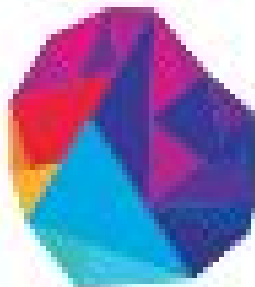


# **ASSESSMENT ON S3, ROUTE 53 AND RDS**

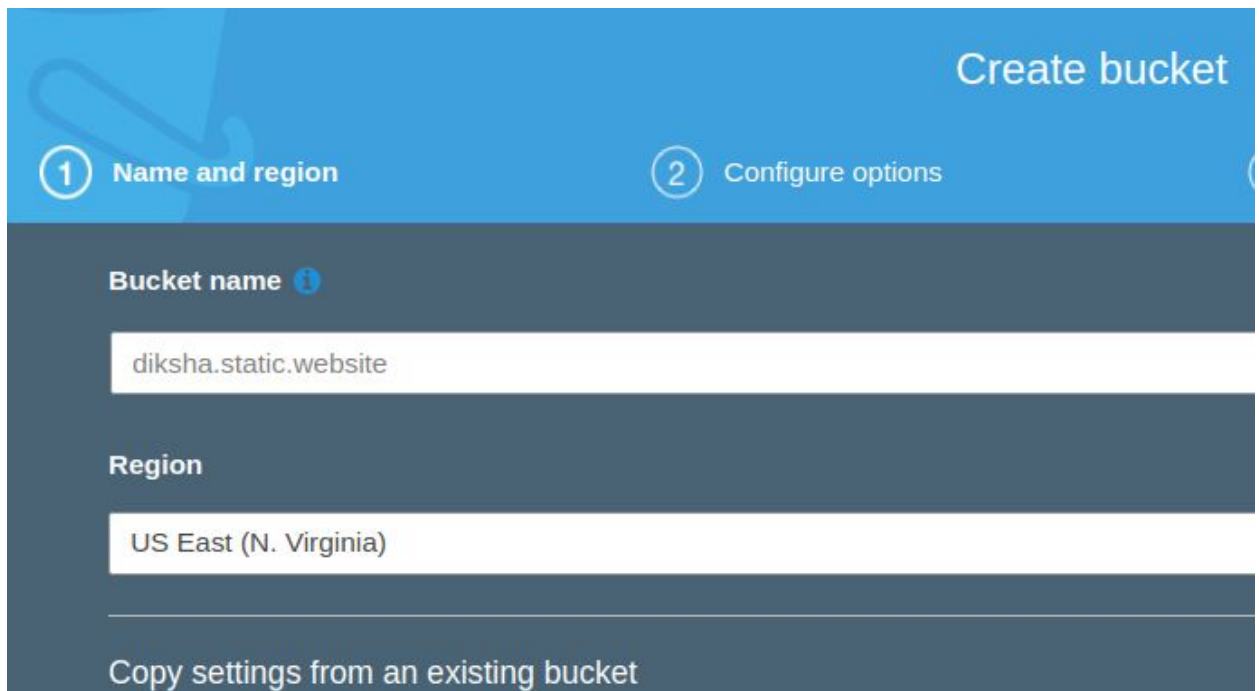
**TO  
THE  
NEW**



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

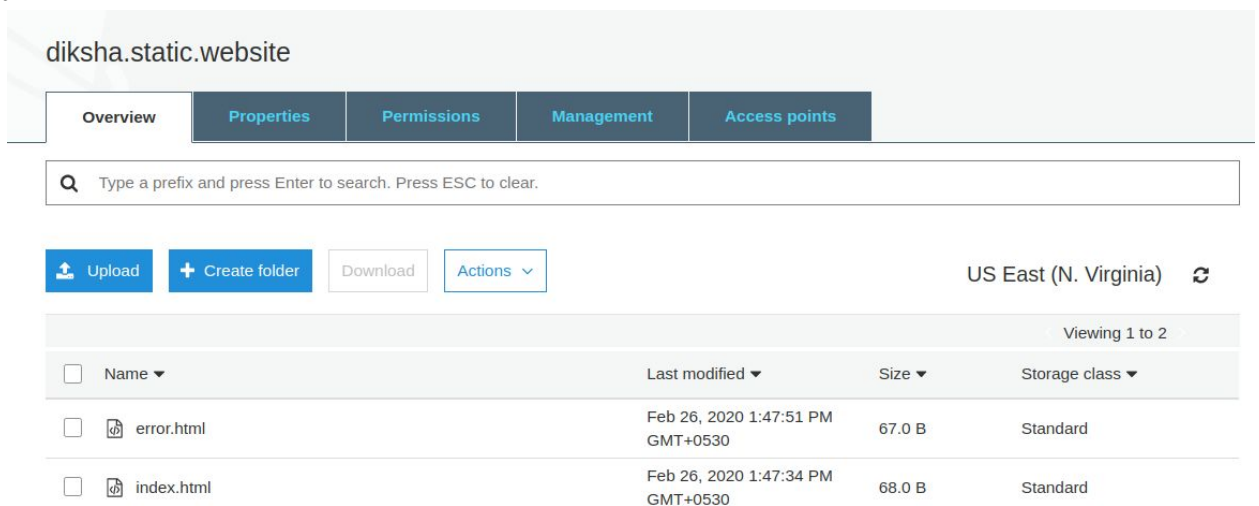
Step 1: Navigate to S3 in the AWS Console.

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



The screenshot shows the 'Create bucket' wizard in the AWS console. The title 'Create bucket' is in the top right. Below it, there are two numbered steps: '1 Name and region' and '2 Configure options'. The 'Name and region' step is active. It contains a 'Bucket name' field with the text 'diksha.static.website' and a 'Region' dropdown menu set to 'US East (N. Virginia)'. At the bottom, there is a link that says 'Copy settings from an existing bucket'.

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



The screenshot shows the AWS S3 console for the bucket 'diksha.static.website'. The 'Overview' tab is selected. Below the tabs, there is a search bar and a list of actions: 'Upload', 'Create folder', 'Download', and 'Actions'. The region 'US East (N. Virginia)' is displayed. A table shows the contents of the bucket:

	Name ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	error.html	Feb 26, 2020 1:47:51 PM GMT+0530	67.0 B	Standard
<input type="checkbox"/>	index.html	Feb 26, 2020 1:47:34 PM GMT+0530	68.0 B	Standard

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

The screenshot shows the Amazon S3 console interface for the bucket 'diksha.static.website'. The breadcrumb navigation at the top reads 'Amazon S3 > diksha.static.website'. Below the bucket name, there are five tabs: 'Overview', 'Properties', 'Permissions' (which is selected), 'Management', and 'Access points'. Under the 'Permissions' tab, there are four buttons: 'Block public access', 'Access Control List', 'Bucket Policy' (which is highlighted in blue), and 'CORS configuration'. Below these buttons, the 'Bucket policy editor' section is visible, showing the ARN 'arn:aws:s3::diksha.static.website' and a prompt to 'Type to add a new policy or edit an existing policy in the text area below.' A code editor displays a JSON policy document with the following content:

