**Experiment-1**

**Launch an instance**

  Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

# Connect to instance

 Info

Connect to your instance i-0a0a299cb7d7fa075 (exp-1) using any of these options

* EC2 Instance Connect
* Session Manager
* SSH client
* EC2 serial console

Instance ID

[i-0a0a299cb7d7fa075](https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-0a0a299cb7d7fa075) (exp-1)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp-1.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

chmod 400 "exp-1.pem"

1. Connect to your instance using its Public DNS:

ec2-3-93-201-212.compute-1.amazonaws.com

Example:

ssh -i "exp-1.pem" ec2-user@ec2-3-93-201-212.compute-1.amazonaws.com

**Note:** In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Aim: Establish an AWS account.

Use the AWS Management Console to launch an EC2 instance and connect to it.

Step 1: Login to the AWS Management Console using Login Credentials

Step 2:Make sure we’re in the N. Virginia (US East) region because we optimized our examples for this region.

Step 3:Click on Services and search for EC2.

Step 4:Click Launch instance to start the wizard for launching a virtual machine as shown in the figure. Step 5: Give a name to the instance Exp1

Step 6: Select the Amazon Linux as our Operating System and select instance type as t2.micro

Step 7:Now create a key pair by clicking on the hyperlink

Step 8:Name the key pair exp1 and click on create key pair keeping other fields default.

Step 9:Click on the checkboxes Allow HTTPS and HTTP traffic from internet along with SSH under Network settings. Then keeping everything else default click on Launch instance.

Step 10:Go to instances and select the given instance and click on connect after we see the instance state as running and 2/2 status checks

Step 11:Go to SSH client and copy the ssh key

Step 12: Now open PowerShell on desktop

Step13: Change the directory where key.pem file is located by using cd and then paste the ssh key.

**Experiment-2**

**AWS S3**

# Create bucket

 Info

Buckets are containers for data stored in S3.

## General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket typeInfo

General purposeRecommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory - *New*Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

## Object Ownership

 Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabledObjects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

Bucket owner preferredIf new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.Object writerThe object writer remains the object owner.

If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads.

## Access control list (ACL)

Grant basic read/write permissions to AWS accounts. [Learn more](https://docs.aws.amazon.com/console/s3/acl-overview)

Grantee

Objects

Object ACL

Object owner (your AWS account)

Canonical ID:

3c42be411f3b3530696f2c5c389b9f81946a54d645ed9d443b56d548ea5f289a

Read

Read

Write

Everyone (public access)

Group:

http://acs.amazonaws.com/groups/global/AllUsers

Read

Read

Write

Authenticated users group (anyone with an AWS account)

Group:

http://acs.amazonaws.com/groups/global/AuthenticatedUsers

Read

Read

Write

When you grant access to the Everyone or Authenticated users group grantees, anyone in the world can access this object.

[Learn more](https://docs.aws.amazon.com/console/s3/set-bucket-acl)

I understand the effects of these changes on this object.

You must select the check box to continue.

Aim : Create our First AWS S3 Bucket and Upload Content to Bucket and Manage their Access and Create Static Website using AWS S3

Step 1: Navigate to AWS console and search for S3 and click on create bucket.

Step 2: Enter the following details

• Bucket type – General purpose

• Bucket name – must be unique within global namespace.

Step 3: Select Object Ownership as ACLs enabled

• Allow all public access

• Click on Create Bucket

Step 4: Click on bucket

• Click on upload and select file to upload and click on upload.

Step 5: Select the file and go to Permissions tab.

Step 6: Click on edit and allow all the Object and Object ACL permissions and click on save changes

Step 7: Select the bucket and go to properties tab.

Step 8: Scroll down for Static web hosting and click on edit. Enable the static web hosting and enter the index document name as “index.html”

Step 9: We can find a link of the static website copy it.

Step 10: Paste it in new tab and we can access our file.

**Experiment-3 Aim: Exemplify the principle of rapid elasticity through a practical exercise involving the setup of an EC2 Amazon instance and the creation of multiple Elastic Block Store (EBS) volumes to be attached to EC2 instance.**

Aim: Exemplify the principle of rapid elasticity through a practical exercise involving the setup of an EC2 Amazon instance and the creation of multiple Elastic Block Store (EBS) volumes to be attached to EC2 instance.

