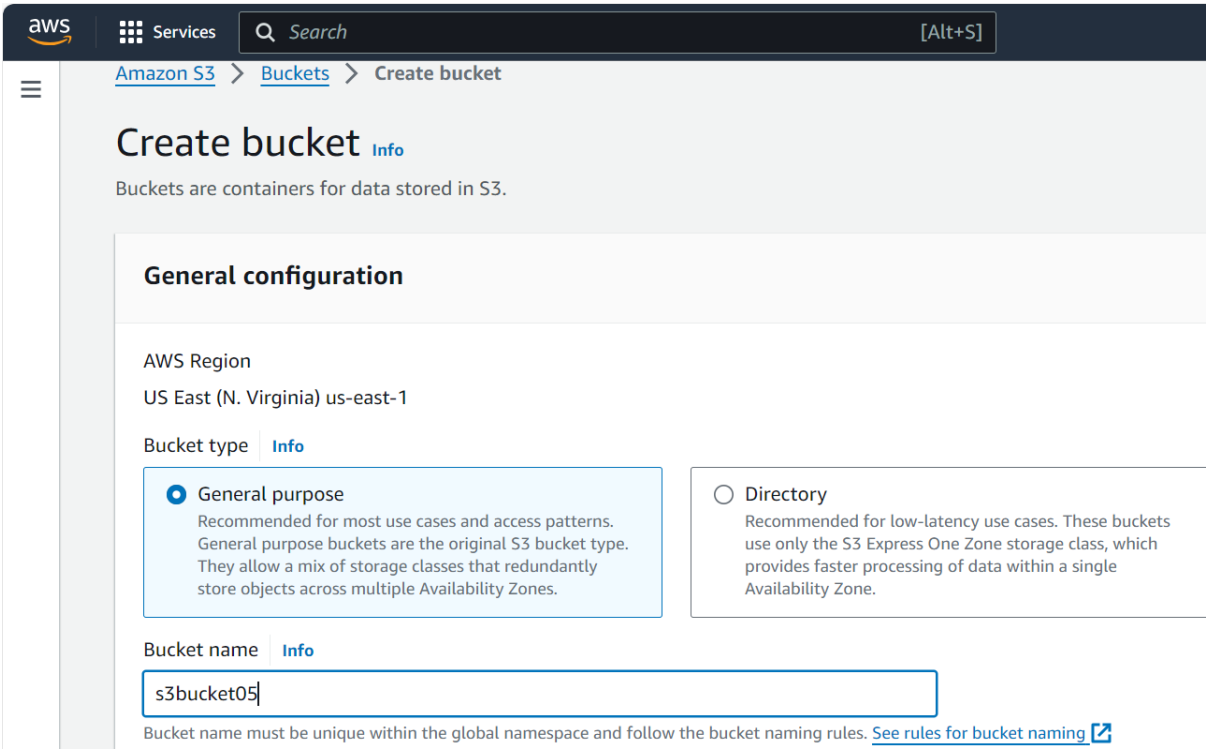
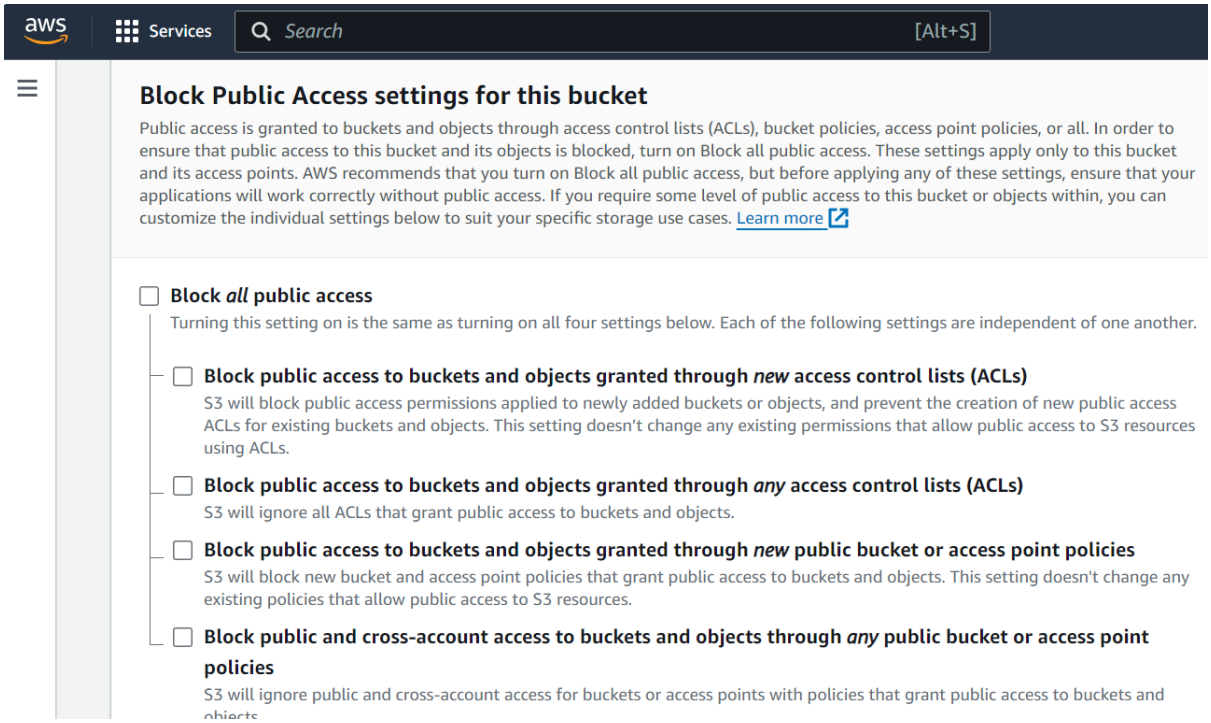
Deploy a static website on AWS S3 bucket using GitHub action.

