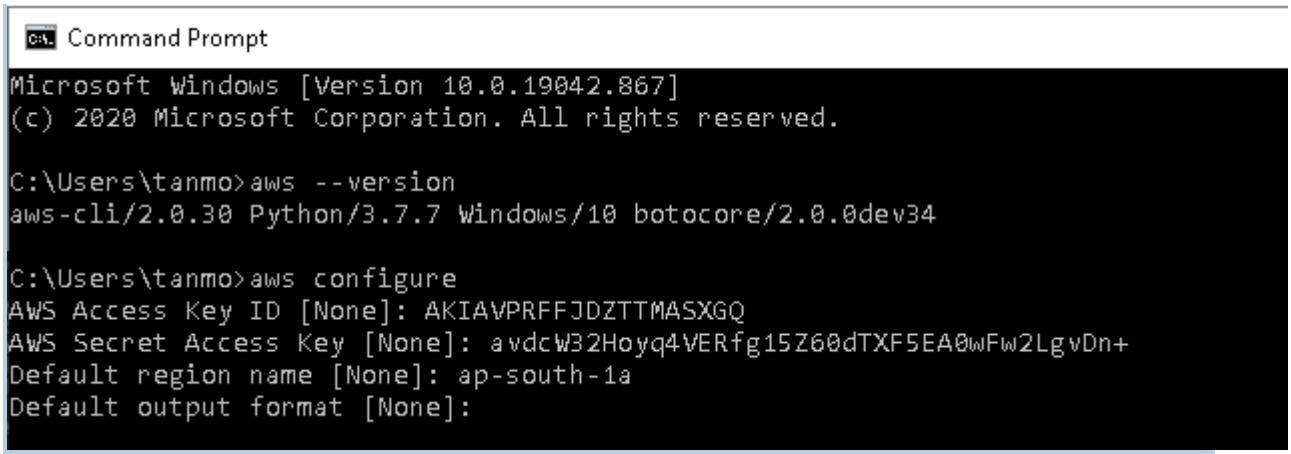
ARTH Task 6 : Creating a high availability architecture with AWS CLI

First we need to install and login to the AWS cli console using the **Access ID** and **Secret ID** provided by **IAM** user

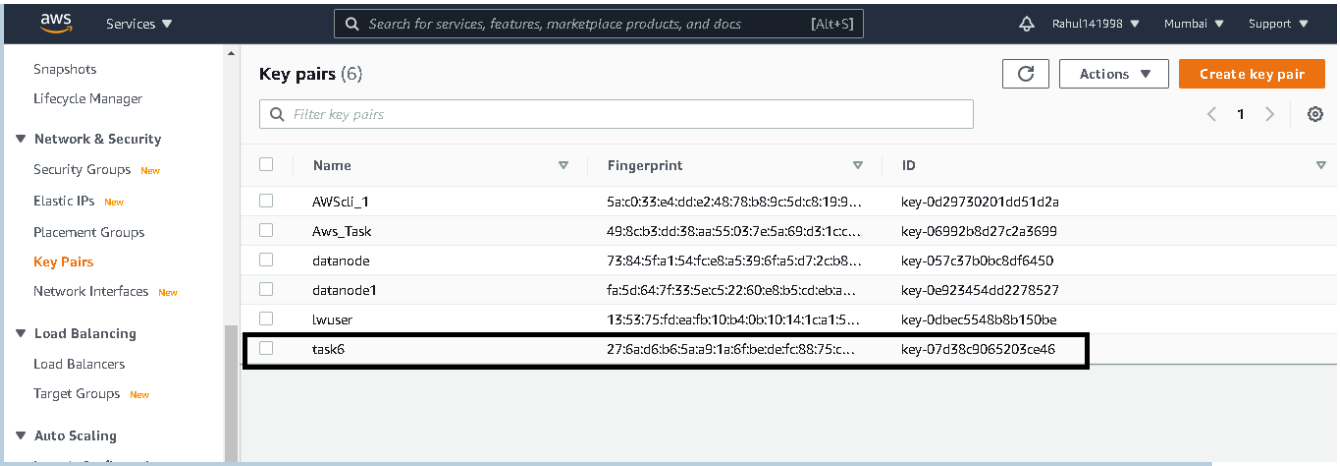
To check AWS cli Version : **aws — version**

