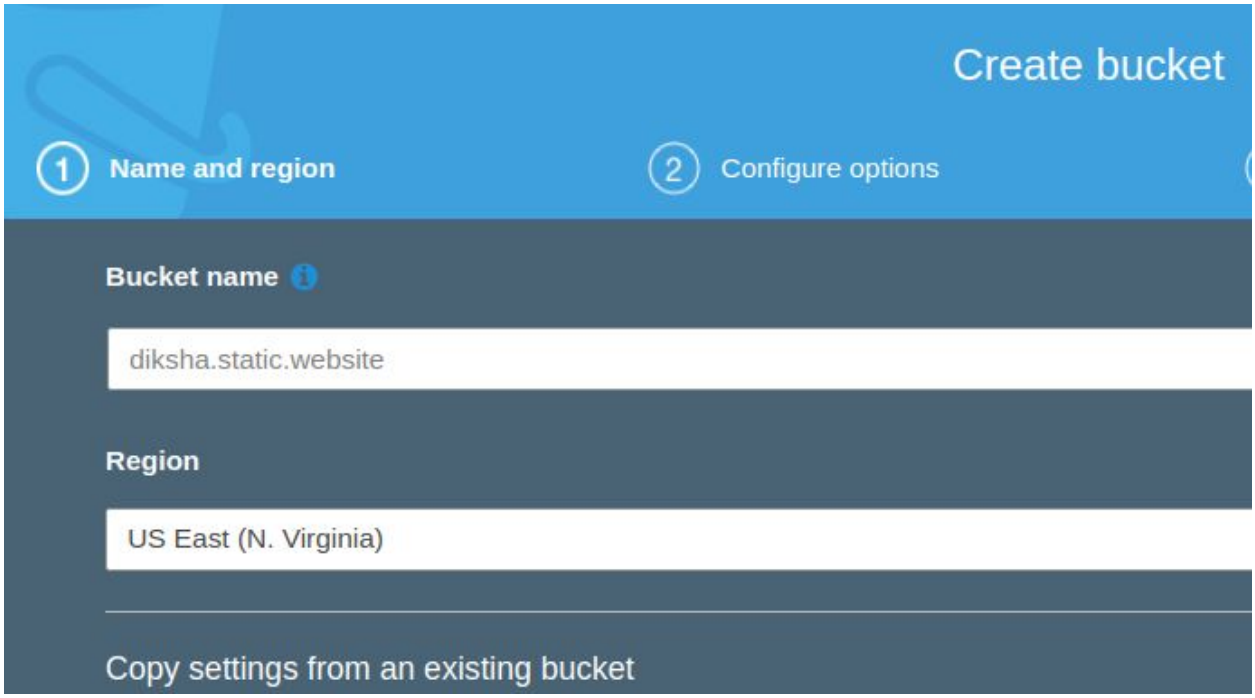
ASSESSMENT ON S3, ROUTE 53 AND RDS

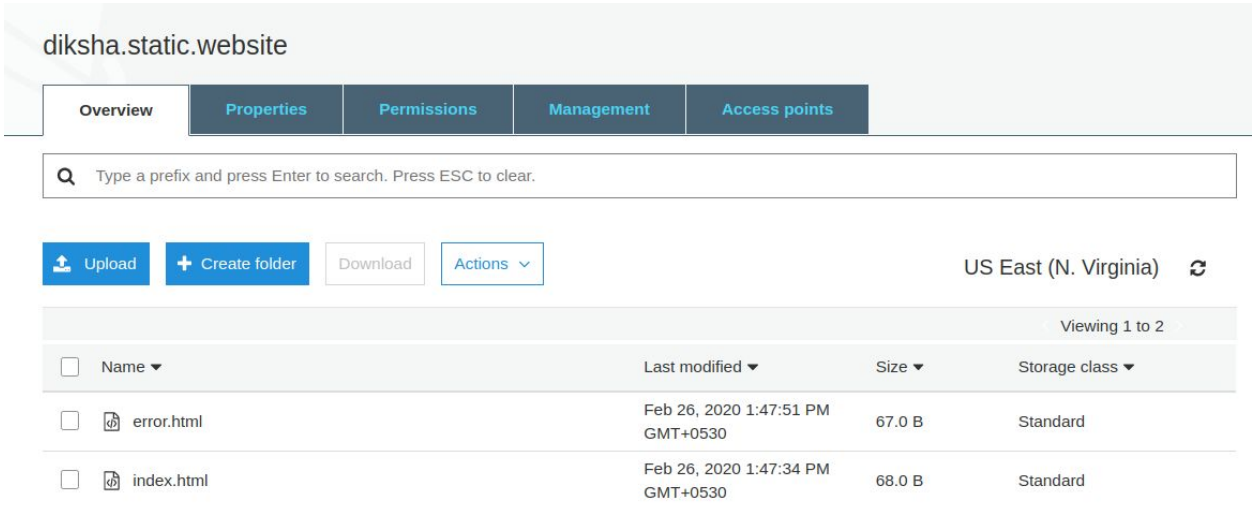
1.Host a static website using s3 (what is index page and error page i.e significance)

Step 1: Navigate to S3 in the AWS Console.

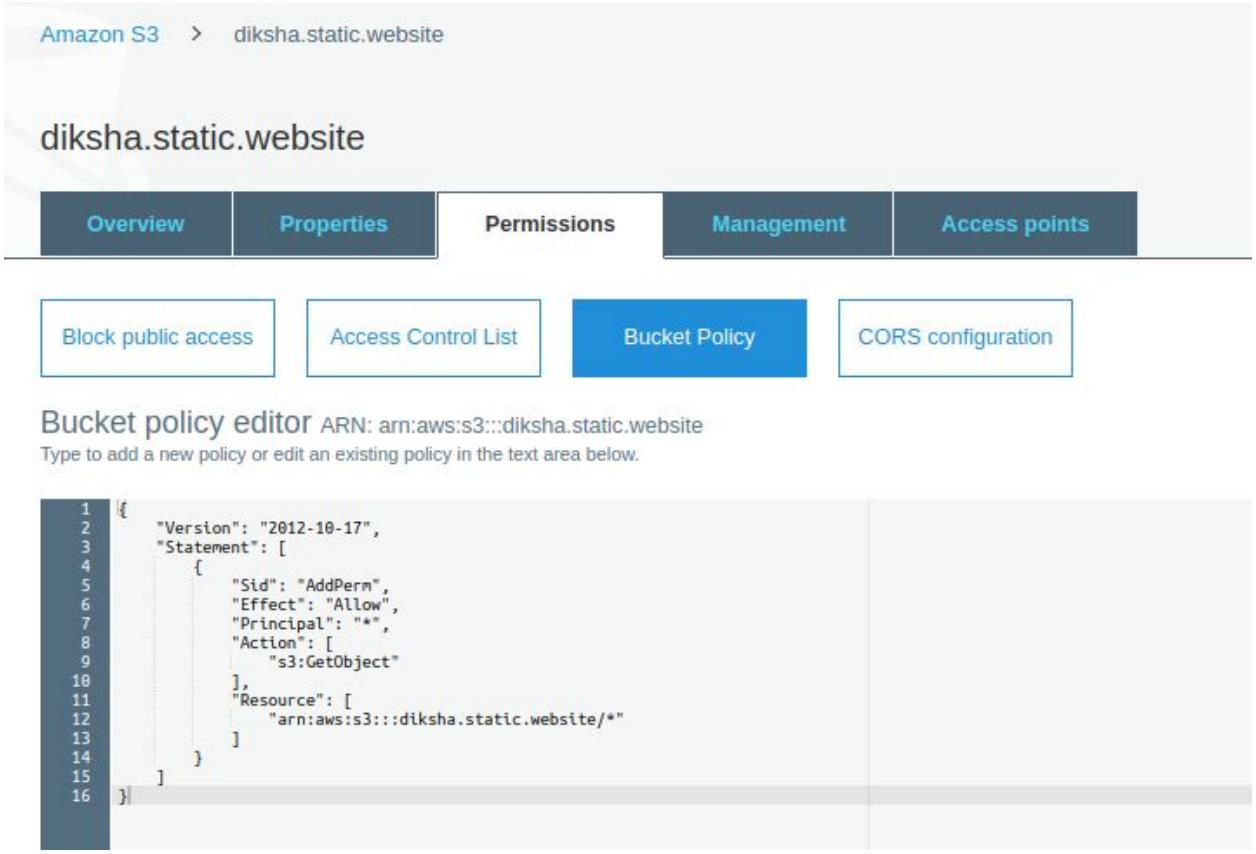
Step 2:: Click Create bucket.Give the bucket a name(it must be unique!!!) and keep clicking next till you get to the review (leave the values as default) then click Create bucket



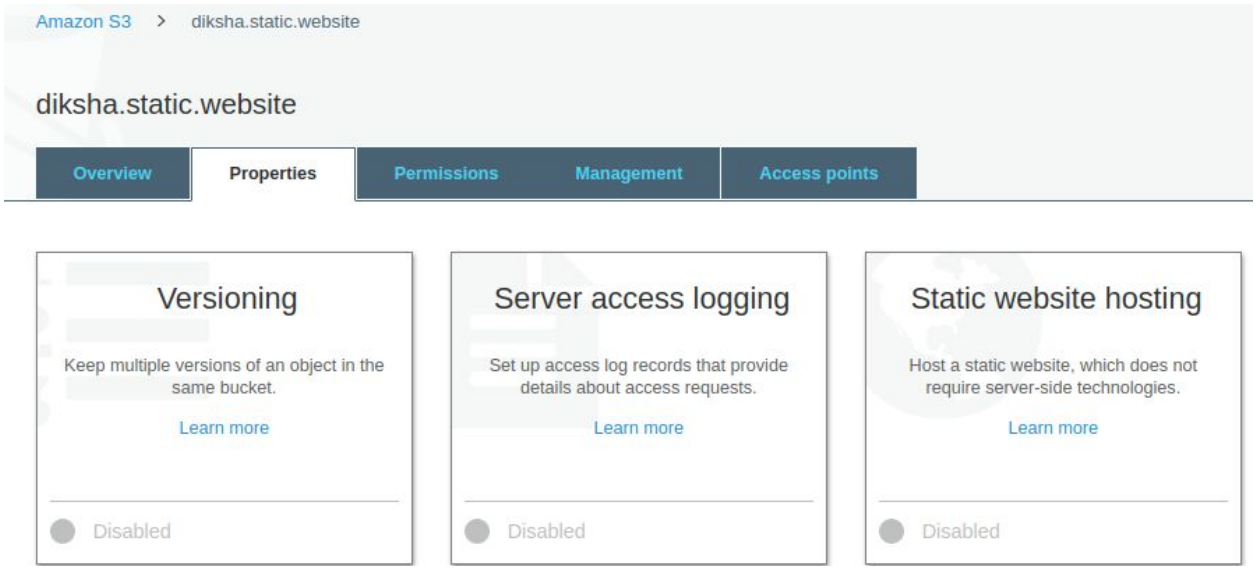
Step 3 : Click on your newly created bucket.Click Upload to start uploading your static web app files to your bucket.