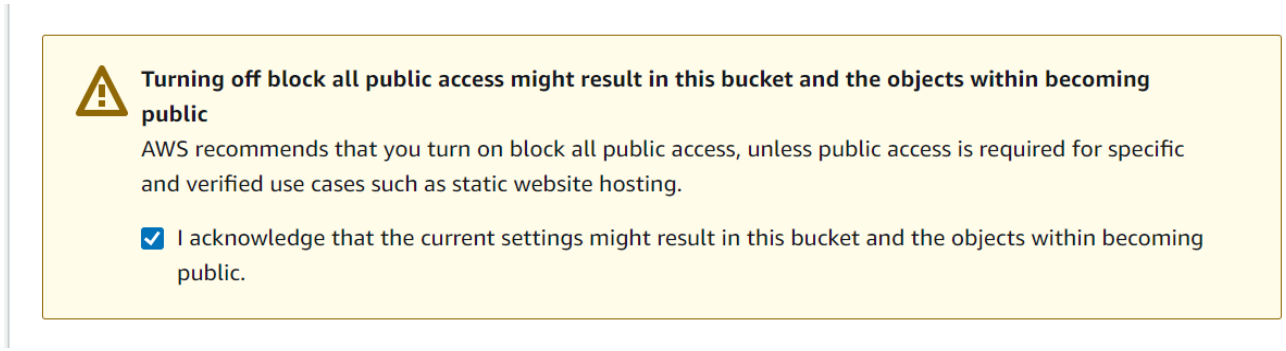
Step 1: Create a bucket



Uncheck the check box for block all public access



Click on the acknowledge checkbox

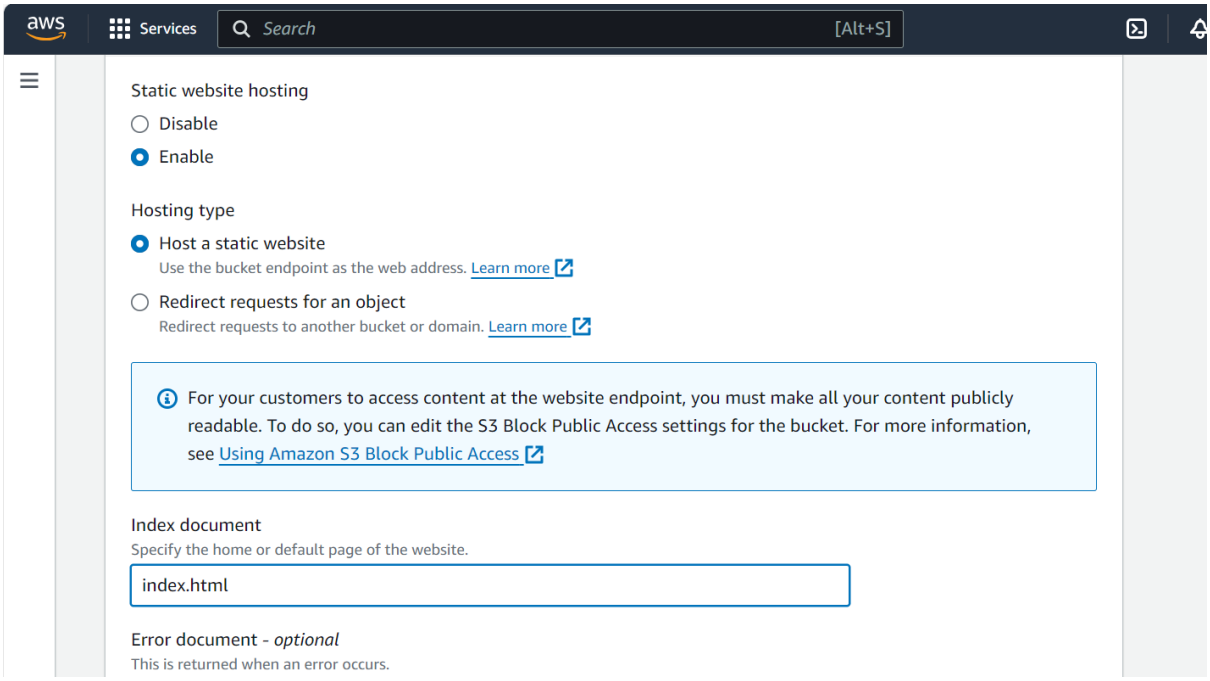


Step 2: Enable static website hosting

1. In the Buckets list, choose the name of the bucket that you want to enable static website hosting for.

2. Choose Properties.

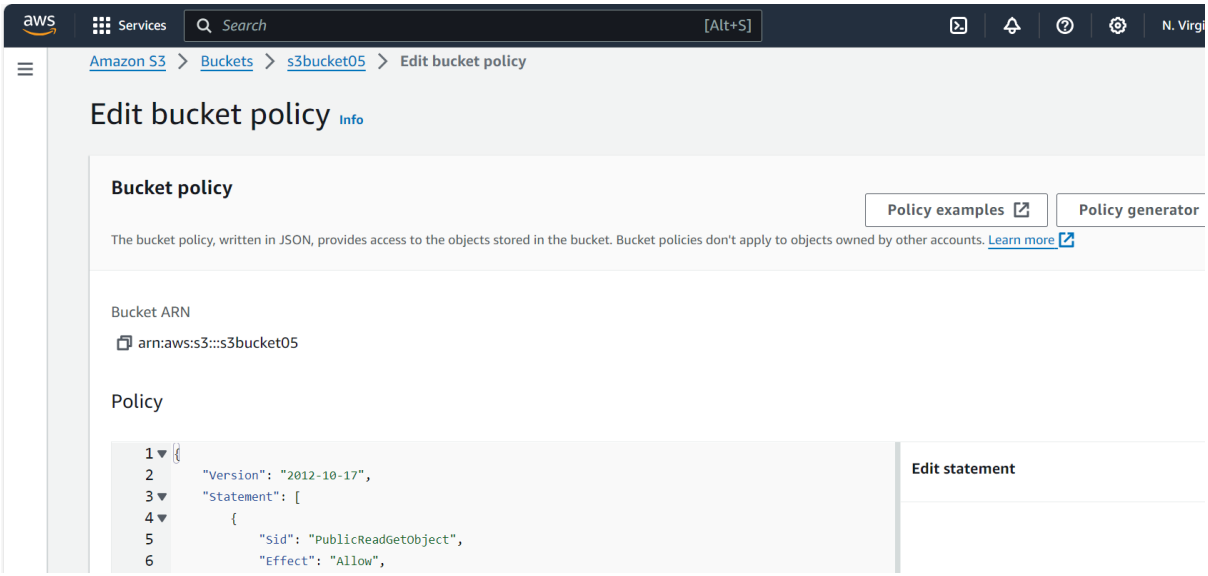
3. **Under Static website hosting, choose Edit.**



