AWS EC2 /ALB/ASG : POC

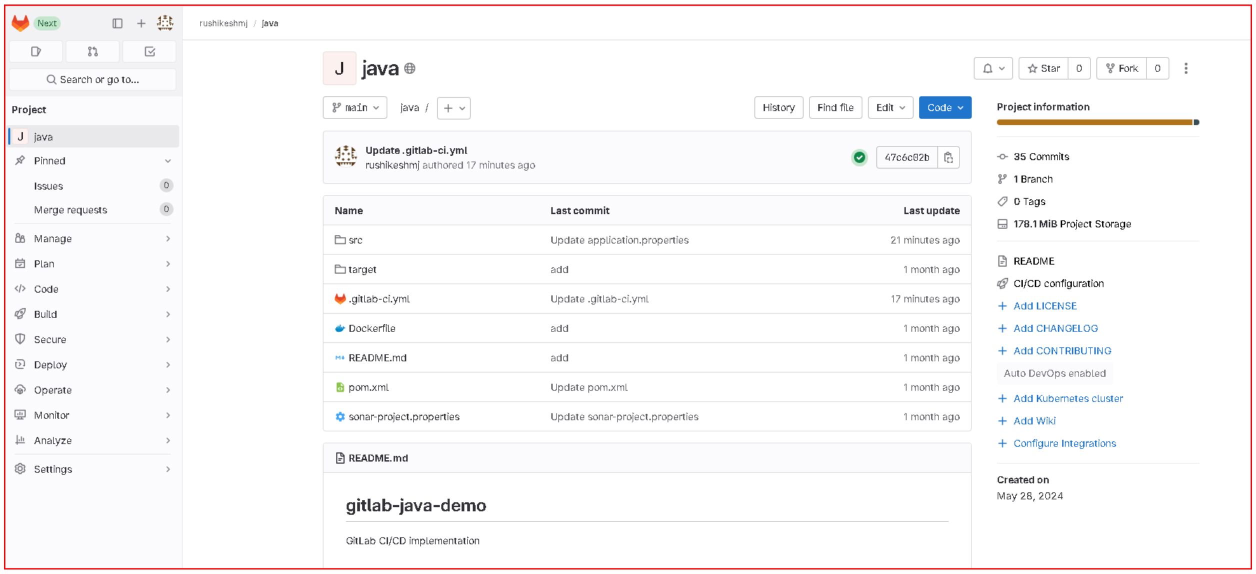
Task 1 : Perform CI/CD Steps & Push Image to AWS ECR

Setup CI/CD Pipeline:

Choose your CI/CD tool - GitLab

Configure your pipeline to build your application, run tests, and build a Docker image.

Repo : https://gitlab.com/rushikeshmj/java.git



rushikeshmj / java

java

Update .gitlab-ci.yml
rushikeshmj authored 17 minutes ago — 47c6c82b

Name	Last commit	Last update
src	Update application.properties	21 minutes ago
target	add	1 month ago
.gitlab-ci.yml	Update .gitlab-ci.yml	17 minutes ago
Dockerfile	add	1 month ago
README.md	add	1 month ago
pom.xml	Update pom.xml	1 month ago
sonar-project.properties	Update sonar-project.properties	1 month ago

Project information
35 Commits
1 Branch
0 Tags
178.1 MiB Project Storage

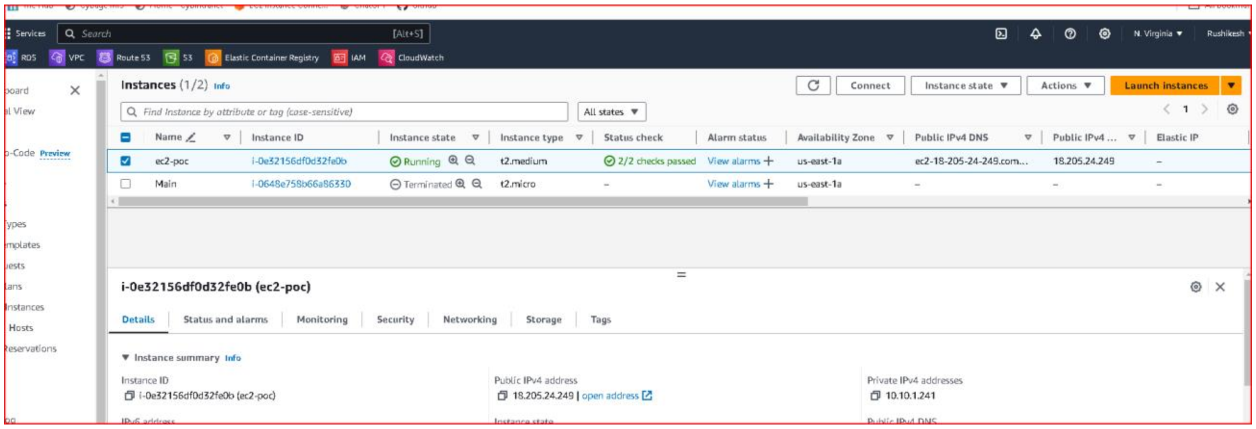
Created on
May 28, 2024

README.md

gitlab-java-demo

GitLab CI/CD implementation

Launch EC2 instance



Instances (1/2)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
ec2-poc	i-0e32156df0d32fe0b	Running	t2.medium	2/2 checks passed	View alarms	us-east-1a	ec2-18-205-24-249.com...	18.205.24.249	–
Main	i-0648e758b66a86330	Terminated	t2.micro	–	View alarms	us-east-1a	–	–	–

