AWS CLI commands EC2

```
aws ec2 describe-instances
```

```
aws ec2 start-instances --instance-ids i-dddddd70
```

```
aws ec2 stop-instances --instance-ids i-5c8282ed
```

```
aws ec2 terminate-instances --dry-run --instance-ids i-dddddd70
```

```
aws ec2 create-tags --resources i-dddddd70 --tags Key=Department,Value=Finance
```

```
aws ec2 describe-volumes
```

```
aws ec2 attach-volume  --volume-id vol-1d5cc8cc --instance-id i-dddddd70 --device
/dev/sdh
```

```
aws ec2 run-instances --dry-run --image-id ami-08111162 --count 1 --instance-type
t1.micro --key-name MyKeyPair --security-groups my-ami-security-group
```

```
aws ec2 reboot-instances --instance-ids i-dddddd70
```

```
aws ec2 modify-instance-attribute --instance-id i-44a44ac3 --instance-type
"{\"Value\": \"m1.small\"}"
```

```
aws ec2 create-image --instance-id i-44a44ac3 --name "Dev AMI" --description "AMI
for development server"



aws ec2 describe-images --image-ids ami-2d574747



aws ec2 deregister-image --image-id ami-2d574747 && aws ec2 delete-snapshot --
snapshot-id snap-4e665454



aws ec2 delete-snapshot --snapshot-id snap-4e665454
```

```
aws ec2 modify-instance-attribute --instance-id i-44a44ac3 --disable-api-
termination



aws ec2 modify-instance-attribute --instance-id i-44a44ac3 --no-disable-api-
termination



aws ec2 get-console-output --instance-id i-44a44ac3



aws ec2 monitor-instances --instance-ids i-44a44ac3



aws ec2 unmonitor-instances --instance-ids i-44a44ac3



aws ec2 describe-key-pairs
```

```
aws ec2 create-key-pair --key-name dev-servers
```

```
aws ec2 delete-key-pair --key-name dev-servers
```

1. View Current Status of an Instance

The following “aws ec2 describe-instances” will display detailed information about all instances that are managed by you. The output will be in JSON format.

```
aws ec2 describe-instances
```

If you have way too many instances, you can use the filter option to view a specific instance.

The following will display only the instance which has the “Name” tag set as “dev-server”.

```
# aws ec2 describe-instances --filter Name=tag:Name,Values=dev-server

..

..

  "State": {

      "Code": 80,

      "Name": "stopped"

  },

..

..

  "InstanceId": "i-e5888e46",
```

```
..
```

From the above output, we can see that this instance is currently “stopped” and is not running.

2. Start an Instance

The following “aws ec2 start-instances” command will start the instance that is specified in the –instance-ids field.

This will also display the current state and the previous state of the instance in the output. As you see from the following output, previously this instance was “stopped” and now it is in “pending” state and will be started soon.

```
# aws ec2 start-instances --instance-ids i-dddddd70

{

 "StartingInstances": [

   {

    "InstanceId": "i-dddddd70",

    "CurrentState": {

      "Code": 0,

      "Name": "pending"

    },

    "PreviousState": {

      "Code": 80,

      "Name": "stopped"

    }

   }
```

```
 ]

}
```

If you want to start multiple instances using a single command, provide all the instance ids at the end as shown below.

```
aws ec2 start-instances --instance-ids i-5c8282ed i-44a44ac3
```

3. Stop an Instance

The following “aws ec2 stop-instances” command will stop the instance that is specified in the –instance-ids field.

As you see from the output, previously this particular instance was in “running” state and currently it is in “stopping” state and will be stopped very soon.

```
# aws ec2 stop-instances --instance-ids i-5c8282ed

{

 "StoppingInstances": [

   {

    "InstanceId": "i-5c8282ed",

    "CurrentState": {

      "Code": 64,

      "Name": "stopping"

    },

    "PreviousState": {

      "Code": 16,

      "Name": "running"
```

```
    }

   }

 ]

}
```

The following are the possible state name and state code for an instance:

- 0 is for pending
- 16 is for running
- 32 is for shutting-down
- 48 is for terminated
- 64 is for stopping
- 80 is for stopped

If you execute the above command on an instance that is already stopped, you'll see both the previous state and the current state as stopped.

To stop multiple instances together, specify one or more instances ids as shown below.

```
aws ec2 stop-instances --instance-ids i-5c8282ed i-e5888e46
```

You can also force an instance to stop. This will not give the system an opportunity to flush the filesystem level cache. Use this only when you know exactly what you are doing.

```
aws ec2 stop-instances --force --instance-ids i-dddddd70
```

4. Terminate an Instance

The following "aws ec2 terminate-instances" command will terminate the instance that is specified in the –instance-ids field.

As you see from the output, previously this particular instance was in "stopped" state and it is not in "terminated" state.

Be very careful when you are terminating an instance, as you can't get your instance back once it is terminated. Terminate is not same as stop.

```
# aws ec2 terminate-instances --instance-ids i-44a44ac3

{

 "TerminatingInstances": [

  {

    "InstanceId": "i-44a44ac3",

    "CurrentState": {

        "Code": 48,

        "Name": "terminated"

    },

    "PreviousState": {

        "Code": 80,

        "Name": "stopped"

    }

  }

 ]

}
```

5. Add Name Tag to an Instance

The following “aws ec2 create-tags” command will add a new tag to the specified instance.