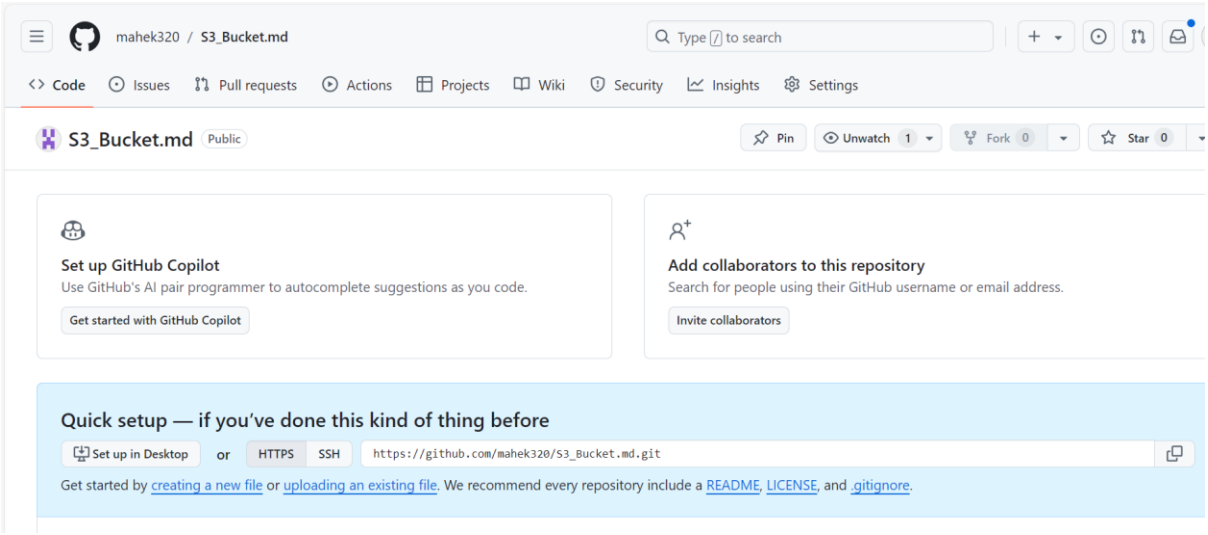
1. **Under Buckets, choose the name of your bucket.**

2. **Choose Permissions.**

3. **Under Bucket Policy, choose Edit.**

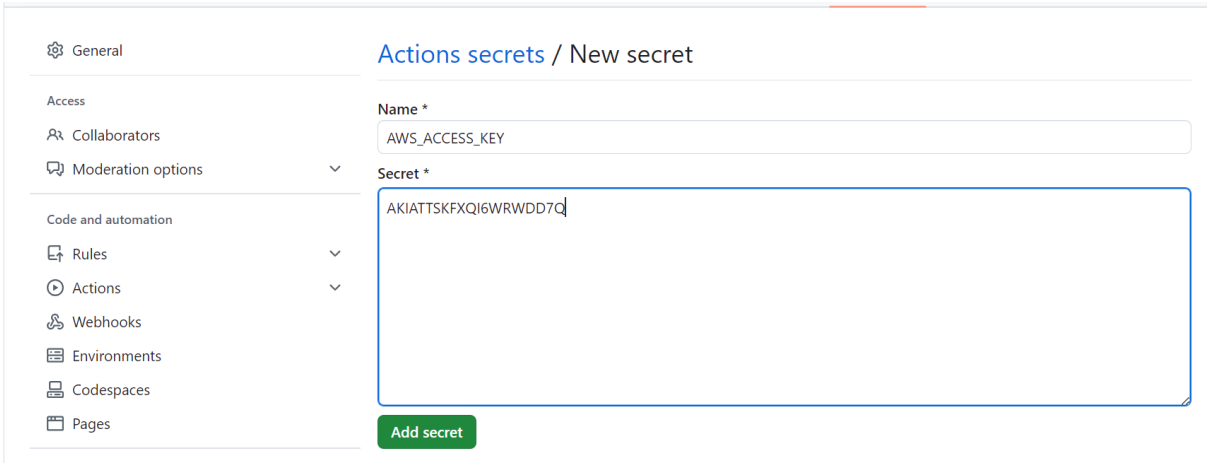


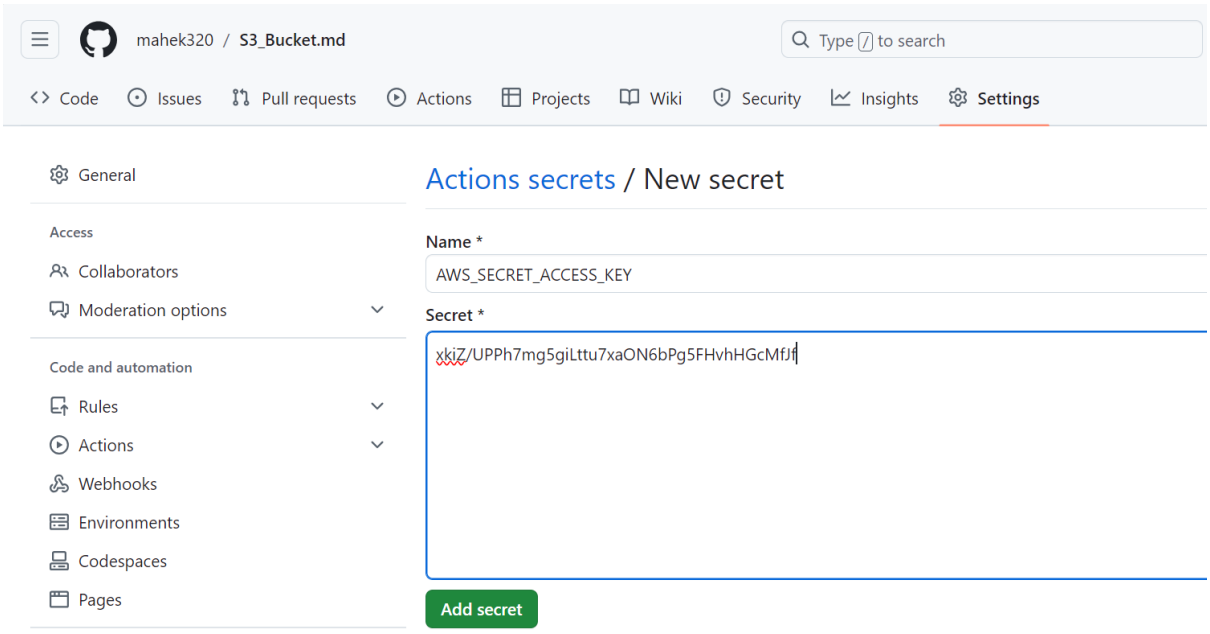
STEP 3: Create a new repository in your GitHub account