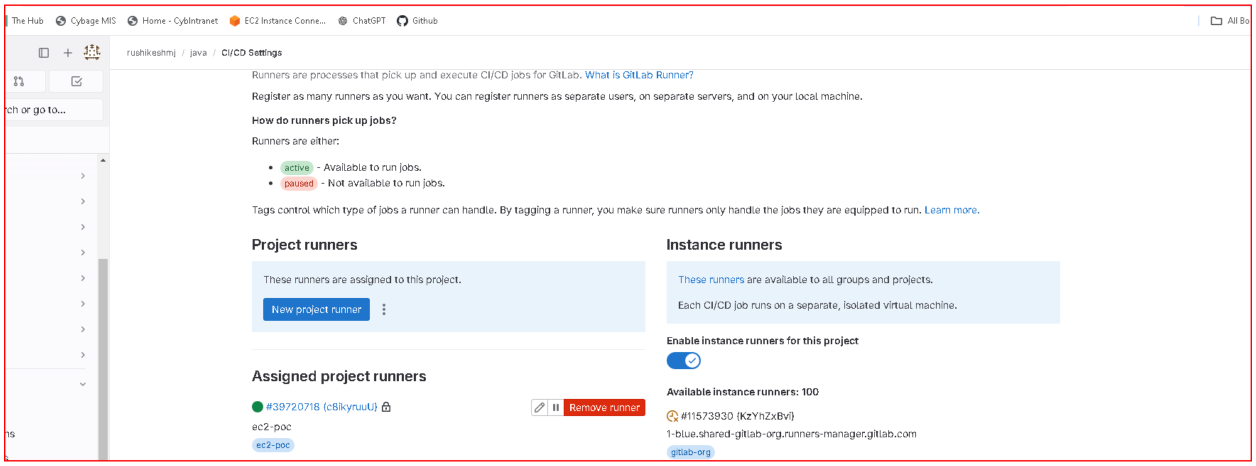
i-0e32156df0d32fe0b (ec2-poc)

Instance summary

Instance ID: i-0e32156df0d32fe0b (ec2-poc)
Public IPv4 address: 18.205.24.249 | open address
Private IPv4 addresses: 10.10.1.241

Run few command on EC2 and configure the runner

```
sudo apt-get update

sudo apt-get install docker.io

# Download the binary for your system

sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

# Give it permission to execute

sudo chmod +x /usr/local/bin/gitlab-runner

# Create a GitLab Runner user

sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

# Install and run as a service

sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner

sudo gitlab-runner start

sudo gitlab-runner register --url https://gitlab.com/ --registration-token GR1348941WyG1vEqykTtxMytRzSdi

vi /etc/gitlab-runner/config.toml

sudo gitlab-runner restart

sudo gitlab-runner status
```



Runners are processes that pick up and execute CI/CD jobs for GitLab. What is GitLab Runner?

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine.

How do runners pick up jobs?

Runners are either:

- active - Available to run jobs.
- paused - Not available to run jobs.

Tags control which type of jobs a runner can handle. By tagging a runner, you make sure runners only handle the jobs they are equipped to run. Learn more.

Project runners

These runners are assigned to this project.

Assigned project runners

● #39720718 (c8ikyruuU)
ec2-poc
ec2-poc

Instance runners

These runners are available to all groups and projects.

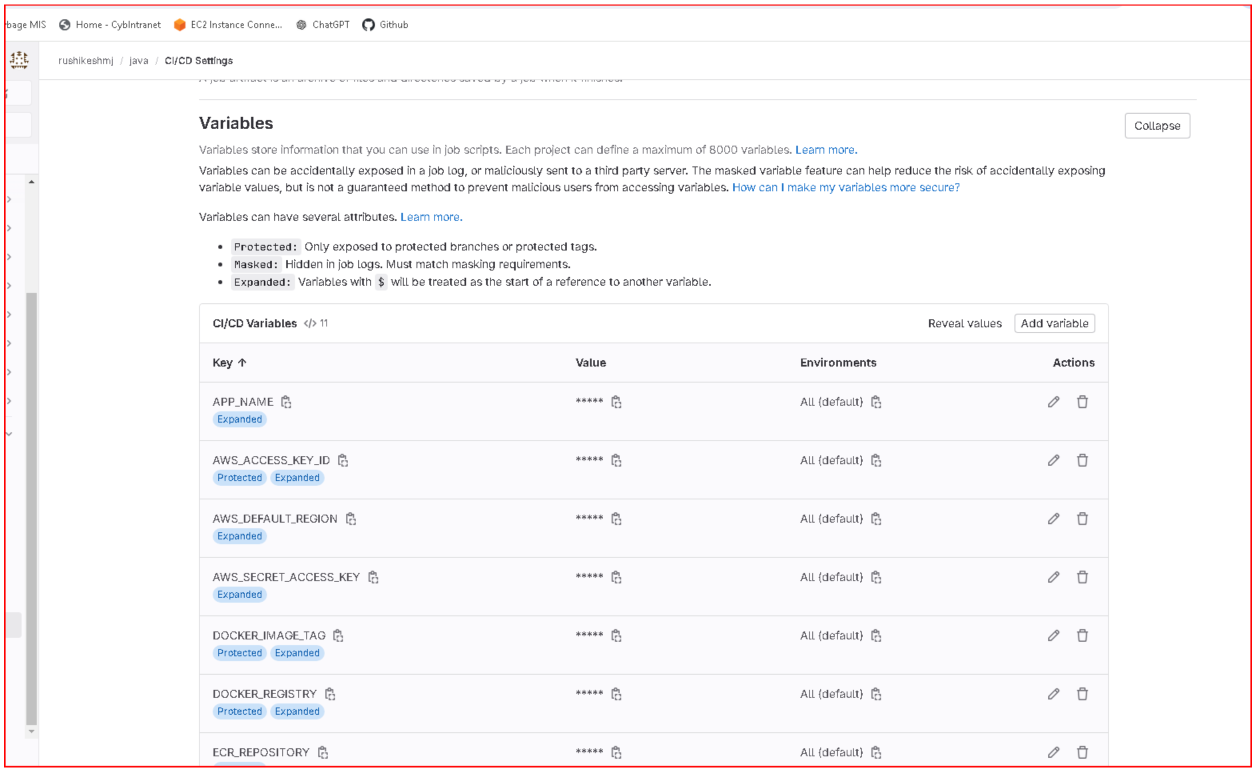
Each CI/CD job runs on a separate, isolated virtual machine.

Enable instance runners for this project ✓

Available instance runners: 100

#11573930 (KzYhZxBvi)
1-blue.shared-gitlab-org.runners-manager.gitlab.com
gitlab-org

Save the ENV variables in CI/CD – Variable section



rushikeshmj / java / CI/CD Settings

Variables

Variables store information that you can use in job scripts. Each project can define a maximum of 8000 variables. Learn more.

Variables can be accidentally exposed in a job log, or maliciously sent to a third party server. The masked variable feature can help reduce the risk of accidentally exposing variable values, but is not a guaranteed method to prevent malicious users from accessing variables. How can I make my variables more secure?