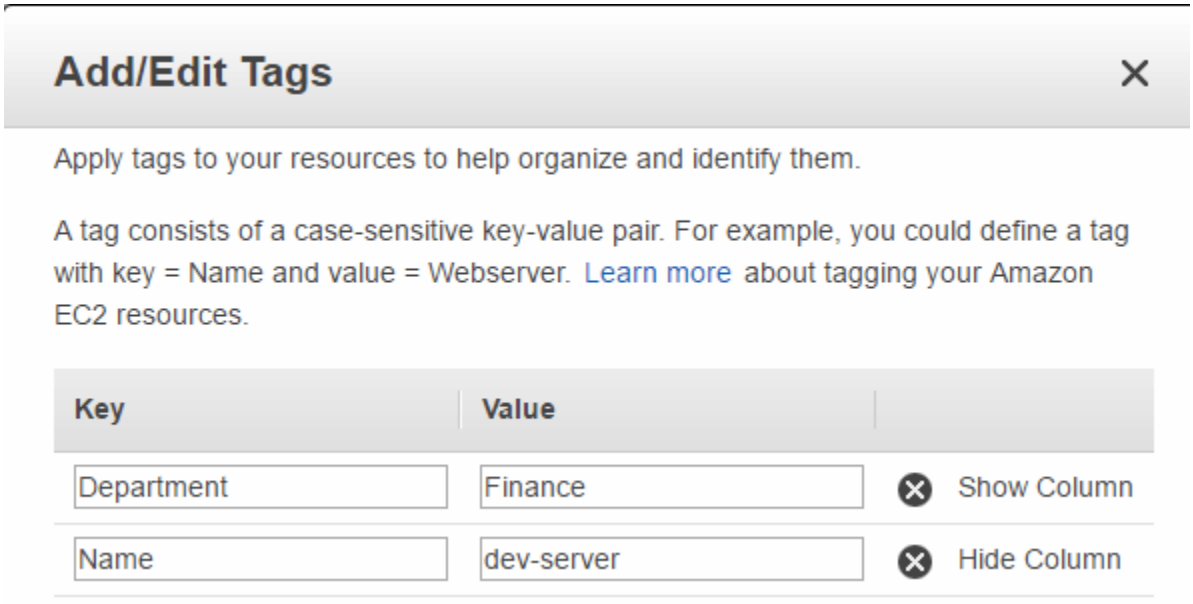
In this example, we are adding a tag with Key as “Department”, and it’s Value as “Finance”

```
aws ec2 create-tags --resources i-dddddd70 --tags Key=Department,Value=Finance
```

Now you'll see that the new Tag has been added.

```
# aws ec2 describe-instances

..

"Tags": [

    {

        "Value": "Finance",

        "Key": "Department"

    },

    {

        "Value": "dev-server",

        "Key": "Name"

    }

],

..
```

You can also verify the TAG from the AWS Management Console GUI as shown below.



6. Add Storage (Block Device) to an Instance

First, use the following command to get a list of all block device volumes that are available for you. Look for those volumes that has the State as “available”

```
aws ec2 describe-volumes

..

{

    "AvailabilityZone": "us-east-1b",

    "Attachments": [],

    "Encrypted": false,

    "VolumeType": "standard",

    "VolumeId": "vol-1d5cc8cc",

    "State": "available",

    "SnapshotId": "",

    "CreateTime": "2016-04-17T15:08:40.469Z",

    "Size": 1
```

```
}
```

..

From the above, get the VolumeId, and use that in the following “aws ec2 attach-volume” command to attach that volume to a particular instance.

In the following command, you should also specify the –device option, which will be the the disk name that will be used at the OS level for this particular volume.

In this example, this volume will be attached as “/dev/sdh” disk.

```
# aws ec2 attach-volume  --volume-id vol-1d5cc8cc --instance-id i-dddddd70 --
device /dev/sdh

{

  "AttachTime": "2016-04-17T15:14:10.144Z",

  "InstanceId": "i-dddddd70",

  "VolumeId": "vol-1d5cc8cc",

  "State": "attaching",

  "Device": "/dev/sdh"

}
```

Note: When you attach a volume to an instance from the AWS management console, by default it will automatically populate the device. But in the AWS EC2 CLI, you have to specify the device name as shown below.

After attaching the device, you’ll notice that the state changed from “available” to “attached” for this particular volume.

```
# aws ec2 describe-volumes
```

..

```
"Attachments": [

  {

    "AttachTime": "2016-04-17T15:14:10.000Z",

    "InstanceId": "i-dddddd70",

    "VolumeId": "vol-1d5cc8cc",

    "State": "attached",

..
```

7. Launch a New EC2 Instance

The following command will create a new AWS EC2 instance for you.

This is equivalent to the “Launch Instance” that you’ll perform the AWS management console.

To launch an instance, use “aws ec2 run-instances” command as shown below.

```
# aws ec2 run-instances --image-id ami-22111148 --count 1 --instance-type t1.micro
--key-name stage-key --security-groups my-aws-security-group
```

In the above command:

- –image-id Specify the image id for the AMI that you want to launch. You can browse the AWS marketplace and choose the correct image that is required for your project.
- –count Specify the number of instance that you want to launch from this image. In this case, we are creating only one new instance.
- –instance-type In this example, I’m launching this instance as a t1.micro type, which doesn’t use have CPU and RAM.
- –key-name Specify the name of the key pair that you want to use this with system. You should create your own key pair before launching your instance.
- –security-groups Specify the name of the security groups. You should create a security group with appropriate firewall rules that are required for your project.