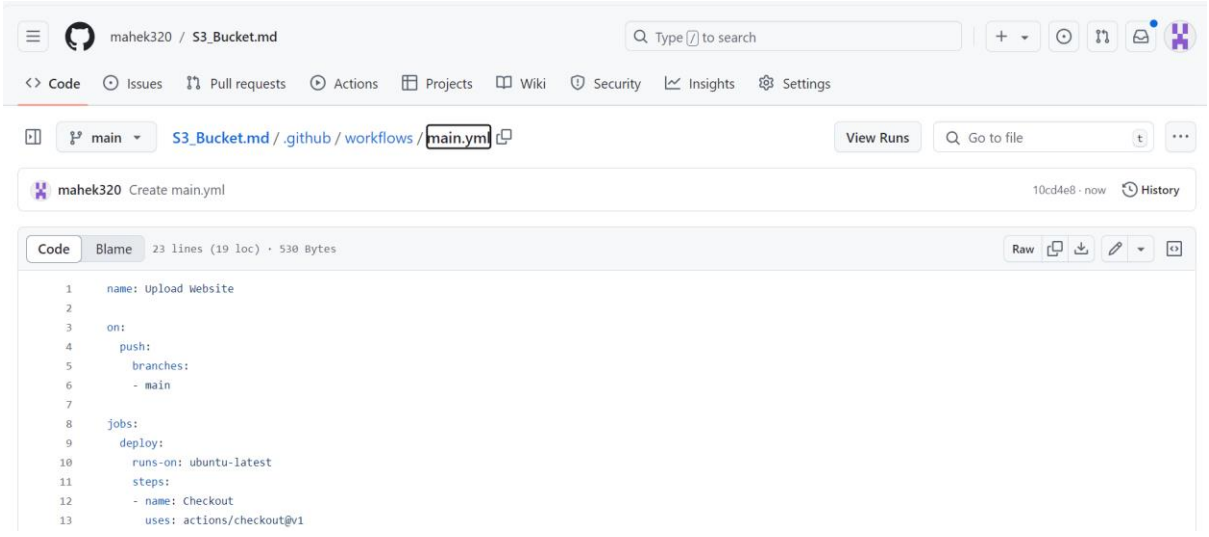


Go to settings >> secrets and variables >> actions >> add new repository secret

Here add access key and secret key


General
Access
Collaborators
Moderation options
Code and automation
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Actions secrets / New secret
Name *
AWS_ACCESS_KEY
Secret *
AKIATTSKFXQI6WRWDD7Q
Add secret

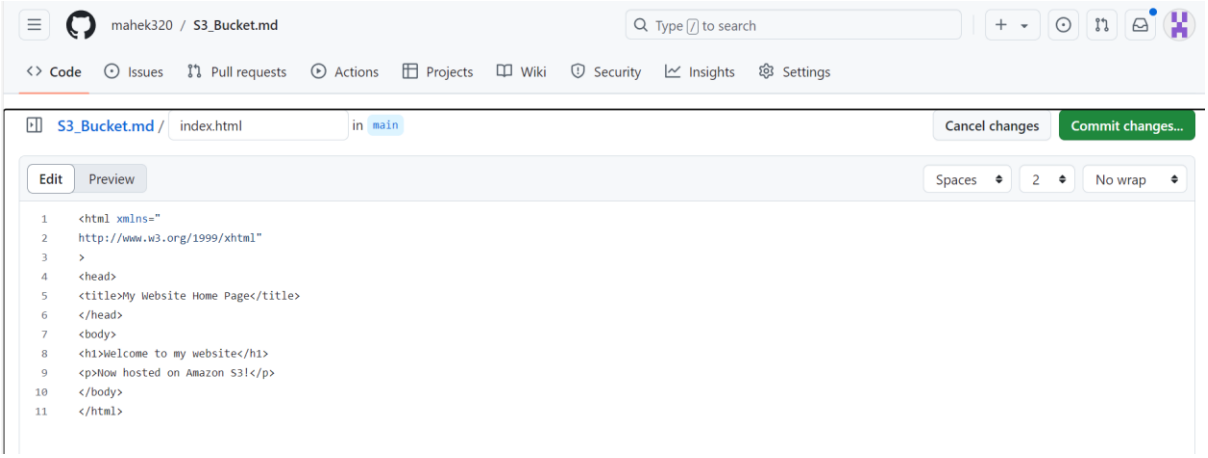

mahek320 / S3_Bucket.md
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Actions secrets / New secret
Name *
AWS_SECRET_ACCESS_KEY
Secret *
xkiZ/UPPh7mg5giLttu7xaON6bPg5FHvhHGcMfJf
Add secret


mahek320 / S3_Bucket.md
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
S3_Bucket.md / .github / workflows / main.yml
View Runs
Go to file
mahek320 Create main.yml
10cd4e8 · now
History
Code
Blame
23 lines (19 loc) · 530 Bytes
Raw
name: Upload Website
on:
push:
branches:
- main
jobs:
deploy:
runs-on: ubuntu-latest
steps:
- name: Checkout
uses: actions/checkout@v1

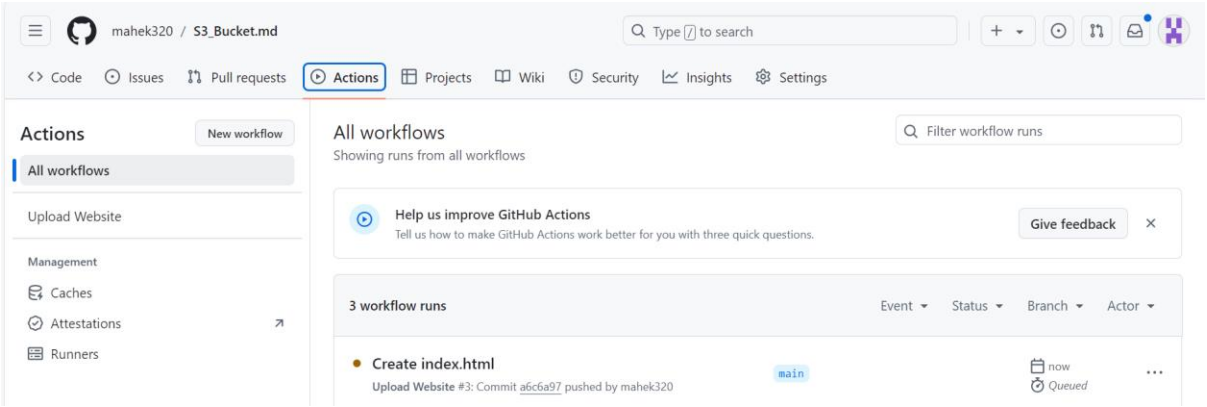
Change the bucket name and region and remove public