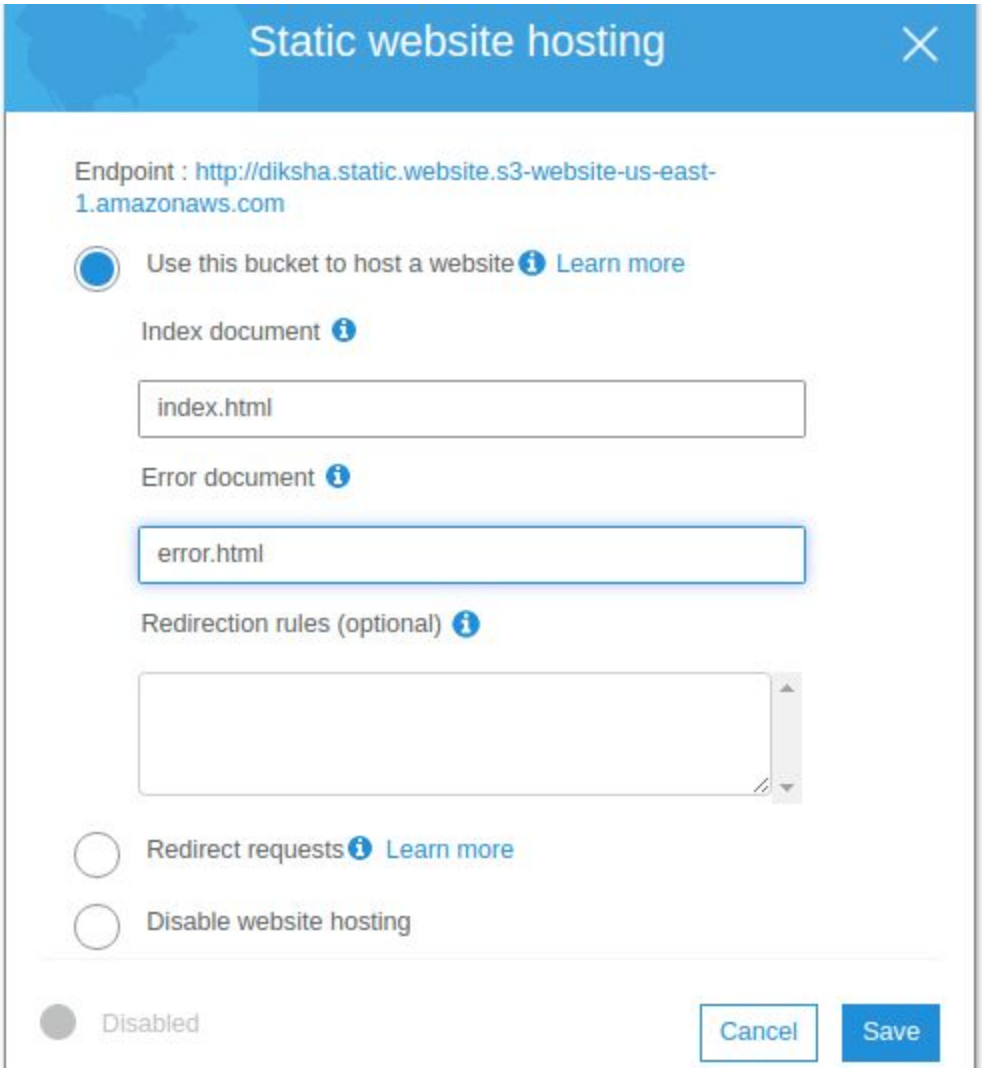
Step 4: Our bucket is only accessible to us so far; let's make it public by changing some policies, Click permissions tab then click on Bucket Policy



Step 5: Time to turn our bucket into a static web hosting server, click on properties then Static website hosting



Step 6: Set the entry page of your static app and any error pages you might have then click save.



Step 7: Click on Static website hosting again (after everything was saved) and you should see an endpoint URL

2.Create an assume role to access s3 using EC2.

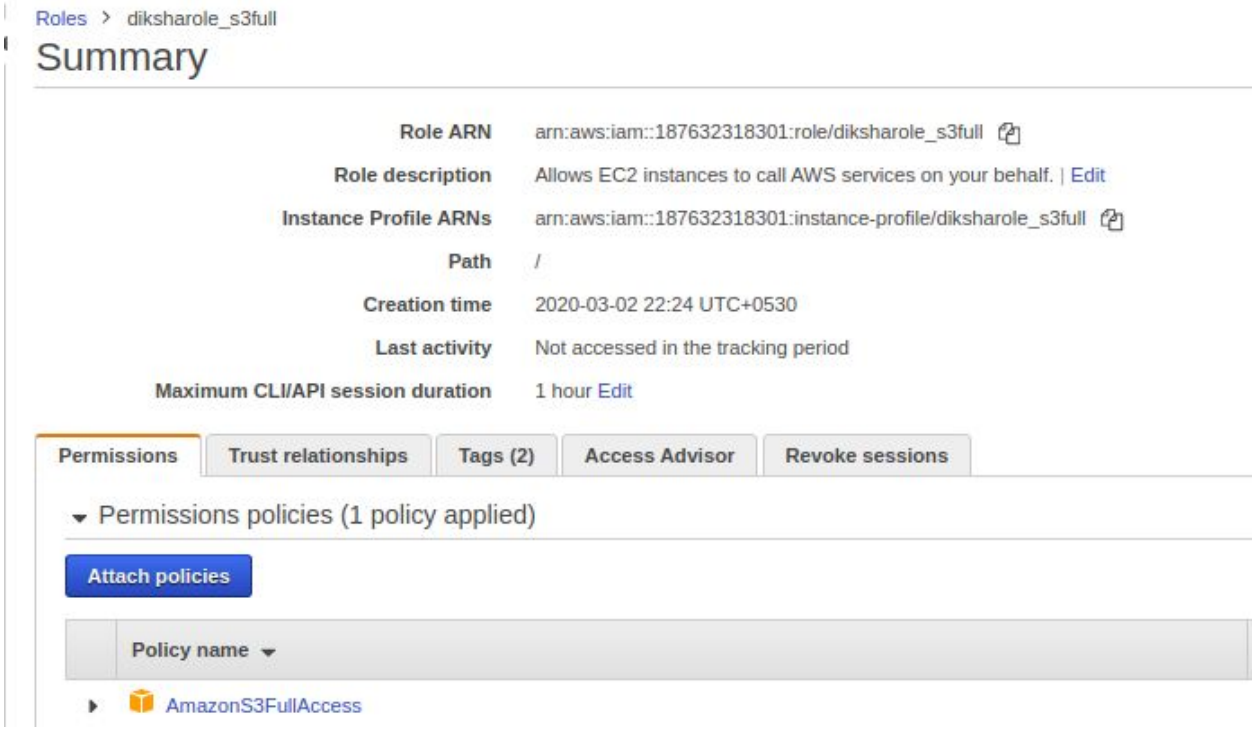
ANS: Returns a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary credentials consist of an access key ID, a secret access key, and a security token. Typically, you use AssumeRole within your account or for cross-account access.

*You cannot use AWS account root user credentials to call AssumeRole. You must use credentials for an IAM user or an IAM role to call AssumeRole.

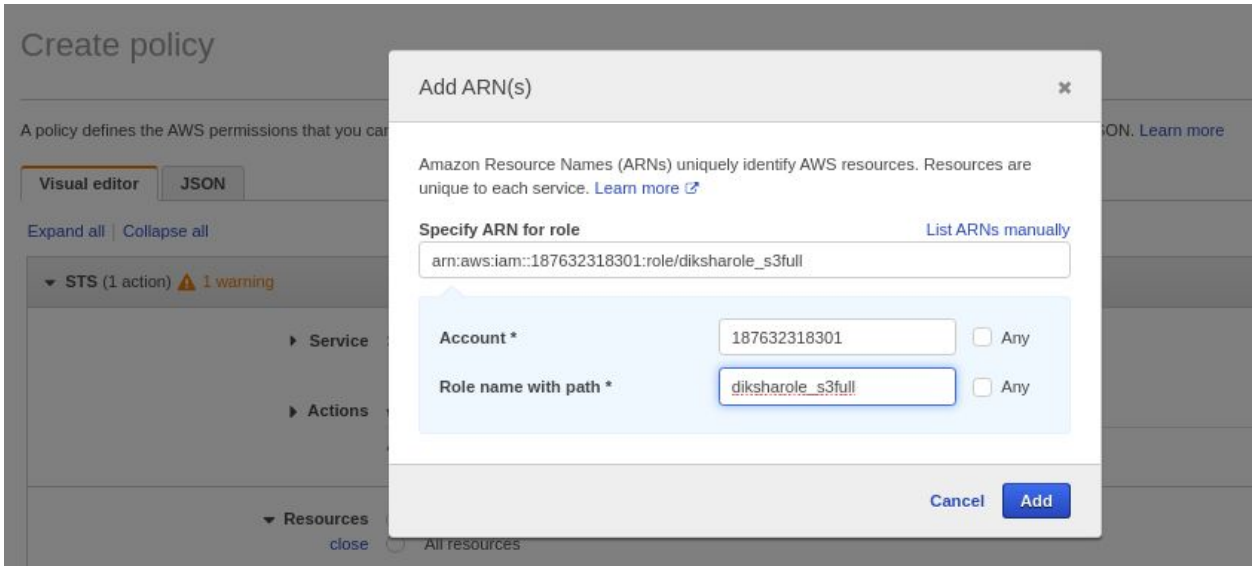
For cross-account access, imagine that you own multiple accounts and need to access resources in each account. You could create long-term credentials in each account to access those resources. However, managing all those credentials and remembering which one can access which account can be time consuming. Instead, you can create one set of long-term credentials in one account. Then use temporary security credentials to access all the other accounts by assuming roles in those accounts.