The following is a sample full output of the above command, which display all the information about the newly launched instance.

```
{
 "OwnerId": "353535354545",
 "ReservationId": "r-d6668103",
 "Groups": [
   {
     "GroupName": "my-aws-security-group",
     "GroupId": "sg-6cbebe01"
   }
 ],
 "Instances": [
   {
     "Monitoring": {
         "State": "disabled"
     },
     "PublicDnsName": "",
     "KernelId": "aki-91afcaf8",
     "State": {
         "Code": 0,
         "Name": "pending"
     },
```

```
    "EbsOptimized": false,
    "LaunchTime": "2016-04-17T19:13:56.000Z",
    "ProductCodes": [],
    "StateTransitionReason": "",
    "InstanceId": "i-44a44ac3",
    "ImageId": "ami-22111148",
    "PrivateDnsName": "",
    "KeyName": "stage-key",
    "SecurityGroups": [
      {
          "GroupName": "my-aws-security-group",
          "GroupId": "sg-6cbebe01"
      }
    ],
    "ClientToken": "",
    "InstanceType": "t1.micro",
    "NetworkInterfaces": [],
    "Placement": {
        "Tenancy": "default",
        "GroupName": "",
        "AvailabilityZone": "us-east-1c"
    },
```

```
     "Hypervisor": "xen",
     "BlockDeviceMappings": [],
     "Architecture": "x86_64",
     "StateReason": {
         "Message": "pending",
         "Code": "pending"
     },
     "RootDeviceName": "/dev/sda1",
     "VirtualizationType": "paravirtual",
     "RootDeviceType": "ebs",
     "AmiLaunchIndex": 0
   }
 ]
}
```

If you get the following error message, then the instance type you've selected is not supported for this AMI. Change the instance type and try again.

```
# aws ec2 run-instances --dry-run --image-id ami-08111162 --count 1 --instance-
type t1.micro --key-name MyKeyPair

A client error (InvalidParameterCombination) occurred when calling the
RunInstances operation: Non-Windows instances with a virtualization type of 'hvm'
are currently not supported for this instance type.
```

The following are additional parameters that you can pass with the "aws ec2run-instances" command

- –subnet-id Use the appropriate subnet id to launch a EC2 VPC instance
- –block-device-mappings file://mymap.json In this JSON file you can specify the volumes that you want to attach to the instance that you want to launch
- –user-data file://myuserdata.txt In this text file you can specify the userdata that need to be executed when the EC2 instance is launched
- –iam-instance-profile Name=myprofile You can also specify your IAM profile that you want to use while launching the instance

8. Reboot an Instance (and General Options)

To reboot an instance, use “aws ec2 reboot-instances” command as shown below.

```
aws ec2 reboot-instances --instance-ids i-dddddd70
```

The are few options that you can use pretty much with most of the AWS EC2 cli commands.

For example, you can use “–dry-run” option pretty much with all the AWS EC2 cli command. As the name suggests, it will not really execute the command. This will only perform a dry-run and display all possible error messages without really doing anything.

For example, the following is a dry-run operation when you want to stop an instance.

```
# aws ec2 stop-instances --dry-run --instance-ids i-dddddd70


A client error (DryRunOperation) occurred when calling the StopInstances
operation: Request would have succeeded, but DryRun flag is set.
```

When you are performing a dry-run the following are the two possible errors:

- If you have appropriate permission, it will display “DryRunOperation” error, and any other real error message that are related to that specific command that you are executing.
- If you don’t have permission to execute that particular command, it will display “UnauthorizedOperation” error

You can also specify the input to the AWS EC2 cli in JSON format using the –cli-input-json option as shown below.

If you don’t know exactly what kind of information needs to passed for a particular EC2 command in JSON format, you can use –generate-cli-skeleton as shown below.

Once you have the JSON output, modify the appropriate values, and use it as an input to –cli-input-json option.

```
# aws ec2 stop-instances --dry-run --force --generate-cli-skeleton  --instance-ids
i-dddddd70

{

    "DryRun": true,

    "InstanceIds": [

        "i-dddddd70"

    ],

    "Force": true

}
```

The following is an example JSON file that can be used as an input to AWS EC2 CLI command.

```
# cat stop.json

{

    "DryRun": true,

    "InstanceIds": [

        "i-dddddd70"

    ],

    "Force": true

}
```

In the following example, we are using the above stop.json file as an value for the –client-input-json option as shown below. Don't forget to give "file://"

```
aws ec2 stop-instances --cli-input-json file://stop.json
```

9. Change Instance Type

Before changing: In this example, the following instance is of type t1.micro

```
# aws ec2 describe-instances

..

"InstanceId": "i-44a44ac3",

..

"InstanceType": "t1.micro",
```

You can change the above instance to a different instance type.

For that, first stop the instance. Without stopping you cannot change the instance type.

```
aws ec2 stop-instances --instance-ids i-44a44ac3
```

The following "aws ec2 modify-instance-attribute" is used to change the instance type. In this example, we are changing the instance type to "m1.small"

```
aws ec2 modify-instance-attribute --instance-id i-44a44ac3 --instance-type
"{\"Value\": \"m1.small\"}"
```

After changing, the following is the instance type.

```
# aws ec2 describe-instances
```

```
..

 "InstanceId": "i-44a44ac3",
```

```
..

 "InstanceType": "m1.small",
```

If an instance type is not supported for your particular image, you'll get the following error message. In this example, t2.nano is not supported for this particular image.

```
# aws ec2 modify-instance-attribute --instance-id i-44a44ac3 --instance-type
"{\"Value\": \"t2.nano\"}"



A client error (InvalidParameterCombination) occurred when calling the
ModifyInstanceAttribute operation: Virtualization type 'hvm' is required for
instances of type 't2.nano'. Ensure that you are using an AMI with virtualization
type 'hvm'.
```

10. Create a New Image

From your particular instance that is running with all the configuration changes that you've done so far, you can create a new image using the following "aws ec2 create-image" command.

```
# aws ec2 create-image --instance-id i-44a44ac3 --name "Dev AMI" --description
"AMI for development server"


{

    "ImageId": "ami-2d574747"

}
```

This is helpful when you want to launch new instance based on this new image that you created which has your changes in it.