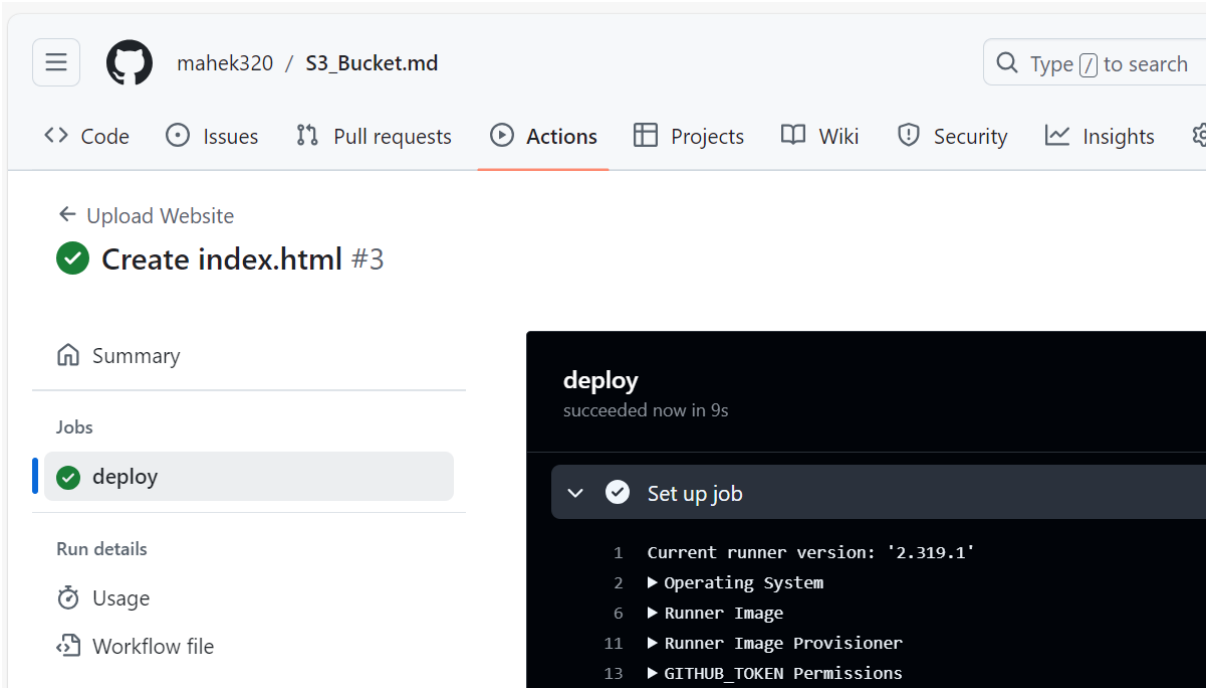
```
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: us-east-1

      - name: Deploy static site to S3 bucket
        run: aws s3 sync . s3://s3bucket05 --delete
```

Create an index file in the same repository



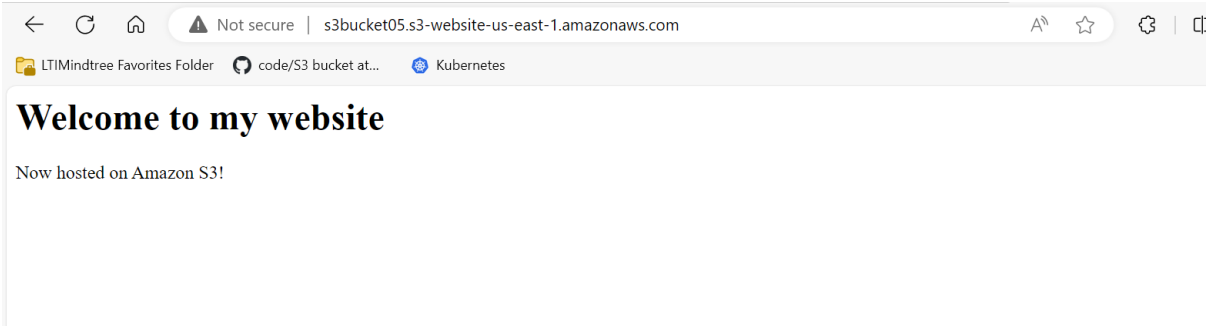
Click on actions in the git repository





Under Static website hosting, click on the Endpoint URL.