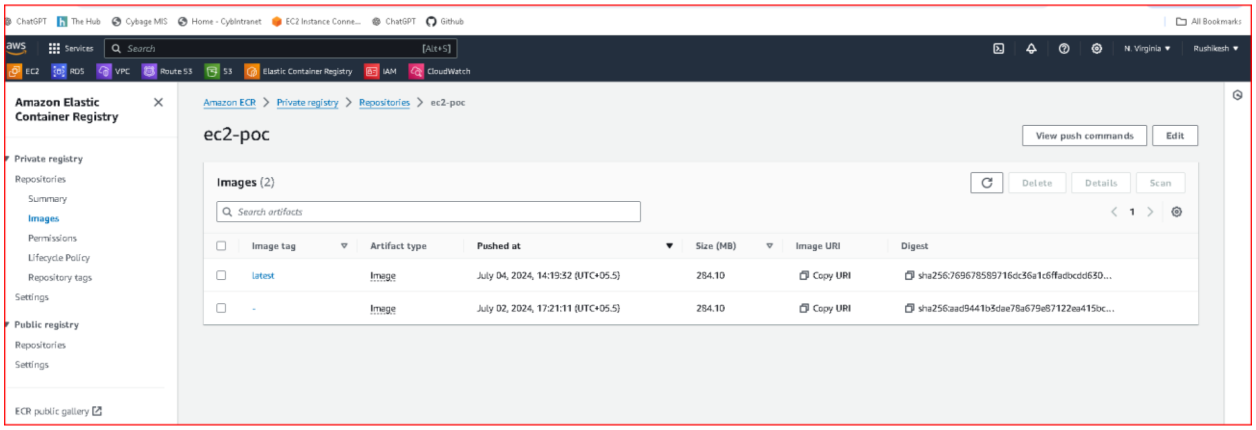
Variables can have several attributes. Learn more.

- `Protected:` Only exposed to protected branches or protected tags.
- `Masked:` Hidden in job logs. Must match masking requirements.
- `Expanded:` Variables with `$` will be treated as the start of a reference to another variable.

CI/CD Variables 11

Key	Value	Environments	Actions
APP_NAME (Expanded)	*****	All (default)	
AWS_ACCESS_KEY_ID (Protected, Expanded)	*****	All (default)	
AWS_DEFAULT_REGION (Expanded)	*****	All (default)	
AWS_SECRET_ACCESS_KEY (Expanded)	*****	All (default)	
DOCKER_IMAGE_TAG (Protected, Expanded)	*****	All (default)	
DOCKER_REGISTRY (Protected, Expanded)	*****	All (default)	
ECR_REPOSITORY	*****	All (default)	

Push Image to AWS ECR:



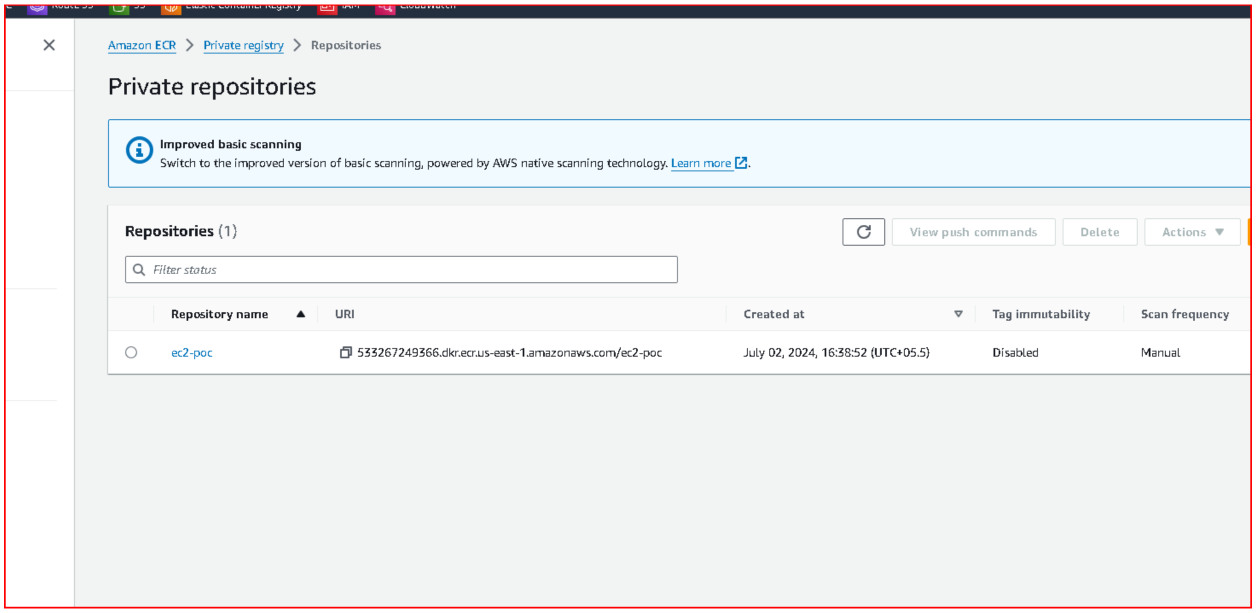
Amazon ECR > Private registry > Repositories > ec2-poc

ec2-poc

Images (2)

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
	latest	Image	July 04, 2024, 14:19:32 (UTC+05.5)	284.10	Copy URI	sha256:769678589716dc36a1c6ffadbcdd630...
	-	Image	July 02, 2024, 17:21:11 (UTC+05.5)	284.10	Copy URI	sha256:aad9441b3dae78a679e87122ea415bc...

Create an AWS ECR repository.