Use the following “aws ec2 describe-images” command to view the details of the new image that you’ve just created.

```
# aws ec2 describe-images --image-ids ami-2d574747

{

 "Images": [

  {

   "VirtualizationType": "paravirtual",

   "Name": "Dev AMI",

   "Hypervisor": "xen",

   "ImageId": "ami-2d574747",

   "RootDeviceType": "ebs",

   "State": "available",

   "BlockDeviceMappings": [

    {

     "DeviceName": "/dev/sda1",

     "Ebs": {

       "DeleteOnTermination": true,

       "SnapshotId": "snap-4e665454",

       "VolumeSize": 8,

       "VolumeType": "standard",

       "Encrypted": false

     }
```

```
    }
   ],
   "Architecture": "x86_64",
   "ImageLocation": "353535354545/Dev AMI",
   "KernelId": "aki-91afcaf8",
   "OwnerId": "353535354545",
   "RootDeviceName": "/dev/sda1",
   "CreationDate": "2016-04-17T19:57:57.000Z",
   "Public": false,
   "ImageType": "machine",
   "Description": "AMI for development server"
  }
 ]
}
```

11. Delete an Image

When you create an image, it also creates a snapshot.

So, when you are deleting your image you have to do two things.

First, use the “aws ec2 deregister-image” command to dereigser the Image.

```
aws ec2 deregister-image --image-id ami-2d574747
```

Next, use the “aws ec2 delete-snapshot” command to delete the snapshot that is associated with your image.

```
aws ec2 delete-snapshot --snapshot-id snap-4e665454
```

12. Enable Instance Termination Protection

It is very easy to delete an running instance by mistake when you execute the terminate command by mistake (Either from UI or from command line).

By default termination protection is turned off. This means that you can delete your instance by mistake.

To enable termination protection for your instance, use the “aws ec2 modify-instance-attribute” command, and pass the “–disable-api-termination” option as shown below.

```
aws ec2 modify-instance-attribute --instance-id i-44a44ac3 --disable-api-termination
```

Later if you want to disable the termination protection, execute the following command.

```
aws ec2 modify-instance-attribute --instance-id i-44a44ac3 --no-disable-api-termination
```

13. Get System Log (View Console Output)

Since you don’t have a physical access to the console for the instances that are running on AWS EC2, use the following command.

This “aws ec2 get-console-output” command will display whatever was sent to the system console for your particular instance.

```
aws ec2 get-console-output --instance-id i-44a44ac3
```

This is very helpful when you are debugging some issues on your system.

14. Enable Cloudwatch Monitoring for an Instance

The following “aws ec2 monitor-instances” command will enable advanced cloudwatch monitoring provided by AWS to your specified instance.

```
# aws ec2 monitor-instances --instance-ids i-44a44ac3

{

 "InstanceMonitorings": [

     {

         "InstanceId": "i-44a44ac3",

         "Monitoring": {

             "State": "enabled"

         }

     }

 ]

}
```

Since there are some cost associated with the monitoring of instance, you may want to enable monitoring temporarily when you are debugging some issue, and later you can disable the montiroing using the following command.

```
# aws ec2 unmonitor-instances --instance-ids i-44a44ac3

{

 "InstanceMonitorings": [

     {

         "InstanceId": "i-44a44ac3",

         "Monitoring": {
```

```
            "State": "disabled"

        }

    }

 ]

}
```

15. AWS EC2 Key Pairs

The following “aws ec2 describe-key-pairs” command will display all keypairs that you’ve created so far in AWS.

```
# aws ec2 describe-key-pairs

{

"KeyPairs": [

 {

   "KeyName": "prod-key",

   "KeyFingerprint": "61:7c:f1:13:53:b0:3a:01:dd:dd:6c:90"

 },

 {

   "KeyName": "stage-key",

   "KeyFingerprint": "41:6c:d1:23:a3:c0:2a:0a:dc:db:60:4c"

 }

]

}
```

To create a new Keypair use the following “aws ec2 create-key-pair” command. In this example, I’m creating a key pair with name “dev-servers”. I’ll be using this key-pair for all my dev instances.

```
# aws ec2 create-key-pair --key-name dev-servers

{

  "KeyName": "dev-servers",

  "KeyMaterial": "-----BEGIN RSA PRIVATE KEY-----\n

  dYXbKYMRlI59J5XKyPgC/67GL8\nXg

  ....

  n-----END RSA PRIVATE KEY-----",

  "KeyFingerprint": "3d:c2:c8:7f:d2:ee:1d:66"

}
```

If you have created a keypair by mistake, use the following command to delete it.

```
# aws ec2 delete-key-pair --key-name dev-servers
```