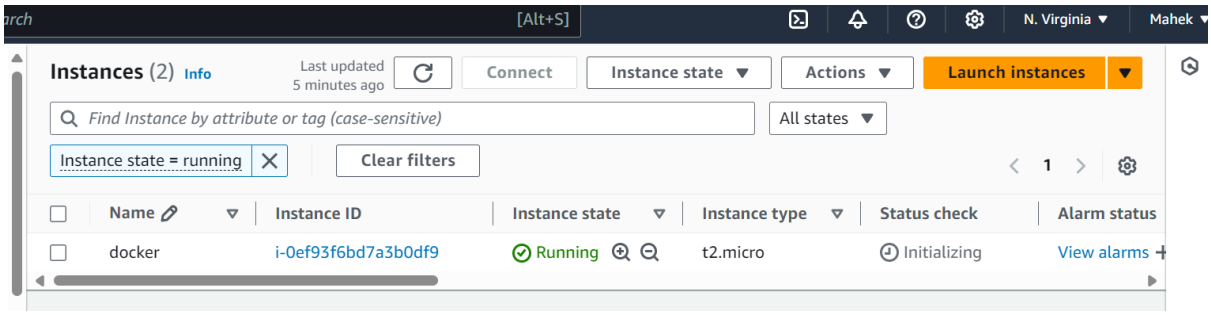
Copy paste the endpoint url



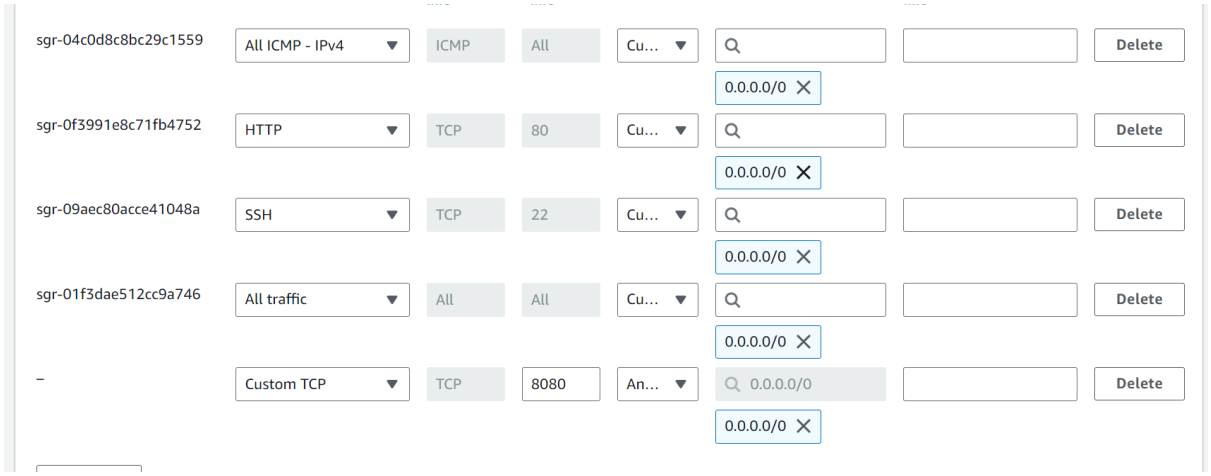
Easy 1

Pull the Ubuntu image from Docker hub and launch a web application in the container on port no. 8080 and this application should be reachable globally.

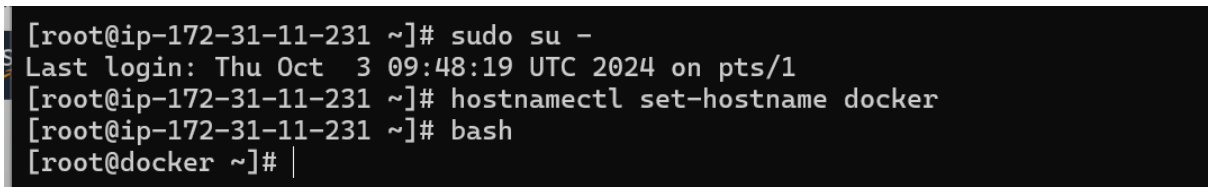
Launch an EC2 instance for docker and connect it on the terminal



Add port 8080 in the security group of the instance



Set the hostname



Run the docker commands on the terminal to install docker

```
[root@docker ~]# sudo yum install -y yum-utils
Last metadata expiration check: 0:33:37 ago on Thu Oct  3 09:16:34 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@docker ~]#
```

```
[root@docker ~]# sudo yum install docker -y
Last metadata expiration check: 0:34:36 ago on Thu Oct  3 09:16:34 2024.
Dependencies resolved.
================================================================================
 Package                         Architecture       Version
```

We will now use the start docker command to start docker

```
[root@docker ~]# sudo systemctl start docker
[root@docker ~]# sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@docker ~]#
```

We will pull the ubuntu image

```
[root@docker ~]# docker run -it --name my-container -p 8080:80 ubuntu:latest /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
eda6120e237e: Pull complete
Digest: sha256:b359f1067efa76f37863778f7b6d0e8d911e3ee8efa807ad01fbf5dc1ef9006b
Status: Downloaded newer image for ubuntu:latest
```

We will install all the packages and update them

```
root@b688afc49cba:/# apt update -y
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [446 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
```

```
root@b688afc49cba:/# apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  adduser apache2-bin apache2-data apache2-utils ca-certificates krb5-loca
  libaprutil1-ldap libaprutil1t64 libbrotli1 libcurl4t64 libexpat1 libgdbm
```

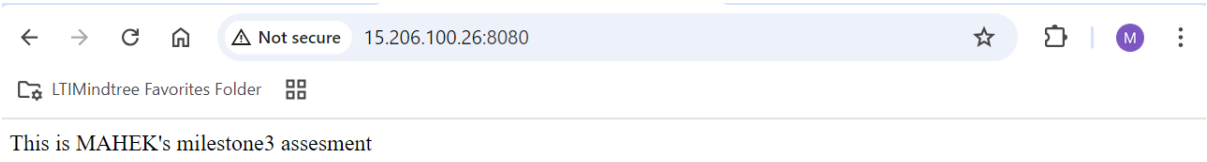
We will create a html file

```
root@b688afc49cba:/# cd /var/www/html
root@b688afc49cba:/var/www/html# echo "This Mahek's Milestone3 Assessment" > index.html
```

We will start the start the apache

```
root@b688afc49cba:/var/www/html# service apache start
apache: unrecognized service
root@b688afc49cba:/var/www/html# [root@docker ~]# docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS         PORTS                                   NAME
S
b688afc49cba   ubuntu:latest   "/bin/bash"   3 minutes ago   Up 3 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-c
ontainer
[root@docker ~]#
```

The image has been successfully hosted on 8080



Medium

Create a custom VPC and create two subnets like public and private in that subnet you need to deploy one web server and another is database server using IAC tool terraform

Created an ec2 instance and connected to terminal

```
[ec2-user@ip-172-31-34-105 ~]$ sudo su -
[root@ip-172-31-34-105 ~]# hostnamectl set-hostname terraform
[root@ip-172-31-34-105 ~]# bash
```

```
[root@terraform ~]# sudo yum install -y yum-utils shadow-utils
Last metadata expiration check: 0:00:27 ago on Thu Oct  3 10:44:09 2024.
Package dnf-utils-4.3.0-13.amzn2023.0.4.noarch is already installed.
Package shadow-utils-2:4.9-12.amzn2023.0.4.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
```

```
[root@terraform ~]# sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
[root@terraform ~]# sudo yum -y install terraform
Hashicorp Stable - x86_64                                   12 MB/s | 1.4 MB     00:00
Last metadata expiration check: 0:00:01 ago on Thu Oct  3 10:44:48 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                 Repository                Size
================================================================================
Installing:
```

```
[root@terraform ~]# terraform -v
Terraform v1.9.7
on linux_amd64
[root@terraform ~]# curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 62.9M  100 62.9M    0     0   141M      0 --:--:-- --:--:-- --:--:--  142M
[root@terraform ~]# unzip awscliv2.zip
Archive:  awscliv2.zip
   creating: aws/
   creating: aws/dist/
  inflating: aws/README.md
  inflating: aws/THIRD_PARTY_LICENSES
  inflating: aws/install
```

We have configured aws and install aws cli

```
[root@terraform ~]# sudo ./aws/install
You can now run: /usr/local/bin/aws --version
[root@terraform ~]# vim provider.tf
[root@terraform ~]# rm -rf provider.tf
[root@terraform ~]# aws configure
AWS Access Key ID [None]: AKIATTSKFXQI6WRWDD7Q
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]: table
[root@terraform ~]# vim vpc.tf
[root@terraform ~]# vim vpc.tf
[root@terraform ~]# terraform init
Initializing the backend...
```