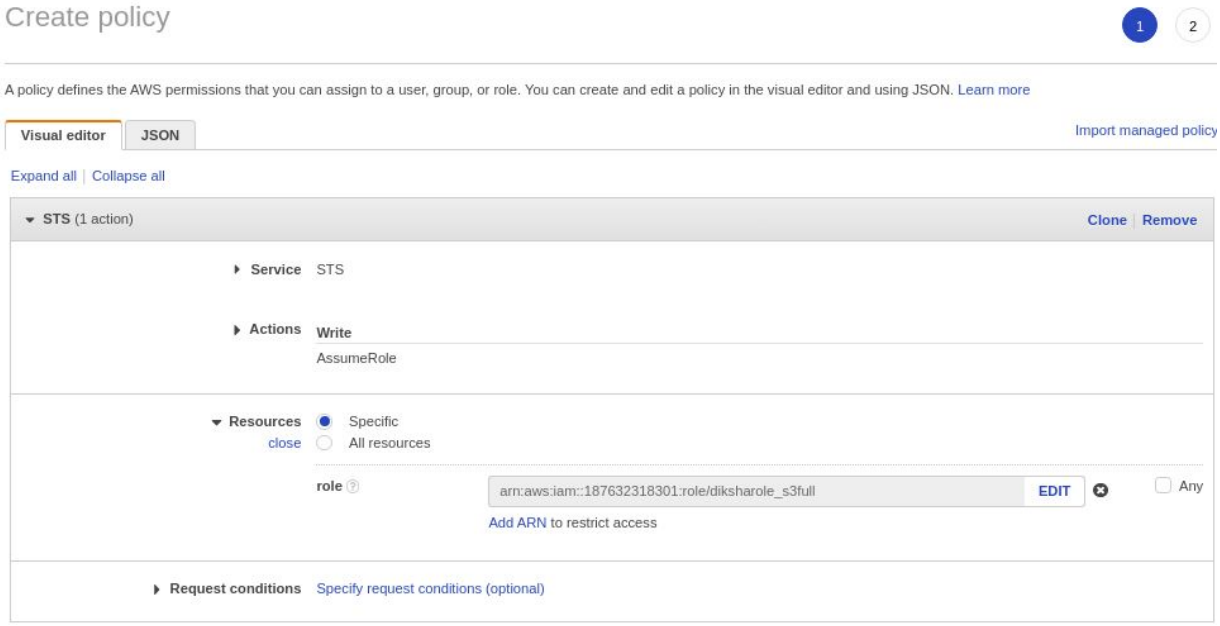
By default, the temporary security credentials created by AssumeRole last for one hour.

STEP 1: Create a role dikshas3_full

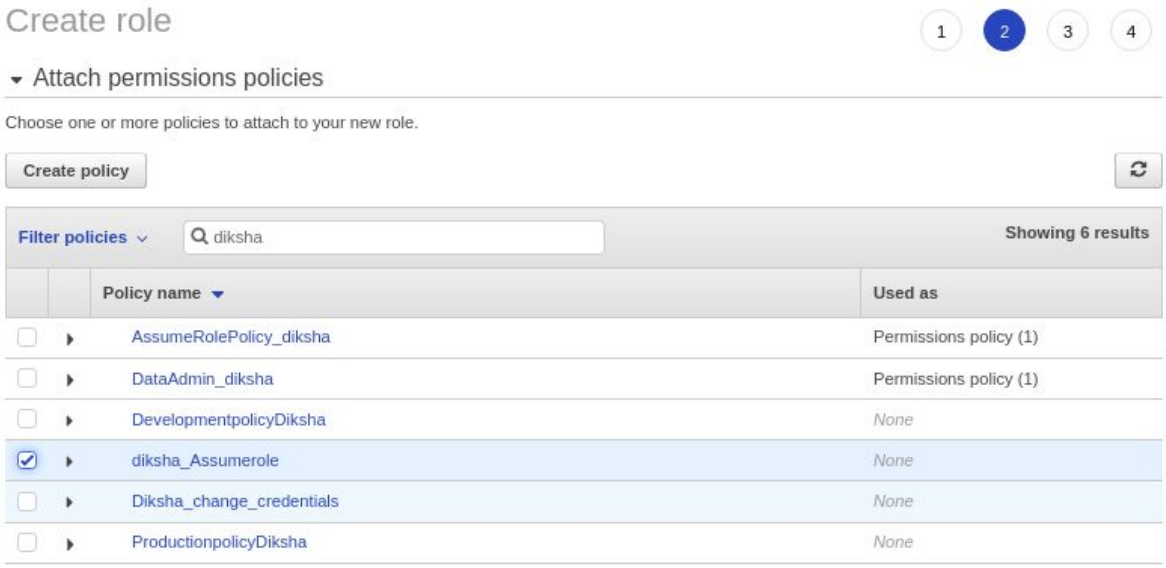


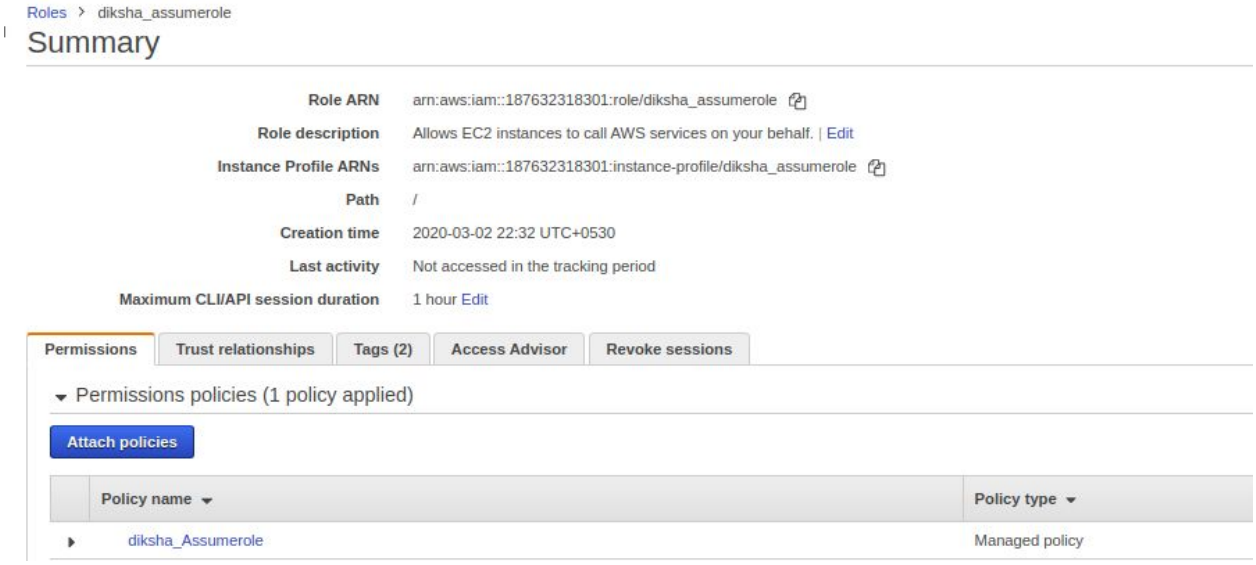
STEP 2: Create a policy “diksha_Assumerole” in which give Service=STS and Action=Assumerole and in Resources=ARN of role “diksharole_s3full”





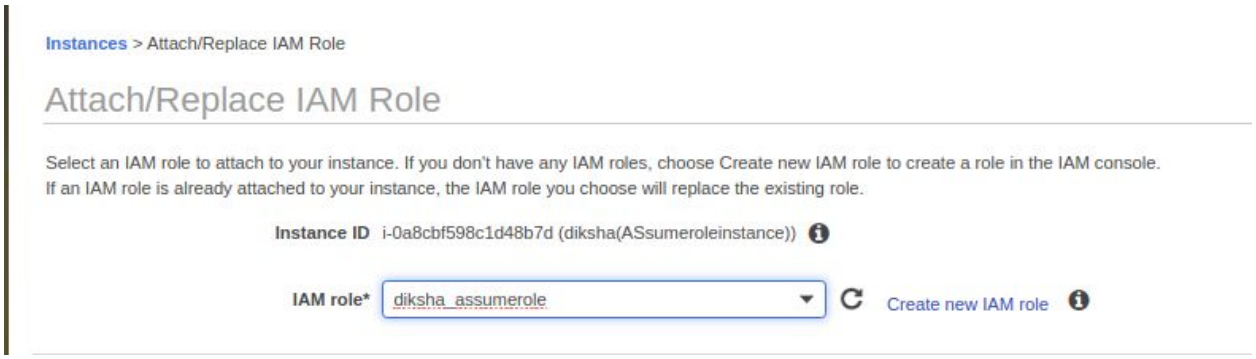
STEP 3: Create a role “diksha_assumerole” and attach the policy created to it