→Attach a new Volume to EC2

Step 1: Launch an EC2 instance

Step 2: Check the Availability Zone === In this case its us-east-1a

• Create a volume

Step 3:Click on volume From Elastic Block Store

Step 4: Click on create Volume

Step 5 :Change the size to required capacity == In this case I am using 15 GB

Step 6: Check the Availability Zone == In this case it should be us-east-1a (as EC2 is launched in that AZ) Step 7: Click on Create Volume Name the Volume == to easily recognize == IN this case I will name as Extra Volume

Step 10: Now attach the volume to the EC2 Instance

1. Select the Extra Volume
2. II. Drop down the Actions
3. III. Click on Attach Volume
4. IV. Select your ec2 instance
5. V. Name the Device Name as /dev/sdf VI. Click on Attach Volume

Step 11: Now Connect to the EC2 Instance named “EC2”

a) Select the instance

b) Click on connect

Step 12: Now run few commands to verify the disks attached to the Ec2 Instance

a) Change over to root user ==== sudo su

b) Check the disks ============ fdisk -l

c) Create a directory ========== mkdir /mnt/test

d) Now format the newly attached disk ============= mkfs /dev/xvdf

e) Now mount the disk on the directory

f) Now change the directory and create a file

→Detach the new volume and Attach to a another EC2 Instance

Step 1: Click on volumes

Step 2: Select the volume then drop down the Actions and click on detach volume then click ondetach and volume state needs to change from in-use to available Step 3: Now go to EC2 dashboard and Launch another new instance named as EC2-New Step 4: Now click again on Volumes Step 5: Select the Extra Volume then drop down the Actions and click on Attach Step 6: Drop down and Select the EC2-New then search for Device name /dev/sdf and select the diskThen click on attach volume Step 7: Select the EC2-New Instance then Click on Connect Step 8: Now change user to root then create a directory and verify the disk →Create a snapshot and copy to another region Step 1: Click on Volumes Step 2: Select the Extra Volume then Drop down the Actions and Click on Create Snapshot Step 3:Click on Create Snapshot Step 4: Click on Snapshots Step 5: Select the Snapshot and then drop-down Actions then click on Copy Snapshot presently weare in N. Virginia Region Step 6:Change the Destination only and only to Us-West-2 for AWS Academy users rest can whohave a free tier account can select any region then click Copy Snapshot →Create volume from the snapshot and attach to an EC2 Instance Step 1: Now change the Region from N. Virginia to Us-West-2 (Oregon) Step 2: Select the Copied Snapshot then drop down the actions and click on Create volume fromsnapshot Step 3: Leave everything as default just click on Create Volume Step 4: Click on Volumes Step 5: Launch an EC2 Instance and attach the volume then connect to Instance

**EFS:Is a network file system where files can be shared among all ec2.**

1)Two security groups

MY-KMIT-EC2-SG ->INBOUND –SSH – Anywhere

MY-KMIT-EFS-SG ->INBOUND – NFS--Custom – MY-KMIT-EC2-SG

2)Create an EFS

Create File System - MY-KMIT-EFS-EC2 🡪 Customize 🡪Security group 🡪MY-KMIT-EFS-SG for all AZ’s

3)Launching 2 EC2 instances in 2 different AZ’s

My server-1🡪Amazon Linux🡪Create new key pair 🡪Edit Network settings🡪Subnet(us-east-1a)🡪 Select existing security group(MY-KMIT-EC2-SG) 🡪Launch instance

My server-2🡪Amazon Linux🡪Create new key pair 🡪Edit Network settings🡪Subnet(us-east-1d)🡪Select existing security group(MY-KMIT-EC2-SG) 🡪Launch instance

4)We will attach EFS with both the EC2 instances

Connect to My-server-1🡪sudo yum -y install amazon-efs-utils🡪sudo mkdir /efs 🡪EFS(my-kmit-efs-ec2) 🡪Attach🡪Copy first command🡪paste it in terminal(:/ /efs at last)🡪cd /efs/🡪sudo mkdir test🡪sudo chown ec2-user test🡪cd test🡪nano kmit.txt🡪Example

Connect to My-server-2 using 🡪sudo yum -y install amazon-efs-utils🡪sudo mkdir /efs 🡪EFS(my-kmit-efs-ec2) Attach🡪Copy first command(:/ /efs at last)🡪cd /efs/🡪ls 🡪cd test 🡪ls 🡪we can see kmit.txt from my-server-1

5)We modify/add files and folders from one EC2

6)We will try to access files/folders from another EC2

**VPC**

VPC will isolate one organization from another.