We created a vim vpc.tf file

```
[root@terraform ~]# vim vpc.tf
[root@terraform ~]# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.69.0...
- Installed hashicorp/aws v5.69.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[root@terraform ~]# terraform fmt
vpc.tf
```

```
resource "aws_vpc" "terraform-vpc" {

  cidr_block = "10.0.0.0/16"

  tags = {

    Name = "terraform-vpc"

  }

}

resource "aws_subnet" "public-subnet" {

  vpc_id = aws_vpc.terraform-vpc.id

  cidr_block = "10.0.0.0/24"

  availability_zone = "ap-south-1a"

  tags = {

    Name = "public-subnet"

  }
```

We have then validated the terraform

```
[root@terraform ~]# vim vpc.tf
[root@terraform ~]# terraform validate
Success! The configuration is valid.
```

We have now used the terraform plan command

```
[root@terraform ~]# terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated
with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_eip.nat will be created
  + resource "aws_eip" "nat" {
      + allocation_id        = (known after apply)
      + arn                  = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = "vpc"
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
```

We have applied all the changes

```
[root@terraform ~]# terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicate
with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_eip.nat will be created
  + resource "aws_eip" "nat" {
      + allocation_id        = (known after apply)
      + arn                  = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = "vpc"
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
      + private_dns          = (known after apply)
```

```
aws_vpc.terraform-vpc: Creating...
aws_eip.nat: Creating...
aws_eip.nat: Creation complete after 0s [id=eipalloc-00df47df7371c7899]
aws_vpc.terraform-vpc: Creation complete after 1s [id=vpc-05a11d2e14aecbe93]
aws_subnet.public-subnet: Creating...
aws_security_group.my-vpc-sg: Creating...
aws_subnet.private-subnet: Creating...
aws_internet_gateway.my-internet-gateway: Creating...
aws_internet_gateway.my-internet-gateway: Creation complete after 0s [id=igw-03b0537303be50d06]
aws_route_table.public-rout: Creating...
aws_subnet.public-subnet: Creation complete after 0s [id=subnet-0024267dac60c4485]
aws_nat_gateway.NAT-gw: Creating...
aws_subnet.private-subnet: Creation complete after 0s [id=subnet-09af744e49af96753]
aws_route_table.public-rout: Creation complete after 1s [id=rtb-09c83ea61c429778b]
aws_route_table_association.association: Creating...
aws_route_table_association.association: Creation complete after 0s [id=rtbassoc-0bd0bca9c45972e24]
aws_security_group.my-vpc-sg: Creation complete after 2s [id=sg-0597faadf82a19975]
aws_instance.my-vpc-private-instance: Creating...
aws_instance.my-vpc-instance: Creating...
aws_nat_gateway.NAT-gw: Still creating... [10s elapsed]
aws_instance.my-vpc-private-instance: Still creating... [10s elapsed]
aws_instance.my-vpc-instance: Still creating... [10s elapsed]
aws_instance.my-vpc-instance: Creation complete after 12s [id=i-06a4c142ae90ff962]
aws_instance.my-vpc-private-instance: Creation complete after 12s [id=i-059a2a84cd7a46bb8]
aws_nat_gateway.NAT-gw: Still creating... [20s elapsed]
```

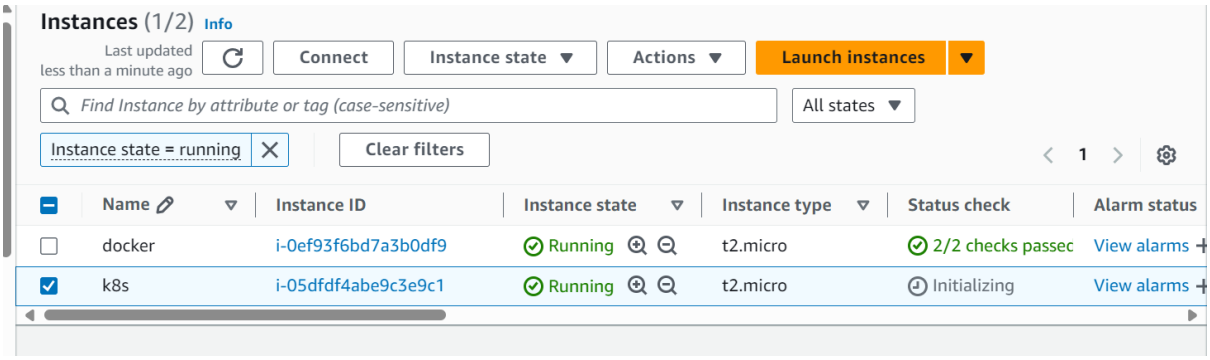
As we can see that all the changes have been successfully implemented

```
aws_route_table.public-rout: Creation complete after 1s [id=rtb-09c83ea61c429778b]
aws_route_table_association.association: Creating...
aws_route_table_association.association: Creation complete after 0s [id=rtbassoc-0bd0bca9c45972e24]
aws_security_group.my-vpc-sg: Creation complete after 2s [id=sg-0597faadf82a19975]
aws_instance.my-vpc-private-instance: Creating...
aws_instance.my-vpc-instance: Creating...
aws_nat_gateway.NAT-gw: Still creating... [10s elapsed]
aws_instance.my-vpc-private-instance: Still creating... [10s elapsed]
aws_instance.my-vpc-instance: Still creating... [10s elapsed]
aws_instance.my-vpc-instance: Creation complete after 12s [id=i-06a4c142ae90ff962]
aws_instance.my-vpc-private-instance: Creation complete after 12s [id=i-059a2a84cd7a46bb8]
aws_nat_gateway.NAT-gw: Still creating... [20s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [30s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [40s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [50s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [1m0s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [1m10s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [1m20s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [1m30s elapsed]
aws_nat_gateway.NAT-gw: Still creating... [1m40s elapsed]
aws_nat_gateway.NAT-gw: Creation complete after 1m44s [id=nat-0fe94e41faa3bd82a]
aws_route_table.private-rout: Creating...
aws_route_table.private-rout: Creation complete after 1s [id=rtb-0b8786017d9c33642]
aws_route_table_association.association-1: Creating...
aws_route_table_association.association-1: Creation complete after 0s [id=rtbassoc-0c6cd9a0613794114]

Apply complete! Resources: 13 added, 0 changed, 0 destroyed.
[root@terraform ~]#
```