To login to IAM user using AWS cli : **aws configure**



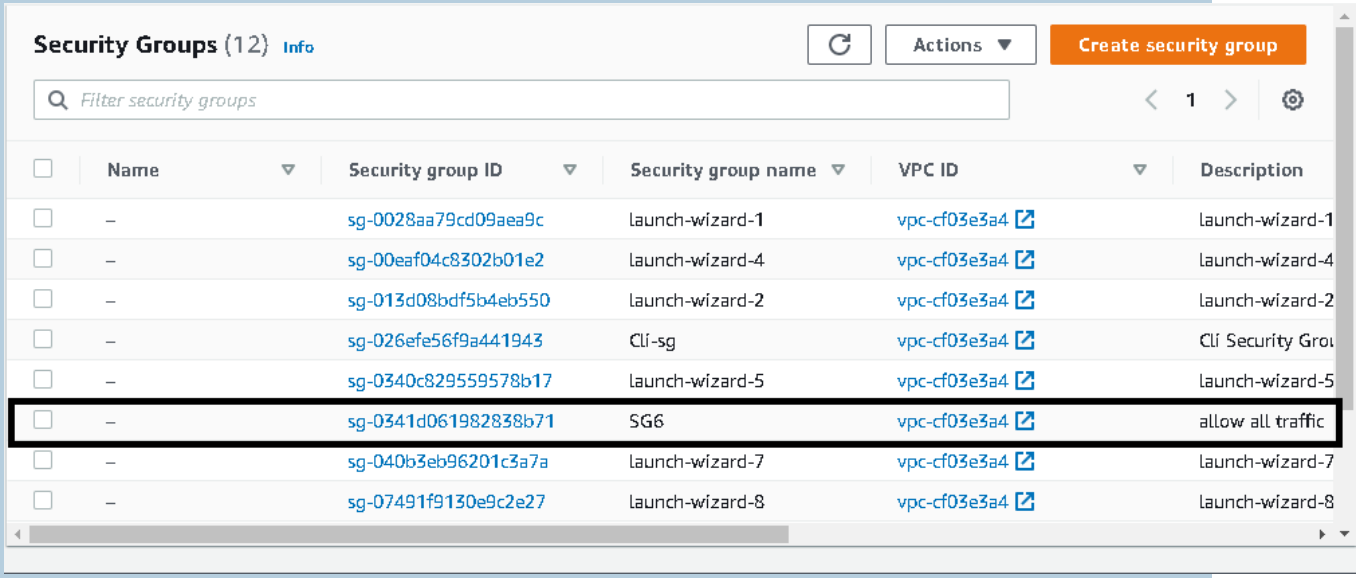
Now we need a key for EC2 instance Authentication

aws ec2 create-key-pair — keyname task 6



Now we need to create a security group : **aws ec2 create-security-group — description “Allow all traffic” — group-name SG6**

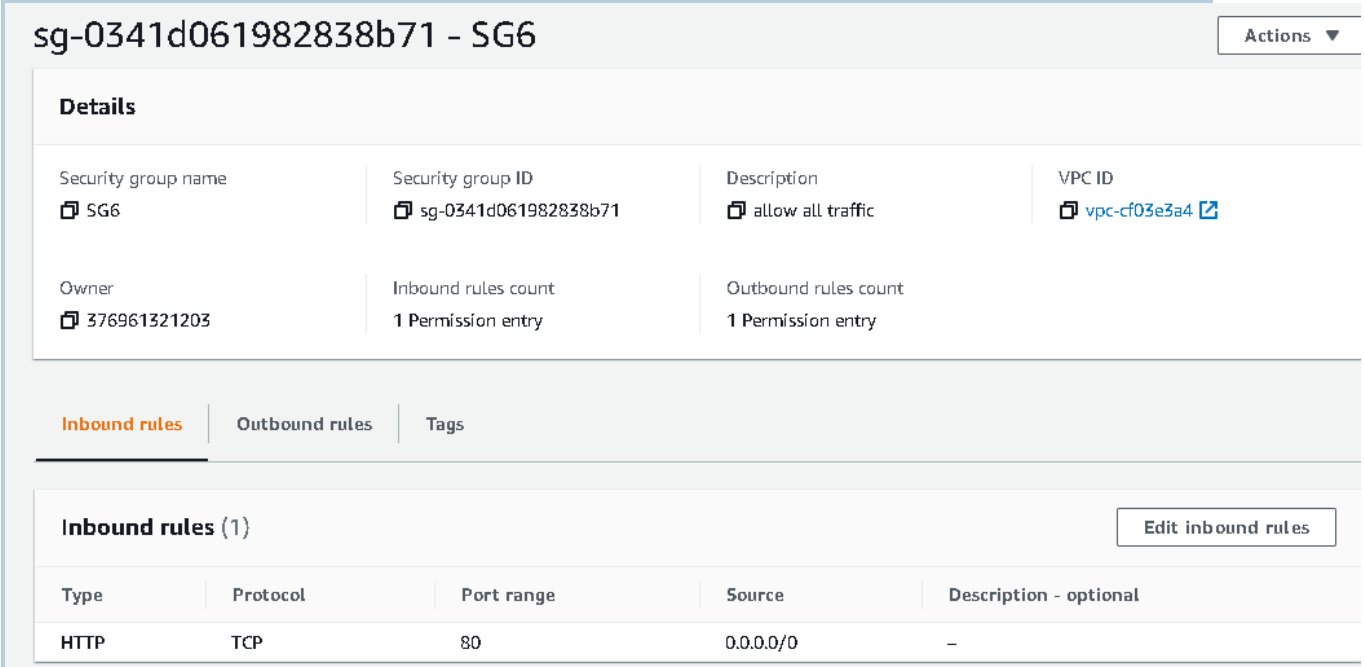
```
C:\Users\tanmo>aws ec2 create-security-group --description "allow all traffic" --group-
{
    "GroupId": "sg-0341d061982838b71"
}
```