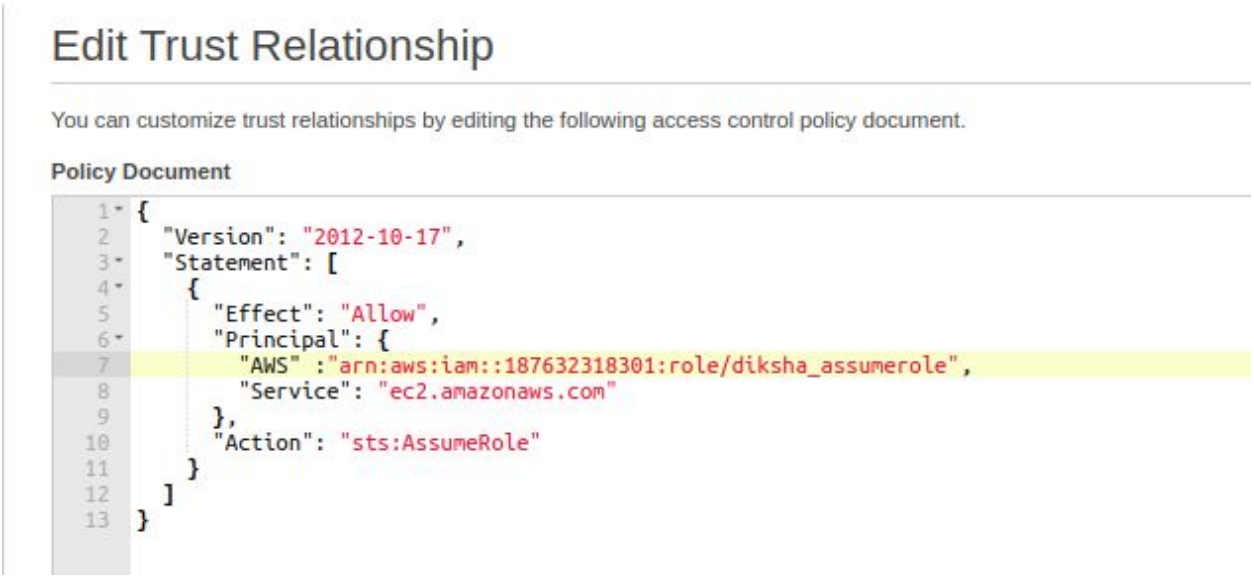


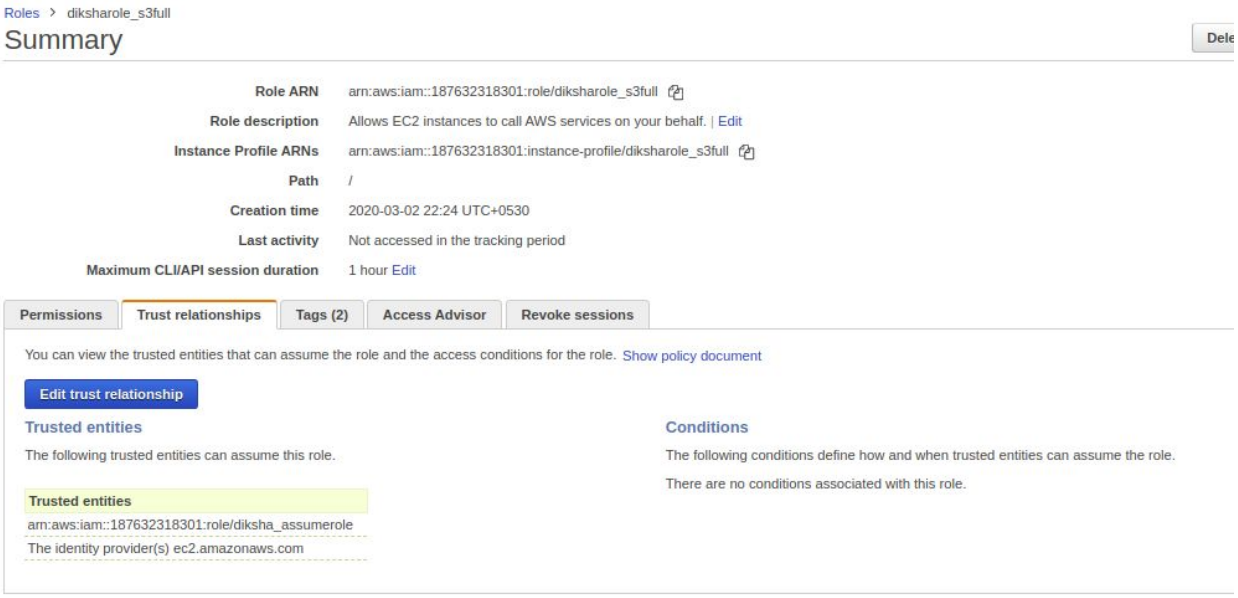
STEP 4: Launch an instance.

STEP 5: Attach the role “diksha_assumerole” to the above created instance.



STEP 6: Now add the arn of new role i.e “diksha_assumerole” to old role “diksharole_s3full in trust relationship





STEP 7: SSH into the instance and get an STS token.

```
ubuntu@ip-172-31-79-130:~$ aws sts assume-role --role-arn arn:aws:iam::187632318
301:role/diksharole_s3full --role-session-name DikshaSession
{
    "Credentials": {
        "AccessKeyId": "ASIASXL6B65O37KSEX57",
        "SecretAccessKey": "SWJ+BOBOYZDOX5q1ZzdAkY9FkZqsLb+nHHKIvHYj",
        "SessionToken": "FwoGZXIvYXdzEGsaDCbQ8g60BLFlrYLkcSKxAaIKA6+75lFTz5YdgAV
HzvB+M+SueIlRaj3zwF5vR8X3b/T1UlBSy3gVKjHjO78EsiZDLxB5Lf/AQkNwYvknaWQIDen6rCjGzci
PIpivw6HBcQe+WywEx3WLCPLmoYKNpJ2qIXImig2gp81hJPNuVjEjhaJC2YZaRKOw/8gL8uOkkKX8DMU
h1v3q/Cx1p5ttngquMJ5Q/YwcfEWORiRZbSOseIQhdAR3QhP4HmtK6EJWiCjLh/XyBTItGRygyISXWzI
iX3MAcnBrSFsS9IgENhssDq2HlQkd3sWIASU7KOWtepPfElEc",
        "Expiration": "2020-03-02T18:35:07Z"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROASXL6B65O3YRM7357A:DikshaSession",
        "Arn": "arn:aws:sts::187632318301:assumed-role/diksharole_s3full/DikshaS
ession"
    }
}
```

STEP 8:Now export it and now ls in s3 you will observe that access is given to you and you can see the buckets listed.

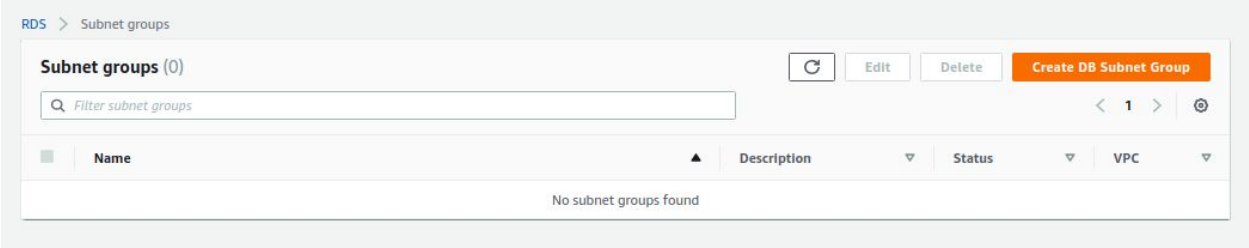
```
ubuntu@ip-172-31-79-130:~$ aws s3 ls

An error occurred (AccessDenied) when calling the ListBuckets operation: Access
Denied
ubuntu@ip-172-31-79-130:~$
ubuntu@ip-172-31-79-130:~$
ubuntu@ip-172-31-79-130:~$
ubuntu@ip-172-31-79-130:~$ export AWS_ACCESS_KEY_ID=ASIASXL6B65O37KSEX57
ubuntu@ip-172-31-79-130:~$ export AWS_SECRET_ACCESS_KEY=SWJ+BOBOYZDOX5q1ZzdAkY9F
kZqsLb+nHHKIvHYj
ubuntu@ip-172-31-79-130:~$ export AWS_SESSION_ACCESS_KEY=FwoGZXIvYXdzEGsaDCbQ8g6
0BLFlrYLkcSKxAaIKA6+75lFTz5YdgAVHzvB+M+SueIlRaj3zwF5vR8X3b/T1UlBSy3gVKjHjO78EsiZ
DLxB5Lf/AQkNwYvknaWQIDen6rCjGzciPIpivw6HBcQe+WywEx3WLCPLmoYKNpJ2qIXImig2gp81hJPN
uVjEjhaJC2YZaRKOw/8gL8uOkkKX8DMUh1v3q/Cx1p5ttngquMJ5Q/YwcfEWORiRZbSOseIQhdAR3QhP
4HmtK6EJWiCjLh/XyBTItGRygyISXWzIiX3MAcnBrSFsS9IgENhssDq2HlQkd3sWIASU7KOWtepPfElE
c
ubuntu@ip-172-31-79-130:~$
```

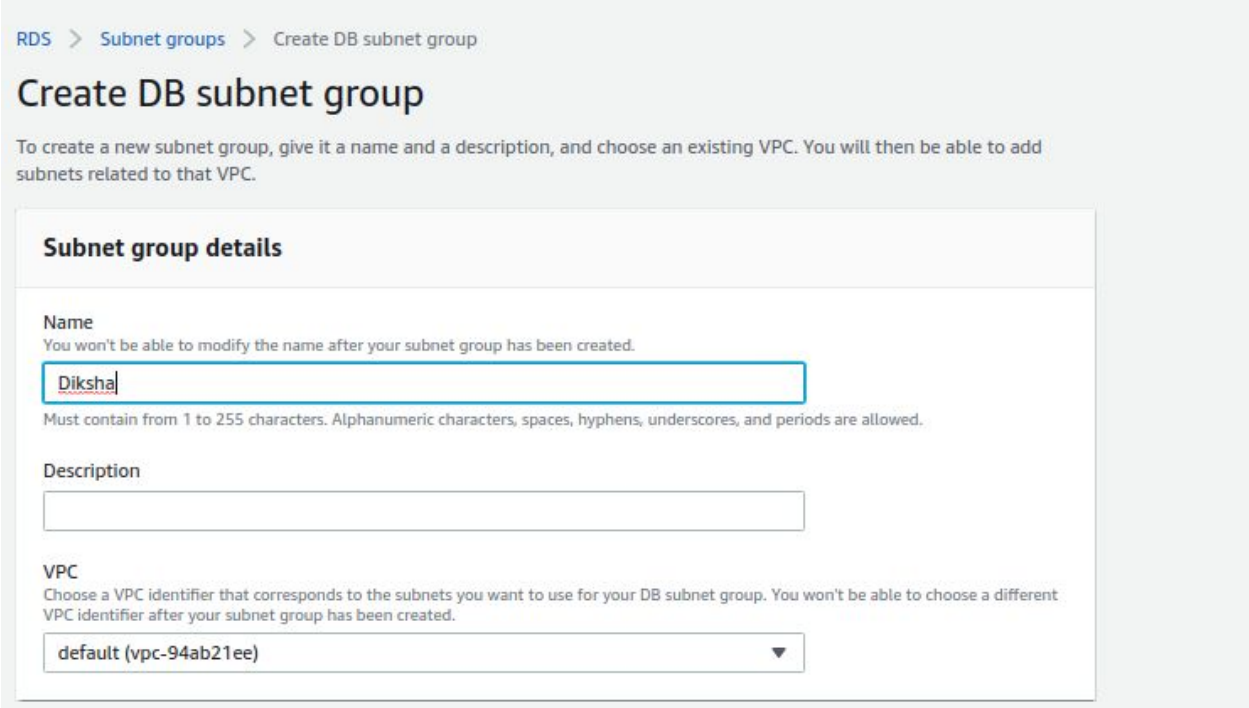
```
ubuntu@ip-172-31-4-230:~$ aws s3 ls
2019-06-26 12:11:08 0testuser11
2018-04-20 16:59:22 187632318301-awsmacietrail-dataevent
2019-04-02 10:11:33 7testdemo
2019-03-11 04:51:59 abhimanyucftemplate
2020-02-28 10:55:02 abhishek-bootcamp
2019-03-04 06:55:23 abneesh1
2019-03-11 11:00:41 adityamun007
2020-02-26 16:26:29 akshaybuck1
2020-02-27 08:55:25 aman-khandelwal-1
2019-03-07 09:40:48 anmol-bootcamp19
2019-03-08 00:25:58 avcabc
2017-09-07 03:41:42 aws-codestar-us-east-1-187632318301
2017-09-07 04:23:01 aws-codestar-us-east-1-187632318301-codestartest2-app
2017-09-07 04:23:07 aws-codestar-us-east-1-187632318301-codestartest2-pipe
2017-09-07 03:41:48 aws-codestar-us-east-1-187632318301-codestarttest-pipe
```