Amazon ECR > Private registry > Repositories

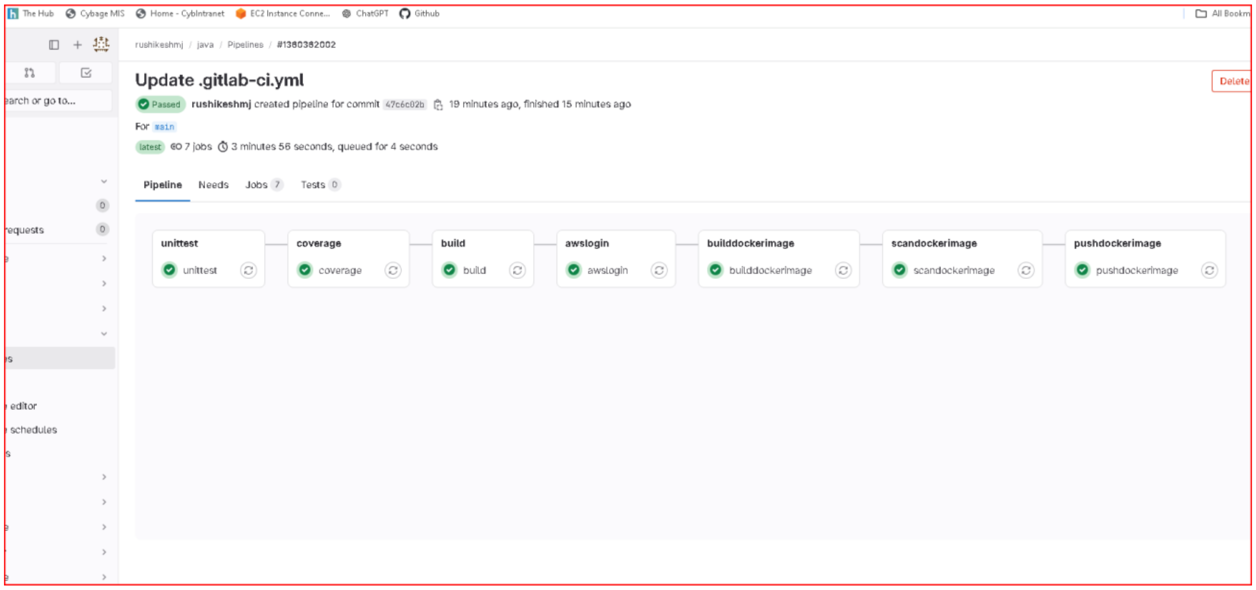
Private repositories

Improved basic scanning
Switch to the improved version of basic scanning, powered by AWS native scanning technology. Learn more.

Repositories (1)

Repository name	URI	Created at	Tag immutability	Scan frequency
ec2-poc	533267249366.dkr.ecr.us-east-1.amazonaws.com/ec2-poc	July 02, 2024, 16:38:52 (UTC+05.5)	Disabled	Manual

Configure your CI/CD pipeline to push the Docker image to ECR after a successful build.



rushikeshmj / java / Pipelines / #1380382002

Update .gitlab-ci.yml

Passed rushikeshmj created pipeline for commit 47c6c02b 19 minutes ago, finished 15 minutes ago

For main

latest 7 jobs 3 minutes 56 seconds, queued for 4 seconds

Pipeline Needs Jobs 7 Tests 0

unittest	coverage	build	awslogin	builddockerimage	scandockerimage	pushdockerimage
✓ unittest	✓ coverage	✓ build	✓ awslogin	✓ builddockerimage	✓ scandockerimage	✓ pushdockerimage

TASK 2 : Perform EC2 Instance Creation Steps

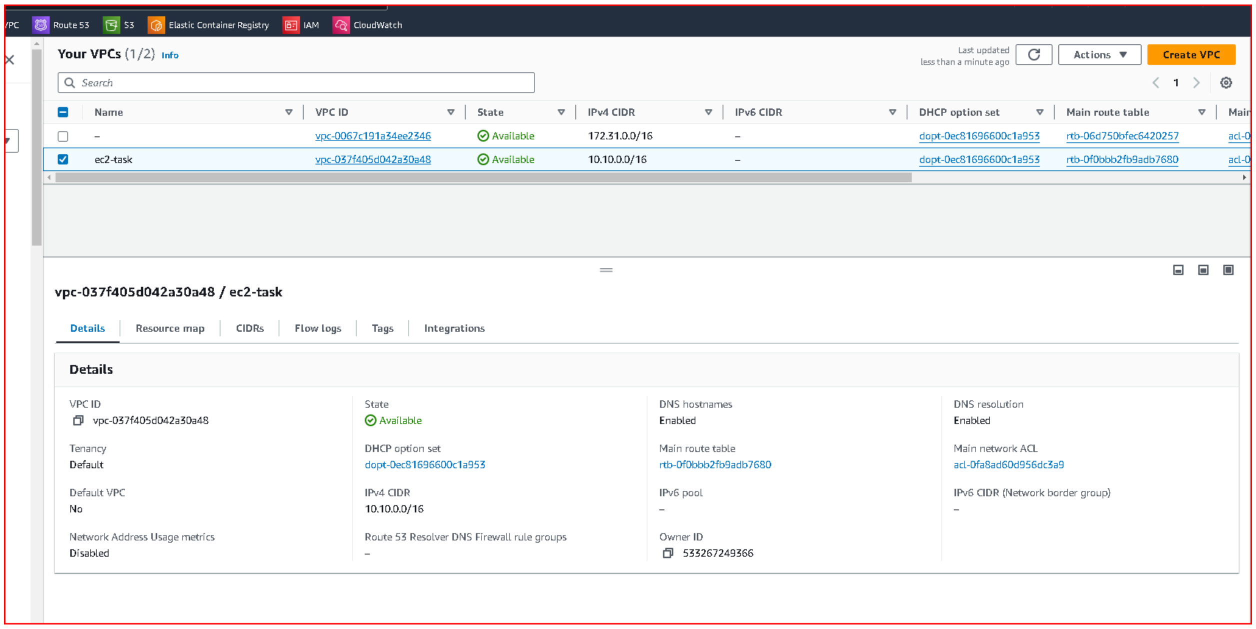
Create VPC and Subnets

VPC Creation:

Navigate to AWS Management Console > VPC > Create VPC.

Define the CIDR block

Create the VPC.



