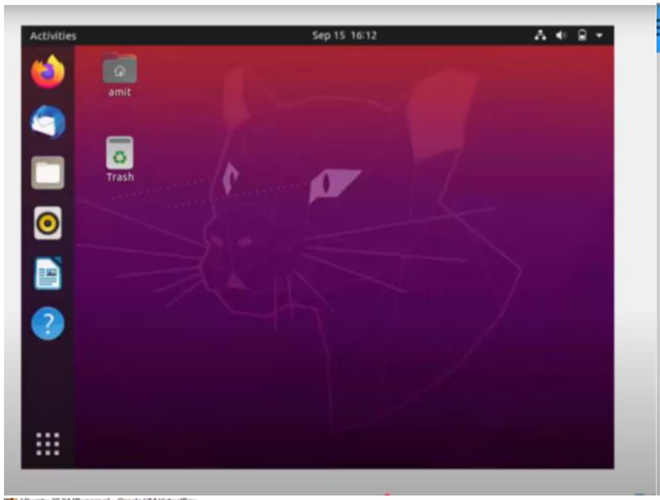
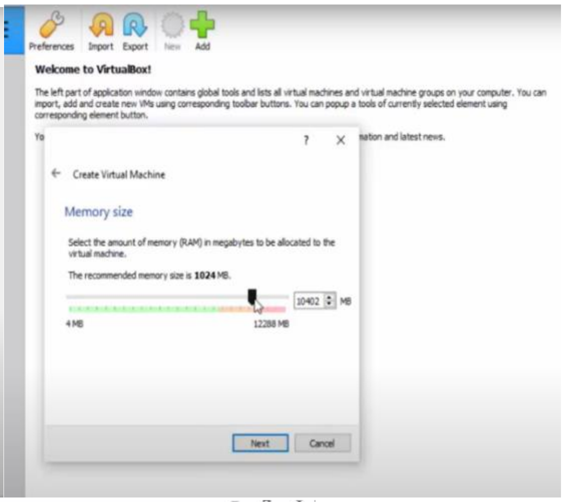




Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

Create Virtual Machine

Memory size

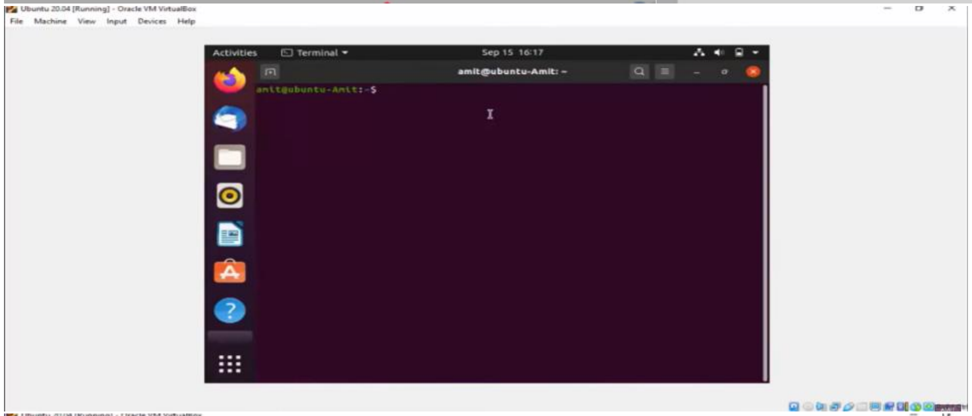
Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