AWS S3 CLI commands

```
# s3 make bucket (create bucket)

aws s3 mb s3://tgsbucket --region us-west-2
```

```
# s3 remove bucket

aws s3 rb s3://tgsbucket

aws s3 rb s3://tgsbucket --force



# s3 ls commands

aws s3 ls

aws s3 ls s3://tgsbucket

aws s3 ls s3://tgsbucket --recursive

aws s3 ls s3://tgsbucket --recursive  --human-readable --summarize



# s3 cp commands

aws s3 cp getdata.php s3://tgsbucket

aws s3 cp /local/dir/data s3://tgsbucket --recursive

aws s3 cp s3://tgsbucket/getdata.php /local/dir/data

aws s3 cp s3://tgsbucket/ /local/dir/data --recursive

aws s3 cp s3://tgsbucket/init.xml s3://backup-bucket

aws s3 cp s3://tgsbucket s3://backup-bucket --recursive



# s3 mv commands

aws s3 mv source.json s3://tgsbucket

aws s3 mv s3://tgsbucket/getdata.php /home/project

aws s3 mv s3://tgsbucket/source.json s3://backup-bucket
```

```
aws s3 mv /local/dir/data s3://tgsbucket/data --recursive

aws s3 mv s3://tgsbucket s3://backup-bucket --recursive



# s3 rm commands

aws s3 rm s3://tgsbucket/queries.txt

aws s3 rm s3://tgsbucket --recursive



# s3 sync commands

aws s3 sync backup s3://tgsbucket

aws s3 sync s3://tgsbucket/backup /tmp/backup

aws s3 sync s3://tgsbucket s3://backup-bucket



# s3 bucket website

aws s3 website s3://tgsbucket/ --index-document index.html --error-document
error.html



# s3 presign url (default 3600 seconds)

aws s3 presign s3://tgsbucket/dnsrecords.txt

aws s3 presign s3://tgsbucket/dnsrecords.txt --expires-in 60
```

1. Create New S3 Bucket

Use mb option for this. mb stands for Make Bucket.

The following will create a new S3 bucket

```
$ aws s3 mb s3://tgsbucket

make_bucket: tgsbucket
```

In the above example, the bucket is created in the us-east-1 region, as that is what is specified in the user's config file as shown below.

```
$ cat ~/.aws/config

[profile ramesh]

region = us-east-1
```

To setup your config file properly, use aws configure command as explained here: 15 AWS Configure Command Examples to Manage Multiple Profiles for CLI
If the bucket already exists, and you own the bucket, you'll get the following error message.

```
$ aws s3 mb s3://tgsbucket

make_bucket failed: s3://tgsbucket An error occurred (BucketAlreadyOwnedByYou)
when calling the CreateBucket operation: Your previous request to create the named
bucket succeeded and you already own it.
```

If the bucket already exists, but owned by some other user, you'll get the following error message.

```
$ aws s3 mb s3://paloalto

make_bucket failed: s3://paloalto An error occurred (BucketAlreadyExists) when
calling the CreateBucket operation: The requested bucket name is not available.
The bucket namespace is shared by all users of the system. Please select a
different name and try again.
```

Under some situation, you might also get the following error message.

```
$ aws s3 mb s3://demo-bucket


make_bucket failed: s3://demo-bucket An error occurred
(IllegalLocationConstraintException) when calling the CreateBucket operation: The
unspecified location constraint is incompatible for the region specific endpoint
this request was sent to.
```

2. Create New S3 Bucket – Different Region

To create a bucket in a specific region (different than the one from your config file), then use the –region option as shown below.

```
$ aws s3 mb s3://tgsbucket --region us-west-2


make_bucket: tgsbucket
```

3. Delete S3 Bucket (That is empty)

Use rb option for this. rb stands for remove bucket.

The following deletes the given bucket.

```
$ aws s3 rb s3://tgsbucket


remove_bucket: tgsbucket
```

If the bucket you are trying to delete doesn't exists, you'll get the following error message.

```
$ aws s3 rb s3://tgsbucket1


remove_bucket failed: s3://tgsbucket1 An error occurred (NoSuchBucket) when
calling the DeleteBucket operation: The specified bucket does not exist
```

4. Delete S3 Bucket (And all its objects)

If the bucket contains some object, you'll get the following error message:

```
$ aws s3 rb s3://tgsbucket

remove_bucket failed: s3://tgsbucket An error occurred (BucketNotEmpty) when
calling the DeleteBucket operation: The bucket you tried to delete is not empty
```

To delete a bucket along with all its objects, use the –force option as shown below.

```
$ aws s3 rb s3://tgsbucket --force

delete: s3://tgsbucket/demo/getdata.php

delete: s3://tgsbucket/ipallow.txt

delete: s3://tgsbucket/demo/servers.txt

delete: s3://tgsbucket/demo/

remove_bucket: tgsbucket
```

5. List All S3 Buckets

To view all the buckets owned by the user, execute the following ls command.

```
$ aws s3 ls

2019-02-06 11:38:55 tgsbucket

2018-12-18 18:02:27 etclinux

2018-12-08 18:05:15 readynas

..

..
```

In the above output, the timestamp is the date the bucket was created. The timezone was adjusted to be displayed to your laptop's timezone.

The following command is same as the above:

```
aws s3 ls s3://
```

6. List All Objects in a Bucket

The following command displays all objects and prefixes under the tgsbucket.

```
$ aws s3 ls s3://tgsbucket

                           PRE config/

                           PRE data/

2019-04-07 11:38:20         13 getdata.php

2019-04-07 11:38:20       2546 ipallow.php

2019-04-07 11:38:20          9 license.php

2019-04-07 11:38:20       3677 servers.txt
```

In the above output:

- Inside the tgsbucket, there are two folders config and data (indicated by PRE)
- PRE stands for Prefix of an S3 object.
- Inside the tgsbucket, we have 4 files at the / level
- The timestamp is when the file was created
- The 2nd column display the size of the S3 object

Note: The above output doesn't display the content of sub-folders config and data

7. List all Objects in a Bucket Recursively

To display all the objects recursively including the content of the sub-folders, execute the following command.

```
$ aws s3 ls s3://tgsbucket --recursive

2019-04-07 11:38:19       2777 config/init.xml
```

```
2019-04-07 11:38:20         52 config/support.txt

2019-04-07 11:38:20       1758 data/database.txt

2019-04-07 11:38:20         13 getdata.php
```

```
2019-04-07 11:38:20       2546 ipallow.php

2019-04-07 11:38:20          9 license.php

2019-04-07 11:38:20       3677 servers.txt
```

Note: When you are listing all the files, notice how there is no PRE indicator in the 2nd column for the folders.

