**Assessment 16 – Doubt Resolving**

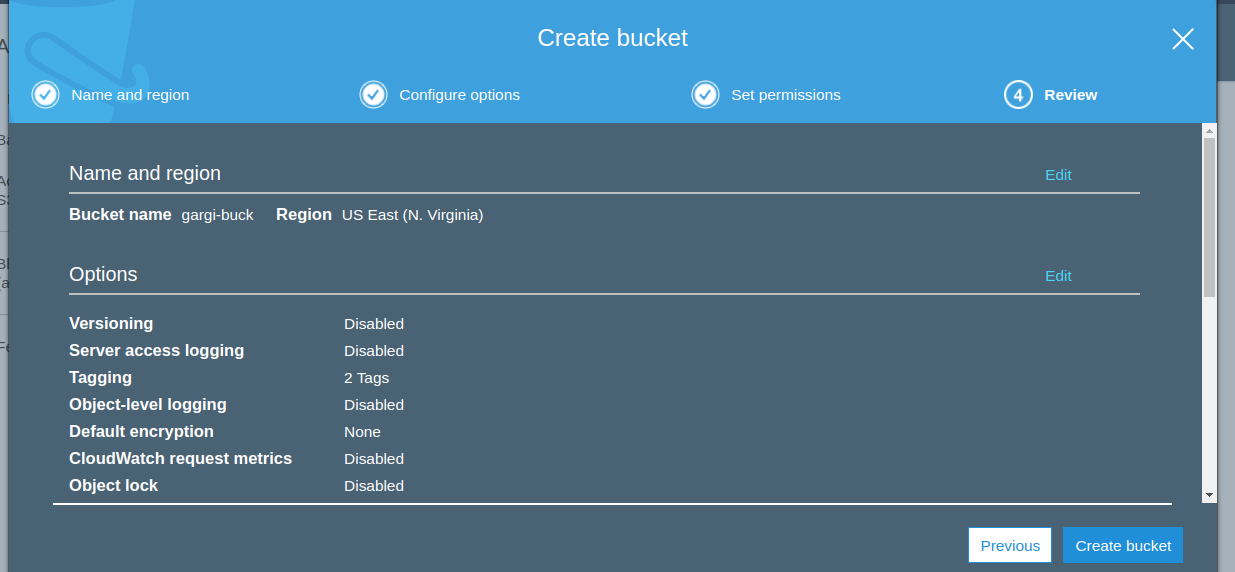
**Trainee Name : Gargi Sharma**

**Mentor Name : Mr. Akansh Gupta**

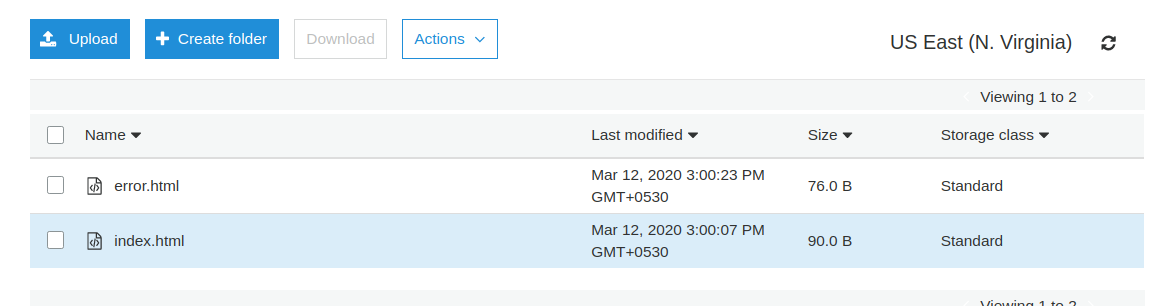
**College Name : UPES**

1. **Static website hosting using s3(what is index and error page)**

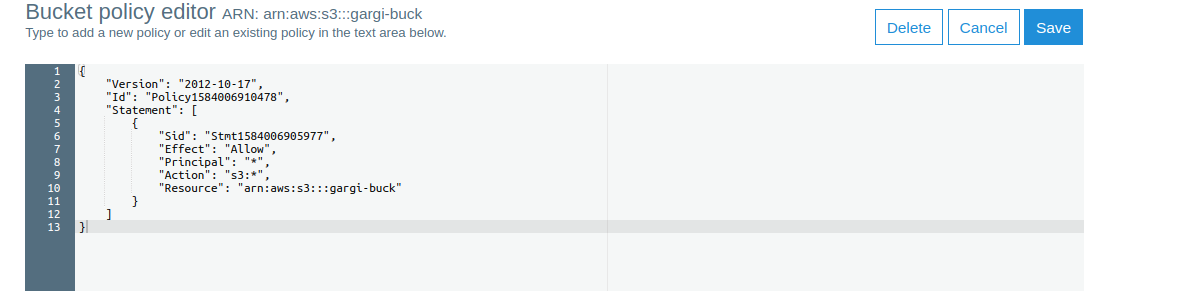
Make a public domain with the domain name you want



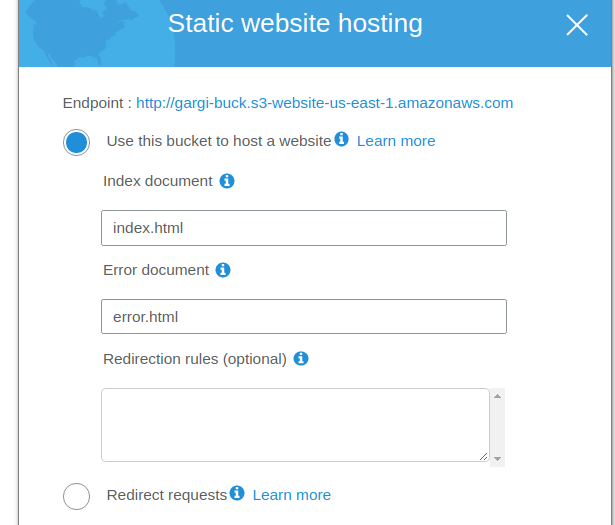
Now upload index.html and error.html in the bucket.



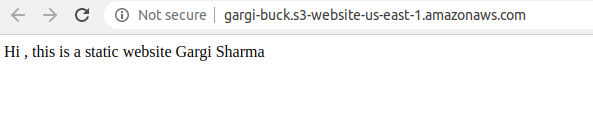
Make the bucket public(go to bucket and edit permissions)



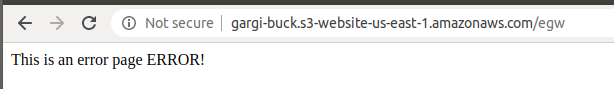
Turn static website hosting on:



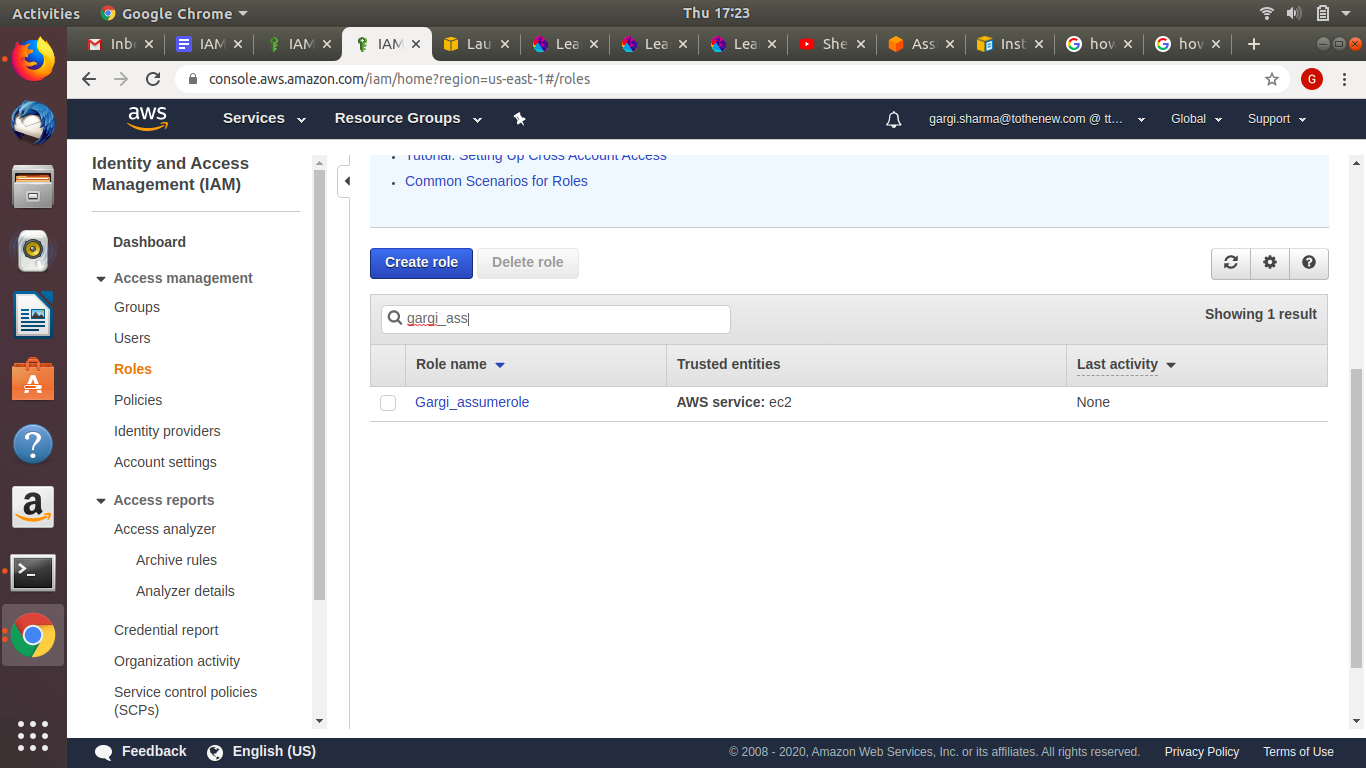
Make the objects public and then Click the URL here.