Your VPCs (1/2) Info

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
–	vpc-0067c191a34ee2346	Available	172.31.0.0/16	–	dopt-0ec81696600c1a953	rtb-06d750bfec6420257
ec2-task	vpc-037f405d042a30a48	Available	10.10.0.0/16	–	dopt-0ec81696600c1a953	rtb-0f0bbb2fb9adb7680

vpc-037f405d042a30a48 / ec2-task

Details

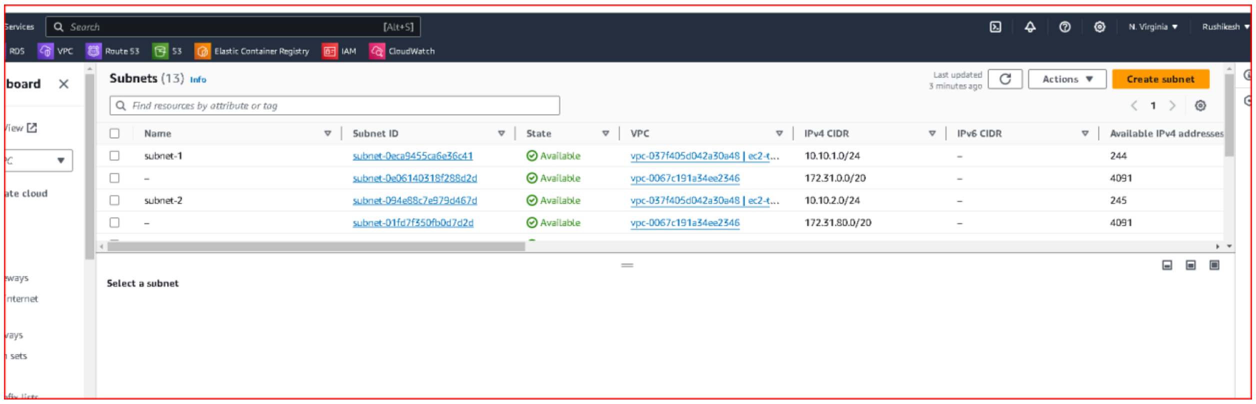
VPC ID: vpc-037f405d042a30a48	State: Available	DNS hostnames: Enabled	DNS resolution: Enabled
Tenancy: Default	DHCP option set: dopt-0ec81696600c1a953	Main route table: rtb-0f0bbb2fb9adb7680	Main network ACL: acl-0fa8ad60d956dc3a9
Default VPC: No	IPv4 CIDR: 10.10.0.0/16	IPv6 pool: –	IPv6 CIDR (Network border group): –
Network Address Usage metrics: Disabled	Route 53 Resolver DNS Firewall rule groups: –	Owner ID: 533267249366	

Subnets Creation:

Inside the VPC dashboard, navigate to Subnets > Create Subnet.

Define subnet CIDR blocks

Create at least two subnets in different AZs for high availability.



Subnets (13) Info

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 addresses
subnet-1	subnet-0eca9455ca6e36c41	Available	vpc-037f405d042a30a48 | ec2-t...	10.10.1.0/24	–	244
–	subnet-0e06140318f288d2d	Available	vpc-0067c191a34ee2346	172.31.0.0/20	–	4091
subnet-2	subnet-094e88c7e979d467d	Available	vpc-037f405d042a30a48 | ec2-t...	10.10.2.0/24	–	245
–	subnet-01fd7f350fb0d7d2d	Available	vpc-0067c191a34ee2346	172.31.80.0/20	–	4091

Select a subnet

Security Groups:

Create security groups to control inbound and outbound traffic to your EC2 instances.

Launch EC2 Instances:

Navigate to EC2 Dashboard > Instances > Launch Instance.

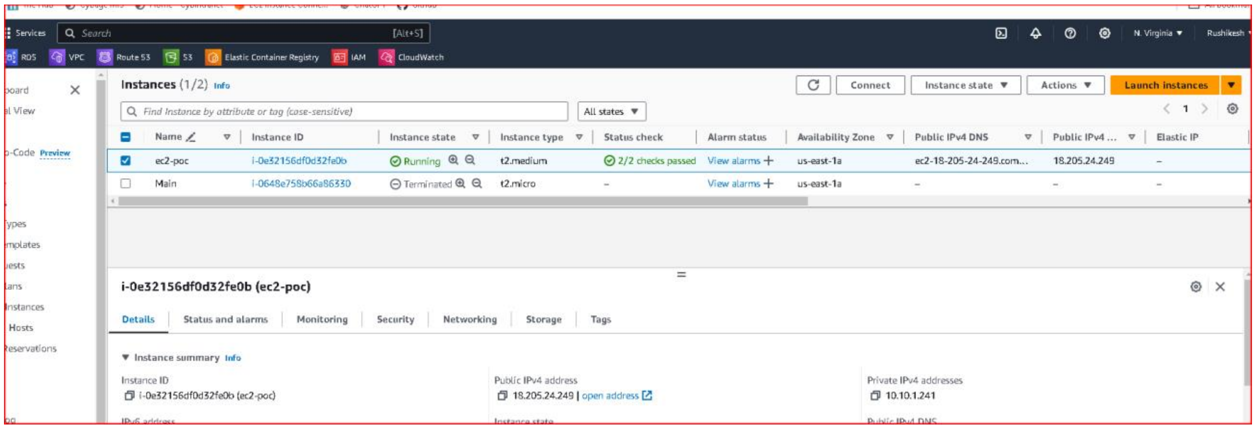
Choose an Amazon Machine Image (AMI).

Select instance type, configure instance details (subnet etc.).

Configure storage and tags as necessary.

Configure security group settings.

Review and launch the instance.



Instances (1/2) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
✓	ec2-poc	i-0e32156df0d32fe0b	Running	t2.medium	2/2 checks passed	View alarms	us-east-1a	ec2-18-205-24-249.com...	18.205.24.249	–
	Main	i-0648e758b66a86330	Terminated	t2.micro	–	View alarms	us-east-1a	–	–	–

i-0e32156df0d32fe0b (ec2-poc)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0e32156df0d32fe0b (ec2-poc)	18.205.24.249 | open address	10.10.1.241

TASK 3 : Creating Load Balancer and Deploying Application

Create Application Load Balancer (ALB):