8. Total Size of All Objects in a S3 Bucket

You can identify the total size of all the files in your S3 bucket by using the combination of following three options: recursive, human-readable, summarize

Note: The following displays both total file size in the S3 bucket, and the total number of files in the s3 bucket

```
$ aws s3 ls s3://tgsbucket --recursive  --human-readable --summarize

2019-04-07 11:38:19    2.7 KiB config/init.xml

2019-04-07 11:38:20   52 Bytes config/support.txt

2019-04-07 11:38:20    1.7 KiB data/database.txt
```

```
2019-04-07 11:38:20   13 Bytes getdata.php

2019-04-07 11:38:20    2.5 KiB ipallow.php

2019-04-07 11:38:20    9 Bytes license.php

2019-04-07 11:38:20    3.6 KiB servers.txt
```

```
Total Objects: 7

   Total Size: 10.6 KiB
```

In the above output:

- recursive option make sure that it displays all the files in the s3 bucket including sub-folders
- human-readable displays the size of the file in readable format. Possible values you'll see in the 2nd column for the size are: Bytes/MiB/KiB/GiB/TiB/PiB/EiB
- summarize options make sure to display the last two lines in the above output. This indicates the total number of objects in the S3 bucket and the total size of all those objects

9. Request Payer Listing

If a specific bucket is configured as requester pays buckets, then if you are accessing objects in that bucket, you understand that you are responsible for the payment of that request access. In this case, bucket owner doesn't have to pay for the access.

To indicate this in your ls command, you'll have to specify –request-payer option as shown below.

```
$ aws s3 ls s3://tgsbucket --recursive --request-payer requester

2019-04-07 11:38:19       2777 config/init.xml

2019-04-07 11:38:20         52 config/support.txt

2019-04-07 11:38:20       1758 data/database.txt

2019-04-07 11:38:20         13 getdata.php

2019-04-07 11:38:20       2546 ipallow.php

2019-04-07 11:38:20          9 license.php

2019-04-07 11:38:20       3677 servers.txt
```

For signed URL, make sure to include x-amz-request-payer=requester in the request

10. Copy Local File to S3 Bucket

In the following example, we are copying getdata.php file from local laptop to S3 bucket.

```
$ aws s3 cp getdata.php s3://tgsbucket

upload: ./getdata.php to s3://tgsbucket/getdata.php
```

If you want to copy the getdata.php to a S3 bucket with a different name, do the following

```
$ aws s3 cp getdata.php s3://tgsbucket/getdata-new.php

upload: ./getdata.php to s3://tgsbucket/getdata-new.php
```

For the local file, you can also specify the full path as shown below.

```
$ aws s3 cp /home/project/getdata.php s3://tgsbucket

upload: ../../home/project/getdata.php to s3://tgsbucket/getdata.php
```

11. Copy Local Folder with all Files to S3 Bucket

In this example, we are copying all the files from the “data” folder that is under /home/projects directory to S3 bucket

```
$ cd /home/projects



$ aws s3 cp data s3://tgsbucket --recursive
```

```
upload: data/parameters.txt to s3://tgsbucket/parameters.txt

upload: data/common.txt to s3://tgsbucket/common.txt

..
```

In the above example, note that only the files from the local data/ folder is getting uploaded. Not the folder “data” itself

If you like to upload the data folder from local to s3 bucket as data folder, then specify the folder name after the bucket name as shown below.

```
$ aws s3 cp data s3://tgsbucket/data --recursive

upload: data/parameters.txt to s3://tgsbucket/data/parameters.txt

upload: data/common.txt to s3://tgsbucket/data/common.txt

..

..
```

12. Download a File from S3 Bucket

To download a specific file from an S3 bucket do the following. The following copies getdata.php from the given s3 bucket to the current directory.

```
$ aws s3 cp s3://tgsbucket/getdata.php .

download: s3://tgsbucket/getdata.php to ./getdata.php
```

You can download the file to the local machine with in a different name as shown below.

```
$ aws s3 cp s3://tgsbucket/getdata.php getdata-local.php

download: s3://tgsbucket/getdata.php to ./getdata-local.php
```

Download the file from S3 bucket to a specific folder in local machine as shown below. The following will download getdata.php file to /home/project folder on local machine.

```
$ aws s3 cp s3://tgsbucket/getdata.php /home/project/
```

```
download: s3://tgsbucket/getdata.php to ../../home/project/getdata.php
```

13. Download All Files Recursively from a S3 Bucket (Using Copy)

The following will download all the files from the given bucket to the current directory on your laptop.

```
$ aws s3 cp s3://tgsbucket/ . --recursive

download: s3://tgsbucket/getdata.php to ./getdata.php

download: s3://tgsbucket/config/init.xml ./config/init.xml

..
```

If you want to download all the files from a S3 bucket to a specific folder locally, please specify the full path of the local directory as shown below.

```
$ aws s3 cp s3://tgsbucket/ /home/projects/tgsbucket --recursive

download: s3://tgsbucket/getdata.php to ../../home/projects/tgsbucket/getdata.php

download: s3://tgsbucket/config/init.xml to
../../home/projects/tgsbucket/config/init.xml

..
```

In the above command, if the tgsbucket folder doesn't exists under /home/projects, it will create it automatically.