3.Create RDS subnet and launch RDS Instance. What is parameter group and Option Group?
ANS:

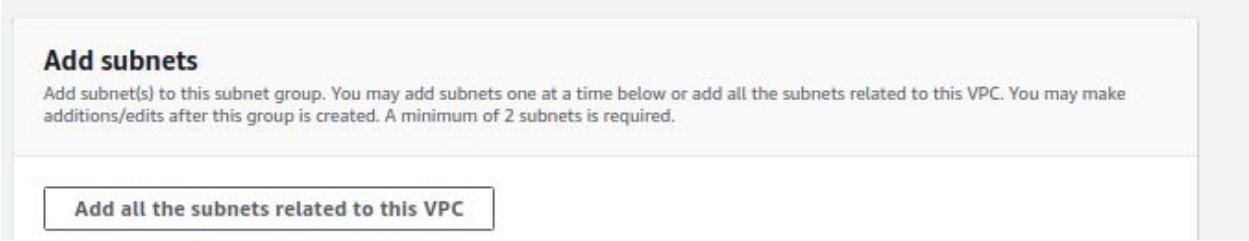
Create a DB Subnet Group

STEP 1: Subnet groups > Choose Create DB Subnet Group. For VPC, choose the VPC that you created.

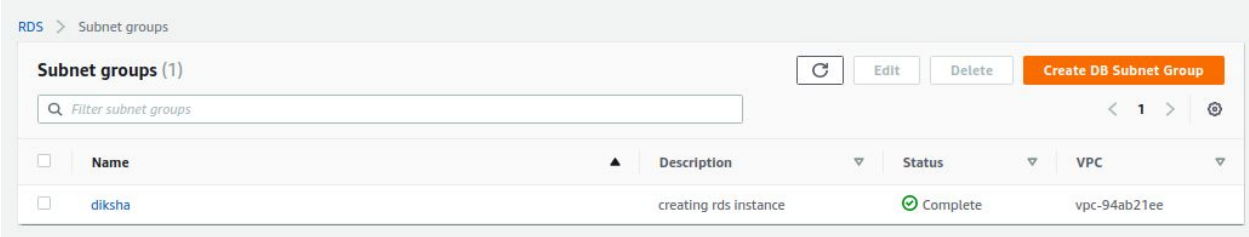




STEP 2: In the Add subnets section, choose Add all the subnets related to this VPC.

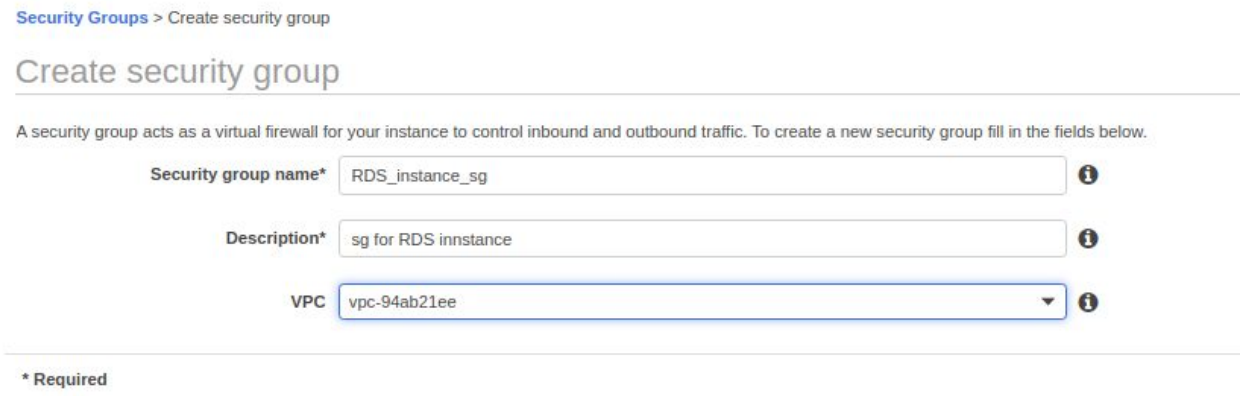


STEP 3: Choose Create.

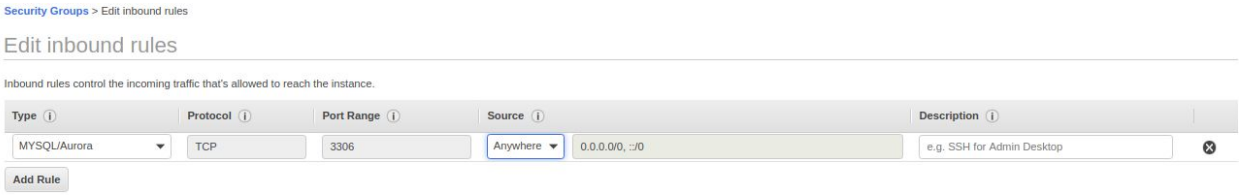


Create a VPC Security Group: Before you create your DB instance, you must create a VPC security group to associate with your DB instance.

STEP 1: Create a VPC Security Group

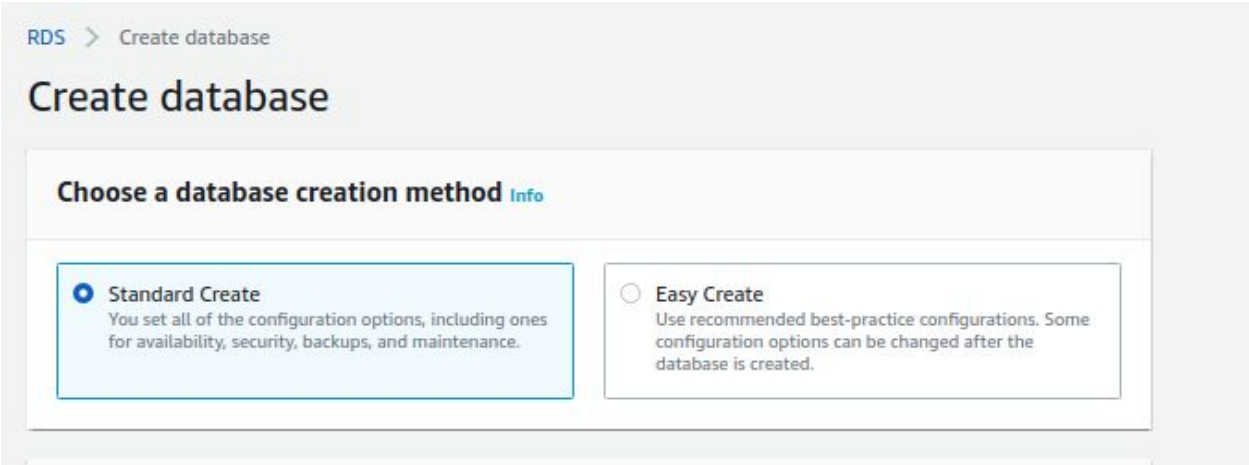


STEP 2: Choose the security group you created and edit inbound rules.
*Set the following values for your new inbound rule to allow MySQL traffic on port 3306 from your EC2 instance. If you do this, you can connect from your web server to your DB instance to store and retrieve data from your web application to your database.



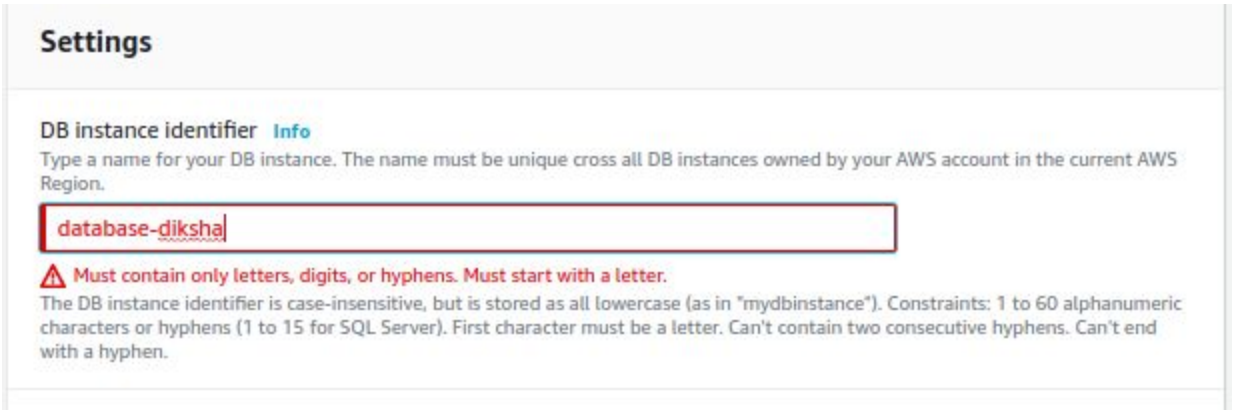
Create a DB Instance in the VPC

STEP 1: **Databases** > Choose **Create database** > In **Choose a database creation method**, choose **Standard Create** I
*Use the VPC name, the DB subnet group, and the VPC security group you created in the previous steps.
*If you want your DB instance in the VPC to be publicly accessible, you must enable the VPC attributes *DNS hostnames* and *DNS resolution*.