Now enter the wrong URL



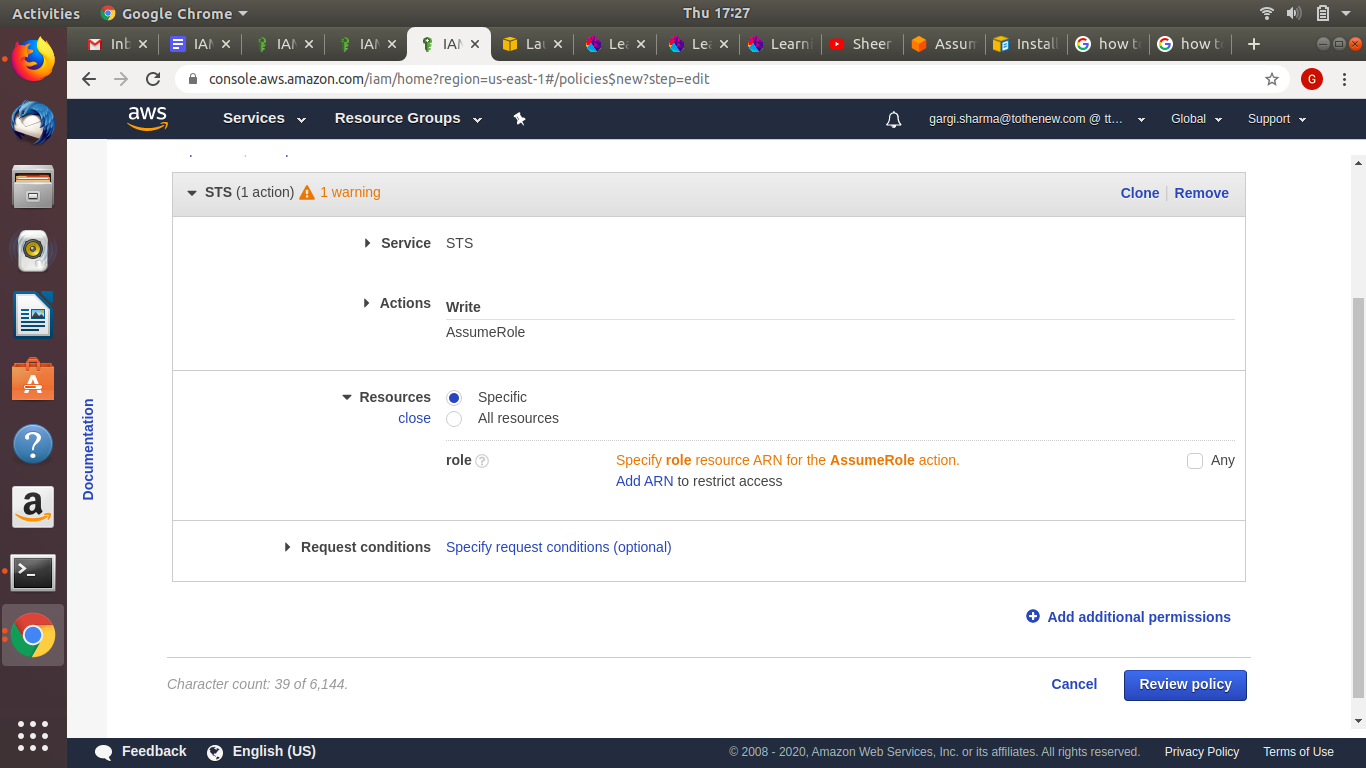
1. **Create an assume role to access s3 using ec2.**