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "AddPerm",
6       "Effect": "Allow",
7       "Principal": "*",
8       "Action": [
9         "s3:GetObject"
10      ],
11      "Resource": [
12        "arn:aws:s3::diksha.static.website/*"
13      ]
14    }
15  ]
16 }
```

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

Amazon S3 > diksha.static.website

## diksha.static.website

[Overview](#)[Properties](#)[Permissions](#)[Management](#)[Access points](#)

### Versioning

Keep multiple versions of an object in the same bucket.

[Learn more](#)

☐ Disabled

### Server access logging

Set up access log records that provide details about access requests.

[Learn more](#)

☐ Disabled

### Static website hosting

Host a static website, which does not require server-side technologies.

[Learn more](#)

☐ Disabled

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

### Static website hosting

Endpoint : <http://diksha.static.website.s3-website-us-east-1.amazonaws.com>

☒ Use this bucket to host a website [Learn more](#)

Index document [i](#)

Error document [i](#)

Redirection rules (optional) [i](#)

☐ Redirect requests [Learn more](#)

☐ Disable website hosting

☐ Disabled

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



**This is my index page**



## **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

Roles > diksharole\_s3full

## Summary

Role ARN	arn:aws:iam::187632318301:role/diksharole_s3full <a href="#">🔗</a>
Role description	Allows EC2 instances to call AWS services on your behalf.   <a href="#">Edit</a>
Instance Profile ARNs	arn:aws:iam::187632318301:instance-profile/diksharole_s3full <a href="#">🔗</a>
Path	/
Creation time	2020-03-02 22:24 UTC+0530
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour <a href="#">Edit</a>

Permissions

Trust relationships

Tags (2)


Access Advisor

Revoke sessions

▼ Permissions policies (1 policy applied)

Attach policies

Policy name ▼

▶  AmazonS3FullAccess

STEP 2: Create a policy “diksha\_Assumerole” in which give Service=STS and Action=Assumerole and in Resources=ARN of role “diksharole\_s3full”

### Create policy

A policy defines the AWS permissions that you can grant to an IAM role or user.

Visual editor

JSON

Expand all | Collapse all

▼ STS (1 action) ⚠️ 1 warning

▶ Service

▶ Actions

▼ Resources

close

All resources

Add ARN(s)

×

Amazon Resource Names (ARNs) uniquely identify AWS resources. Resources are unique to each service. [Learn more](#)

Specify ARN for role [List ARNs manually](#)

arn:aws:iam::187632318301:role/diksharole\_s3full

Account \*

187632318301

☐ Any

Role name with path \*

diksharole\_s3full

☐ Any

Cancel

Add

## Create policy

1

2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor

JSON

[Import managed policy](#)

[Expand all](#) | [Collapse all](#)

▼ STS (1 action)

Clone Remove

▶ Service

STS

▶ Actions

Write

AssumeRole

▼ Resources

☒ Specific

☐ All resources

close

role ?

arn:aws:iam::187632318301:role/diksharole\_s3full

EDIT

⊗

☐ Any

Add ARN to restrict access

▶ Request conditions

[Specify request conditions \(optional\)](#)

**STEP 3:** Create a role “diksha\_assumerole” and attach the policy created to it

## Create role

1

2

3

4

### ▼ Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy



Filter policies ▼		Q diksha	Showing 6 results
	Policy name ▼	Used as	
<input type="checkbox"/>	▶ AssumeRolePolicy_diksha	Permissions policy (1)	
<input type="checkbox"/>	▶ DataAdmin_diksha	Permissions policy (1)	
<input type="checkbox"/>	▶ DevelopmentpolicyDiksha	None	
<input checked="" type="checkbox"/>	▶ diksha_Assumerole	None	
<input type="checkbox"/>	▶ Diksha_change_credentials	None	
<input type="checkbox"/>	▶ ProductionpolicyDiksha	None	



Roles > diksha\_assumerole

## Summary

Role ARN	arn:aws:iam::187632318301:role/diksha_assumerole
Role description	Allows EC2 instances to call AWS services on your behalf.   <a href="#">Edit</a>
Instance Profile ARNs	arn:aws:iam::187632318301:instance-profile/diksha_assumerole
Path	/
Creation time	2020-03-02 22:32 UTC+0530
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour <a href="#">Edit</a>

Permissions Trust relationships Tags (2) Access Advisor Revoke sessions

▼ Permissions policies (1 policy applied)

Attach policies

Policy name ▼	Policy type ▼
▶ diksha_Assumerole	Managed policy

**STEP 4:** Launch an instance.

Launch Instance ▼ Connect Actions ▼

search : diksha Add filter

Name	Purpose	Instance ID	Instance Type	Availability Zone	Instance State
diksha(ASsumerolesinstance)	Assume role implementation	i-0a8cbf598c1d48b7d	t2.micro	us-east-1c	running

**STEP 5:** Attach the role “diksha\_assumerole” to the above created instance.

Instances > Attach/Replace IAM Role

Attach/Replace IAM Role

Select an IAM role to attach to your instance. If you don't have any IAM roles, choose Create new IAM role to create a role in the IAM console.  
If an IAM role is already attached to your instance, the IAM role you choose will replace the existing role.

Instance ID i-0a8cbf598c1d48b7d (diksha(ASsumerolesinstance)) ⓘ

IAM role\*  ⓘ Create new IAM role ⓘ

**STEP 6:** Now add the arn of new role i.e “diksha\_assumerole” to old role “diksharole\_s3full in trust relationship

# Edit Trust Relationship

You can customize trust relationships by editing the following access control policy document.

## Policy Document

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "AWS": "arn:aws:iam::187632318301:role/diksha_assumerole",  
8       },  
9       "Service": "ec2.amazonaws.com"  
10    },  
11    "Action": "sts:AssumeRole"  
12  ]  
13 }
```