STEP 2: In **Engine options**, choose **MySQL**.

STEP 3: In **Templates**, choose the template that matches your use case.

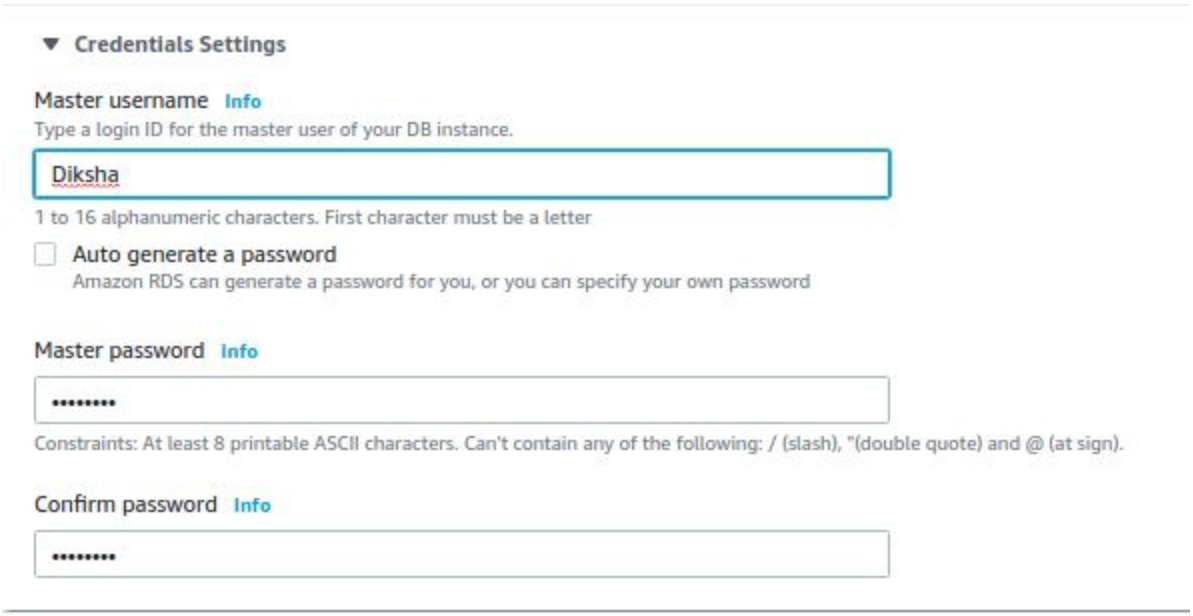


STEP 4: To enter your master password, do the following:

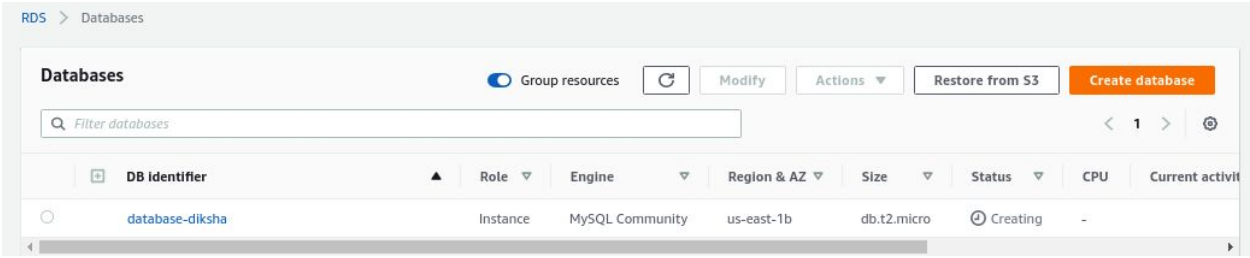
a. In the **Settings** section, open **Credential Settings**.

b. Clear the **Auto generate a password** check box.

c. (Optional) Change the **Master username** value and enter the same password in **Master password** and **Confirm password**.



STEP 5: Your database has been created.



Parameter group : For AWS RDS instances, you manage your database engine configuration through the use of parameters in a DB parameter group. DB parameter groups act as a container for engine configuration values that are applied to one or more DB instances.
Option Group: An *option group* can specify features, called options, that are available for a particular Amazon RDS DB instance. Options can have settings that specify how the option works. When you associate a DB instance with an option group, the specified options and option settings are enabled for that DB instance.

Amazon RDS supports options for the following database engines:

Database Engine	Relevant Documentation
`MariaDB`	**Options for MariaDB Database Engine**
`Microsoft SQL Server`	**Options for the Microsoft SQL Server Database Engine**
`MySQL`	**Options for MySQL DB Instances**
`Oracle`	**Options for Oracle DB Instances**

4.ACL, Bucket policy, IAM policy in context of S3?
ANS:
The Access Control List (ACL): is used to define other users' access permissions for your file and folder objects. The Access Permissions that you set using the ACL determine what a user can and cannot do with your file and folder objects. For example, you can set permissions on a file object to let one user read the contents of a file (read access) and let another user make changes to the file (write access). In Amazon S3 you will first add grants to objects and then set the permissions for the grant.

There are 4 types of grants:

1. An Owner grant - which defines the permissions the owner of the object has.

2. Authenticated Users – which are all Amazon S3 storage users that have an account with S3.

3. Public – which means any anonymous user that you have provided the URL to.

4. Email-ID – which is an email address of specific S3 customers that have S3 accounts, not general public emails. The email given must match exactly the email address the S3 user signed up with and can only match one user account.

Bucket Policies: bucket Policies are similar to IAM policies in that they allow access to resources via a JSON script. However, Bucket policies are applied to Buckets in S3, where as IAM policies are assigned to user/groups/roles and are used to govern access to any AWS resource through the IAM service.

When a bucket policy is applied the permissions assigned apply to all objects within the Bucket. The policy will specify which 'principles' (users) are allowed to access which resources. The use of Principles within a Bucket policy differs from IAM policies, Principles within IAM policies are defined by who is associated to that policy via the user and group element. As Bucket policies are assigned to Buckets, there is this need of an additional requirement of 'Principles'.

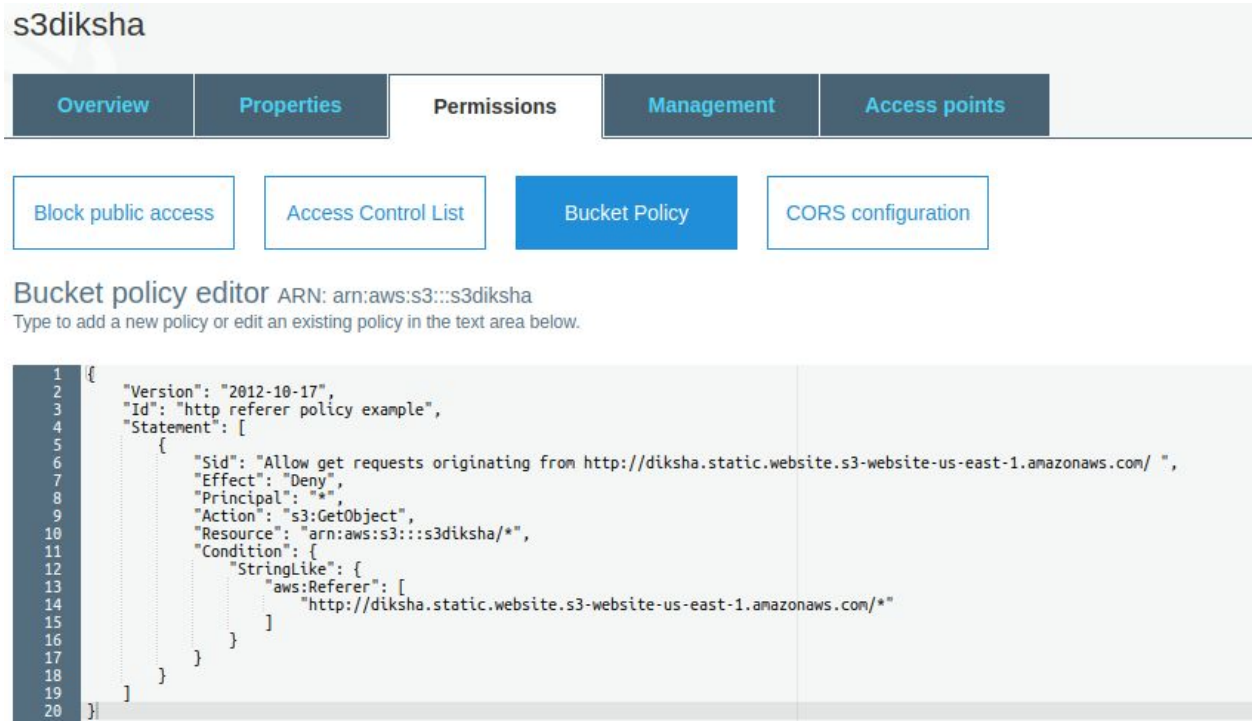
IAM POLICY :A policy is an entity that, when attached to an identity or resource, defines their permissions. A policy that is attached to an identity in IAM is known as an *identity-based policy*. Identity-based policies can include AWS managed policies, customer managed policies, and inline policies. AWS managed policies are created and managed by AWS. You can use them, but you can't manage them. An inline policy is one that you create and embed directly to an IAM group, user, or role. Inline policies can't be reused on other identities or managed outside of the identity where it exists.

5.Block S3 access on the basis of:

a. **Ip**

```
{
 "Version": "2012-10-17",
 "Id": "S3PolicyId1",

 "Statement": [

   {
     "Sid": "IPAllowdiksha",
     "Effect": "Deny",
     "Principal": "*",
     "Action": "s3:*",
     "Resource": "arn:aws:s3:::s3diksha/*",
     "Condition": {
        "NotIpAddress": {"aws:SourceIp": "10.1.210.205"}

     }
   }
 ]
}
```

b. Domain



c. Pre-signed URL(time based)

ANS: A presigned URL is a URL that you can provide to your users to grant temporary access to a specific S3 object. Using the URL, a user can either READ the object or WRITE an Object (or update an existing object). The URL contains specific parameters which are set by your application.