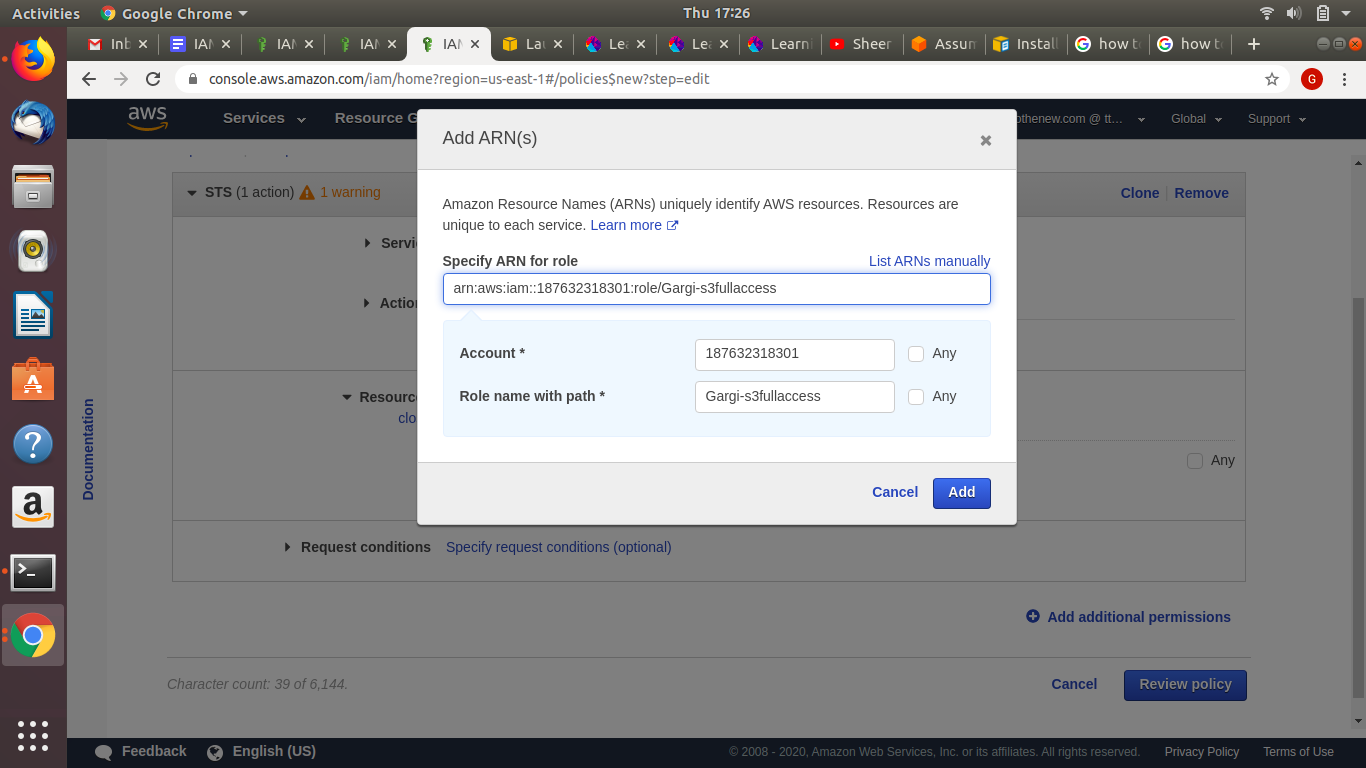
Create a new role****

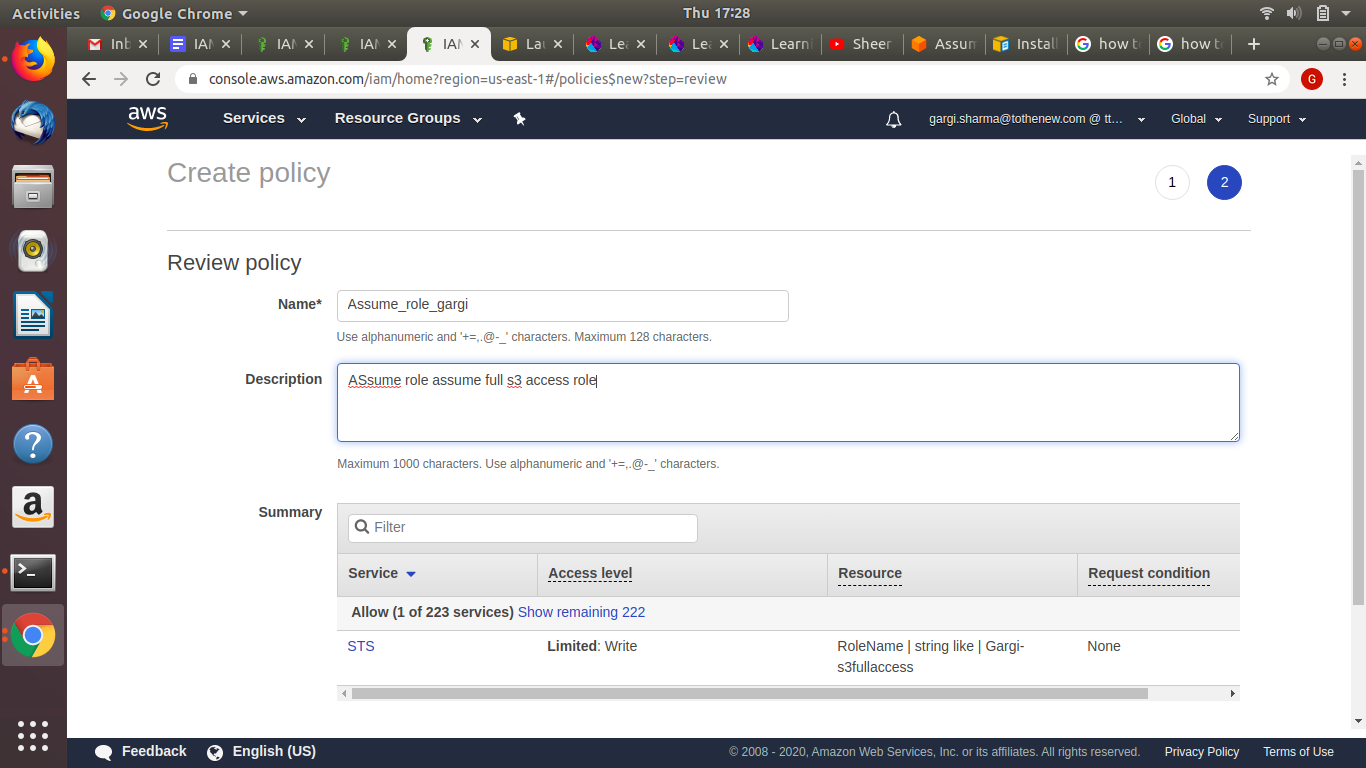
Create a new policy

Select service STS and action assume role

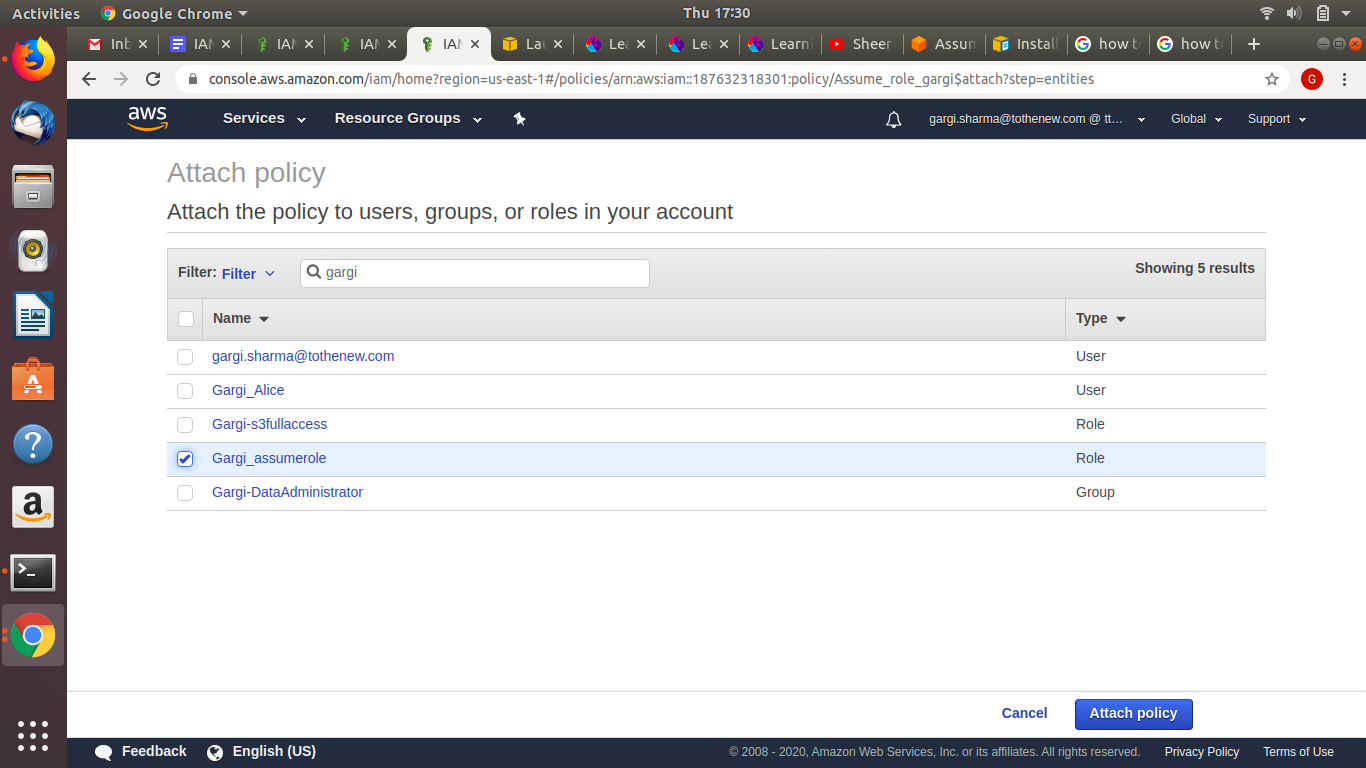
Go to resources(specific) and Copy the ARN of s3 full access and paste

****

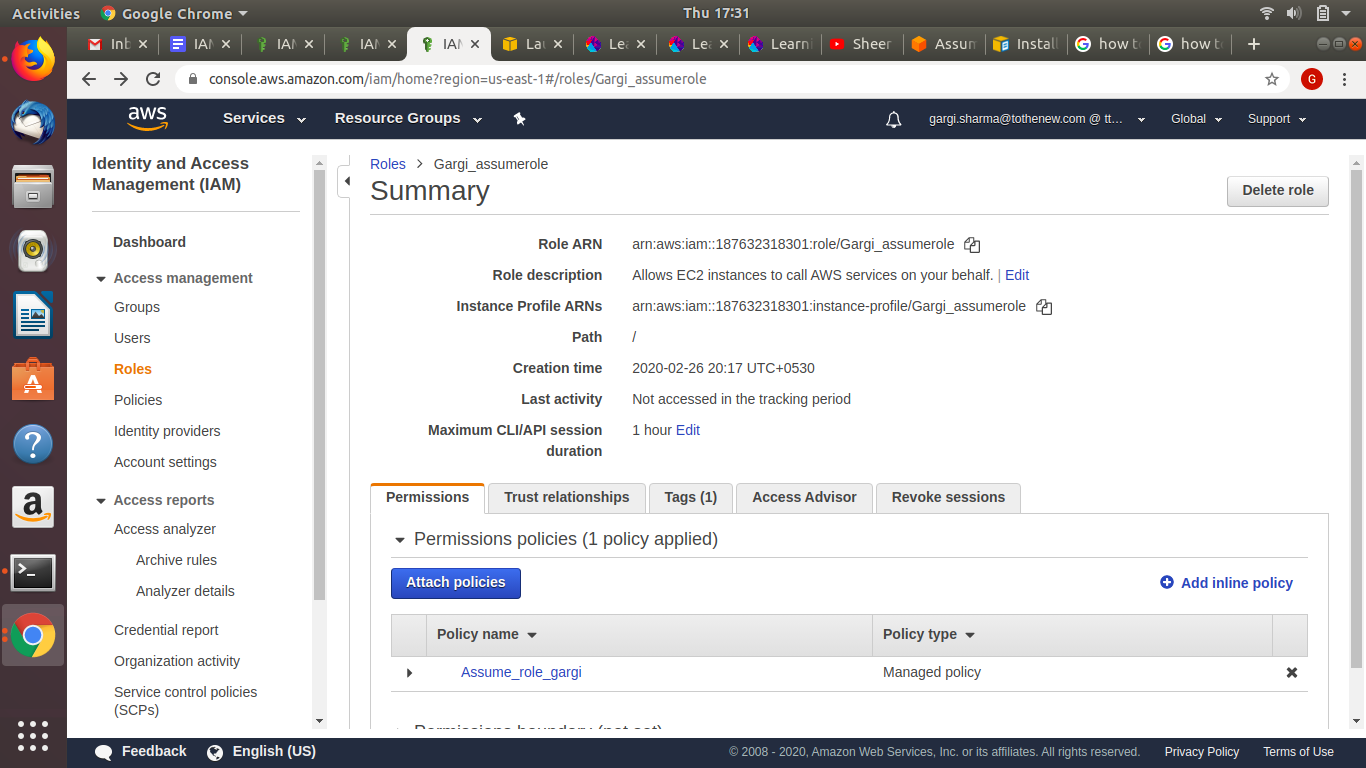
****

****

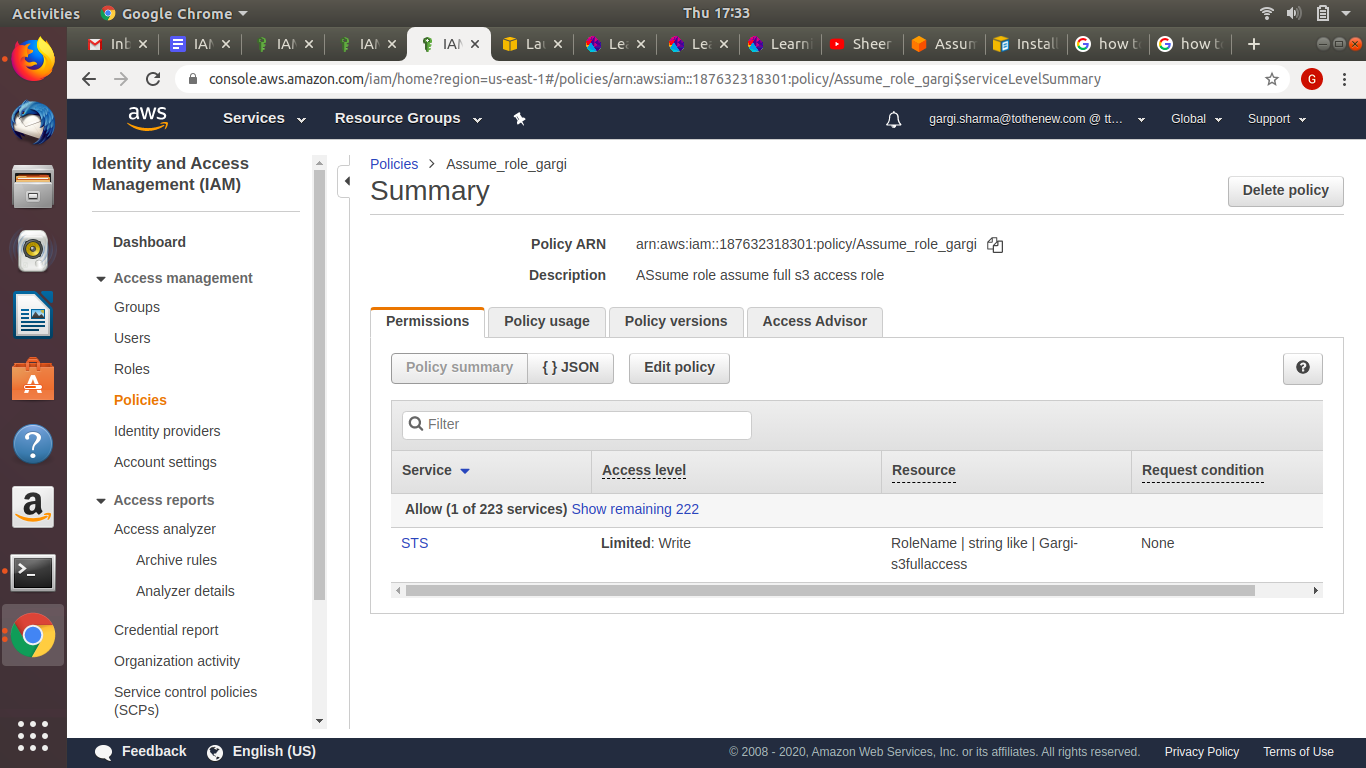
Attach the policarn:aws:iam::187632318301:role/arn:aws:iam::187632318301:role/arn:aws:iam::187632318301:role/y to the new role created