Also we can verify our security group using command : **aws ec2 authorize-security-group — ingress — group-id**

[security-group id] — protocol [protocolname] — port [Number] — cidr 0.0.0.0/0

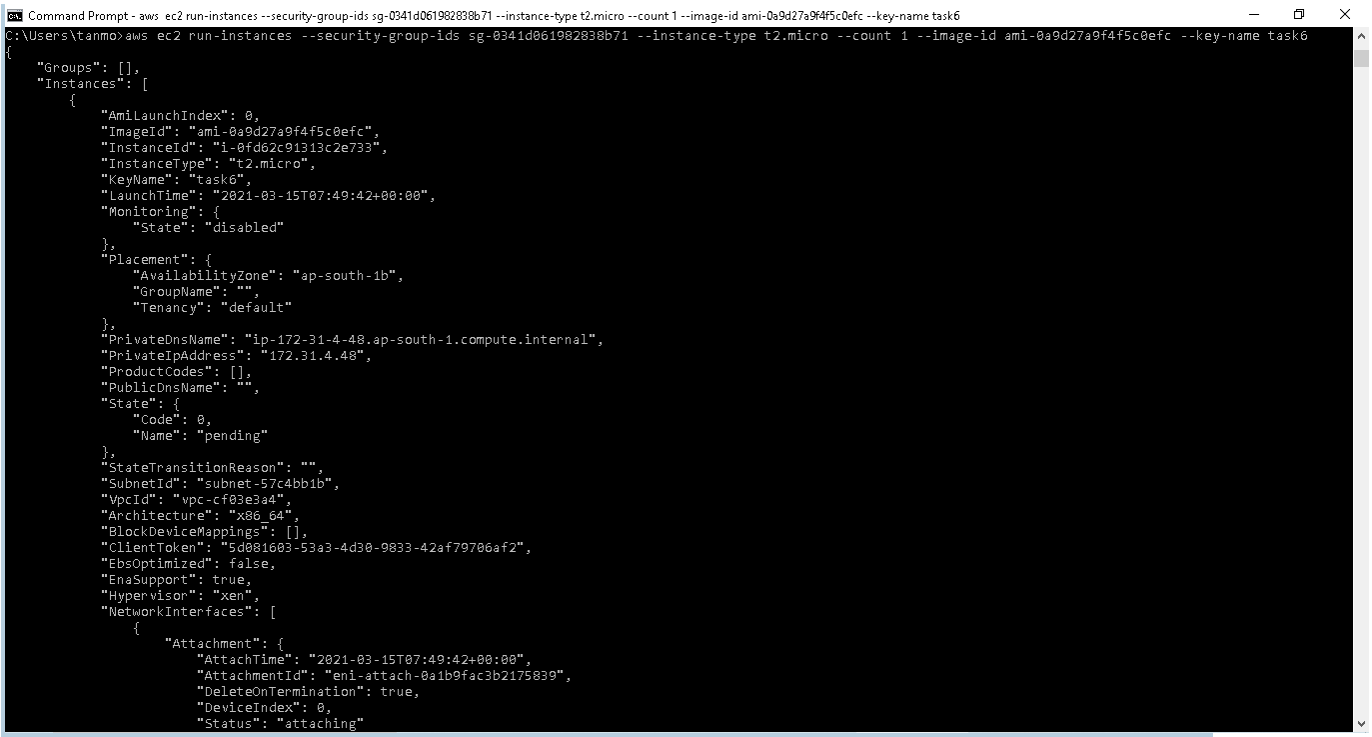
```
C:\Users\tanmo>aws ec2 authorize-security-group-ingress --group-id sg-0341d061982838b71 --protocol tcp --port 80 --cidr
0.0.0.0/0


C:\Users\tanmo>
```

