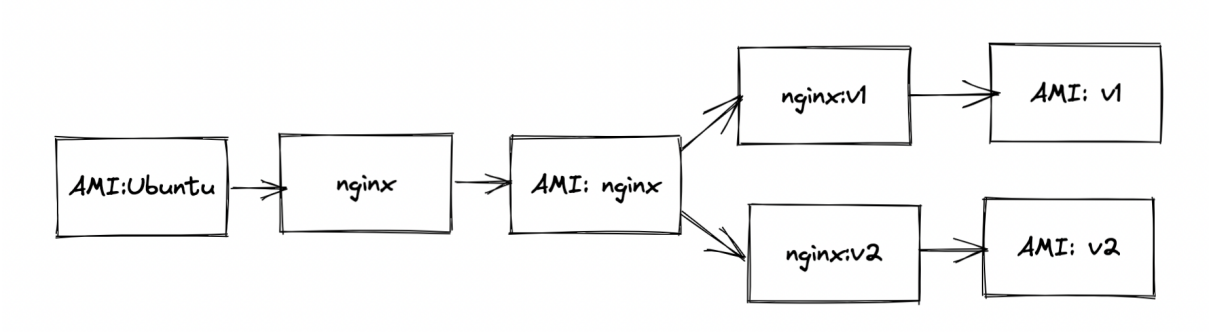
AWS Assignments

ASSIGNMENT-1

```
Assignment 1

In this assignment you have to learn about deployment strateg
    - Recreate deployment
    - Rolling deployment
    - Blue-green deployment
    - A/B  deployment
    - Canary deployment

After this you need to implement these 2 below deployment str
Must Do
    - Recreate deployment
    - Rolling deployment
Good to  Do
    - Blue Green deployment
    - Canary deployment

Recommendation
    - Don't straight forward jump into creating the utility,

github-link --> https://github.com/OT-TRAINING/DeploymentStra

Hint :
1. Use ASG
2. Use LB
3. Use EC2 service
4. Use AMI Snapshot
```