****

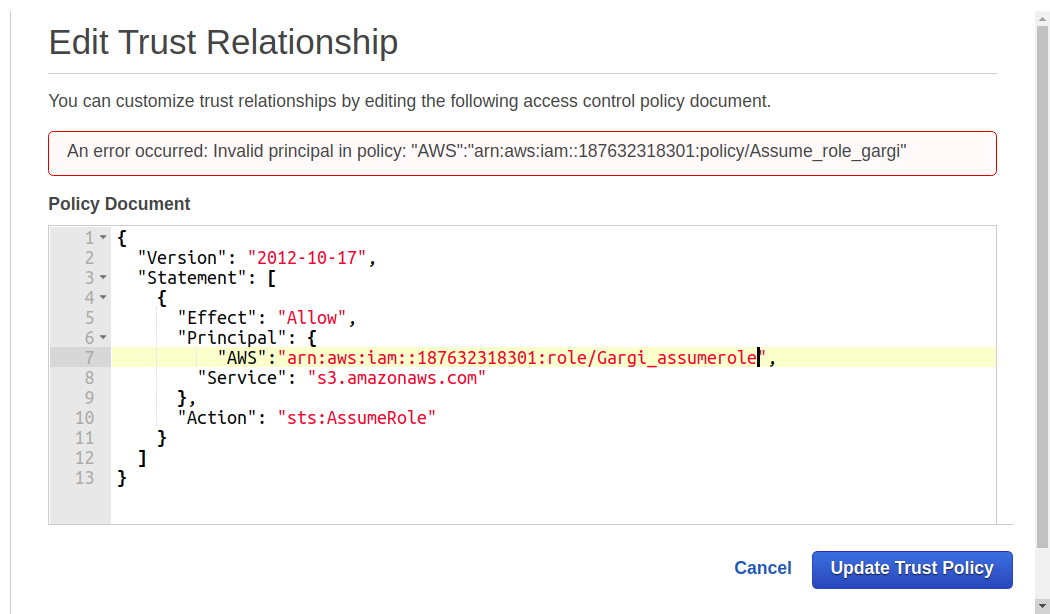
Now open the newly created role and check for the assume role

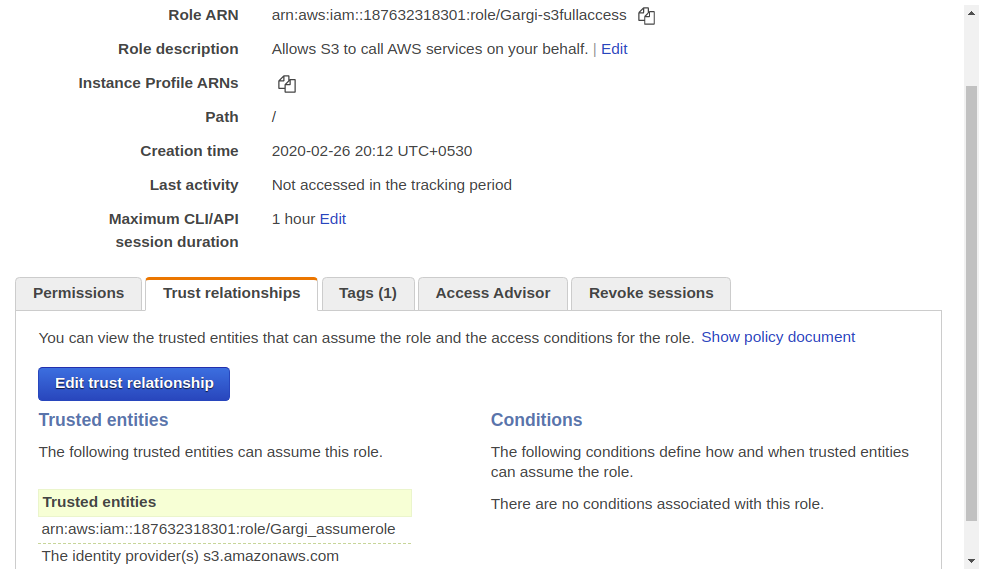
****

Go inside Assume\_role\_gargi(policy name)

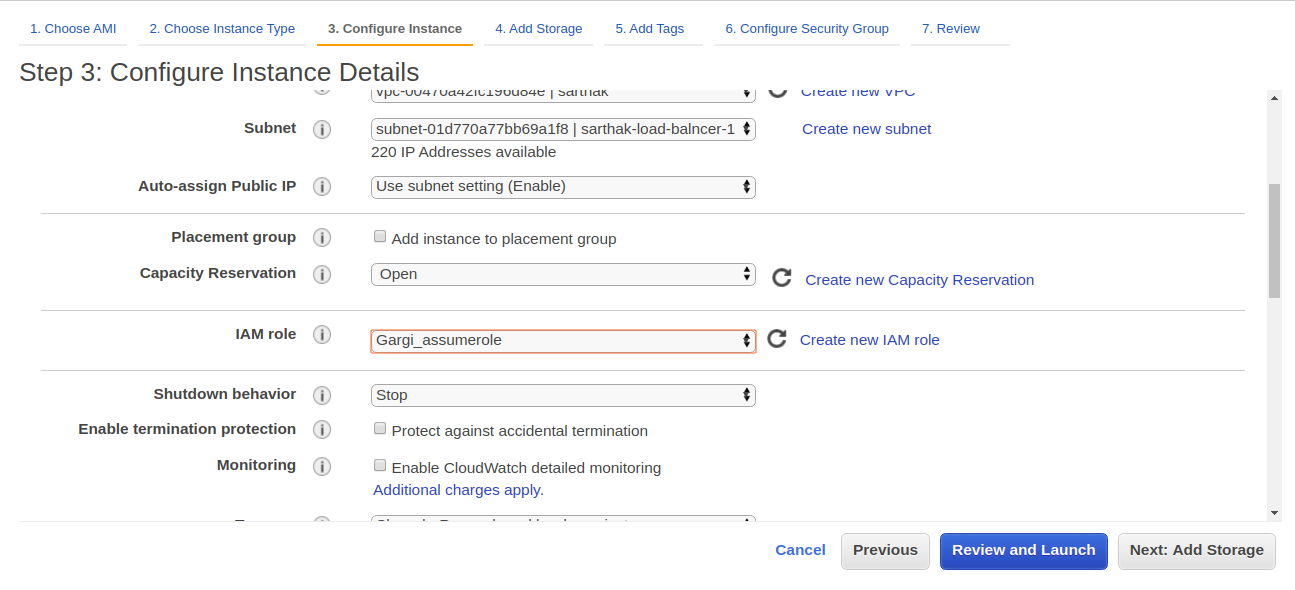
****

Go to the newly created role(Gargi\_assumerole) and copy the ARN. Now go to the old role(GargiS3fullaccess) and edit trust relationships. Then paste the ARN as follows.

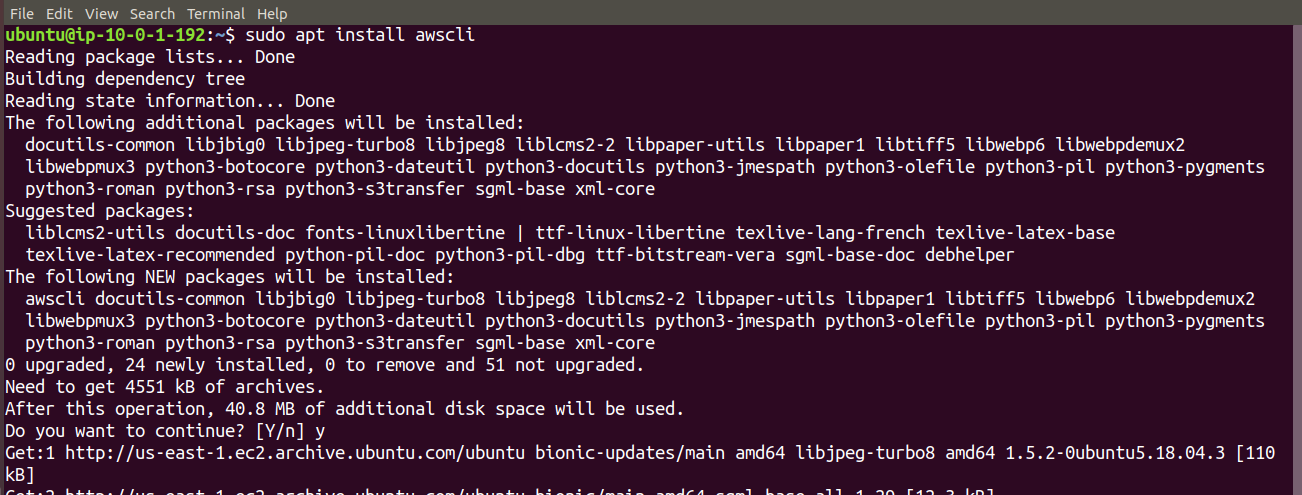
****

****

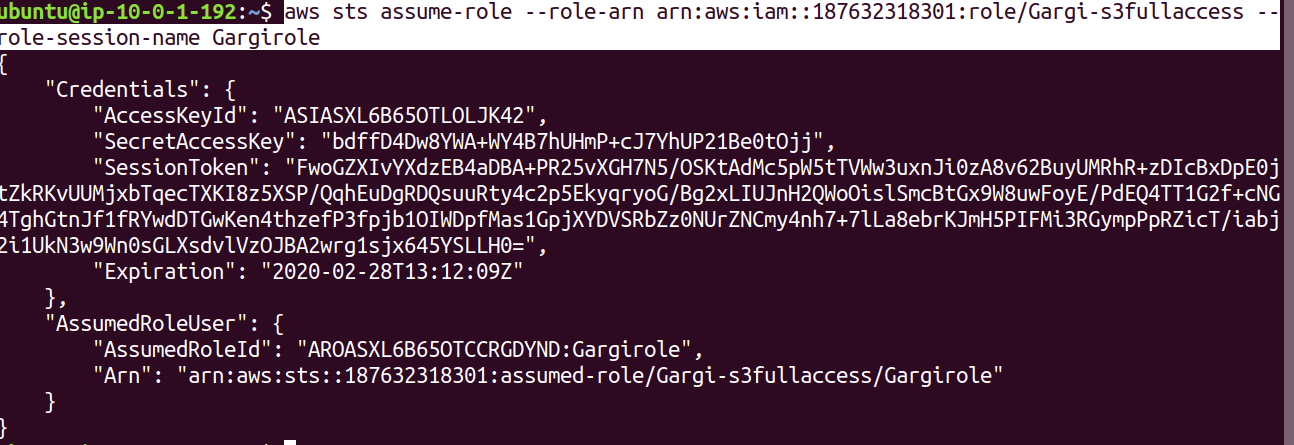
Create a new instance and then attach the new role(Gargi\_assume\_role)

****

SSh into the instance and install awscli

****

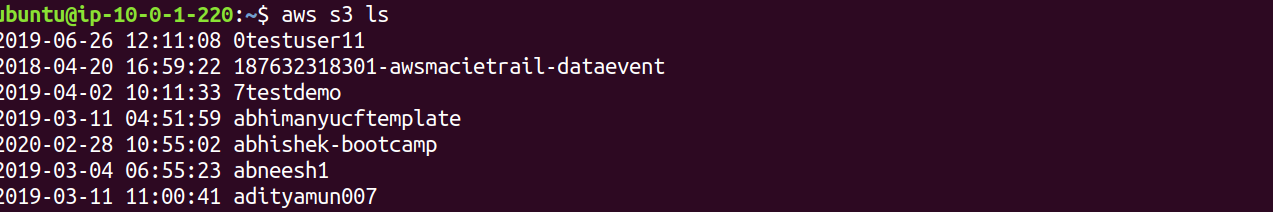
Now execute the following command :aws sts assume-role --role-arn arn:aws:iam::187632318301:role/Gargi-s3fullaccess --role-session-name Gargirole to generate the sts token.