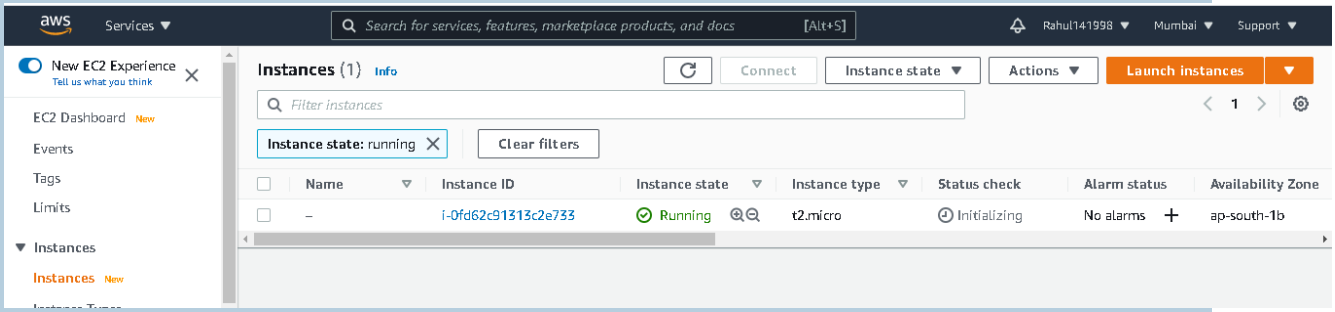


To view the security group details : **aws ec2 describe-security-group — group-id [security group id]**

```
C:\Users\tanmo>aws ec2 describe-security-groups --group-id sg-0341d061982838b71
{
    "SecurityGroups": [
        {
            "Description": "allow all traffic",
            "GroupName": "SG6",
            "IpPermissions": [
                {
                    "FromPort": 80,
                    "IpProtocol": "tcp",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "ToPort": 80,
                    "UserIdGroupPairs": []
                }
            ],
            "OwnerId": "376961321203",
            "GroupId": "sg-0341d061982838b71",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "UserIdGroupPairs": []
                }
            ],
            "VpcId": "vpc-cf03e3a4"
        }
    ]
}
```

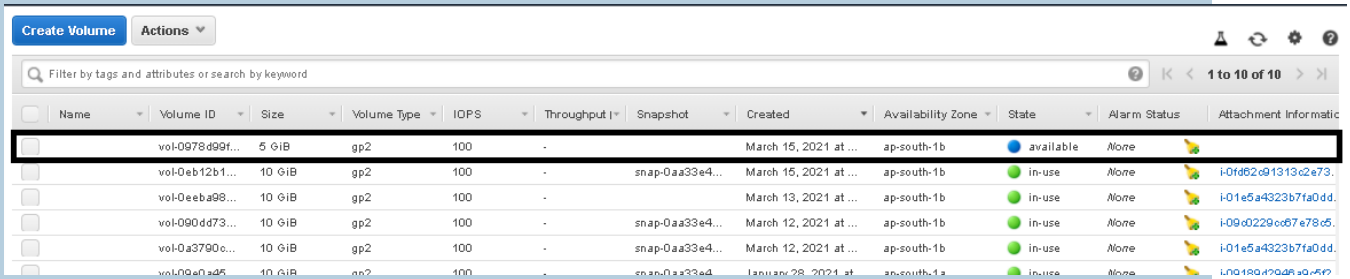
Now launching instance : **aws ec2 run-instances — security-group[sg id] — instance-type [type] — count [number] — image-id [Image id] — key-name [name]**





Now creating EBS volume : **aws ec2 create-volume — availability-zone [AZ of instance] — volume-type [Type] — size [value] -tag-specifications**

```
C:\Users\tanmo>aws ec2 create-volume --availability-zone ap-south-1b --volume-type gp2 --size 5 --tag-specifications
{
    "AvailabilityZone": "ap-south-1b",
    "CreateTime": "2021-03-15T07:51:50+00:00",
    "Encrypted": false,
    "Size": 5,
    "SnapshotId": "",
    "State": "creating",
    "VolumeId": "vol-0978d99fc18f9b8f8",
    "Iops": 100,
    "Tags": [],
    "VolumeType": "gp2",
    "MultiAttachEnabled": false
}
```



Now we need to attach the EBS volume to the running instance : **aws ec2 attach-volume — instance-id [Instance] — volume-id [Volume] — device /attaching directory**

```
C:\Users\tanmo>aws ec2 attach-volume --instance-id i-0fd62c91313c2e733 --volume-id vol-
dh
{
    "AttachTime": "2021-03-15T07:55:24.261000+00:00",
    "Device": "/dev/sdh",
    "InstanceId": "i-0fd62c91313c2e733",
    "State": "attaching",
    "VolumeId": "vol-0978d99fc18f9b8f8"
}
```