Subnet will isolate one department from another in same organization. (1st 2 octets should be same as VPC)

Create VPC – Large Network – 192.168.0.0/16

Create Public Subnet – 192.168.1.0/24

Create Private Subnet – 192.168.2.0/24

2 services to access EC2 instance deployed in public IP:

Internet Gateway(Exposing to public network)

Route tables(Mapping of public and private IP)

Steps:  
Create VPC🡪VPC settings🡪Resources to create🡪VPC only🡪Name tag:My-VPC-KMIT(remember)🡪IPV4 CIDR(192.168.0.0/16)

VPC🡪Subnets🡪Create subnet🡪Select VPC(My-VPC-KMIT)🡪Subnet name:Public subnet🡪Select AZ(us-east-1a)🡪IPV4 subnet CIDR block(192.168.1.0/24)

Subnets🡪 Create subnet🡪Select VPC(My-VPC-KMIT)🡪Subnet name:Private subnet🡪Select AZ(us-east-1b)🡪IPV4 subnet CIDR block(192.168.2.0/24)

Launch Instance🡪My-Public-EC2🡪Select AMI(Windows)🡪Network settings🡪VPC(my-vpc-kmit)🡪Subnets(Public-Subnet)🡪Auto-assign public IP(Enable)

Connect🡪RDB client🡪Download rdb file🡪Click on it🡪Error(No Gateway and route table)

VPC🡪Internet Gateways🡪Create Internet Gateway🡪Name:my-IGW🡪Create🡪Attach to a VPC🡪Select my-vpc-kmit🡪Attach Internet Gateway

VPC🡪Route tables🡪Create route tables🡪Name(my-igw-rt)🡪VPC(my-vpc-kmit)🡪Create route table🡪Subnet association🡪Edit subnet association🡪Select Public-subnet🡪Save association🡪Routes🡪Edit routes🡪Add route🡪0.0.0.0🡪Internet Gateway(my-igw)

Connect🡪rdb file🡪It asks for password🡪EC2🡪RDB client🡪Get password🡪Upload private key file🡪Decrypt the password🡪Enter and connect

EC2🡪 Private-EC2🡪Windows🡪Network settings🡪VPC🡪Subnet(private-subnet)🡪Launch

Connect🡪 RDB client🡪Get password🡪Upload private key file🡪Decrypt the password🡪Copy🡪In public-ec2🡪Search remote desktop connection🡪Computer:(Copy private ip)🡪Connect🡪User name(Administrator)🡪Password

**Experiment-9 Launch a NoSQL database using Amazon DynamoDB.**

**Amazon DynamoDB**

A fast and flexible NoSQL database service for any scale

DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.

How it works

DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit-millisecond latency at any scale. Its flexible data model and reliable performance make DynamoDB a great fit for mobile, web, gaming, advertising technology, Internet of Things, and other applications.

## Benefits and features

### Performance at scale

DynamoDB supports some of the world’s largest-scale applications by providing consistent, single-digit-millisecond response times at any scale. You can build applications with virtually unlimited throughput and storage.

### No servers to manage

DynamoDB is serverless. You don't have to provision, patch, or manage servers, and you don't have to install, maintain, or operate software. DynamoDB automatically scales tables up and down to adjust for capacity and maintain performance.

### Enterprise ready

DynamoDB is built for mission-critical workloads, including support for ACID transactions for a broad set of applications that require complex business logic.

## Use cases

### Ad tech

Advertising technology (ad tech) companies use DynamoDB as a key-value store for storing various kinds of marketing data, such as user profiles, user events, clicks, and visited links.

[Learn more](https://aws.amazon.com/blogs/database/amazon-dynamodb-ad-tech-use-cases-and-design-patterns/)

### Gaming

Gaming companies use DynamoDB in all capabilities of game platforms, including game state, player data, session history, and leaderboards.

[Learn more](https://aws.amazon.com/blogs/database/amazon-dynamodb-gaming-use-cases-and-design-patterns/)

### Retail

Many retail companies use common DynamoDB design patterns to deliver consistently low latency for mission-critical use cases such as shopping carts, workflow engines, inventory tracking and fulfillment, customer profiles, and accounts.