****

Once the token is generated, export variables:

****

Now we can execute the “aws s3 ls” command to list all buckets.

****

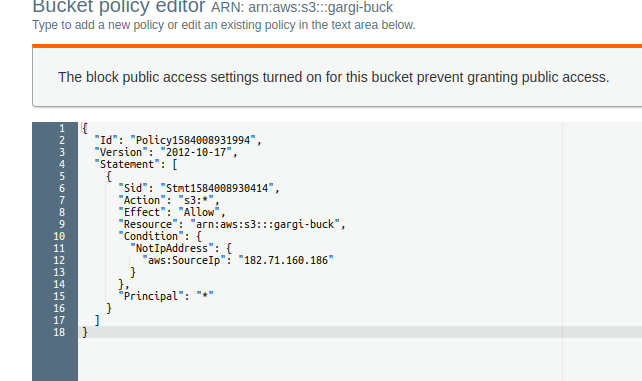
1. **Block s3 access on the basis of**

**i. IP**

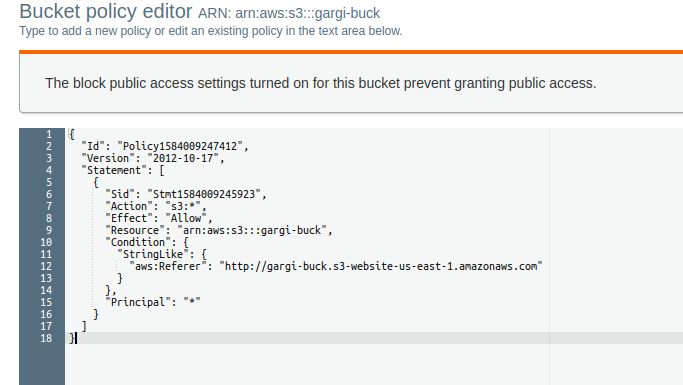
**ii. Domain**

**iii. Pre-signed URL(Time based)**

Edit the bucket policy (based on IP) using the policy generator tool.



Edit the bucket policy(based on domain)



Edit the bucket policy(based on pre-signed URL-time based)

A presigned URL is a URL that you can provide to your users to grant temporary access to a

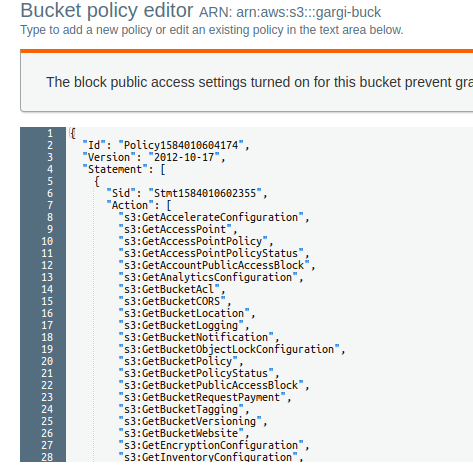
specific S3 object.

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.





1. **ACL, Bucket policy, IAM Policy.**

**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 Policy**: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, whereas 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’.

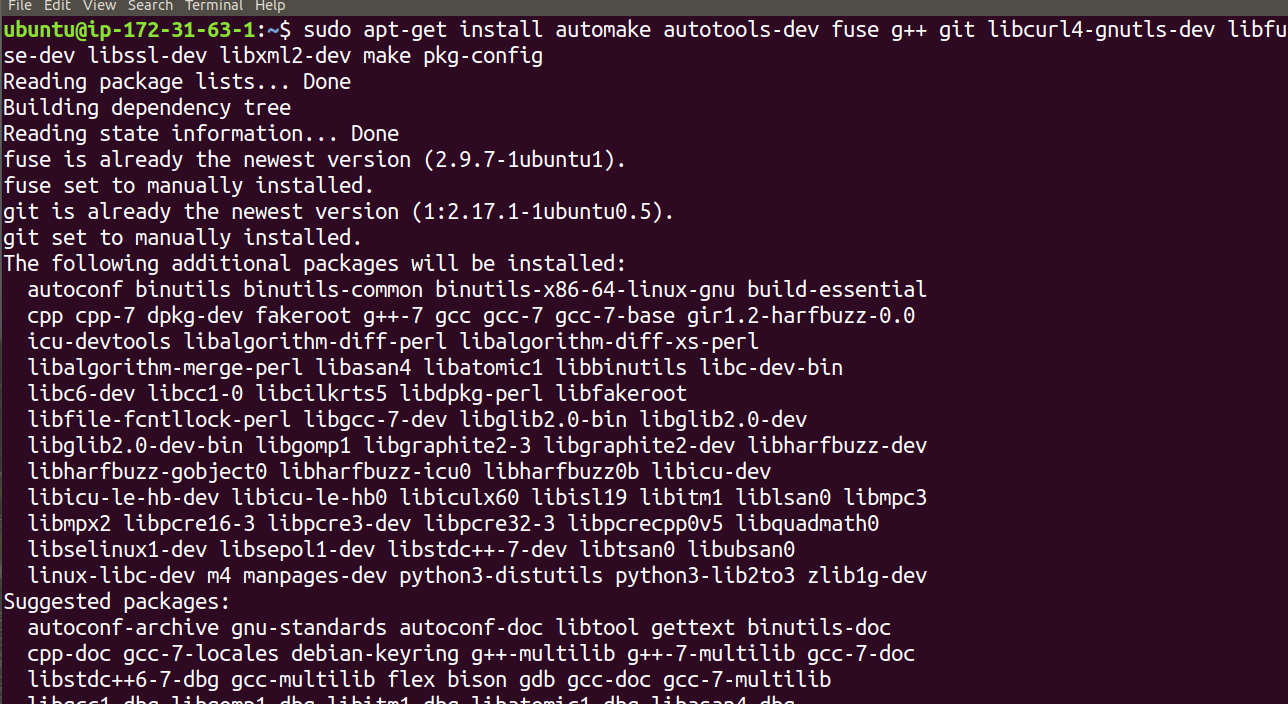
**IAM Policy:** A policy is an entity that, when attached to an identity or resource, defines their permissions. A policy that is attached to an identity in IAM is known as an identity-based policy.Identity-based policies can include AWS managed policies, customer managed policies, and inline policies. AWS managed policies are created and managed by AWS. You can use them,but you can&#39;t manage them. An inline policy is one that you create and embed directly to an IAM group, user, or role. Inline policies can be reused on other identities or managed outside of the identity where it exists.

1. Mount S3 to an EC2 instance.

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 users 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.

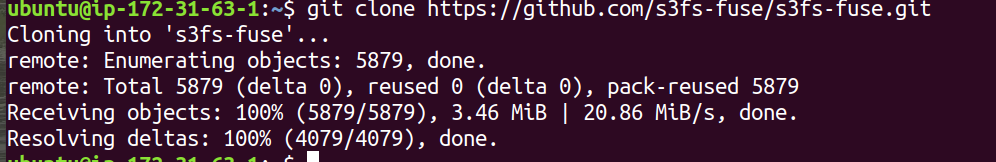
Install all the dependencies

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



Clone S3fs sourced code from git.

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



Now change the source code,directory, and compile and install the code

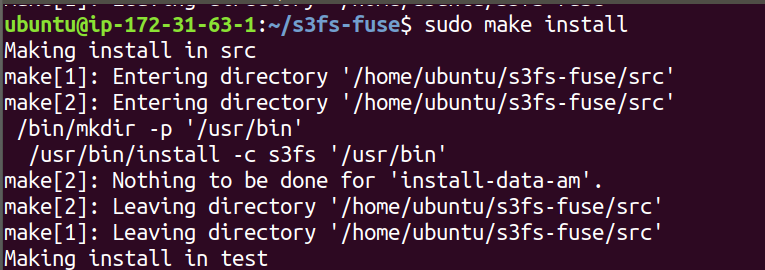
cd s3fs-fuse

./autogen.sh

./configure --prefix=/usr --with-openssl

Make

sudo make install

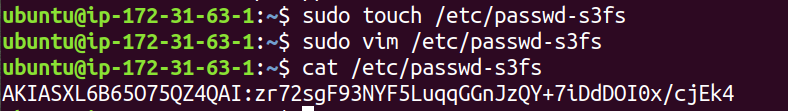


Now check where s3fs is placed in OS and if the installation is ok

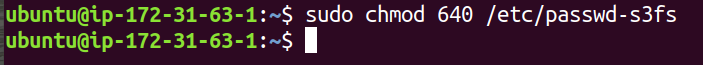
which s3fs



Create a role giving EC2 full access for s3 and attach it to the instance.

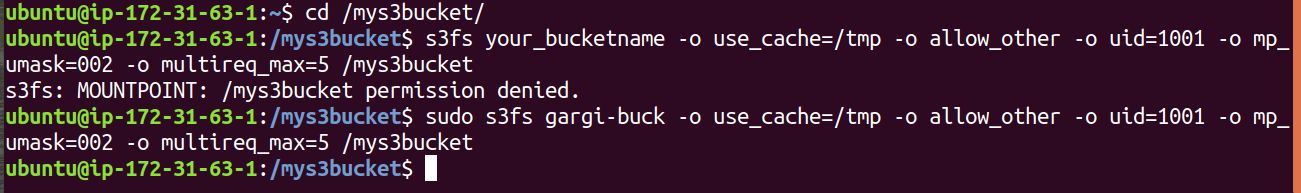
Now get the access key and secret key . Create a file in /etc and paste the access key and secret key in that file

Change the permission of the file.