A pre-signed URL uses three parameters to limit the access to the user;

- Bucket: The bucket that the object is in (or will be in)
- Key: The name of the object

- Expires: The amount of time that the URL is valid

Bucket policy editor ARN: arn:aws:s3:::diksha.static.website

Type to add a new policy or edit an existing policy in the text area below.

```
{
    "Id": "Policy1583297551962",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "presigned url",
            "Action": [
                "s3:Get*"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:s3:::diksha.static.website/*",
            "Condition": {
                "StringEquals": {
                    "s3:authtype": "REST-QUERY-STRING"
                }
            },
            "Principal": "*"
        }
    ]
}
```

6.Mount S3 to an EC2 Instance

ANS:A S3 bucket can be mounted in a AWS instance as a file system known as S3fs. S3fs is a FUSE file-system that allows you to mount an Amazon S3 bucket as a local file-system. It behaves like a network attached drive, as it does not store anything on the Amazon EC2, but user can access the data on S3 from EC2 instance.

Filesystem in Userspace (FUSE) is a simple interface for userspace programs to export a virtual file-system to the Linux kernel. It also aims to provide a secure method for non privileged users to create and mount their own file-system implementations.

Step-1:- If you are using a ubuntu instance. Update the system.

Step-2:- Install the dependencies.

$ sudo apt-get install automake autotools-dev fuse g++ git libcurl4-gnutls-dev libfuse-dev libssl-dev libxml2-dev make pkg-config

Step-3:- Clone s3fs source code from git.

git clone https://github.com/s3fs-fuse/s3fs-fuse.git

Step-4:- Now change to source code directory, and compile and install the code with the following commands:

```
cd s3fs-fuse
./autogen.sh
./configure --prefix=/usr --with-openssl
make
sudo make install
```

```
ubuntu@ip-172-31-79-130:~$ cd s3fs-fuse
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ ./autogen.sh
--- Make commit hash file -------
--- Finished commit hash file ---
--- Start autotools -------------
configure.ac:30: installing './compile'
configure.ac:26: installing './config.guess'
configure.ac:26: installing './config.sub'
configure.ac:27: installing './install-sh'
configure.ac:27: installing './missing'
src/Makefile.am: installing './depcomp'
parallel-tests: installing './test-driver'
--- Finished autotools ----------
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ ./configure --prefix=/usr --with-openssl
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking target system type... x86_64-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
```

```
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ make
make  all-recursive
make[1]: Entering directory '/home/ubuntu/s3fs-fuse'
Making all in src
make[2]: Entering directory '/home/ubuntu/s3fs-fuse/src'
g++ -DHAVE_CONFIG_H -I. -I..  -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse -I/usr/
include/x86_64-linux-gnu -I/usr/include/libxml2    -g -O2 -Wall -D_FILE_OFFSET_B
ITS=64 -D_FORTIFY_SOURCE=2 -MT s3fs.o -MD -MP -MF .deps/s3fs.Tpo -c -o s3fs.o s3
fs.cpp

mv -f .deps/s3fs.Tpo .deps/s3fs.Po
g++ -DHAVE_CONFIG_H -I. -I..  -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse -I/usr/
include/x86_64-linux-gnu -I/usr/include/libxml2    -g -O2 -Wall -D_FILE_OFFSET_B
ITS=64 -D_FORTIFY_SOURCE=2 -MT curl.o -MD -MP -MF .deps/curl.Tpo -c -o curl.o cu
rl.cpp
mv -f .deps/curl.Tpo .deps/curl.Po
g++ -DHAVE_CONFIG_H -I. -I..  -D_FILE_OFFSET_BITS=64 -I/usr/include/fuse -I/usr/
include/x86_64-linux-gnu -I/usr/include/libxml2    -g -O2 -Wall -D_FILE_OFFSET_B
ITS=64 -D_FORTIFY_SOURCE=2 -MT cache.o -MD -MP -MF .deps/cache.Tpo -c -o cache.o
 cache.cpp
```

Step-5:- Check where s3fs command is placed in O.S. It will also tell you the installation is ok.

```
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ which s3fs
/usr/bin/s3fs
ubuntu@ip-172-31-79-130:~/s3fs-fuse$
```

Step-6:- Getting the access key and secret key.

Step-7 :- Create a new file in /etc with the name passwd-s3fs and Paste the access key and secret key in the below format :

Your_accesskey:Your_secretkey

```
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ touch /etc/passwd-s3fs
touch: cannot touch '/etc/passwd-s3fs': Permission denied
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ sudo !!
sudo touch /etc/passwd-s3fs
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ sudo vim /etc/passwd-s3fs
ubuntu@ip-172-31-79-130:~/s3fs-fuse$
```

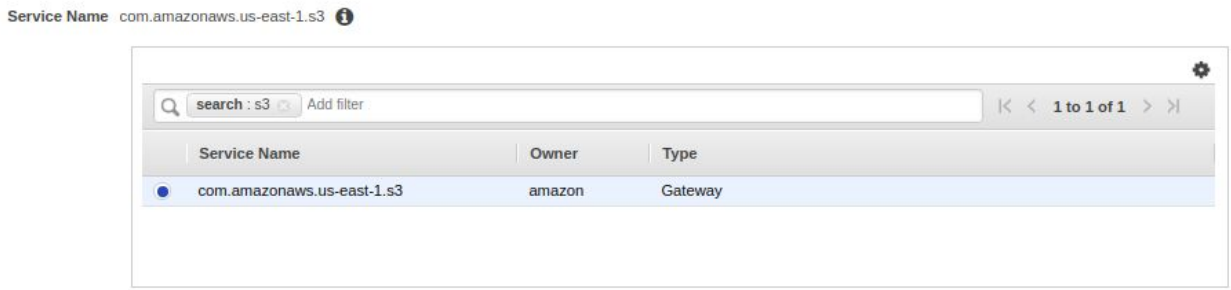
Step-8:- change the permission of file

```
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ sudo chmod 640 /etc/passwd-s3fs
ubuntu@ip-172-31-79-130:~/s3fs-fuse$
```

Step-9:- Now create a directory or provide the path of an existing directory and mount S3bucket in it.

```
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ sudo s3fs diksha.static.website mys3bucket/
-o use_cache=/tmp -o allow_other -o uid=1001 -o mp_umask=002 -o multireq_max=5 -o
 use_path_request_style -o url=https://s3-us-east-1.amazonaws.com
ubuntu@ip-172-31-79-130:~/s3fs-fuse$ df -Th
Filesystem     Type       Size  Used Avail Use% Mounted on
udev           devtmpfs   480M     0  480M   0% /dev
tmpfs          tmpfs       99M  744K   98M   1% /run
/dev/xvda1     ext4       7.7G  1.9G  5.8G  25% /
tmpfs          tmpfs      492M     0  492M   0% /dev/shm
tmpfs          tmpfs      5.0M     0  5.0M   0% /run/lock
tmpfs          tmpfs      492M     0  492M   0% /sys/fs/cgroup
/dev/loop0     squashfs    18M   18M     0 100% /snap/amazon-ssm-agent/1480
/dev/loop1     squashfs    90M   90M     0 100% /snap/core/8268
/dev/loop2     squashfs    92M   92M     0 100% /snap/core/8689
tmpfs          tmpfs       99M     0   99M   0% /run/user/1000
s3fs           fuse.s3fs  7.7G  1.9G  5.8G  25% /home/ubuntu/s3fs-fuse/mys3bucket
ubuntu@ip-172-31-79-130:~/s3fs-fuse$
```

***You can make an entry in /etc/rc.local to automatically remount after reboot. Find the s3fs binary file by "which" command and make the entry before the "exit 0"**



Step-10:- Check mounted s3 bucket.

$ df -Th

7.Change content type using S3

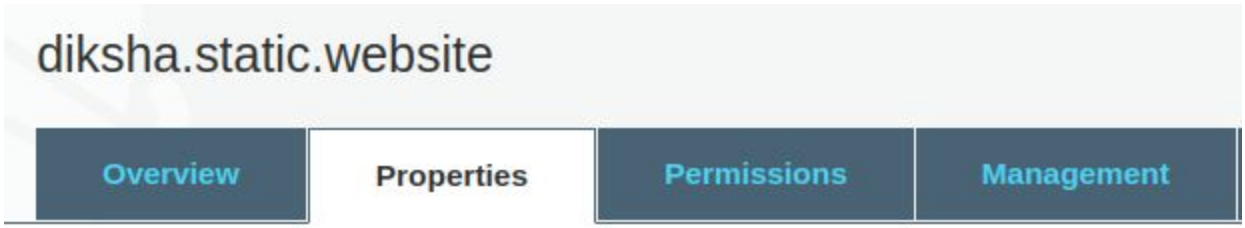
ANS:

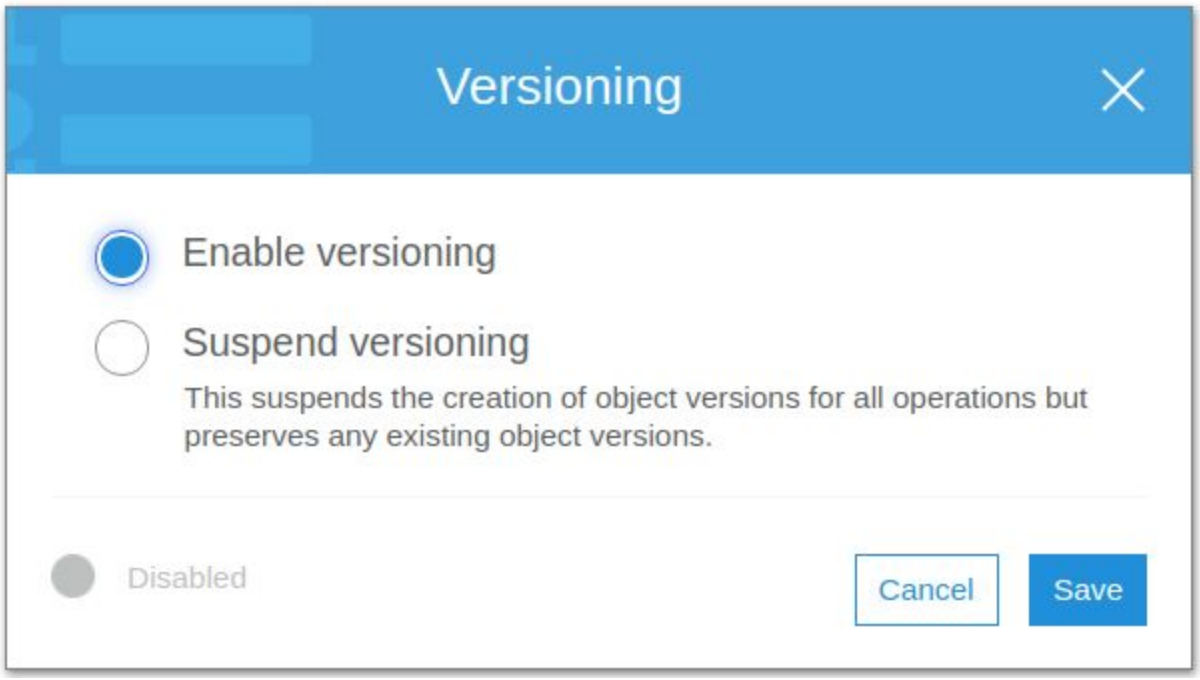
STEP 1: This is how you can set Content-Type for all files of type *.png