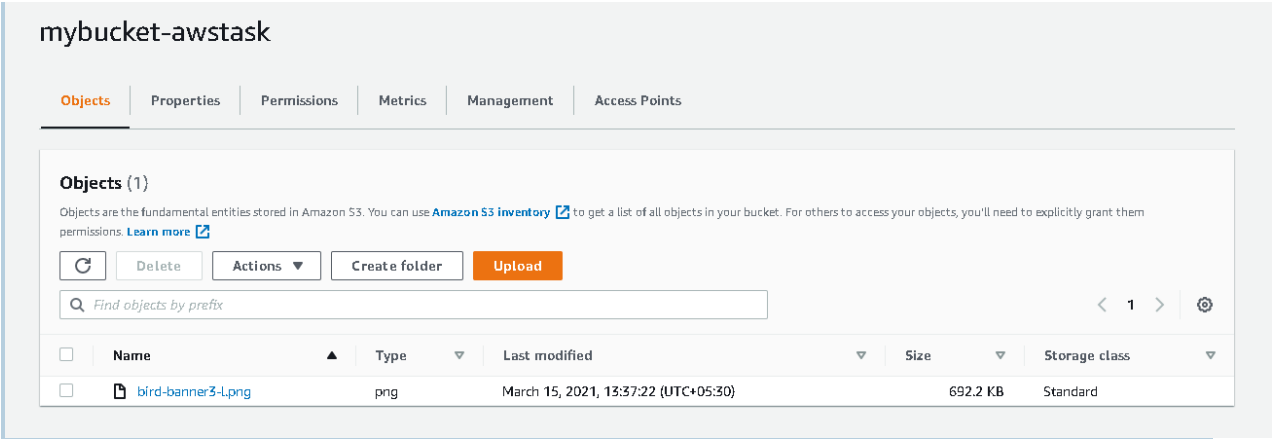
To confirm the attaching of the instance : **aws ec2 describe-volumes**

```
:\Users\tanmo>aws ec2 describe-volumes
{
    "Volumes": [
        {
            "Attachments": [
                {
                    "AttachTime": "2021-01-28T09:42:37+00:00",
                    "Device": "/dev/sda1",
                    "InstanceId": "i-01c50cde1ba8d0c19",
                    "State": "attached",
                    "VolumeId": "vol-0d455b2ee08e270c9",
                    "DeleteOnTermination": true
                }
            ],
            "AvailabilityZone": "ap-south-1b",
            "CreateTime": "2021-01-28T09:42:37.856000+00:00",
            "Encrypted": false,
            "Size": 10,
            "SnapshotId": "snap-0aa33e40a60b0122d",
            "State": "in-use",
            "VolumeId": "vol-0d455b2ee08e270c9",
            "Iops": 100,
            "VolumeType": "gp2",
            "MultiAttachEnabled": false
        },
        {
            "Attachments": [
                {
                    "AttachTime": "2021-01-28T09:50:34+00:00",
                    "Device": "/dev/sda1",
                    "InstanceId": "i-06a5a429ac1941862",
                    "State": "attached",
                    "VolumeId": "vol-037ff041c5a5b0db9",
                    "DeleteOnTermination": true
                }
            ],
            "AvailabilityZone": "ap-south-1b",
            "CreateTime": "2021-01-28T09:50:34.850000+00:00",
            "Encrypted": false,
```

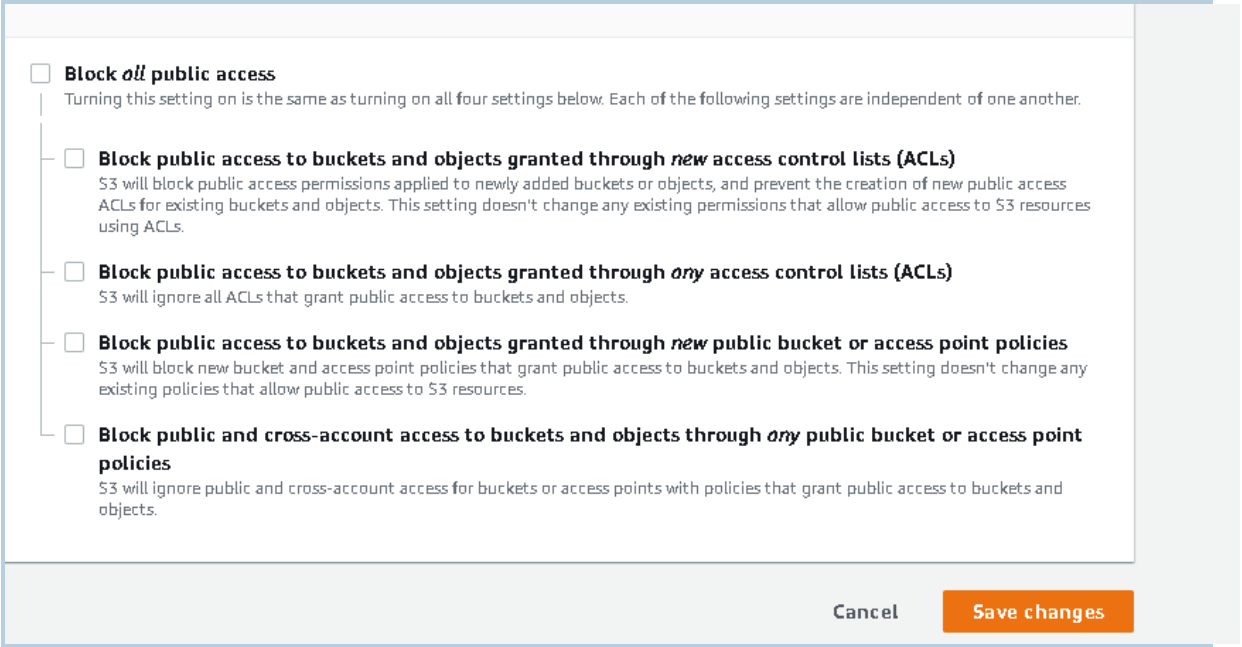
Now we need to create an S3 bucket : **aws s3 mb s3://[bucket-name] — region [region]**

```
C:\Users\tanmo>aws s3 mb s3://mybucket-awstask --region ap-south-1
make_bucket: mybucket-awstask
```