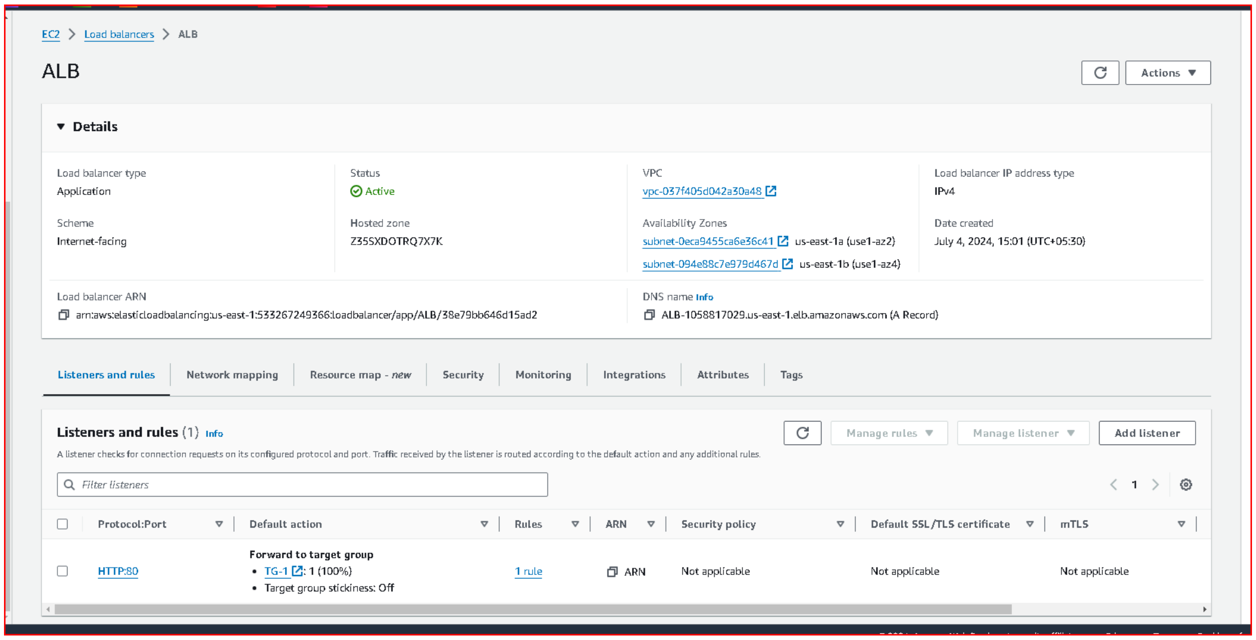
Navigate to EC2 Dashboard > Load Balancers > Create Load Balancer.

Choose Application Load Balancer.

Configure listeners (HTTP/HTTPS), configure security settings.

Select VPC and subnets.

Configure routing and health checks.


EC2 > Load balancers > ALB

ALB

Details

Load balancer type Application	Status Active	VPC vpc-037f405d042a30a48	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0eca9455ca6e36c41 us-east-1a (use1-az2) subnet-094e88c7e979d467d us-east-1b (use1-az4)	Date created July 4, 2024, 15:01 (UTC+05:30)

Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:533267249366:loadbalancer/app/ALB/38e79bb646d15ad2

DNS name Info
ALB-1058817029.us-east-1.elb.amazonaws.com (A Record)

Listeners and rules (1) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

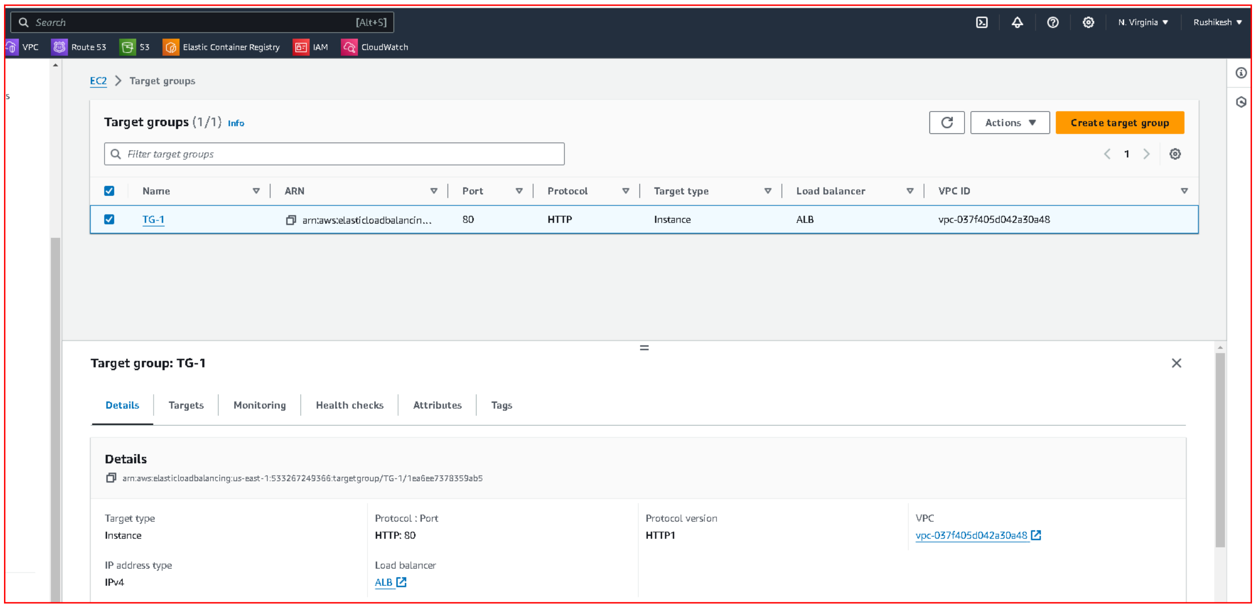
Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS
HTTP:80	Forward to target group • TG-1: 1 (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable

Set Up Target Groups:

Define target groups for your EC2 instances.

Specify instance protocol and port.

Register instances from the EC2 dashboard or via Auto Scaling.


EC2 > Target groups

Target groups (1/1) Info

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
TG-1	arn:aws:elasticloadbalancin...	80	HTTP	Instance	ALB	vpc-037f405d042a30a48

Target group: TG-1

Details

arn:aws:elasticloadbalancing:us-east-1:533267249366:targetgroup/TG-1/1ea6ee7378359ab5

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-037f405d042a30a48
IP address type IPv4	Load balancer ALB		