[Learn more](https://www.twitch.tv/videos/470470827)

### Media and entertainment

Media and entertainment companies use DynamoDB when they require extreme scale of throughput and concurrency, low latency, and reliability. DynamoDB scales elastically to handle the load and maintains low latency that is critical for real-time scenarios, such as video streaming and interactive content.

[Learn more](https://www.twitch.tv/videos/470470827)

### Software and internet

Software and internet companies require the ability to accommodate extreme concurrency, request rates, and spikes in traffic. This concurrency is measured in millions of users and connections, and request rates can easily reach millions per second. DynamoDB has a proven record of being able to handle internet-scale requirements while maintaining consistent, single-digit-millisecond latency.

[Learn more](https://www.twitch.tv/videos/470470827)

### Banking and finance

As companies in banking and finance build more cloud-native applications, they seek to use fully managed services to increase agility, reduce time to market, and minimize operational overhead. Common use cases include user transactions, event-driven transaction processing, fraud detection, mainframe offloading, and change data capture.

# Create table

Top of Form

## Table details

 Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (\_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

String

1 to 255 characters and case sensitive.

Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

**Experiment-10 Migrate a website from local server to cloud**

8. Migrate a website from local server to Cloud using Docker.

Migrate a website from local server to Cloud

==============================================

1) Launch an EC2 Instance : t2.large, Network setting: Add security group: all traffic anywhere,

2) Connect to EC2 Instance

3) clone the git repos

backend

============

tinyurl.com/cs1bekmit

git clone https://github.com/procareer3fwd/realgrandebackend.git

frontend

==============

tinyurl.com/cs1fekmit

git clone https://github.com/procareer3fwd/realgrandefrontend.git

4) Update the Ubuntu

sudo apt update

5) Install the Docker

sudo apt -y install docker.io

6)  Check the docker images

sudo docker images

7) change the directory to backend

cd realgrandebackend/

8) create an .env file

nano .env

9) Copy Paste the lines in .env file

MONGODBURL="mongodb+srv://fsd04.2hxrdca.mongodb.net/realgrande?retryWrites=true&w=majority"

DBUSERNAME=procareer3

DBPASSWORD=ISobjBDohsFqEAqg

FRONTENDURI="http://3.82.156.186"

10) Build the docker image for backend

 sudo docker build -t backend\_server .

11) Check for any images running in the container

sudo docker ps

12) Now run the backend\_server docker image in the container

sudo docker run -d -p 2001:5000 backend\_server

13) Now try to access the backend\_server on the browser

<EC2\_PUBLIC\_IP\_ADDRESS>:2001/api

14) Now change the directory to frontend

ubuntu@ip-172-31-1-17:~/realgrandebackend$ cd

ubuntu@ip-172-31-1-17:~$ cd realgrandefrontend/

ubuntu@ip-172-31-1-17:~/realgrandefrontend$

15) Now create a .env file

nano .env

16) Copy paste the line in .env file

REACT\_APP\_BACKEND\_URL="http://44.203.169.59:2001/api"

17) Build the docker image for frontend

 sudo docker build -t frontend .

18) Check for any images running in the container

sudo docker ps

19) Now run the backend\_server docker image in the container

sudo docker run -d -p 80:3000 frontend

20) Now try to access the frontend on the browser

<EC2\_PUBLIC\_IP\_ADDRESS>

**Experiment-11**

**Amazon Simple Notification Service**

Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

## Benefits and features

### Reliably deliver messages with durability

Amazon SNS uses cross availability zone message storage to provide high message durability. Amazon SNS reliably delivers messages to valid AWS endpoints, such as Amazon SQS queues and AWS Lambda functions.

### Automatically scale your workload

Amazon SNS leverages the proven AWS cloud to dynamically scale with your application. Amazon SNS is a fully managed service, taking care of the heavy lifting related to capacity planning, provisioning, monitoring, and patching.

### Simplify your architecture with Message Filtering

Amazon SNS helps you simplify your pub/sub messaging architecture by offloading the message filtering logic from your subscriber systems, and message routing logic from your publisher systems.

### Keep messages private and secure

Amazon SNS topic owners can set topic policies that restrict who can publish and subscribe to a topic. Amazon SNS also ensures that data is encrypted in transit and at rest, and provides VPC endpoints for message privacy.