Roles > diksharole\_s3full

## Summary

Delete

Role ARN	arn:aws:iam::187632318301:role/diksharole_s3full <a href="#">🔗</a>
Role description	Allows EC2 instances to call AWS services on your behalf.   <a href="#">Edit</a>
Instance Profile ARNs	arn:aws:iam::187632318301:instance-profile/diksharole_s3full <a href="#">🔗</a>
Path	/
Creation time	2020-03-02 22:24 UTC+0530
Last activity	Not accessed in the tracking period
Maximum CLI/API session duration	1 hour <a href="#">Edit</a>

- Permissions
- Trust relationships
- Tags (2)
- Access Advisor
- Revoke sessions

You can view the trusted entities that can assume the role and the access conditions for the role. [Show policy document](#)

Edit trust relationship

### Trusted entities

The following trusted entities can assume this role.

**Trusted entities**  
arn:aws:iam::187632318301:role/diksha\_assumerole  
The identity provider(s) ec2.amazonaws.com

### Conditions

The following conditions define how and when trusted entities can assume the role.  
There are no conditions associated with this role.

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::187632318301:role/diksharole_s3full --role-session-name DikshaSession
{
  "Credentials": {
    "AccessKeyId": "ASIASXL6B65037KSEX57",
    "SecretAccessKey": "SWJ+B0B0YZD0X5q1ZzdAkY9FkZqsLb+nHHKIvHYj",
    "SessionToken": "FwoGZXIvYXdzEGsaDCbQ8g60BLFlrYLkcSKxAaIKA6+75lFTz5YdgAVHvzB+M+SueIlRaj3zwF5vR8X3b/T1UlBSy3gVKjHj078EsiZDLxB5Lf/AQkNwYvknaWQIDen6rCjGzciPiPivw6HBcQe+WywEx3WLCPLmoYKNpJ2qIXImig2gp81hJPNuVjEjhaJC2YZaRK0w/8gL8u0kkKX8DMUhi1v3q/Cx1p5ttngquMJ5Q/YwcfEWORiRZbS0seIQhdAR3QhP4HmtK6EJWiCjLh/XyBTItGRygyISXWzIiX3MACnBrSFsS9IgENhssDq2HlQkd3sWIASU7KOWtepPFELEc",
    "Expiration": "2020-03-02T18:35:07Z"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROASXL6B6503YRM7357A:DikshaSession",
    "Arn": "arn:aws:sts::187632318301:assumed-role/diksharole_s3full/DikshaSession"
  }
}
```

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=ASIASXL6B65037KSEX57
ubuntu@ip-172-31-79-130:~$ export AWS_SECRET_ACCESS_KEY=SWJ+B0B0YZD0X5q1ZzdAkY9FkZqsLb+nHHKIvHYj
ubuntu@ip-172-31-79-130:~$ export AWS_SESSION_ACCESS_KEY=FwoGZXIvYXdzEGsaDCbQ8g60BLFlrYLkcSKxAaIKA6+75lFTz5YdgAVHvzB+M+SueIlRaj3zwF5vR8X3b/T1UlBSy3gVKjHj078EsiZDLxB5Lf/AQkNwYvknaWQIDen6rCjGzciPiPivw6HBcQe+WywEx3WLCPLmoYKNpJ2qIXImig2gp81hJPNuVjEjhaJC2YZaRK0w/8gL8u0kkKX8DMUhi1v3q/Cx1p5ttngquMJ5Q/YwcfEWORiRZbS0seIQhdAR3QhP4HmtK6EJWiCjLh/XyBTItGRygyISXWzIiX3MACnBrSFsS9IgENhssDq2HlQkd3sWIASU7KOWtepPFELEc
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

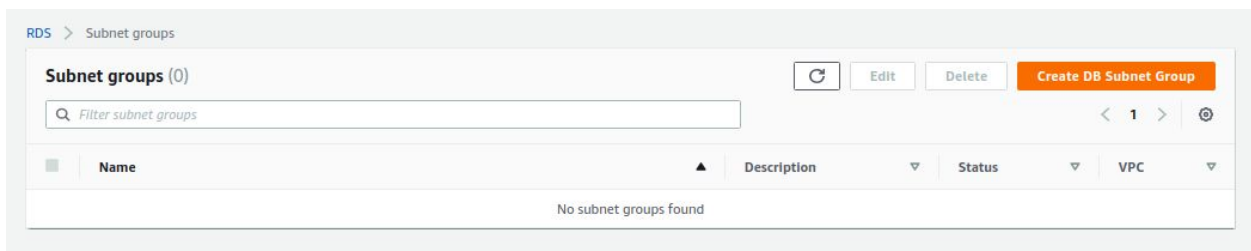
```

**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.





## Create DB subnet group

To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.