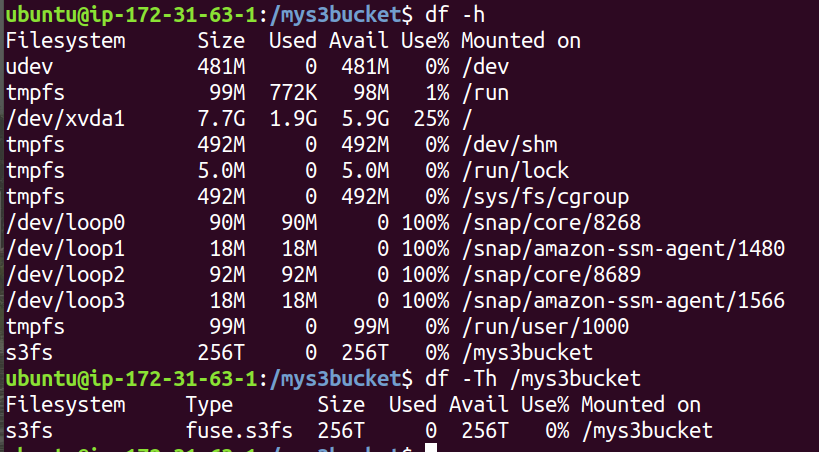


Now create a directory and mount the s3 bucket there.

Mkdir mys3bucket



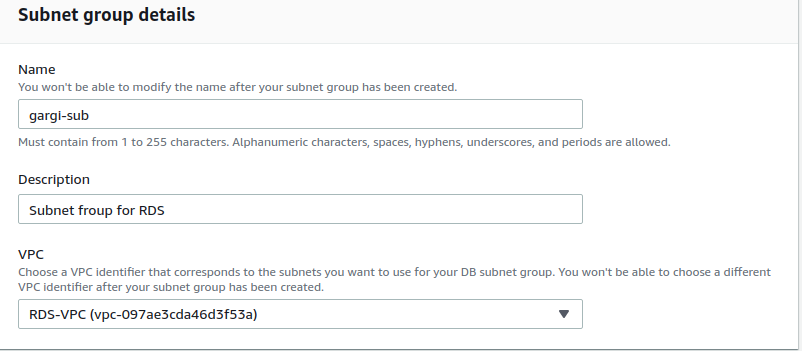
Now check if the bucket is mounted by “df -h”

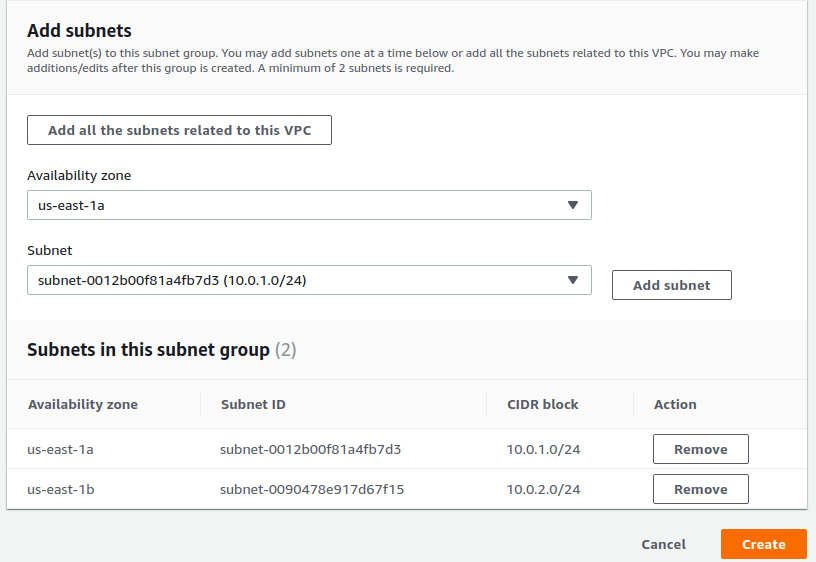


1. Create RDS subnet and launch RDS instance, what is parameter group and option

Group?

Go to Amazon RDS, build a db-subnet group



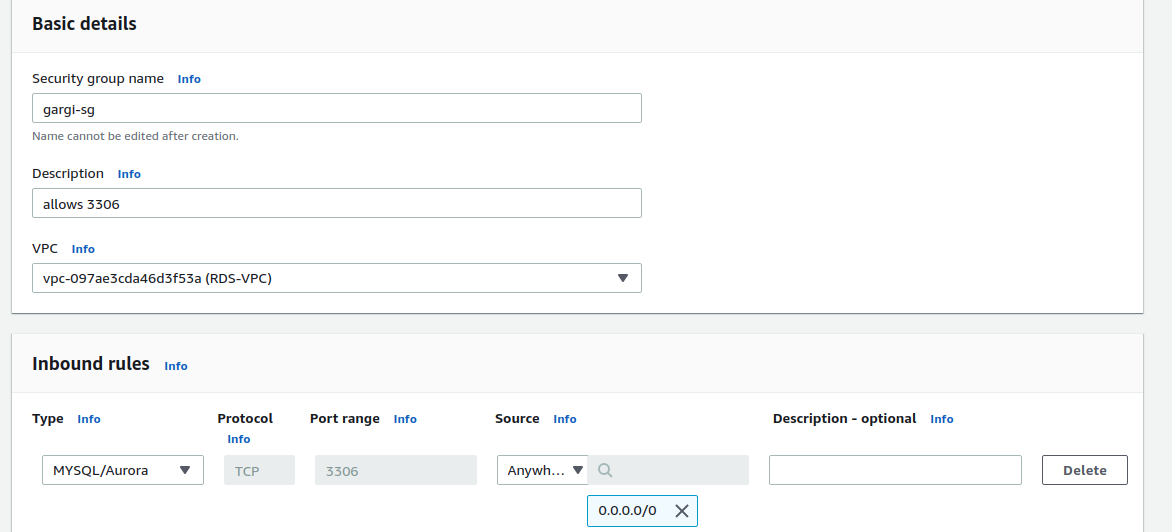


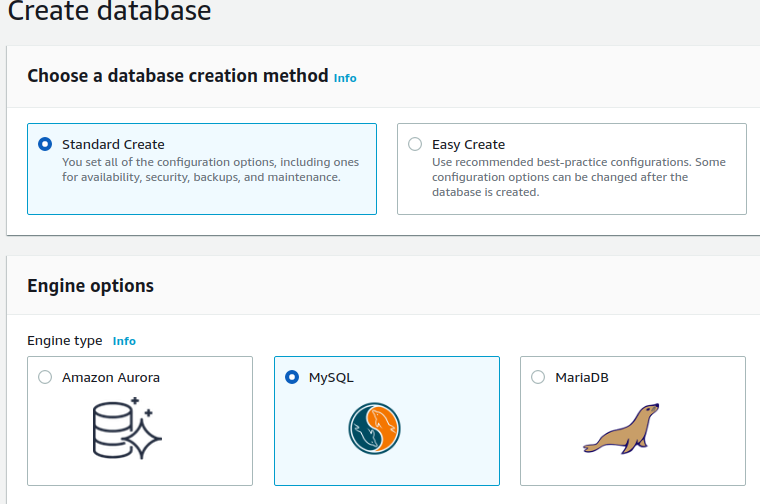
Create a VPC Security Group: Before you create your DB instance, you must create a VPC

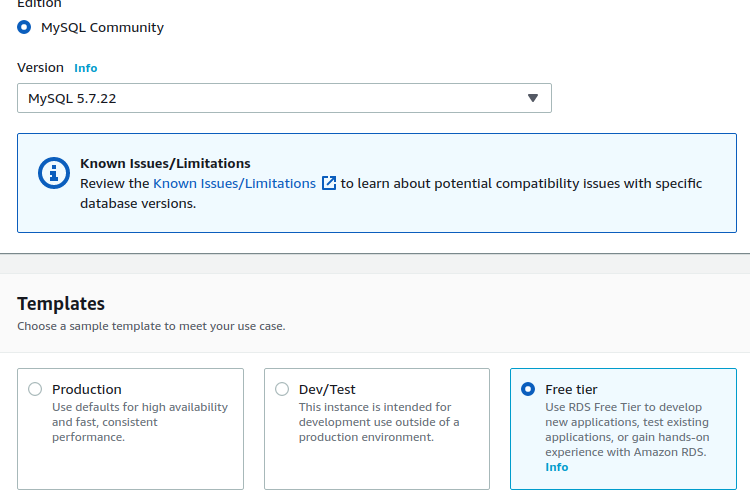
security group to associate with your DB instance. Choose the security group you created and edit inbound rules.

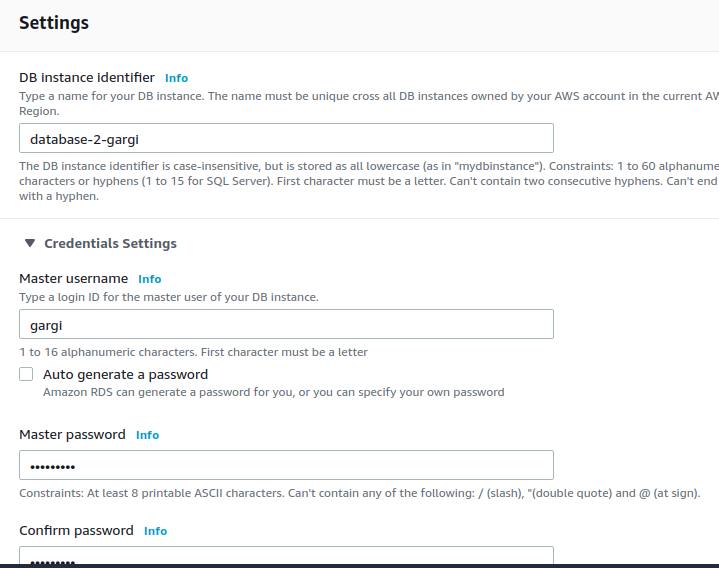
Set the following values for your new inbound rule to allow MySQL traffic on port 3306 from

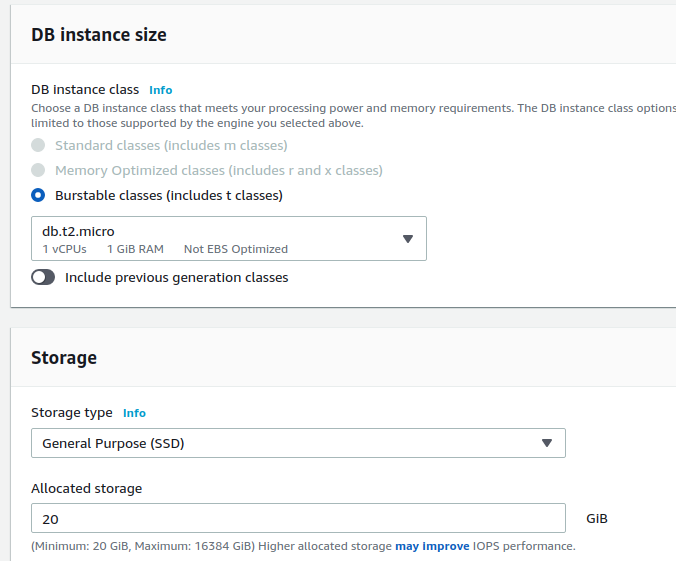
your EC2 instance. If you do this, you can connect from your web server to your DB instance to store and retrieve data from your web application to your database.