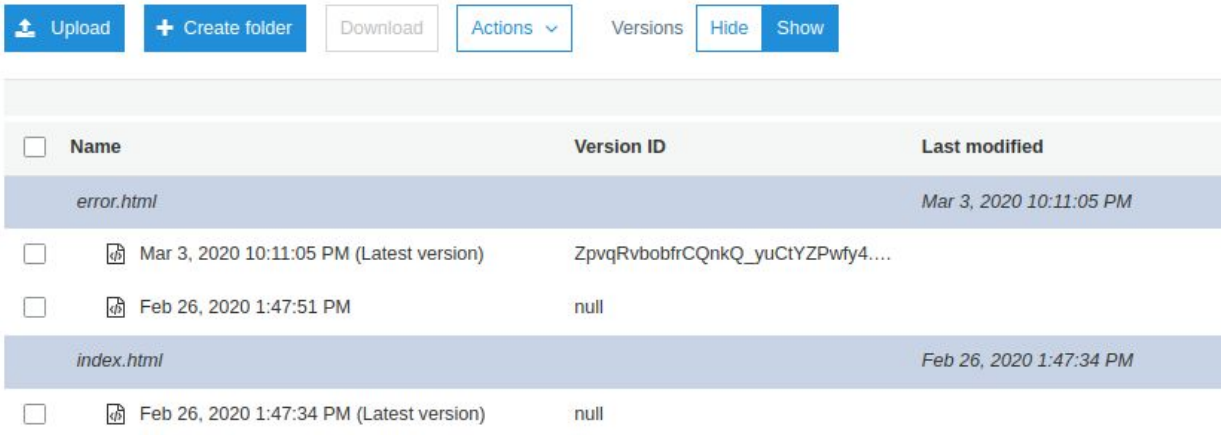
```
diksha@diksha:~$ aws s3api get-object --bucket s3diksha --key error.html data.tx
t
{
    "AcceptRanges": "bytes",
    "LastModified": "2020-03-04T04:58:20+00:00",
    "ContentLength": 9,
    "ETag": "\"fa56e451eb3f9c7e0c3c8116526689fb\"",
    "ContentType": "text/html",
    "Metadata": {}
}
diksha@diksha:~$ aws s3 cp        s3://s3diksha/        s3://s3diksha/        --
exclude '*'        --include '*.html'        --no-guess-mime-type        --conte
nt-type="text/plain"        --metadata-directive="REPLACE"        --recursive
copy: s3://s3diksha/error.html to s3://s3diksha/error.html
diksha@diksha:~$ aws s3api get-object --bucket s3diksha --key error.html data.tx
t
{
    "AcceptRanges": "bytes",
    "LastModified": "2020-03-04T05:04:24+00:00",
    "ContentLength": 9,
    "ETag": "\"fa56e451eb3f9c7e0c3c8116526689fb\"",
    "ContentType": "text/plain",
    "Metadata": {}
}
diksha@diksha:~$
```

8.Retrieve previous version of S3.Enabling versioning

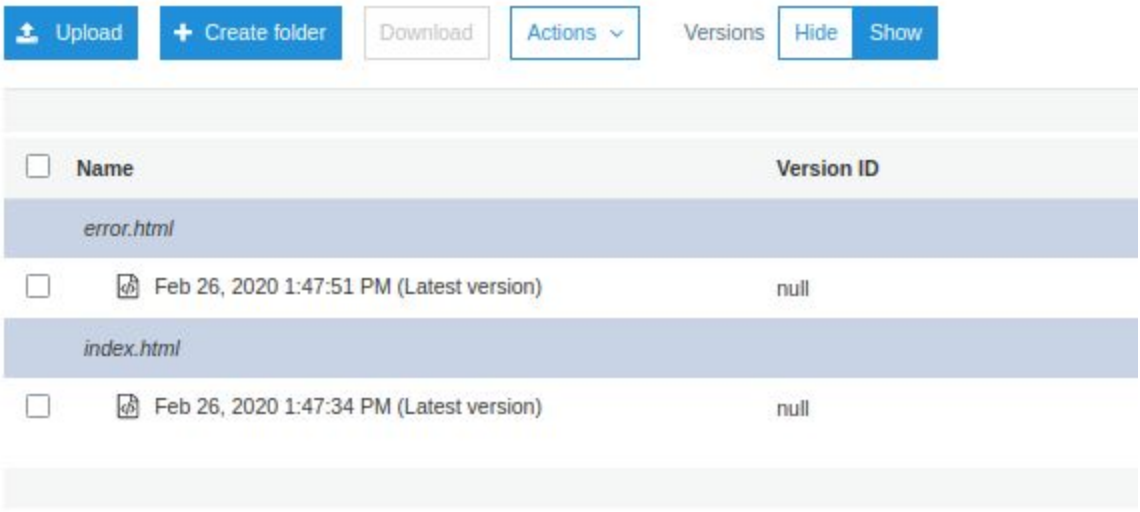
ANS:







Delete the latest version to retrieve previous version.



9.What is VPC endpoint ?Enable it.

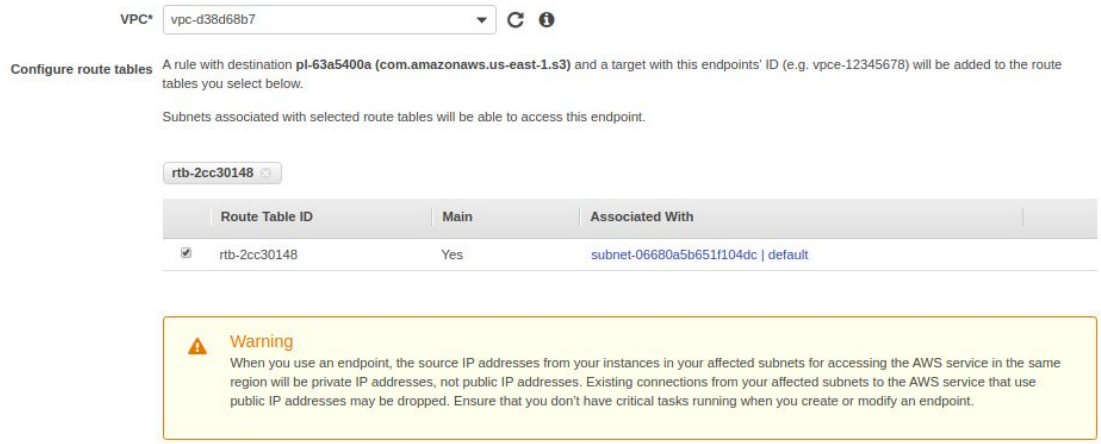
ANS: VPC endpoint enables a user to connect with AWS services that are outside the VPC through a private link. VPC endpoints use AWS PrivateLinks in the backend with which users will be able to connect to AWS services without using public IP's. Thus the traffic will not leave the Amazon network. AWS PrivateLinks are highly available, redundant and scalable technology.

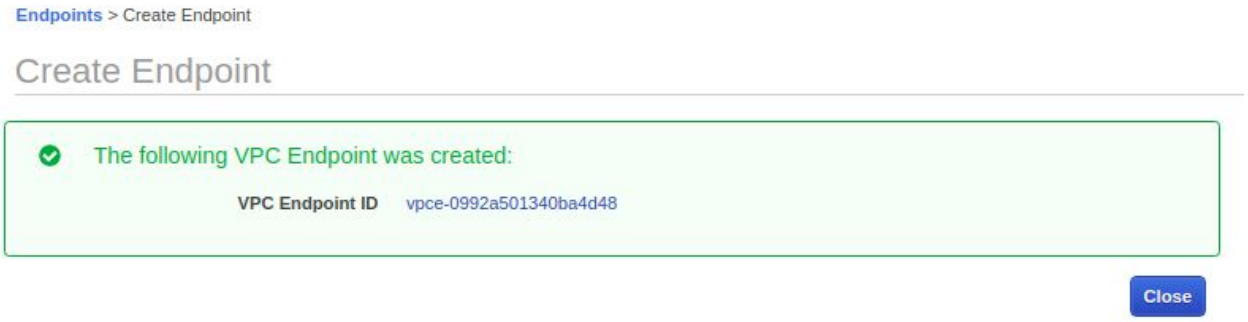
There are two types of VPC endpoints Interface Endpoints and Gateway Endpoints:

Interface Endpoints are Elastic Network Interfaces (ENI) with private IP addresses. ENI will act as the entry point for the traffic that is destined to a particular service. Services such as Amazon CloudWatch Logs, Amazon SNS, etc. are supported.

Gateway endpoints is a gateway targeted for a specific route in the routCreate another role which has the policy to assume the previous Roleeing table. They can be used to route traffic to a destined AWS service. As of now, Amazon S3 and DynamoDB are the only services that are supported by gateway endpoints.

STEP 1: Go to VPC > Endpoint > Create Endpoint and mention the service eg. S3





10.CORS, Enabling CORS for 2 specific website

ANS: Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell browsers to give a web application running at one origin, access to selected resources from a different origin. A web application executes a cross-origin HTTP request when it requests a resource that has a different origin (domain, protocol, or port) from its own.
An example of a cross-origin request: the front-end JavaScript code served from `https://domain-a.com` uses `XMLHttpRequest` to make a request for `https://domain-b.com/data.json`.

<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration
xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<CORSRule>
<AllowedOrigin>https://website-1.com</AllowedOrigin>

<AllowedOrigin>https://website-2.com</AllowedOrigin>
<AllowedMethod>GET</AllowedMethod>
<AllowedMethod>POST</AllowedMethod>
<AllowedMethod>PUT</AllowedMethod>
<MaxAgeSeconds>3000</MaxAgeSeconds>
<AllowedHeader>Authorization</AllowedHeader>
</CORSRule>
</CORSConfiguration>