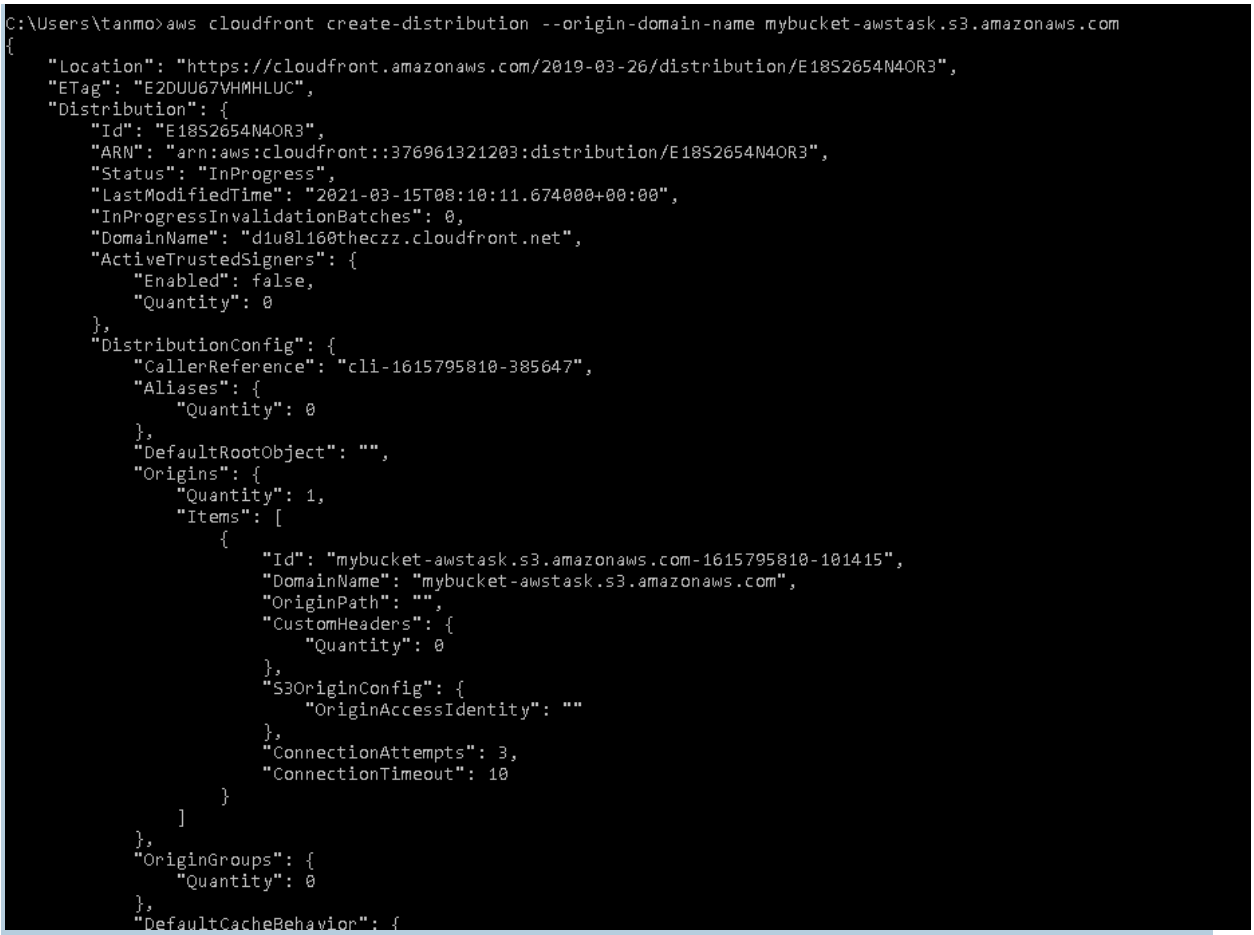
Then we upload a file that we need there