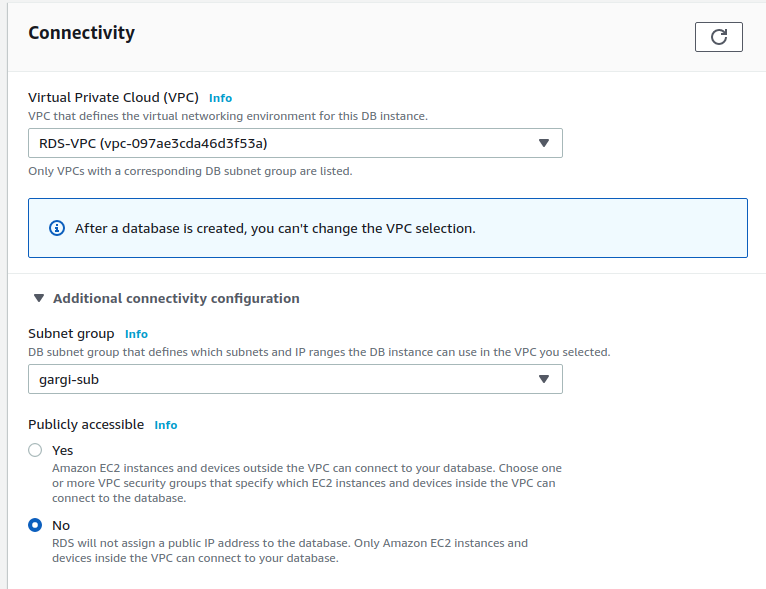


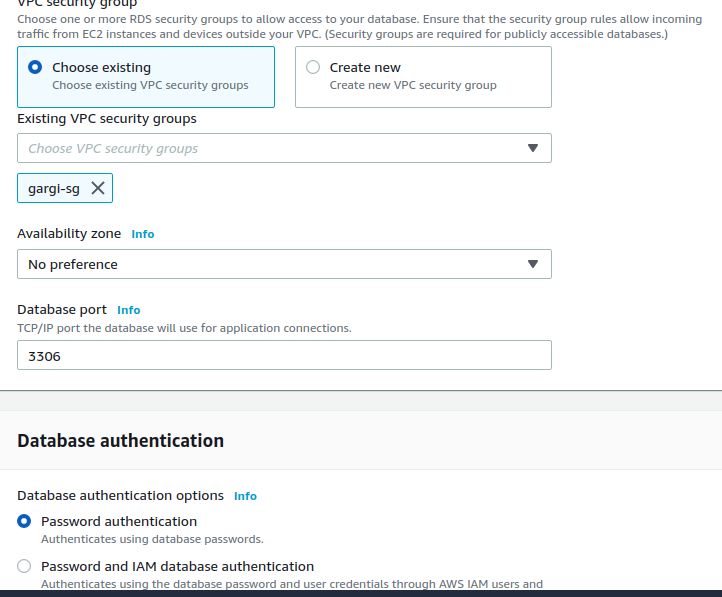












**Parameter group :**For AWS RDS instances, you manage your database engine configuration

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

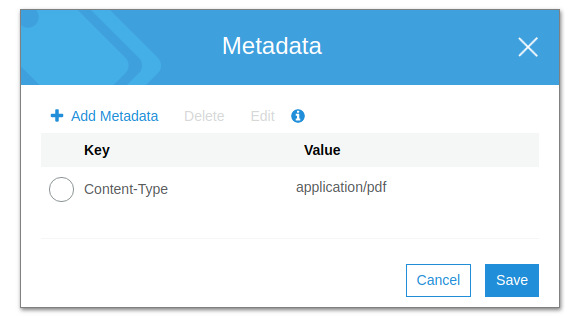
works. When you associate a DB instance with an option group, the specified options and

option settings are enabled for that DB instance.

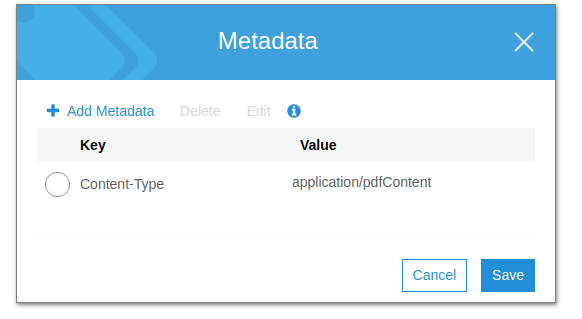
1. Change content type using S3

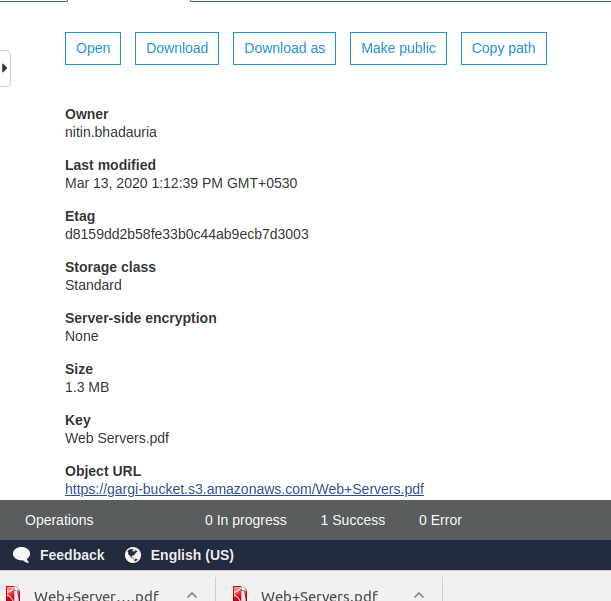
Goto Properties -> Metadata of S3 Object -> Add metadata -> Select content type

* application/pdf -> to render the pdf in the browser
* application/pdfContent -> to download the file



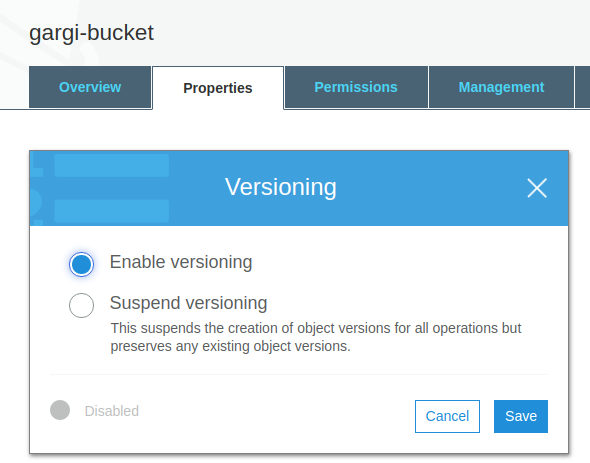
Delete the content type and add a new content type. The new content type will download the pdf file when clicked on the URL of the object.

Also make the object “webserver” public. Now when you click on the object URL, the file is downloaded.

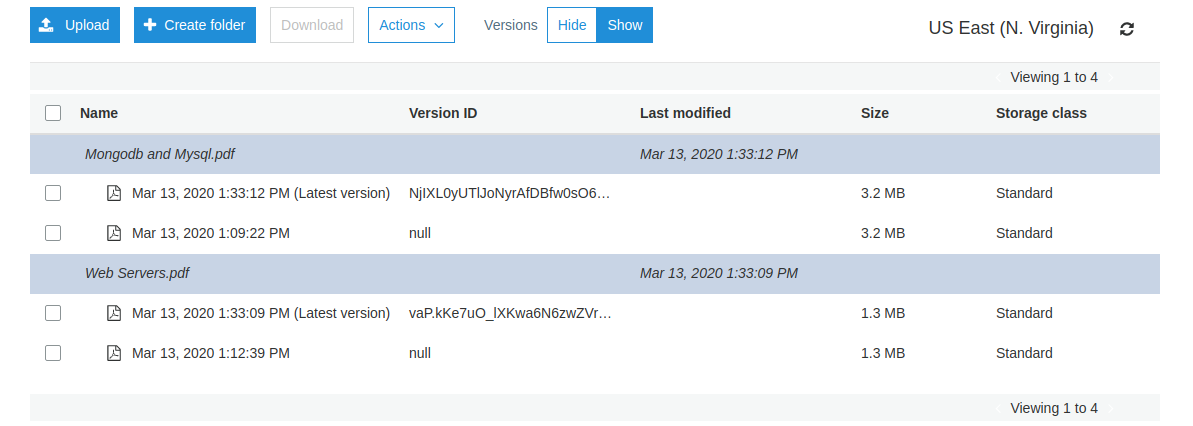


1. Retrieve previous version of S3 (enable versioning).

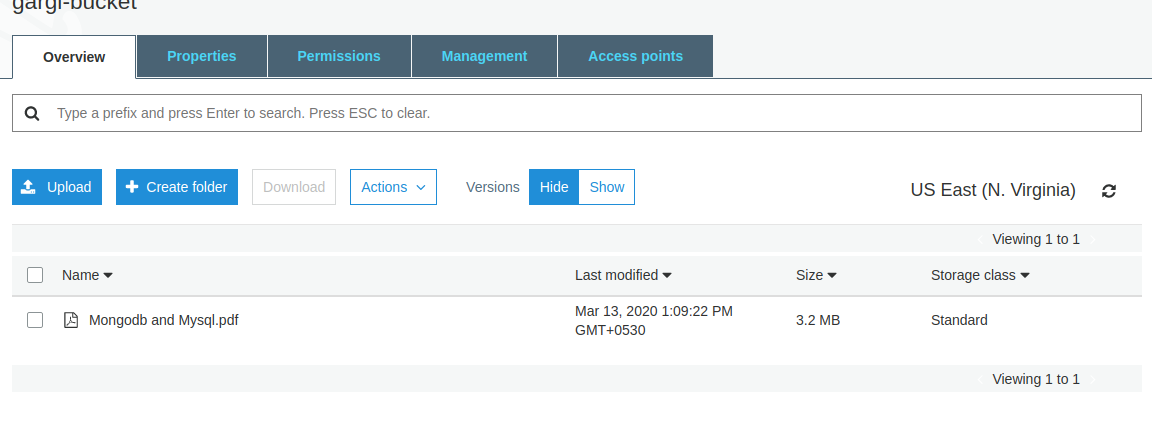
Enable versioning in the bucket properties.