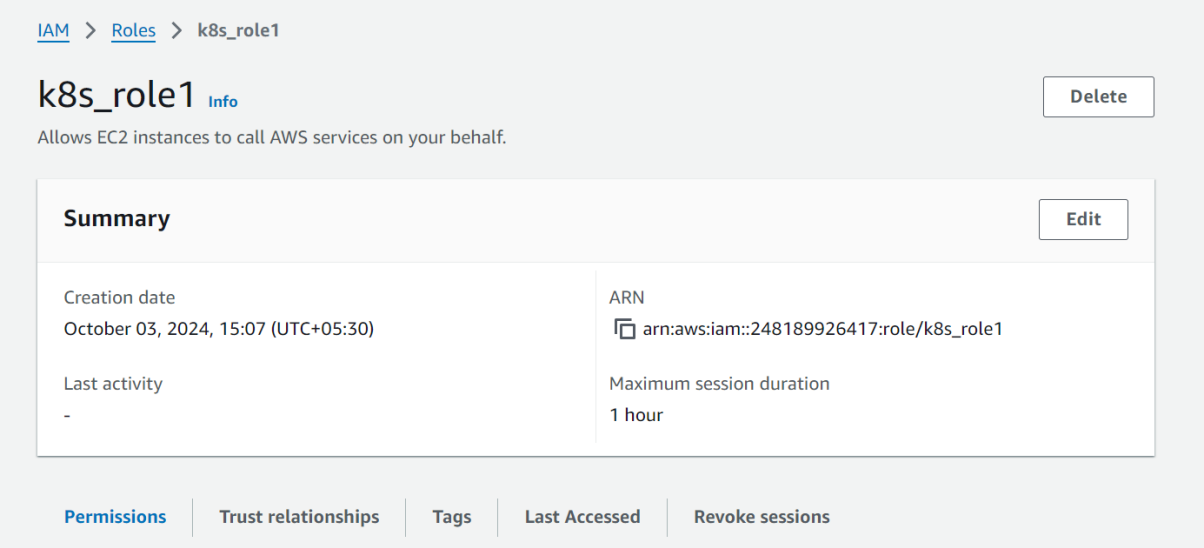
Medium 1

Deploy a web application in the Kubernetes pod. And create a replica set. In any case load is going to increase on your replica set. increase the number of replicas of the pods.

Create an instance



Create a role



Connect to the terminal

```
ubuntu@ip-172-31-8-172:~$ sudo su -
root@ip-172-31-8-172:~# hostnamectl set-hostname cluster
root@ip-172-31-8-172:~# bash
root@cluster:~#
```

Unzip the necessary files for installation

```
root@cluster:~# apt install unzip -y
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Install aws

```
root@cluster:~# sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```

Configure aws

```
root@cluster:~# aws configure
AWS Access Key ID [None]: AKIATTSKFXQI6WRWDD7Q
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: table
```

Install EKS tool

```
root@cluster:~# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -
s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
0.191.0
```

Install Kubectl

```
root@cluster:~# curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl https://storage.googleapis.com
/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
```

Create a cluster

```
root@cluster:~# eksctl create cluster --name my-newcluster --region us-east-1 --version 1.29 --vpc-public-subnets subnet
-03a62d6d9c8eecbd2,subnet-018fe897927b207d1 --without-nodegroup
2024-10-03 09:41:23 [ℹ]  eksctl version 0.191.0
2024-10-03 09:41:23 [ℹ]  using region us-east-1
2024-10-03 09:41:24 [✔]  using existing VPC (vpc-0d7f2d060395c3cd4) and subnets (private:map[] public:map[us-east-1a:{su
bnet-03a62d6d9c8eecbd2 us-east-1a 172.31.0.0/20 0 } us-east-1b:{subnet-018fe897927b207d1 us-east-1b 172.31.80.0/20 0 }])
2024-10-03 09:41:24 [!]  custom VPC/subnets will be used; if resulting cluster doesn't function as expected, make sure t
```

Creating node group

```
root@cluster:~/.ssh# eksctl create nodegroup   --cluster my-cluster   --region us-east-1   --name my-node-group   --node
-ami-family Ubuntu2004   --node-type t2.small   --subnet-ids subnet-03a62d6d9c8eecbd2,subnet-018fe897927b207d1   --nodes
 3   --nodes-min 2   --nodes-max 4   --ssh-access   --ssh-public-key /root/.ssh/id_rsa.pub
2024-10-03 10:22:14 [ℹ]  will use version 1.29 for new nodegroup(s) based on control plane version
2024-10-03 10:22:14 [ℹ]  nodegroup "my-node-group" will use "ami-0edeff0632655c0de" [Ubuntu2004/1.29]
2024-10-03 10:22:14 [ℹ]  using SSH public key "/root/.ssh/id_rsa.pub" as "eksctl-my-cluster-nodegroup-my-node-group-84:2
0:11:43:04:8e:cb:5f:17:f1:4b:bd:65:c7:80:df"
2024-10-03 10:22:15 [ℹ]  1 nodegroup (my-node-group) was included (based on the include/exclude rules)
2024-10-03 10:22:15 [ℹ]  will create a CloudFormation stack for each of 1 managed nodegroups in cluster "my-cluster"
2024-10-03 10:22:15 [!]  "aws-node" was not found
2024-10-03 10:22:15 [ℹ]
2 sequential tasks: { fix cluster compatibility, 1 task: { 1 task: { create managed nodegroup "my-node-group" } }
}
```

Create a vim replicaset.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 4
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: apache-app
        image: nginx
~
~
```

Here we see that the pods are created

```
root@cluster:~# vim replicaset.yml
root@cluster:~# kubectl apply -f replicaset.yml
replicaset.apps/frontend created
root@cluster:~# kubectl get pods
NAME             READY   STATUS              RESTARTS   AGE
frontend-d5zkk   0/1     ContainerCreating   0          7s
frontend-l7nnx   1/1     Running             0          7s
frontend-lnf4t   0/1     ContainerCreating   0          7s
frontend-m2z4r   0/1     ContainerCreating   0          7s
root@cluster:~#
```

We can increase or decrease the number of replicaset by the below command automatically

```
root@cluster:~# kubectl autoscale rs frontend --max=10 --min=3
horizontalpodautoscaler.autoscaling/frontend autoscaled
```

Inorder to increase it manually

```
root@cluster:~# kubectl get rs
NAME       DESIRED   CURRENT   READY   AGE
frontend   9         9         9       2m21s
root@cluster:~#
```