, 12:12 PM -08:00. The bottom of the page includes standard AWS footer links: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

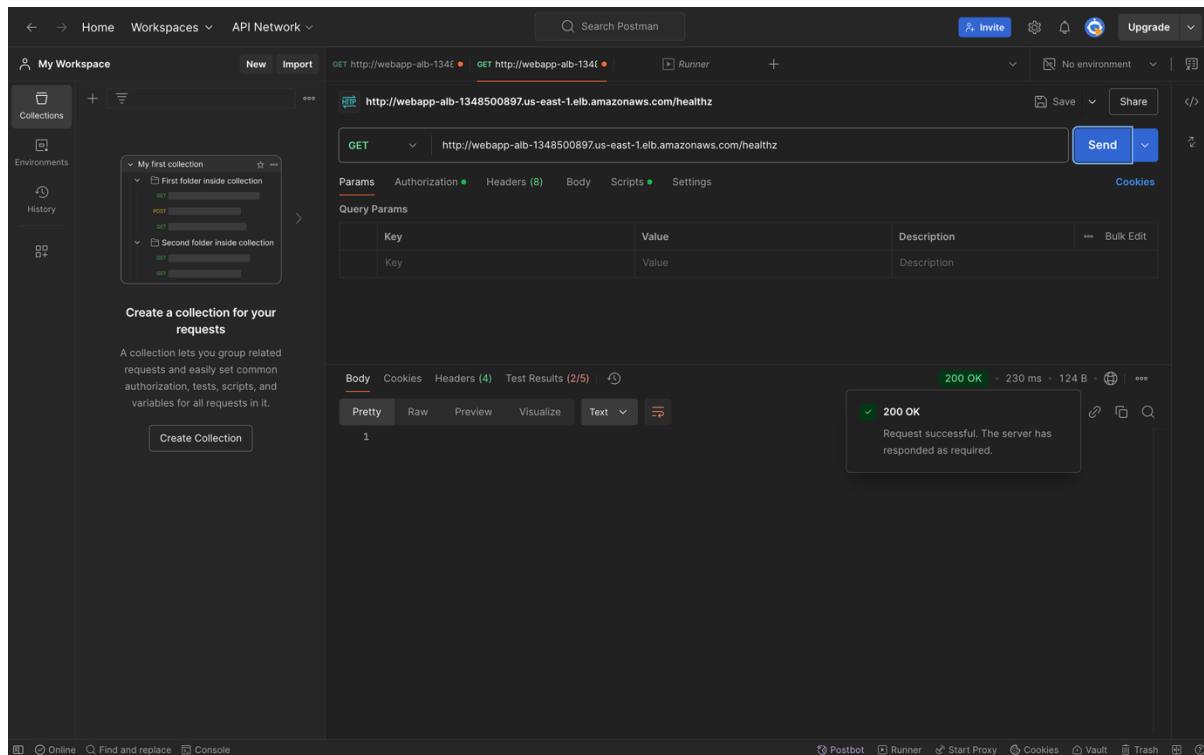
The screenshot displays CloudWatch monitoring details, showing key metrics such as CPU utilization, network traffic, and instance performance over time.

This screenshot shows the AWS EC2 Auto Scaling Groups interface with the 'Monitoring' tab selected. The left sidebar is collapsed, and the main navigation bar includes 'Search' and 'Option+S'. The top navigation bar shows 'N. Virginia' and 'Yash Tambakhe'. The main content area has tabs: 'Details', 'Integrations - new', 'Automatic scaling', 'Instance management', 'Instance refresh', 'Activity' (disabled), and 'Monitoring' (selected). The 'Monitoring' section is titled 'CloudWatch monitoring details' and includes tabs for 'Auto Scaling' (selected) and 'EC2'. It shows configuration for 'Auto Scaling group metrics collection': 'Enable' (unchecked), 'All times shown are in UTC.' (checked), and 'View all CloudWatch metrics' (link). There is also an 'Alarm recommendations' section with a dropdown for 'UTC timezone' set to '3h 1d 1w' and a 'UTC timezone' dropdown. Below these are several input fields for CloudWatch Metrics: 'Minimum Group Size...', 'Maximum Group Size...', 'Desired Capacity (Co...)', 'In Service Instances (...)', 'Pending Instances (C...)', 'Standby Instances (C...)', 'Terminating Instance...', and 'Total Instances (Count)'. Each field has a range slider and a timestamp (05:48 to 08:47). The bottom of the page includes standard AWS footer links: CloudShell, Feedback, © 2024, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

The screenshot shows the Amazon CloudWatch Alarms console monitoring a metric alarm named cpu-utilization-high, which tracks CPU utilization exceeding 70% for 2 data points within 10 minutes. The alarm is currently in the OK state, with a graph displaying CPU usage below the threshold.



The screenshot shows a GET request to the /healthz endpoint, which returned a 200 OK response. This indicates that the server is running and healthy.



The screenshot shows a successful POST request to create a user. The server responded with a 201 Created status, and the response body contains the user details in JSON format, confirming that the user was created successfully.

The screenshot displays the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are visible. The main workspace is titled 'My Workspace'. A collection named 'New Collection' is selected, containing a single request labeled 'GET New Request'. The request URL is set to `http://webapp-alb-1348500897.us-east-1.elb.amazonaws.com/v1/user`. The method is set to 'POST'. The 'Authorization' tab is selected, showing 'Basic Auth' as the type, with 'Username' set to `sam.email2@example.com` and 'Password' set to `samplePassword1234`. The 'Body' tab is selected, showing a JSON response with the following content:

```
1 {
2     "ID": "abc4fba3-35b8-44f3-ae92-329efbc60974",
3     "email": "sample.email2@example.com",
4     "first_name": "Sample",
5     "last_name": "User",
6     "account_created": "2024-12-08T03:25:44.760147072Z",
7     "account_updated": "2024-12-08T03:25:44.760147072Z"
8 }
```

The status bar at the bottom indicates a '201 Created' response with a duration of 292 ms and a size of 362 B. Other options like 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and 'Help' are also visible.