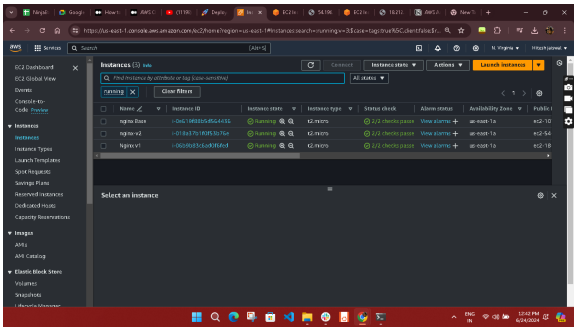
PART-1 SERVER, IMAGE AND SNAPSHOT



Server Base

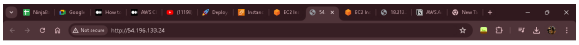
nginx-v1
AMI nginx-v1
AMI Snapshot

Version 1

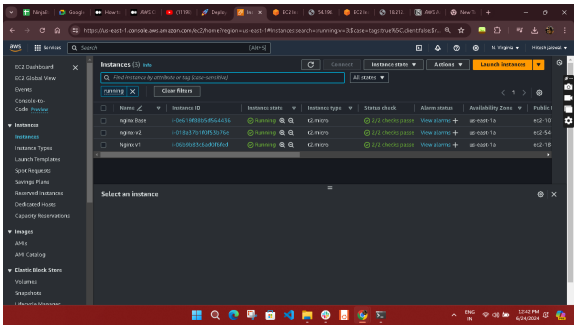


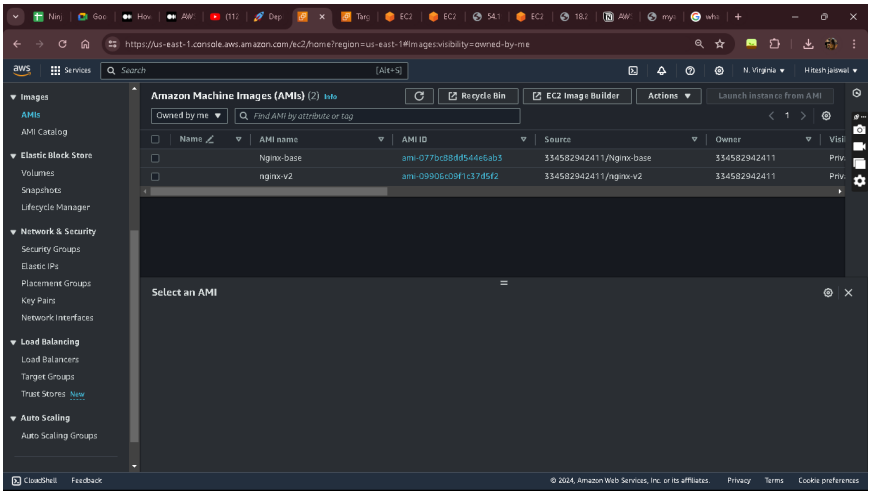
Server Base

nginx-v2
AMI nginx-v2
AMI Snapshot



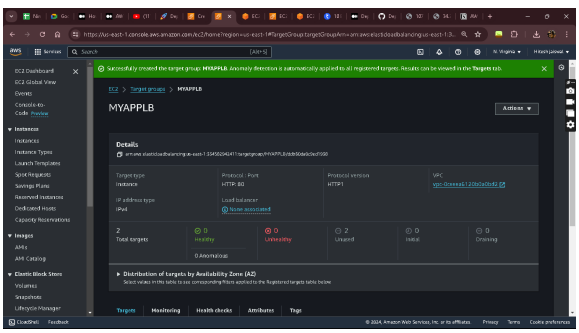
Version 2

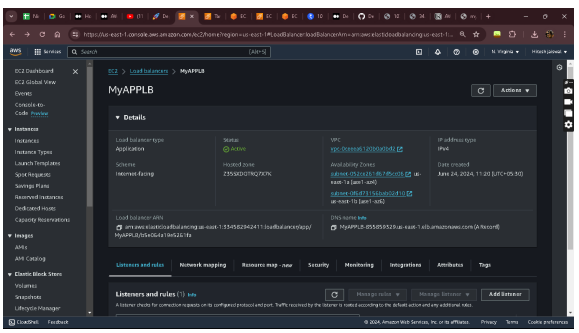


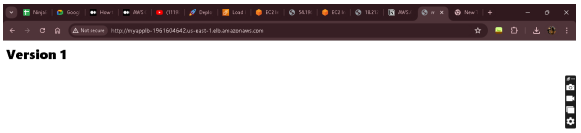


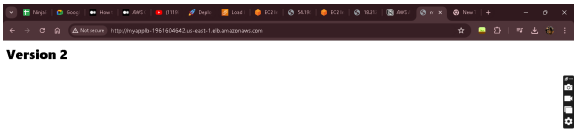
PART -2 Application Load Balancer

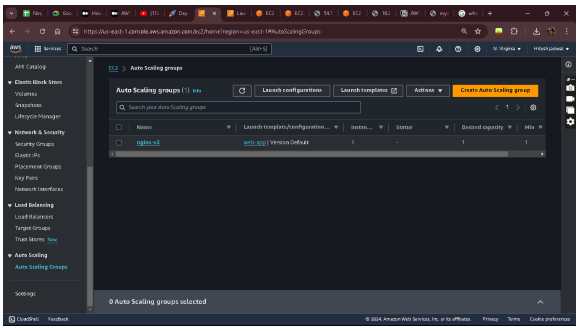
Target Group & Load Balancer











Auto-Scaling Group

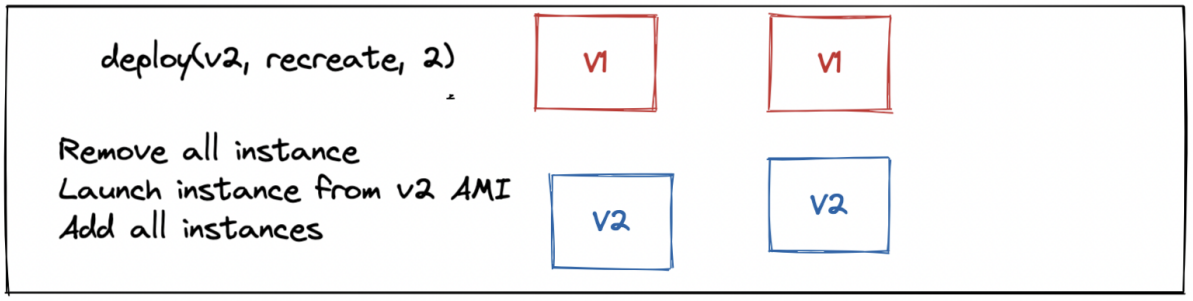
nginx v1 Removal - Deleted via Auto Scaling group
nginx v2 Addition Template - Adding Via Auto Scaling

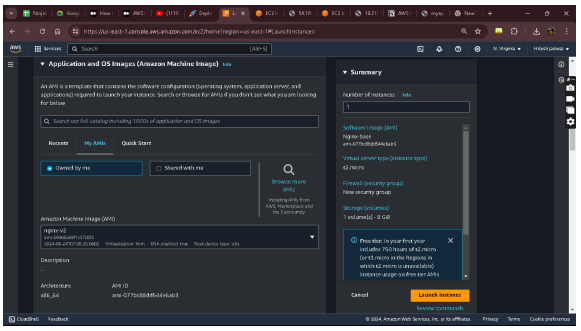
PART-3

Deployment Strategy

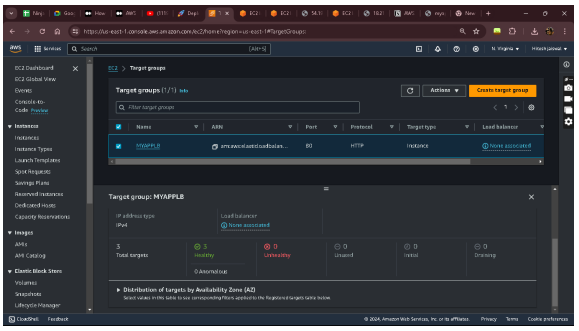
1. Recreate Strategy :- a basic deployment pattern which simply shuts down all the old pods and replaces them with new ones
. We don’t use it much as the downtime is high in this.

Removed all Instance associated with v1 at one time





Through Version 2 AMI

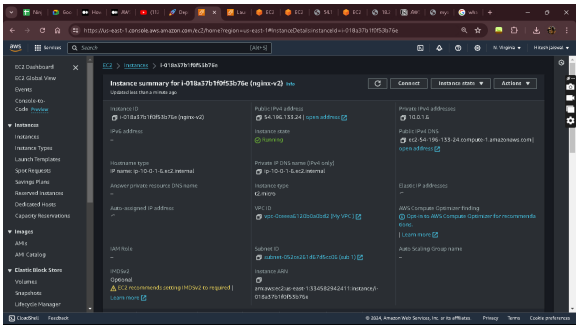


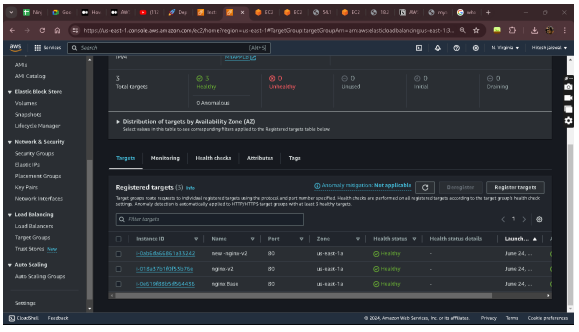
Health Checks for Targets

2. Rolling Development Strategy :- A rolling deployment is a deployment strategy that slowly replaces previous versions of an application with new versions of an application by completely replacing the infrastructure on which the application is running.