14. Copy a File from One Bucket to Another Bucket

The following command will copy the config/init.xml from tgsbucket to backup bucket as shown below.

```
$ aws s3 cp s3://tgsbucket/config/init.xml s3://backup-bucket

copy: s3://tgsbucket/config/init.xml to s3://backup-bucket/init.xml
```

In the above example, eventhough init.xml file was under config folder in the source bucket, on the destination bucket, it copied the init.xml file to the top-level / in the backup-bucket.

If you want to copy the same folder from source and destination along with the file, specify the folder name in the desintation bucketas shown below.

```
$ aws s3 cp s3://tgsbucket/config/init.xml s3://backup-bucket/config

copy: s3://tgsbucket/config/init.xml to s3://backup-bucket/config/init.xml
```

If the destination bucket doesn't exist, you'll get the following error message.

```
$ aws s3 cp s3://tgsbucket/test.txt s3://backup-bucket-777

copy failed: s3://tgsbucket/test.txt to s3://backup-bucket-777/test.txt An error occurred (NoSuchBucket) when calling the CopyObject operation: The specified bucket does not exist
```

15. Copy All Files Recursively from One Bucket to Another

The following will copy all the files from the source bucket including files under sub-folders to the destination bucket.

```
$ aws s3 cp s3://tgsbucket s3://backup-bucket --recursive

copy: s3://tgsbucket/getdata.php to s3://backup-bucket/getdata.php

copy: s3://tgsbucket/config/init.xml s3://backup-bucket/config/init.xml

..
```

16. Move a File from Local to S3 Bucket

When you move file from Local machine to S3 bucket, as you would expect, the file will be physically moved from local machine to the S3 bucket.

```
$ ls -l source.json
```

```
-rw-r--r--  1 ramesh  sysadmin  1404 Apr  2 13:25 source.json
```

```
$ aws s3 mv source.json s3://tgsbucket

move: ./source.json to s3://tgsbucket/source.json
```

As you see the file doesn't exists on the local machine after the move. Its only on S3 bucket now.

```
$ ls -l source.json

ls: source.json: No such file or directory
```

17. Move a File from S3 Bucket to Local

The following is reverse of the previou example. Here, the file will be moved from S3 bucket to local machine.

As you see below, the file now exists on the s3 bucket.

```
$ aws s3 ls s3://tgsbucket/getdata.php

2019-04-06 06:24:29       1758 getdata.php
```

Move the file from S3 bucket to /home/project directory on local machine.

```
$ aws s3 mv s3://tgsbucket/getdata.php /home/project

move: s3://tgsbucket/getdata.php to ../../../home/project/getdata.php
```

After the move, the file doesn't exists on S3 bucketanymore.

```
$ aws s3 ls s3://tgsbucket/getdata.php
```

18. Move a File from One S3 Bucket to Another S3 Bucket

Before the move, the file source.json is in tgsbucket.

```
$ aws s3 ls s3://tgsbucket/source.json

2019-04-06 06:51:39       1404 source.json
```

This file is not in backup-bucket.

```
$ aws s3 ls s3://backup-bucket/source.json

$
```

Move the file from tgsbucketto backup-bucket.

```
$ aws s3 mv s3://tgsbucket/source.json s3://backup-bucket

move: s3://tgsbucket/source.json to s3://backup-bucket/source.json
```

Now, the file is only on the backup-bucket.

```
$ aws s3 ls s3://tgsbucket/source.json

$
```

```
$ aws s3 ls s3://backup-bucket/source.json

2019-04-06 06:56:00       1404 source.json
```

19. Move All Files from a Local Folder to S3 Bucket

In this example, the following files are under data folder.

```
$ ls -1 data

dnsrecords.txt

parameters.txt

dev-setup.txt

error.txt
```

The following moves all the files in the data directory on local machine to tgsbucket

```
$ aws s3 mv data s3://tgsbucket/data --recursive

move: data/dnsrecords.txt to s3://tgsbucket/data/dnsrecords.txt

move: data/parameters.txt to s3://tgsbucket/data/parameters.txt

move: data/dev-setup.txt to s3://tgsbucket/data/dev-setup.txt

move: data/error.txt to s3://tgsbucket/data/error.txt
```

20. Move All Files from S3 Bucket to Local Folder

In this example, the localdata folder is currently empty.

```
$ ls -1 localdata

$
```

The following will move all the files in the S3 bucketunder data folder to localdata folder on your local machine.

```
$ aws s3 mv s3://tgsbucket/data/ localdata --recursive

move: s3://tgsbucket/data/dnsrecords.txt to localdata/dnsrecords.txt

move: s3://tgsbucket/data/parameters.txt to localdata/parameters.txt
```

```
move: s3://tgsbucket/data/dev-setup.txt to localdata/dev-setup.txt

move: s3://tgsbucket/data/error.txt to localdata/error.txt
```

Here is the output after the above move.

```
$ aws s3 ls s3://tgsbucket/data/

$



$ ls -1 localdata

dnsrecords.txt

parameters.txt

dev-setup.txt

error.txt
```

21. Move All Files from One S3 Bucket to Another S3 Bucket

Use the recursive option to move all files from one bucket to another as shown below.

```
$ aws s3 mv s3://tgsbucket s3://backup-bucket --recursive

move: s3://tgsbucket/dev-setup.txt to s3://backup-bucket/dev-setup.txt

move: s3://tgsbucket/dnsrecords.txt to s3://backup-bucket/dnsrecords.txt

move: s3://tgsbucket/error.txt to s3://backup-bucket/error.txt

move: s3://tgsbucket/parameters.txt to s3://backup-bucket/parameters.txt
```