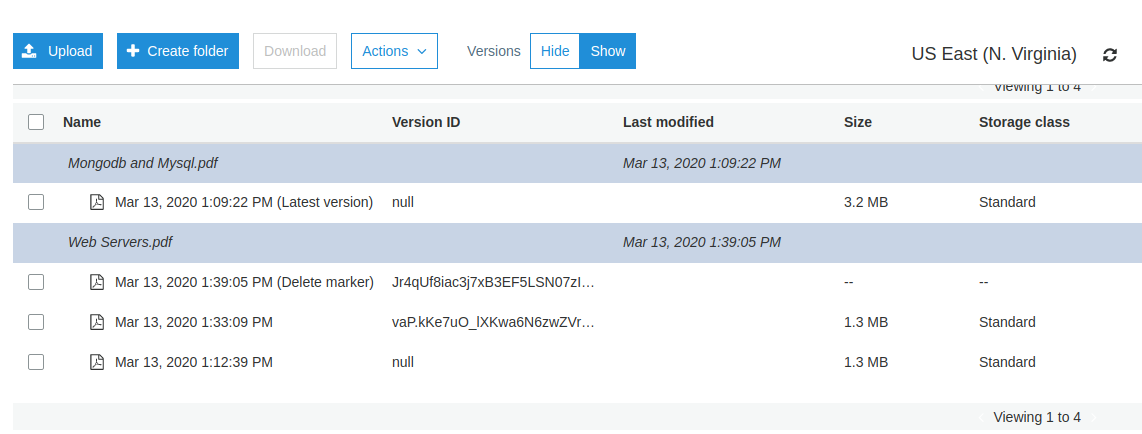
We can now see two versions of the object in the bucket.



Delete the web servers.pdf to check versioning.



Now show versions and we can see all the previous versions of webserver.pdf . Click and delete the version you want. The rest will be retrieved.



1. **S3 VPC endpoint.**

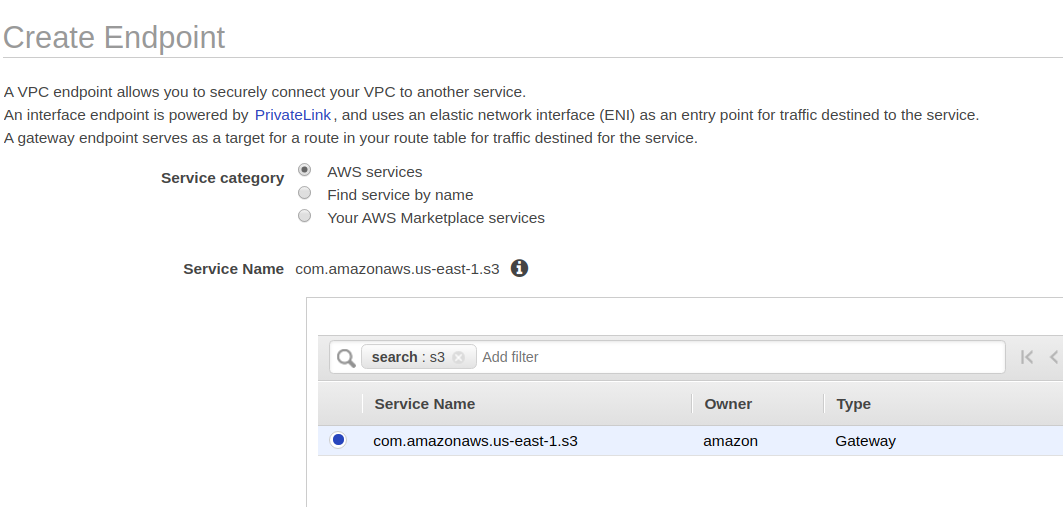
VPC endpoint enables a user to connect with AWS services that are outside the VPC through a private link. VPC endpoints use AWS PrivateLinks in the backend with which users will be able to connect to AWS services without using public IP’s. Thus the traffic will not leave the Amazon network. AWS PrivateLinks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

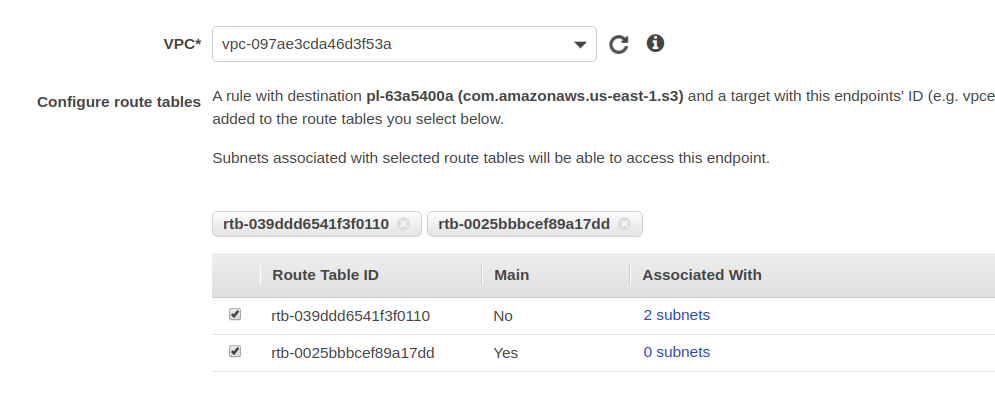
There are two types of VPC endpoints Interface Endpoints and Gateway Endpoints:

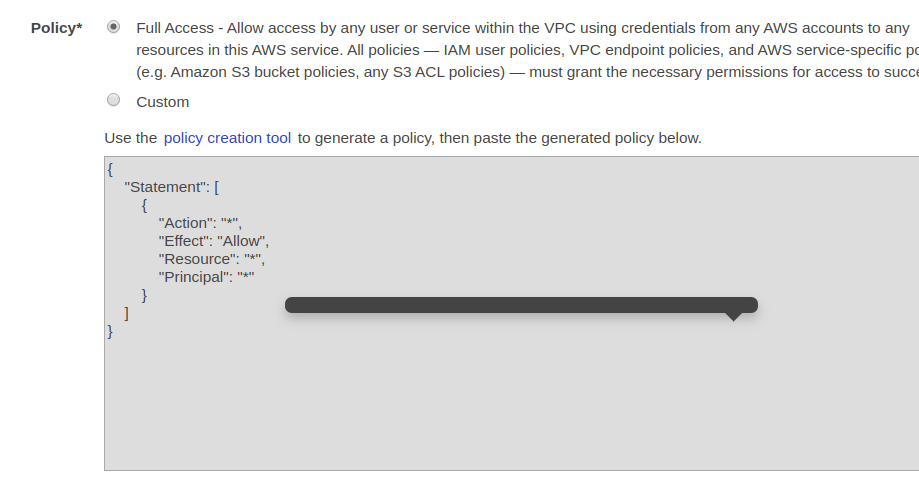
**Interface Endpoints** are Elastic Network Interfaces (ENI) with private IP addresses. ENI will act as the entry point for the traffic that is destined to a particular service. Services such as Amazon CloudWatch Logs, Amazon SNS, etc. are supported.

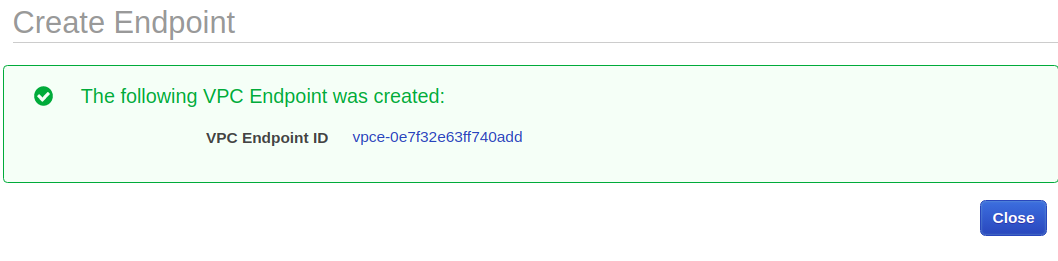
**Gateway endpoints** is a gateway targeted for a specific route in the routCreate another role which has the policy to assume the previous Roleeing table. They can be used to route traffic to a destined AWS service. As of now, Amazon S3 and DynamoDB are the only services that are supported by gateway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

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



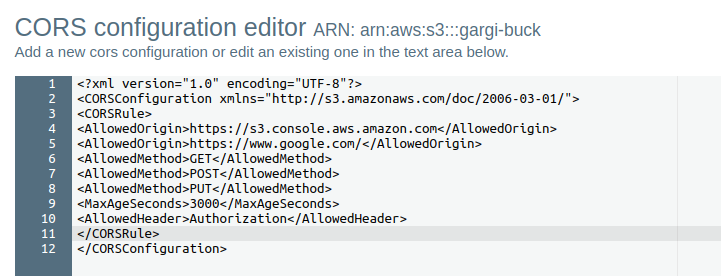






1. **CORS, Enabling CORS for 2 specific website**

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.



The above two URLs can now access this bucket and it’s objects regardless of the bucket region.