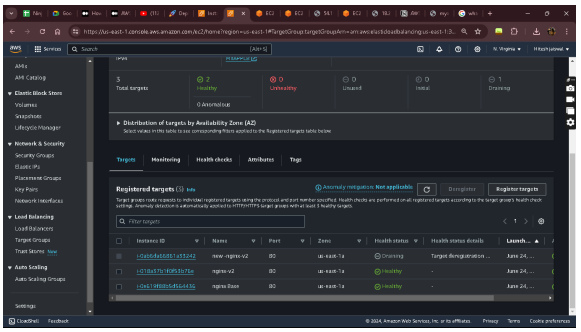
   Deregister nginx v2 AMI associated instance 1-by -1

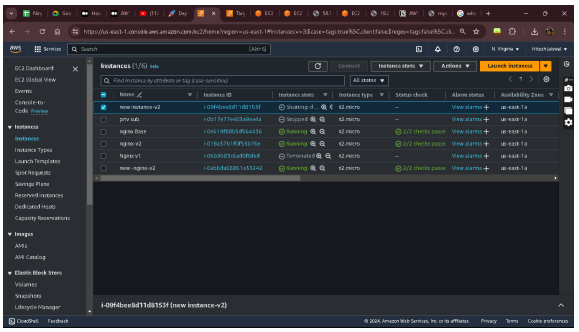
   Added nginx v2 and New-nginx-v2





Now deregistering new-nginx-v2 and Deleting



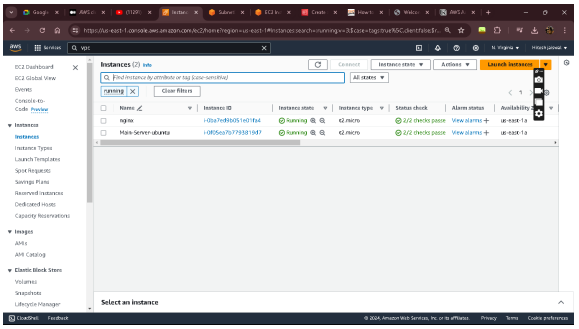


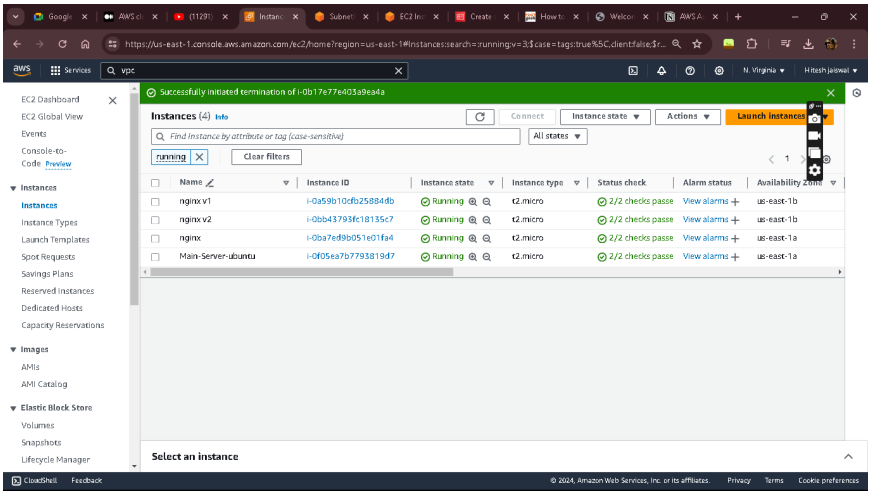
NOW THROUGH AWS CLI

1. Launch Instance nginx - We had launched one main ubuntu AMI then using that AMI we launch AMI nginx as base server, now we will launch nginx v1 and nginx v2 using base server

Command :-

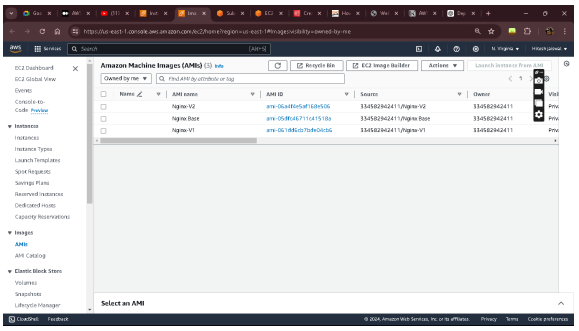
```
aws ec2 run-instances \
--image-id ami-0e001c9271cf7f3b9 \
--count 2 \
--instance-type t2.micro \
--key-name aws \
--security-group-ids sg-04caf7a20c859a1e3 \
--subnet-id subnet-052ce261d67d5cc06 \
--associate-public-ip-address
```



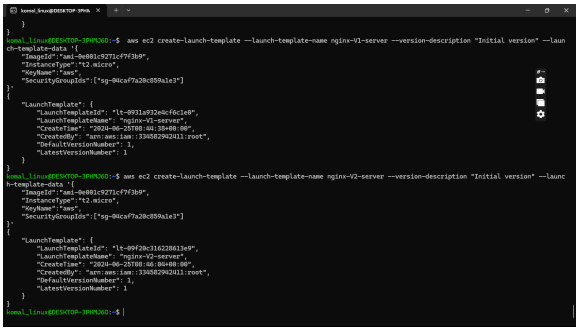


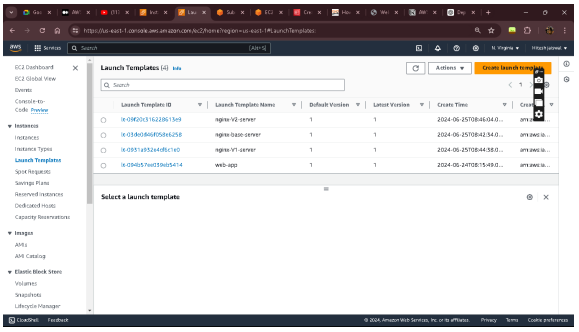
2. Create Image & Snapshot and Templates of Instance

Image & Snapshot



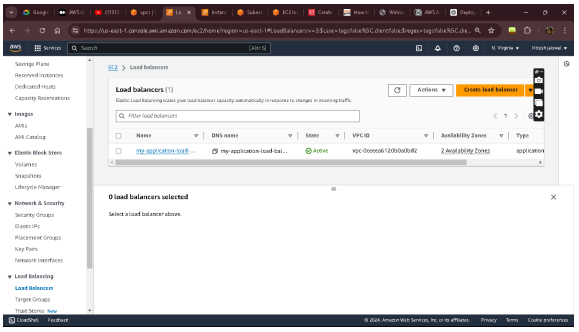
Template Launched





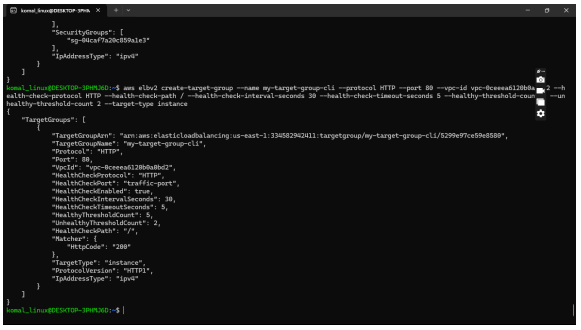
3. Create Load Balancer, Target Group & Auto Scaling