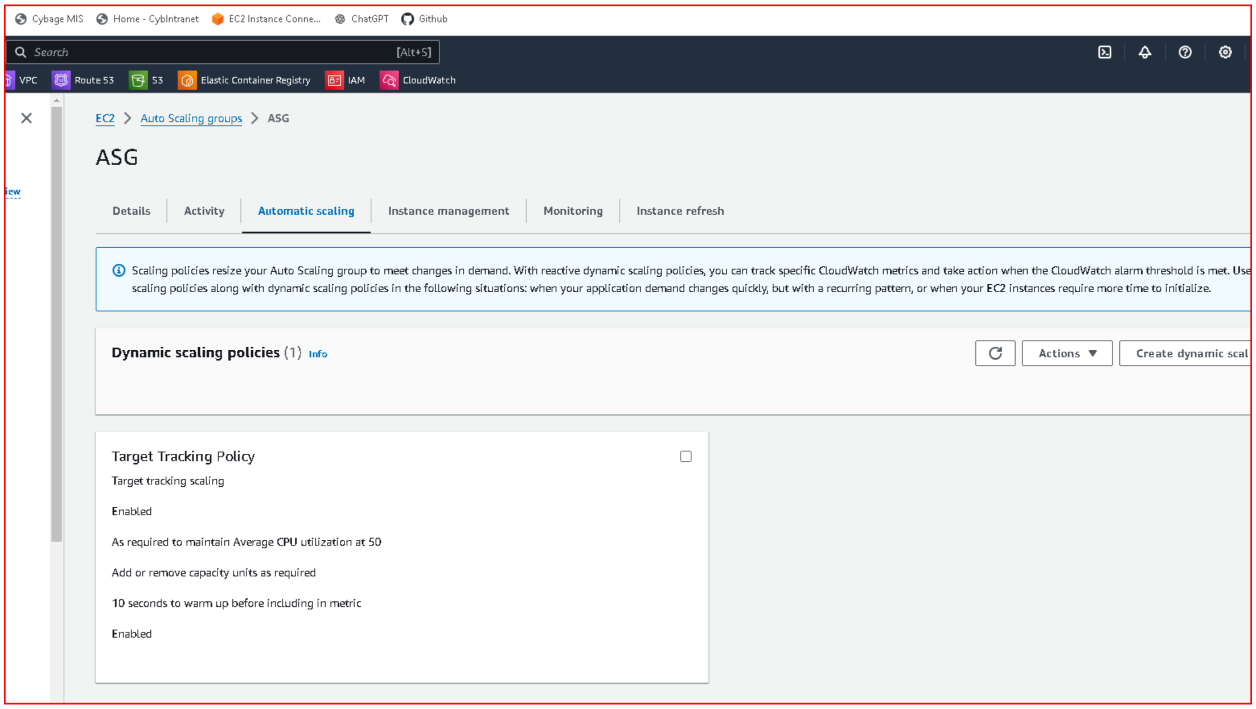
Auto Scaling Group

Navigate to EC2 Dashboard > Auto Scaling > Auto Scaling Groups.

Create a new Auto Scaling group.

Configure launch configuration (AMI, instance type, key pair, etc.).

Define scaling policies based on metrics like CPU utilization or request count.



ASG

Scaling policies resize your Auto Scaling group to meet changes in demand. With reactive dynamic scaling policies, you can track specific CloudWatch metrics and take action when the CloudWatch alarm threshold is met. Use scaling policies along with dynamic scaling policies in the following situations: when your application demand changes quickly, but with a recurring pattern, or when your EC2 instances require more time to initialize.

Dynamic scaling policies (1) Info

Target Tracking Policy

Target tracking scaling

Enabled

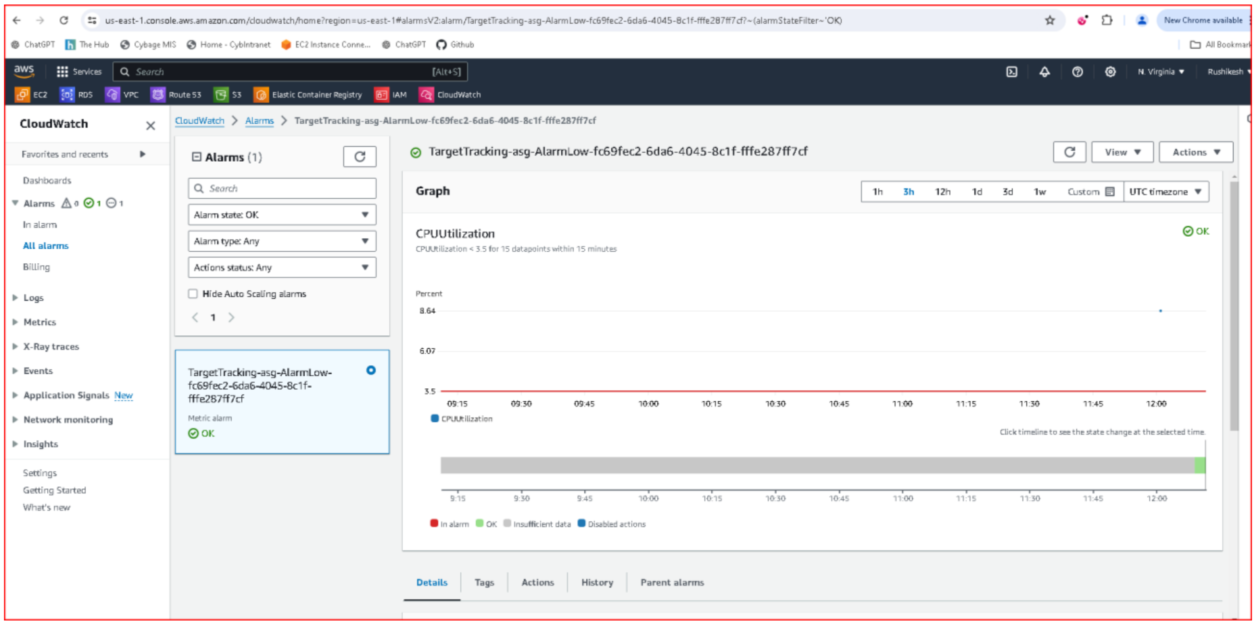
As required to maintain Average CPU utilization at 50

Add or remove capacity units as required

10 seconds to warm up before including in metric

Enabled

Cloudwatch Alarms



us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#alarmsV2:alarm/TargetTracking-asg-AlarmLow-fc69fec2-6da6-4045-8c1f-fffe287ff7cf?~(alarmStateFilter~'OK)