### Subnet group details

#### Name

You won't be able to modify the name after your subnet group has been created.

Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.

#### Description

#### VPC

Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.

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

### Add subnets

Add subnet(s) to this subnet group. You may add subnets one at a time below or add all the subnets related to this VPC. You may make additions/edits after this group is created. A minimum of 2 subnets is required.

Add all the subnets related to this VPC

**STEP 3:** Choose Create.

### Subnet groups (1)



Edit

Delete

Create DB Subnet Group

<

1

>



<input type="checkbox"/>	Name	Description	Status	VPC
<input type="checkbox"/>	diksha	creating rds instance	Complete	vpc-94ab21ee

**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

[Security Groups](#) > Create security group

## Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group fill in the fields below.

Security group name*	<input type="text" value="RDS_instance_sg"/>	
Description*	<input type="text" value="sg for RDS innstance"/>	
VPC	<input type="text" value="vpc-94ab21ee"/>	

\* Required

**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.

[Security Groups](#) > Edit inbound rules

### Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Type	Protocol	Port Range	Source	Description
MySQL/Aurora	TCP	3306	Anywhere	0.0.0.0/0, ::0
e.g. SSH for Admin Desktop				

## Create a DB Instance in the VPC

**STEP 1:** **Databases** > Choose **Create database** > In **Choose a database creation method**, choose **Standard Create** |

\*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*.

## Create database

### Choose a database creation method [Info](#)

☒ **Standard Create**

You set all of the configuration options, including ones for availability, security, backups, and maintenance.

☐ **Easy Create**

Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

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

### Engine options

#### Engine type [Info](#)

☐ **Amazon Aurora**



☒ **MySQL**



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

### Templates

Choose a sample template to meet your use case.

☐ **Production**

Use defaults for high availability and fast, consistent performance.

☐ **Dev/Test**

This instance is intended for development use outside of a production environment.

☒ **Free tier**

Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. [Info](#)

## Settings

**DB instance identifier** [Info](#)

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

**database-diksha**

**⚠ Must contain only letters, digits, or hyphens. Must start with a letter.**

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

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

- In the **Settings** section, open **Credential Settings**.
- Clear the **Auto generate a password** check box.
- (Optional) Change the **Master username** value and enter the same password in **Master password** and **Confirm password**.

### ▼ Credentials Settings

**Master username** [Info](#)

Type a login ID for the master user of your DB instance.

**Diksha**

1 to 16 alphanumeric characters. First character must be a letter

☐ **Auto generate a password**  
Amazon RDS can generate a password for you, or you can specify your own password

**Master password** [Info](#)

\*\*\*\*\*

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), " (double quote) and @ (at sign).

**Confirm password** [Info](#)

\*\*\*\*\*

**STEP 5:** Your database has been created.

RDS > Databases

**Databases**
Group resources
Refresh
Modify
Actions
Restore from S3
Create database

Filter databases

	DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current activity
	database-diksha	Instance	MySQL Community	us-east-1b	db.t2.micro	Creating	-	



**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

```

1 {
2   "Version": "2012-10-17",
3   "Id": "S3PolicyId1",
4
5   "Statement": [
6
7     {
8       "Sid": "IPAllowdiksha",
9       "Effect": "Deny",
10      "Principal": "*",
11      "Action": "s3:*",
12      "Resource": "arn:aws:s3:::s3diksha/*",
13      "Condition": {
14        "NotIpAddress": {"aws:SourceIp": "10.1.210.205"}
15      }
16    }
17  ]
18 }
19

```

## b. Domain

s3diksha

Overview	Properties	Permissions	Management	Access points
----------	------------	-------------	------------	---------------

Block public access	Access Control List	Bucket Policy	CORS configuration
---------------------	---------------------	---------------	--------------------