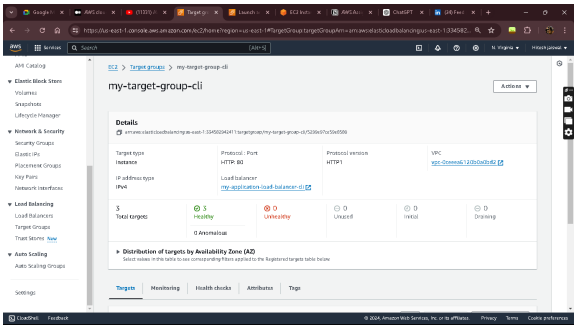
Load Balancer



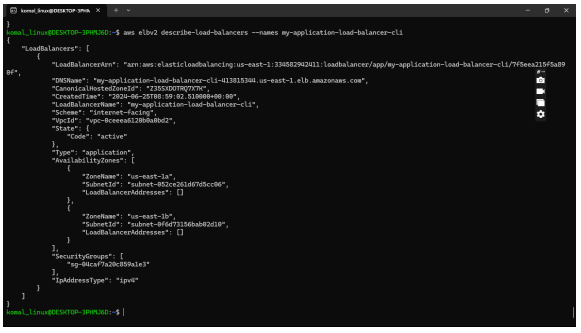
Target Group

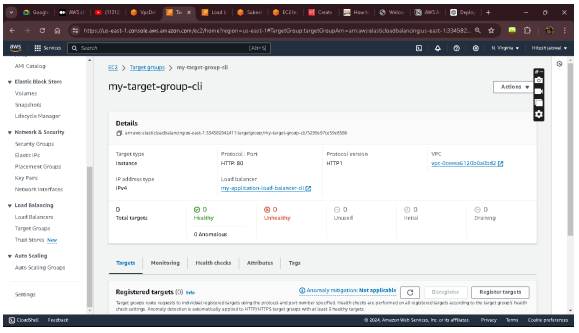
Created TG





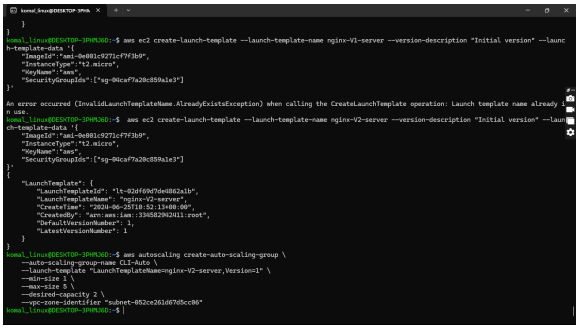
Attach to Load Balancer

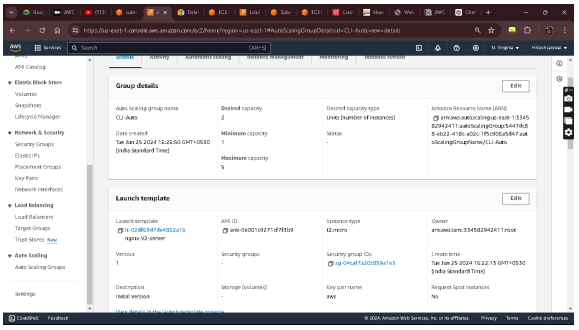


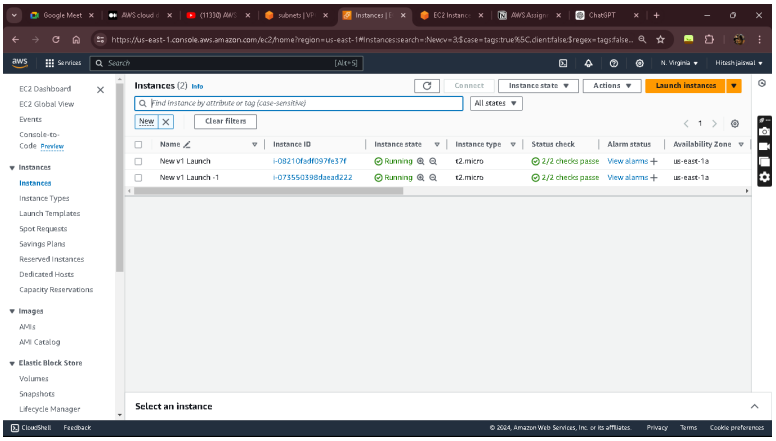


Attach Listener

1. Recreate Strategy - using AMI of V1 we will launch template associate it with auto scaling group so when V1 will be deleted V2 will launch itself with new version. and it will be updated. As I have done through cli in above screenshot (Launched Template with Auto Scaling group ami is of V1)







Command to terminate
aws ec2 terminate-instances --instance-ids $i-0492bd517bfb25fc8

We face downtime as every version deletes and it will time to make new version

2. Rolling Instance

In this we can roll back to previous version we can simply deregister using command

aws elbv2 deregister-targets \
--target-group-arn arn:aws:elasticloadbalancing:us-east-1:334582942411:targetgroup/my-target-group-cli/5299e97ce59e8580 \
--targets Id=$id of TG

we remove version line by line so we don’t face downtime

ASSIGNMENT-2

```
Assignment 2

"""Suppose your client has infrastructure running over the
AWS cloud, but for the past few days, the client applicatio
n has not been able to handle enough traffic and load, so h
e is thinking of setting up any reverse proxy for handling
the load as middleware. He called you from the DevOps team
```