## Use cases

The NASA image & video library provides easy access to thousands of images, audio recordings and videos, documenting NASA’s more than half a century of achievements in exploring the vast unknown.

.

PlayOn! operates a comprehensive technology platform and serverless video processing pipeline, enabling high-quality, low-cost productions of live sporting events.

Topic name

A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

TypeInfo

Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

* Strictly-preserved message ordering
* Exactly-once message delivery
* High throughput, up to 300 publishes/second
* Subscription protocols: SQS

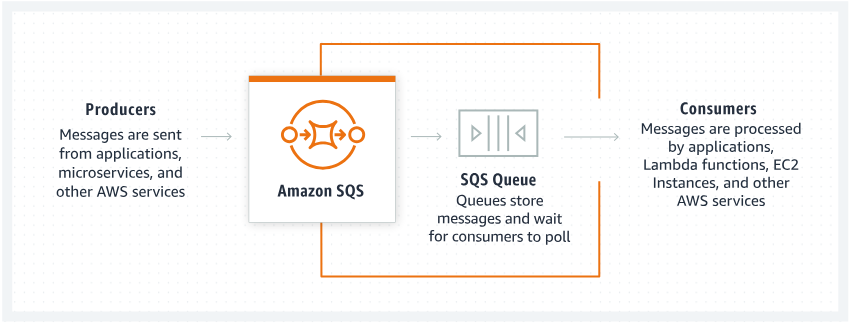
Standard

* Best-effort message ordering
* At-least once message delivery
* Highest throughput in publishes/second
* Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Display name - *optional*Info

* To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.
* **Amazon SQS**
* A message queuing service
* Amazon SQS provides queues for high-throughput, system-to-system messaging. You can use queues to decouple heavyweight processes and to buffer and batch work. Amazon SQS stores messages until microservices and serverless applications process them.

## How it works



Amazon SQS allows producers to send messages to a queue. Messages are then stored in an SQS Queue. When consumers are ready to process new messages they poll them from the queue. Applications, microservices, and multiple AWS services can take the role of producers or consumers.

## Benefits and features

### Highly scalable Standard and FIFO queues

Queues scale elastically with your application. Nearly unlimited throughput and no limit to the number of messages per queue in Standard queues. First-In-First-Out delivery and exactly once processing in FIFO queues.

### Durability and availability

Your queues are distributed on multiple servers. Redundant infrastructure provides highly concurrent access to messages.

### Security

Protection in transit and at rest. Transmit sensitive data in encrypted queues. Send messages in a Virtual Private Cloud.

### Batching

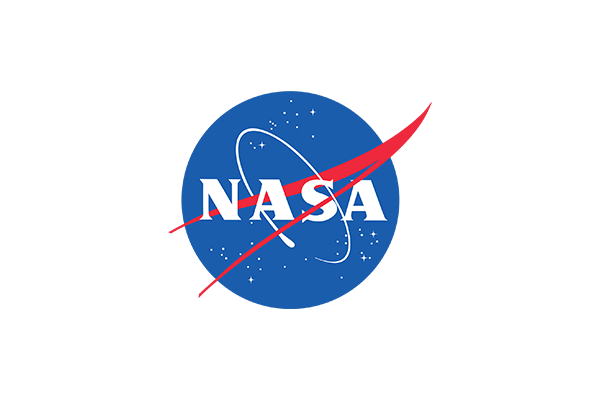
Send, receive, or delete messages in batches of up to 10 messages or 256KB to save costs.

## Use cases



### EMS

Petroleum retailers in Australia are improving the performance and safety of their service stations with an AWS IoT–enabled solution called Fuelsuite from EMS. Fuelsuite schedules messages to and from the edge devices by using Amazon SQS.



### NASA

The NASA image & video library provides easy access to thousands of images, audio recordings and videos, documenting NASA’s more than half a century of achievements in exploring the vast unknown. The architecture includes Amazon SQS to decouple incoming jobs from pipeline processes.

Type

Choose the queue type for your application or cloud infrastructure.

Standard Info

At-least-once delivery, message ordering isn't preserved

* At-least once delivery
* Best-effort ordering

FIFO Info

First-in-first-out delivery, message ordering is preserved

* First-in-first-out delivery
* Exactly-once processing