Bucket policy editor ARN: arn:aws:s3:::s3diksha

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

```

1 {
2   "Version": "2012-10-17",
3   "Id": "http referer policy example",
4   "Statement": [
5     {
6       "Sid": "Allow get requests originating from http://diksha.static.website.s3-website-us-east-1.amazonaws.com/ ",
7       "Effect": "Deny",
8       "Principal": "*",
9       "Action": "s3:GetObject",
10      "Resource": "arn:aws:s3:::s3diksha/*",
11      "Condition": {
12        "StringLike": {
13          "aws:Referer": [
14            "http://diksha.static.website.s3-website-us-east-1.amazonaws.com/*"
15          ]
16        }
17      }
18    }
19  ]
20 }

```

## 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.

```

1  {
2    "Id": "Policy1583297551962",
3    "Version": "2012-10-17",
4    "Statement": [
5      {
6        "Sid": "presigned url",
7        "Action": [
8          "s3:Get*"
9        ],
10       "Effect": "Deny",
11       "Resource": "arn:aws:s3:::diksha.static.website/*",
12       "Condition": {
13         "StringEquals": {
14           "s3:authType": "REST-QUERY-STRING"
15         }
16       },
17       "Principal": "*"
18     }
19   ]
20 }
```

### 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:

```
1 cd s3fs-fuse
2 ./autogen.sh
3 ./configure --prefix=/usr --with-openssl
4 make
5 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”**

Service Name com.amazonaws.us-east-1.s3 ⓘ

search : s3	Add filter	1 to 1 of 1
Service Name	Owner	Type
com.amazonaws.us-east-1.s3	amazon	Gateway

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.txt
{
  "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 --content-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.txt
{
  "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:

# diksha.static.website

Overview

Properties

Permissions

Management

## Versioning

☒ Enable versioning

☐ Suspend versioning

This suspends the creation of object versions for all operations but preserves any existing object versions.

☐ Disabled

Cancel

Save

Upload

+ Create folder




Download

Actions

Versions

Hide

Show

<input type="checkbox"/>	Name	Version ID	Last modified
	error.html		Mar 3, 2020 10:11:05 PM
<input type="checkbox"/>	 Mar 3, 2020 10:11:05 PM (Latest version)	ZpvqRvbobfrCQnkQ_yuCtYZPwfy4....	
<input type="checkbox"/>	 Feb 26, 2020 1:47:51 PM	null	
	index.html		Feb 26, 2020 1:47:34 PM
<input type="checkbox"/>	 Feb 26, 2020 1:47:34 PM (Latest version)	null	

Delete the latest version to retrieve previous version.



Upload

Create folder

Download

Actions

▼

Versions

Hide

Show

☐

Name

Version ID

*error.html*

☐

Feb 26, 2020 1:47:51 PM (Latest version)

null

*index.html*

☐

Feb 26, 2020 1:47:34 PM (Latest version)

null

## 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 routeCreate another role which has the policy to assume the previous Roleeeing 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

[Endpoints](#) > Create Endpoint

## Create Endpoint

A VPC endpoint allows you to securely connect your VPC to another service.

An interface endpoint is powered by [PrivateLink](#), and uses an elastic network interface (ENI) as an entry point for traffic destined to the service.

A gateway endpoint serves as a target for a route in your route table for traffic destined for the service.

Service category ☒ AWS services  
☐ Find service by name  
☐ Your AWS Marketplace services

Service Name  Select a service

VPC\*
vpc-d38d68b7

Configure route tables

A rule with destination **pl-63a5400a (com.amazonaws.us-east-1.s3)** and a target with this endpoints' ID (e.g. vpce-12345678) will be added to the route tables you select below.

Subnets associated with selected route tables will be able to access this endpoint.

rtb-2cc30148

	Route Table ID	Main	Associated With
<input checked="" type="checkbox"/>	rtb-2cc30148	Yes	subnet-06680a5b651f104dc   default

**Warning**

When you use an endpoint, the source IP addresses from your instances in your affected subnets for accessing the AWS service in the same region will be private IP addresses, not public IP addresses. Existing connections from your affected subnets to the AWS service that use public IP addresses may be dropped. Ensure that you don't have critical tasks running when you create or modify an endpoint.

Endpoints > Create Endpoint

## Create Endpoint



The following VPC Endpoint was created:

VPC Endpoint ID **vpce-0992a501340ba4d48**

Close

### 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>
```