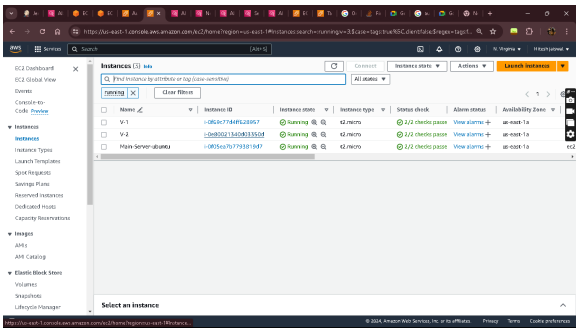
```
and explained the problem statement. You have explained ver
y well about the plan and design that you are going to perf
orm."""

You were told that nginx needs to be set up in between your
client and application service, which can handle the reques
t, but we need to setup one more thing, which is the basic
HA (high availability) and DR (disaster recovery) of that m
iddleware setup. Tasks would be done day-wise.

Day 1:
- Create instance and install nginx on that, after that cre
ate an AMI-1
- Create instance from the above AMI-1 and make V1
- Do some changes in same instance and  again create AMI-2
- Create instance from the above AMI-2 and make V2
- Make both V1 and V2 AMI's
- Now you have 2 AMI's total

You have to continue the strategy with autoscaling group, a
ttach asg on LB and follow the same strategy and make this
highly available by testing load test on the nginx server s
o that it would autoscale at the time of load.

Attach the policy to asg and increase the stress according
to policy mentioned then analyse the result. ( Try every po
licy )
    - Avg CPU utilization
    - Network bytes in/out
    - ALB requests count per target.
Somehow client needs version upgrade in Nginx so please upg
rade V1 to V2 that you have made

"""But the client has complained to us that our new version
V2 is not compatible with the service, so please revert bac
k. You need to revert back to the previous version. """

Day 2: """Client has told you about the change in it's Ngin
```

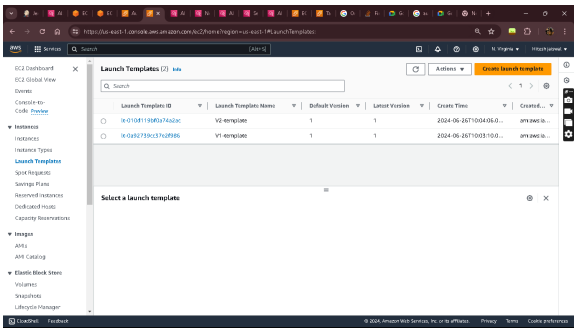
```
x as he needs this tool as hosting webpage too, so you need
to plan accordingly. Also, he has specially mentioned that
all tasks should be performed from their AWS environment, w
hich means all O/P would be done through EC2 only. """

- You need to pull image content which would be display at
the time of webpage hosting from VCS git and push your imag
es on S3 bucket using aws-cli. (Do not use secrete and acce
ss keys)

- You need to create a frontend including images in nginx p
age which will be fetched directly from S3 bucket.

Day3:
"""Now client has again told us to test it continuously to
see if ASG is working or not if my server got unhealthy."""

Enter in the server and make some changes in server in orde
r to make server unhealthy. Now analyze the result to see h
ow ASG can help you to maintain your instance desire state.

==========================================
(Remember the special points from client)
NOTE 1: You need to create the utility in such a way that i
t will make AMI of specific version and attach to the asg a
nd perform the rolling deployment strategie. In case of rev
ert back you should also have the function of revert back f
eature.

NOTE 2: But always remember first do it all the tasks manua
lly.
==========================================

GOOD TO DO:
Do the whole automation with Blue Green Deployment and add
CloudFront in front of Load Balancer to access the website
in order to overcome latency.
Continue to the previous assignment 1 and use your both ngi
```

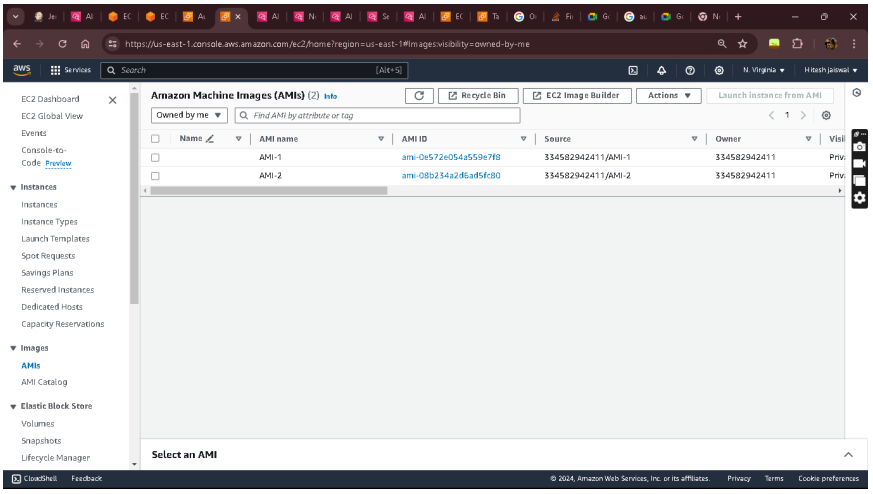
```
nx AMI's for displaying webpage and use your images from S3
bucket.

Day4:
"""Client called devOps and asked if it was possible to get
the two different details with single LB DNS, as he has two
webpages and wants to pass the traffic through a single ALB
but with a different path, like app.opstree.com/path1 or ap
p.opstree.com/path2. """

You have told him yes, it is possible with path-based routi
ng in the ALB. Now do the following:

## Use existing Nginx and there should be 1 ec2 in each pri
vate subnet .
    - 1 bastion host in public subnet .
    - port 22 of bastion host should only accessible from y
our public ip.
    - nginx welcome page with path '/ninja1' on first ec2 s
hould display `Image-1`.
    - nginx welcome page with path '/ninja2' on second ec2
should display `Image-2`.
    - port 22 of  both the nginx servers should only be acc
essible from  bastion host.

## Create target group for each nginx server. (2 servers)
## Create Application load balancer.
    - port 80 open of nginx server should only be accessibl
e from your ALB.
    - port 80 open of ALB should only be accessible from yo
ur public IP.
    - ALB should only be accessible though your public IP n
one else.
    - create listener rule so that {YOUR-ALB-DNS-NAME}/ninj
a1 should display welcome page of first ec2.
    - create listener rule so that {YOUR-ALB-DNS-NAME}/ninj
a2 should display welcome page of second ec2.
```

```
## Push all the updated images of the webpage to S3 bucket
defined folders through any EC2 instance and maintain repo
on that server only.

Day5:
"""Client asked for S3 service and bucket to be deployed in
the US-East-1 region and segregate the folders with env nam
es with proper restrictions."""

## You need to create a IAM user and a S3 Bucket on any of
the region, Kindly follow as mentioned below :
 - Create 2 folders prod and nonprod inside that bucket.
 - Upload any different image files inside both folders.
 - Create a role for the above specific task and it should
only be access S3 full access
 - A bucket should only be accessible from your both root,
IAM user and nginx application.
 - Restrict the access of IAM user to not access prod folde
r but able to access the nonprod folder.
 - Set IAM and bucket policies in such a way that it accomp
lished the above points.

Day6:
"""Client facing issue with some policy in IAM. Earlier,  h
e assigned one policy to S3, but if he assigned the same to
CDN, then it showed an error. He called the devOps for this
anonymous issue."""

DevOps told him that it needs to be validated through a tru
st relationship in the IAM service

## suppose organization wants to get the page access quick,
so you need to implement CDN and fetch the images through C
DN instead S3, use same role in the CDN which you are using
in EC2 without any modification of policy.

## Make your own IAM user and assign minimal permissions to
yourself for this task.
```