22. Delete a File from S3 Bucket

To delete a specific file from a S3 bucket, use the rm option as shown below. The following will delete the queries.txt file from the given S3 bucket.

```
$ aws s3 rm s3://tgsbucket/queries.txt

delete: s3://tgsbucket/queries.txt
```

23. Delete All Objects from S3 buckets

When you specify rm option just with a bucket name, it doesn't do anything. This will not delete any file from the bucket.

```
aws s3 rm s3://tgsbucket
```

To delete all the files from a S3 bucket, use the –recursive option as show nbelow.

```
$ aws s3 rm s3://tgsbucket --recursive

delete: s3://tgsbucket/dnsrecords.txt

delete: s3://tgsbucket/common.txt

delete: s3://tgsbucket/parameters.txt

delete: s3://tgsbucket/config/init.xml

..
```

24. Sync files from Laptop to S3 Bucket

When you use sync command, it will recursively copies only the new or updated files from the source directory to the destination.

The following will sync the files from backup directory in local machine to the tgsbucket.

```
$ aws s3 sync backup s3://tgsbucket
```

```
upload: backup/docker.sh to s3://tgsbucket/docker.sh

upload: backup/address.txt to s3://tgsbucket/address.txt

upload: backup/display.py to s3://tgsbucket/display.py

upload: backup/getdata.php to s3://tgsbucket/getdata.php
```

If you want to sync it to a subfolder called backup on the S3 bucket, then include the folder name in the s3 bucket as shown below.

```
$ aws s3 sync backup s3://tgsbucket/backup

upload: backup/docker.sh to s3://tgsbucket/backup/docker.sh

upload: backup/address.txt to s3://tgsbucket/backup/address.txt

upload: backup/display.py to s3://tgsbucket/backup/display.py

upload: backup/getdata.php to s3://tgsbucket/backup/getdata.php
```

Once you do the sync once, if you run the command immediately again, it will not do anything, as there is no new or updated files on the local backup directory.

```
$ aws s3 sync backup s3://tgsbucket/backup

$
```

Let us create a new file on the local machine for testing.

```
echo "New file" > backup/newfile.txt
```

Now when you execute the sync, it will sync only this new file to the S3 bucket.

```
$ aws s3 sync backup s3://tgsbucket/backup
```

```
upload: backup/newfile.txt to s3://tgsbucket/backup/newfile.txt
```

25. Sync File from S3 bucket to Local

This is reverse of the previous example. Here, we are syncing the files from the S3 bucket to the local machine.

```
$ aws s3 sync s3://tgsbucket/backup /tmp/backup

download: s3://tgsbucket/backup/docker.sh to ../../tmp/backup/docker.sh

download: s3://tgsbucket/backup/display.py to ../../tmp/backup/display.py

download: s3://tgsbucket/backup/newfile.txt to ../../tmp/backup/newfile.txt

download: s3://tgsbucket/backup/getdata.php to ../../tmp/backup/getdata.php

download: s3://tgsbucket/backup/address.txt to ../../tmp/backup/address.txt
```

26. Sync Files from one S3 Bucket to Another S3 Bucket

The following example syncs the files from one tgsbucket to backup-bucket

```
$ aws s3 sync s3://tgsbucket s3://backup-bucket

copy: s3://tgsbucket/backup/newfile.txt to s3://backup-bucket/backup/newfile.txt

copy: s3://tgsbucket/backup/display.py to s3://backup-bucket/backup/display.py

copy: s3://tgsbucket/backup/docker.sh to s3://backup-bucket/backup/docker.sh

copy: s3://tgsbucket/backup/address.txt to s3://backup-bucket/backup/address.txt

copy: s3://tgsbucket/backup/getdata.php to s3://backup-bucket/backup/getdata.php
```

27. Set S3 bucket as a website

You can also make S3 bucket to host a static website as shown below. For this, you need to specify both the index and error document.

```
aws s3 website s3://tgsbucket/ --index-document index.html --error-document
error.html
```

This bucket is in us-east-1 region. So, once you've done the above, you can access the tgsbucket as a website using the following URL: http://tgsbucket.s3-website-us-east-1.amazonaws.com/

For this to work properly, make sure public access is set on this S3 bucket, as this acts as a website now.

28. Presign URL of S3 Object for Temporary Access

When you presign a URL for an S3 file, anyone who was given this URL can retrieve the S3 file with a HTTP GET request.

For example, if you want to give access to the dnsrecords.txt file to someone temporarily, presign this specific S3 object as shown below.

```
$ aws s3 presign s3://tgsbucket/dnsrecords.txt

https://tgsbucket.s3.amazonaws.com/error.txt?AWSAccessKeyId=AAAAAAAAAAAAAAAAAAAA&E
xpires=1111111111&Signature=ooooooooooo%2Babcdefghijlimmm%3A
```

The output of the above command will be a HTTPS url, which you can hand it out someone who should be able to download the dnsrecords.txt file from your S3 bucket.

The above URL will be valid by default for 3600 seconds (1 hour).

If you want to specify a short expirty time, use the following expires-in option. The following will create a presigned URL that is valid only for 1 minute.
–expires-in (integer) Number of seconds until the pre-signed URL expires. Default is 3600 seconds.

```
$ aws s3 presign s3://tgsbucket/dnsrecords.txt --expires-in 60

https://tgsbucket.s3.amazonaws.com/error.txt?AWSAccessKeyId=AAAAAAAAAAAAAAAAAAAA&E
xpires=1111
```