Alarm state: OK

Alarm type: Any

Actions status: Any

TargetTracking-asg-AlarmLow-fc69fec2-6da6-4045-8c1f-fffe287ff7cf

Metric alarm

OK

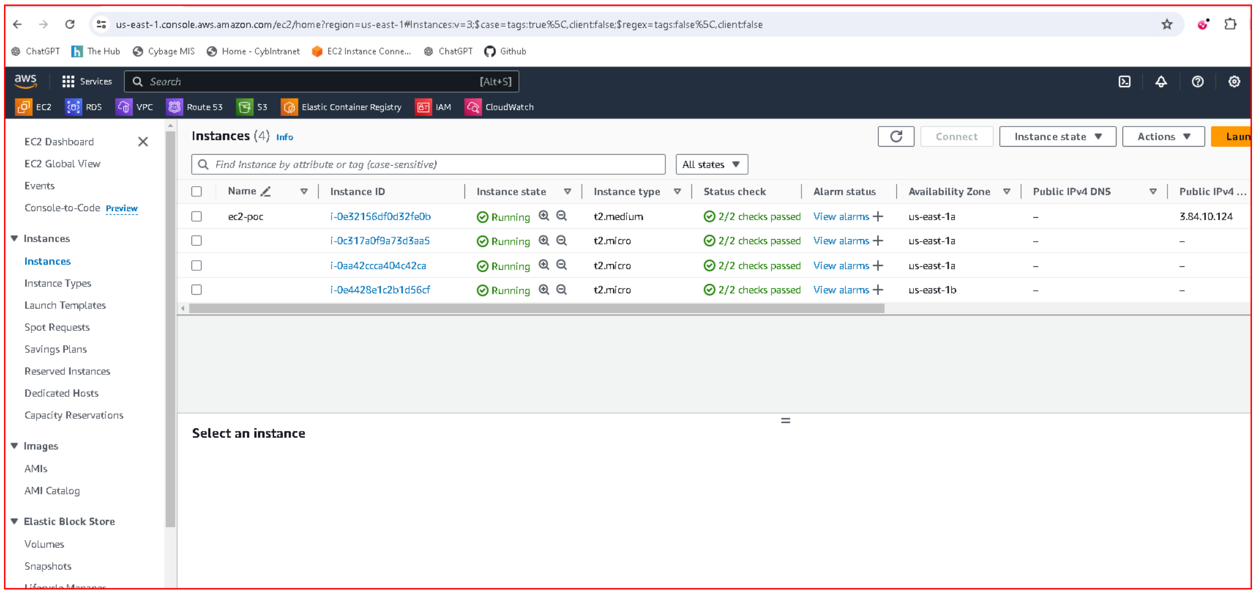
TargetTracking-asg-AlarmLow-fc69fec2-6da6-4045-8c1f-fffe287ff7cf

CPUUtilization

CPUUtilization < 3.5 for 15 datapoints within 15 minutes

OK

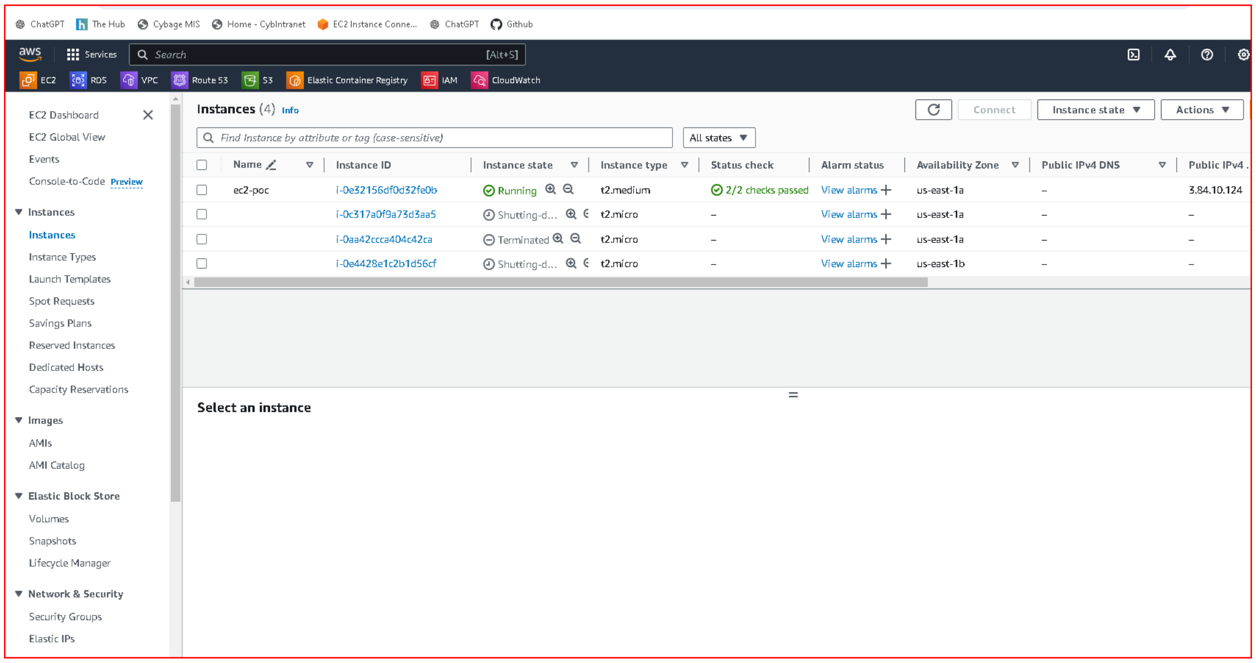
After increase load on instance Autoscaling launches instances



Instances (4) Info

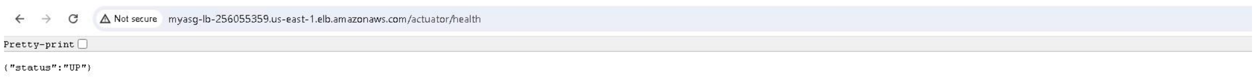
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
ec2-poc	i-0e32156df0d32fe0b	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1a	–	3.84.10.124
	i-0c317a0f9a73d3aa5	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	–	–
	i-0aa42ccca404c42ca	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	–	–
	i-0e4428e1c2b1d56cf	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	–	–

After decrease load on instance Autoscaling terminates instances



Instances (4) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 .
ec2-poc	i-0e32156df0d32fe0b	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1a	–	3.84.10.124
	i-0c317a0f9a73d3aa5	Shutting-d...	t2.micro	–	View alarms +	us-east-1a	–	–
	i-0aa42ccca404c42ca	Terminated	t2.micro	–	View alarms +	us-east-1a	–	–
	i-0e4428e1c2b1d56cf	Shutting-d...	t2.micro	–	View alarms +	us-east-1b	–	–


Not secure
myasg-lb-256055359.us-east-1.elb.amazonaws.com/actuator/health
Pretty-print

```
{"status":"UP"}
```