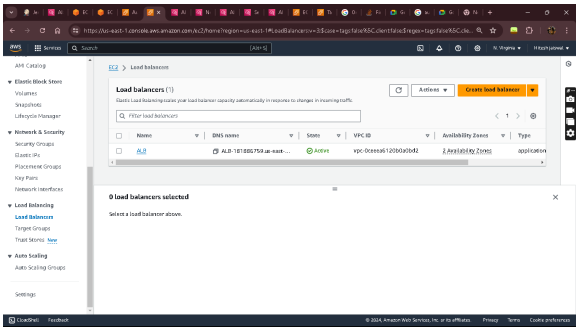
Instance with version and AMI, Template

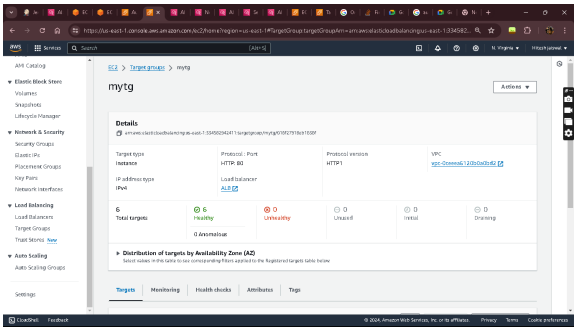






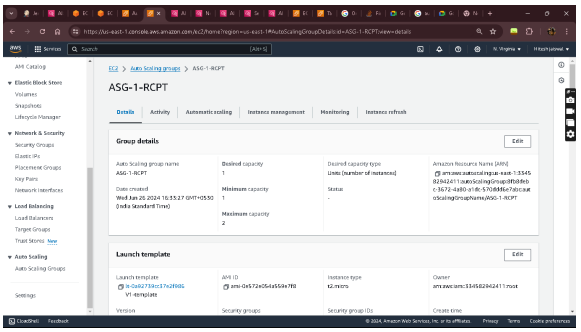
Load Balancer, Target Group

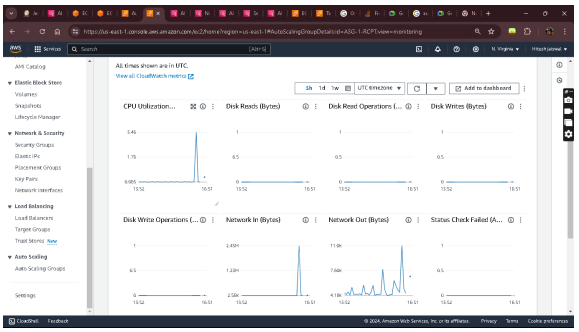


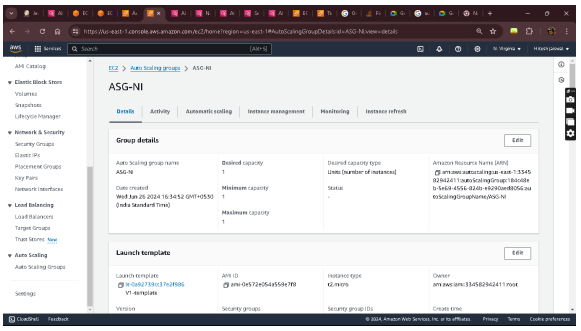


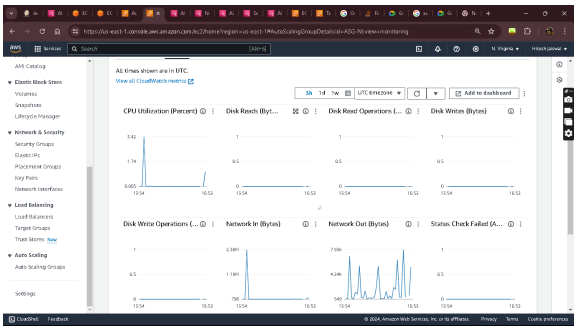
Auto Scaling Group For CPU Optimization, Network Inbound, Network Outbound , ALB Request Per Target with Monitoring and AWS CloudWatch Alarm