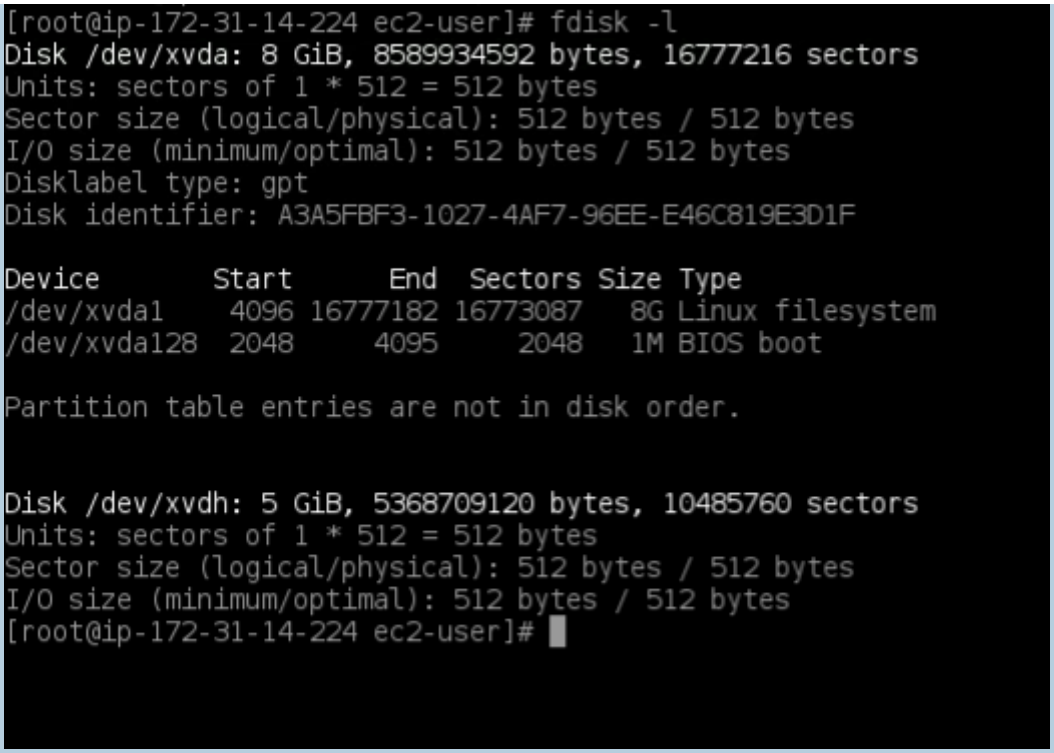
We need to make the file in the S3 bucket public to make it accessible to everyone.



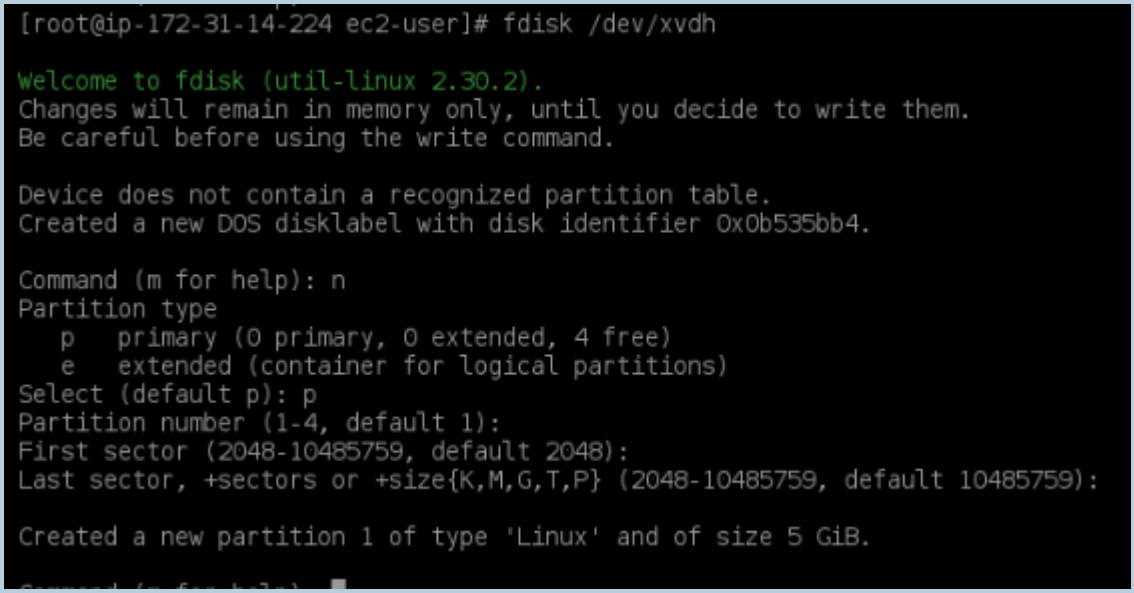
Now we created cloudfront web distribution for faster content delivery across the world from any of the location using origin as s3



Now we connect to the Instance and check the partitions using **fdisk -l**



We create a partition to use the attached EBS volume : **fdisk /directory**



Now we need to format the partition pre usage : **mkfs.ext4 /directory**

```
[root@ip-172-31-14-224 ec2-user]# mkfs.ext4 /dev/xvdh
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
327680 inodes, 1310720 blocks
65536 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1342177280
40 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Before Mounting we need to install the webserver : **yum install httpd -y**

```
[root@ip-172-31-14-224 etc]# yum install httpd -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.46-1.amzn2 will be installed
--> Processing Dependency: httpd-tools = 2.4.46-1.amzn2 for package: httpd-2.4.46-1.amzn2.x86_64
--> Processing Dependency: httpd-filesystem = 2.4.46-1.amzn2 for package: httpd-2.4.46-1.amzn2.x86
--> Processing Dependency: system-logos-httpd for package: httpd-2.4.46-1.amzn2.x86_64
--> Processing Dependency: mod_http2 for package: httpd-2.4.46-1.amzn2.x86_64
--> Processing Dependency: httpd-filesystem for package: httpd-2.4.46-1.amzn2.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.46-1.amzn2.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.46-1.amzn2.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.46-1.amzn2.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.6.3-5.amzn2.0.2 will be installed
---> Package apr-util.x86_64 0:1.6.1-5.amzn2.0.2 will be installed
--> Processing Dependency: apr-util-bdb(x86-64) = 1.6.1-5.amzn2.0.2 for package: apr-util-1.6.1-5.
---> Package generic-logos-httpd.noarch 0:18.0.0-4.amzn2 will be installed
---> Package httpd-filesystem.noarch 0:2.4.46-1.amzn2 will be installed
---> Package httpd-tools.x86_64 0:2.4.46-1.amzn2 will be installed
---> Package mailcap.noarch 0:2.1.41-2.amzn2 will be installed
---> Package mod_http2.x86_64 0:1.15.14-2.amzn2 will be installed
--> Running transaction check
---> Package apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