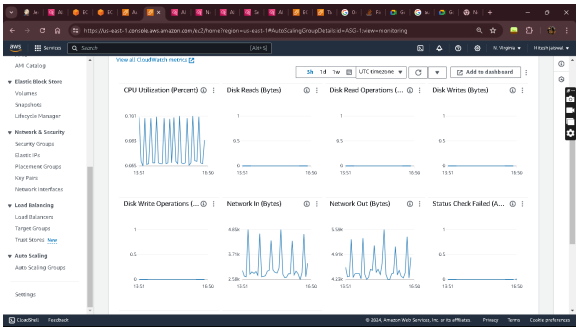
For Version-1 Template & Version 2

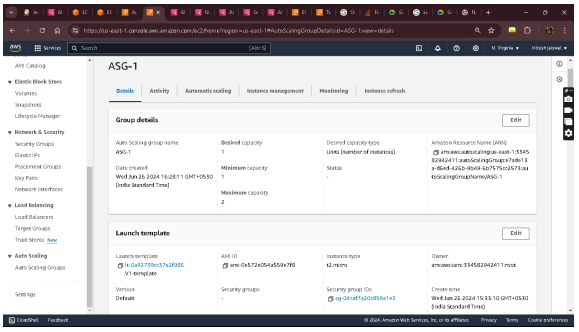


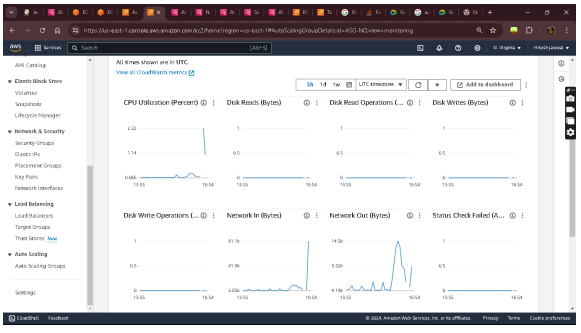


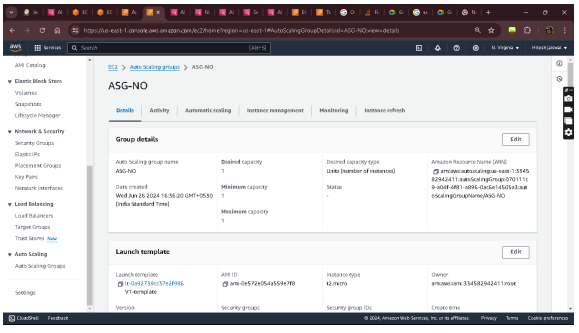


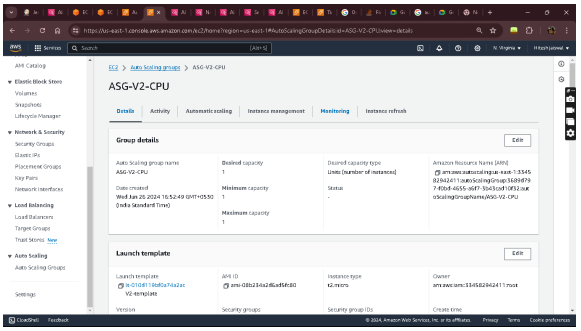


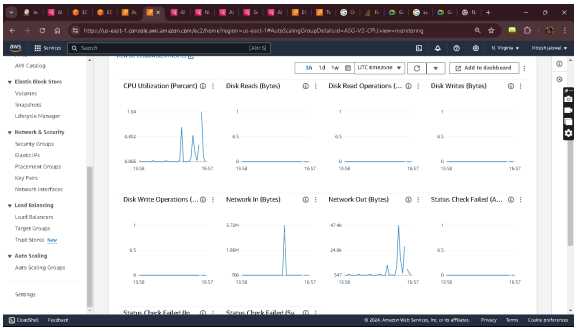




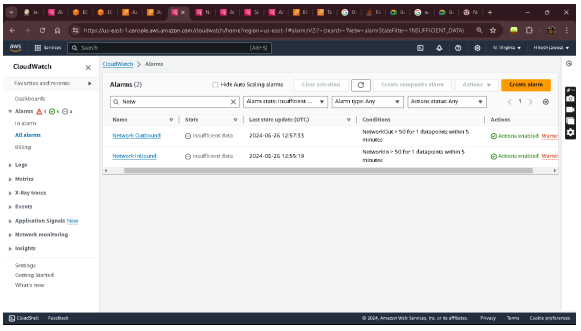


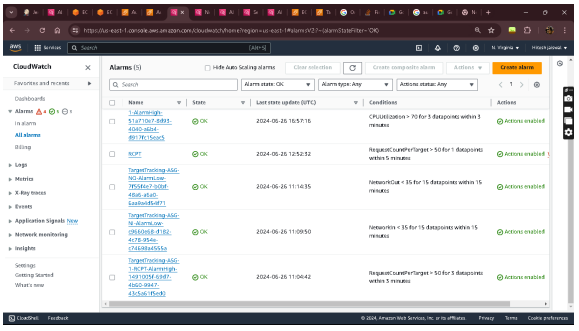


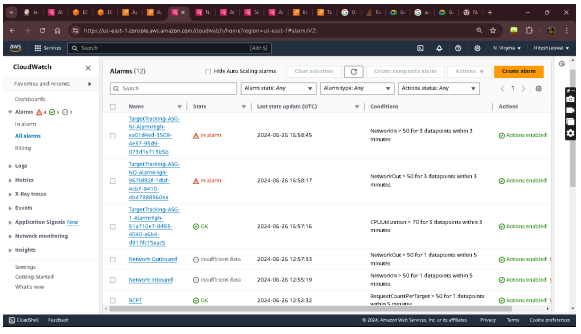


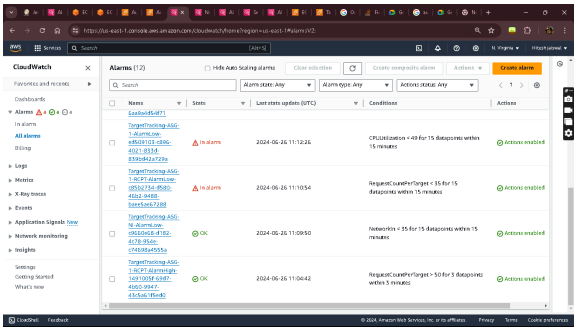


Alarm Cloud Watch

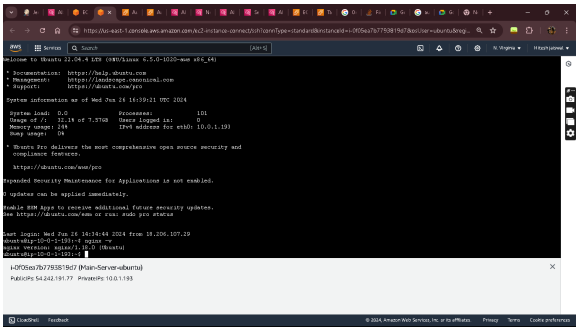


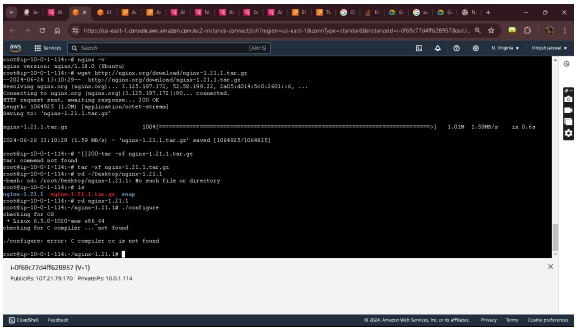




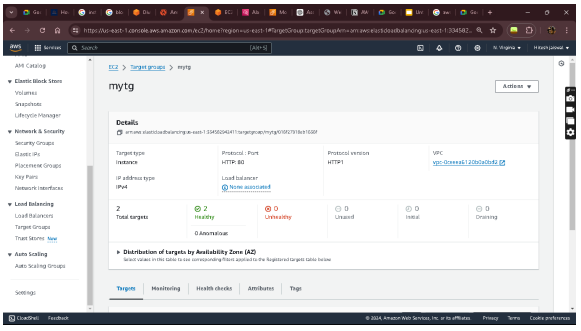


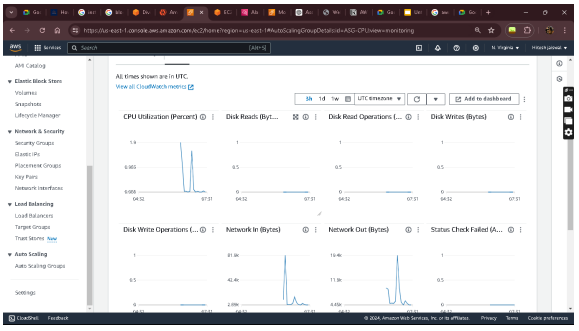
Revert Back to version 1 of Nginx 1.18.0 and Delete the 2nd Version of Nginx 1.21.1 we downloaded using wget Now we deleted v2 and just launch the instance through AMI V1

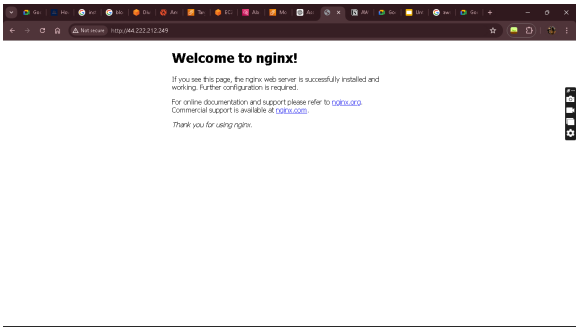


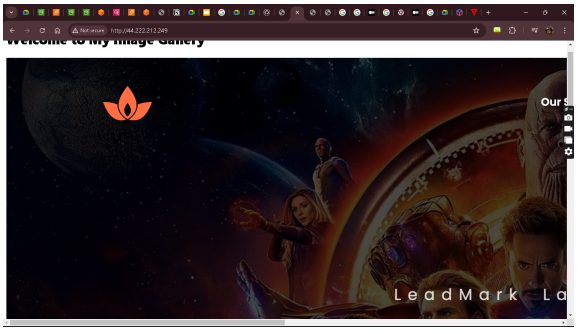


Healthy

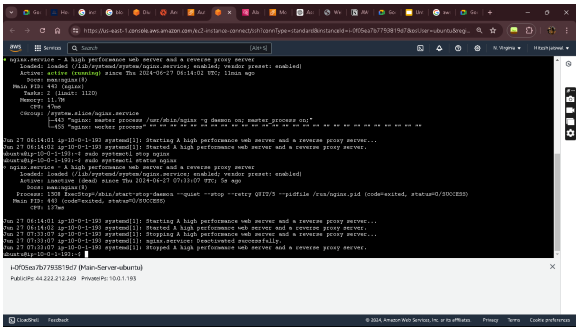


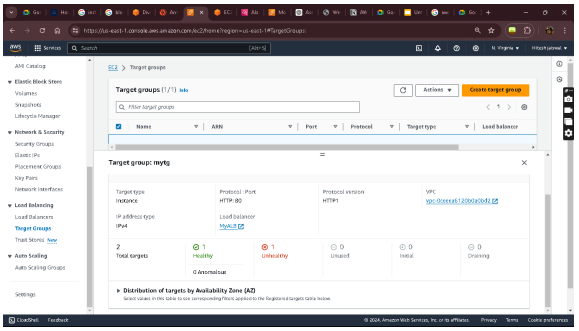




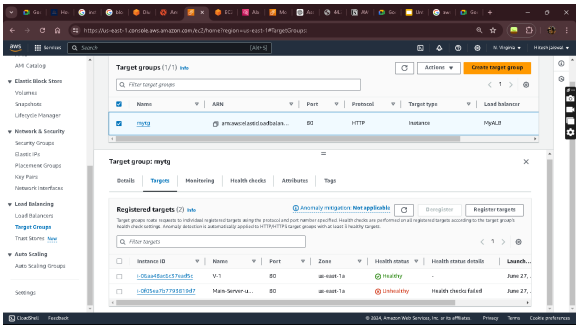


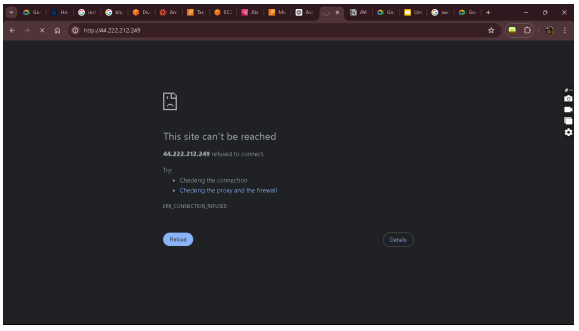
To check the monitory I made server unhealthy by stopping the nginx

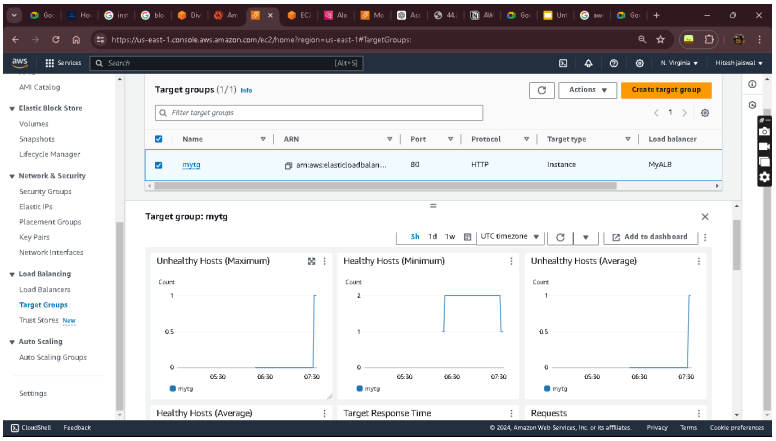




I stopped Nginx nginx on my main server It is showing Unhealthy to that Not the V1







Make IAM user and give it s3 permission and do ssh through that so we don’t need access and secret key we can check that while using curl website of that instance . Now S3 made

Images are Displayed using nginx the images are in S3 bucket

In Nginx html I have given image source of S3 bucket

I have also made customized bucket policy