now changing the directory to the main webserver directory : **cd /var/www/html** and mount the partition there : **mount /partitiondirectory /mounting directory**

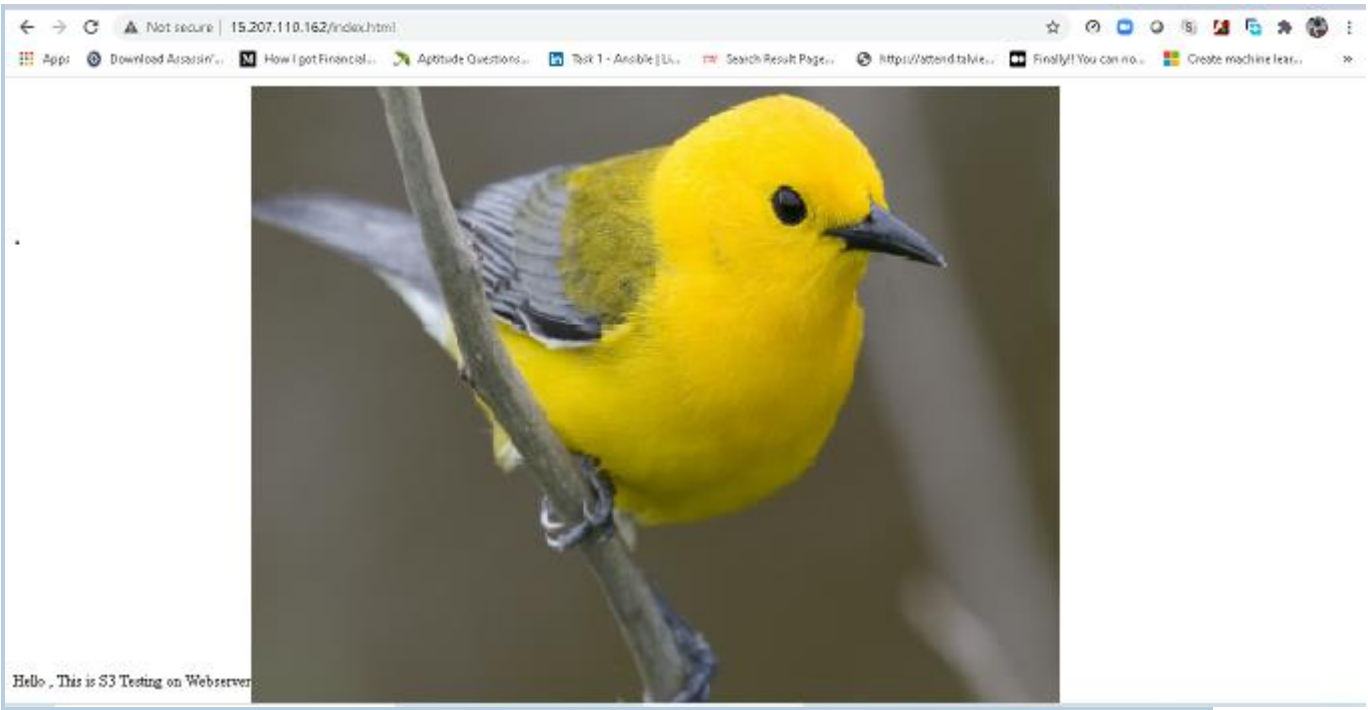
```
[root@ip-172-31-14-224 etc]# cd /var/www/html
[root@ip-172-31-14-224 html]# mkdir /web
[root@ip-172-31-14-224 html]# mount /dev/xvdh /var/www/html/web/
mount: /var/www/html/web/: mount point does not exist.
[root@ip-172-31-14-224 html]# mount /dev/xvdh /var/www/html
[root@ip-172-31-14-224 html]#
```

Making a file index.html and entering the code there

```
Hello , This is S3 Testing on Webserver
<img src="https://mybucket-awstask.s3.ap-south-1.amazonaws.com/bird-banner3-l.png"'height=400 width=600"/>
</body>
~
"index.html" 4L, 163C written
[root@ip-172-31-14-224 html]# vi index.html
```

Starting the webserver : **systemctl start httpd**

And the going to the browser and entering in the address bar **: [public ip of the instance/filename]**



Thus the Page runs successfully and 4 different services of AWS are used: **EC2:** for Running OS

EBS: for Attaching Extra storage

S3: for storing the file

Cloudfront : For providing high availability