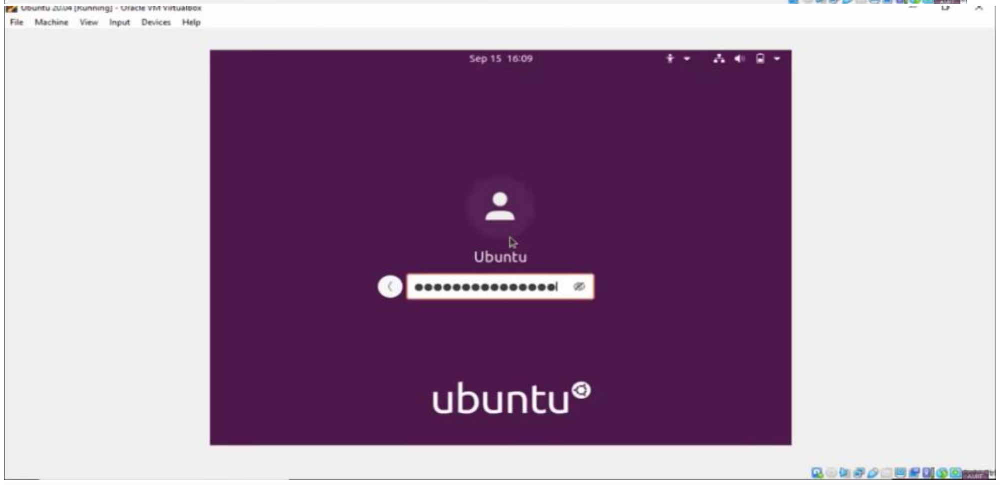
10402 MB

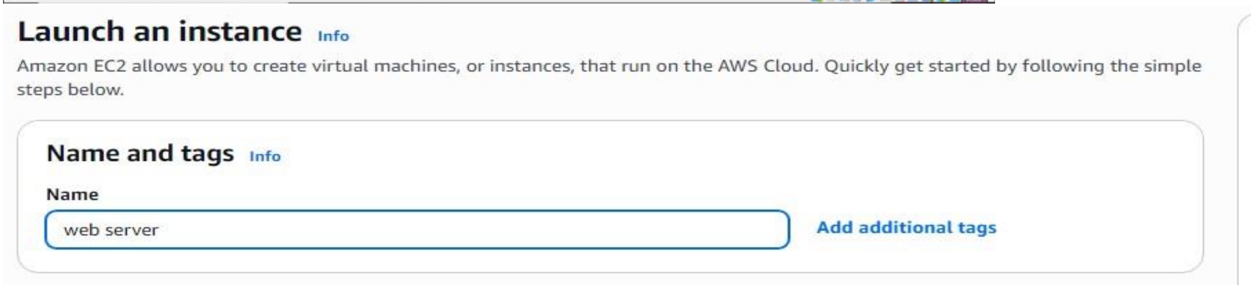
4 MB 12288 MB

Next Cancel


Ubuntu 20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Sep 15 16:17
amit@ubuntu-Amit: ~

```
amit@ubuntu-Amit:~$
```


Ubuntu 20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Sep 15 16:09
Ubuntu
ubuntu



Launch an instance Info

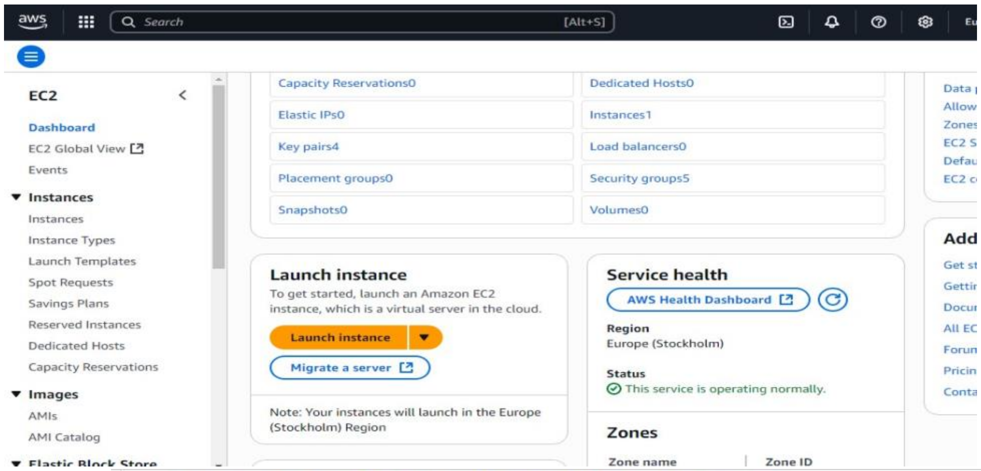
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

web server

Add additional tags



Capacity Reservations0	Dedicated Hosts0
Elastic IPs0	Instances1
Key pairs4	Load balancers0
Placement groups0	Security groups5
Snapshots0	Volumes0

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the Europe (Stockholm) Region

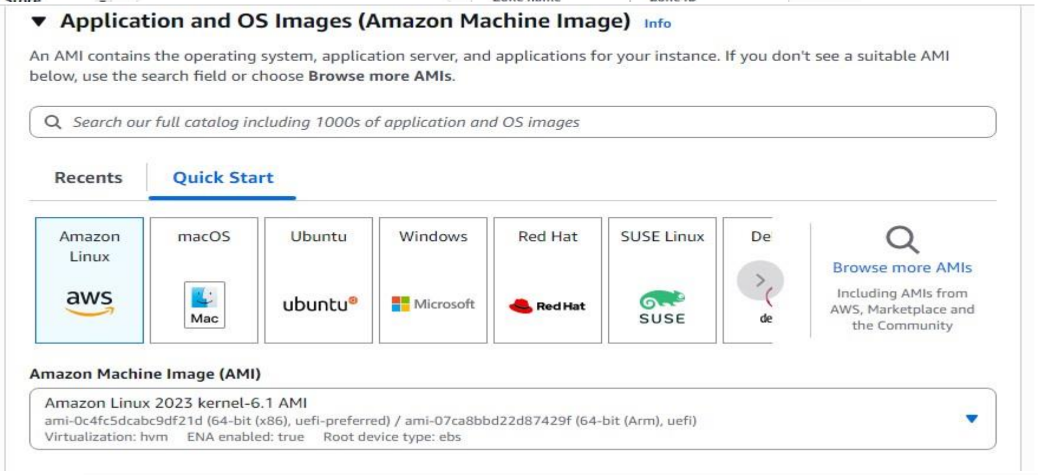
Service health

Region
Europe (Stockholm)

Status
This service is operating normally.

Zones

Zone name | Zone ID

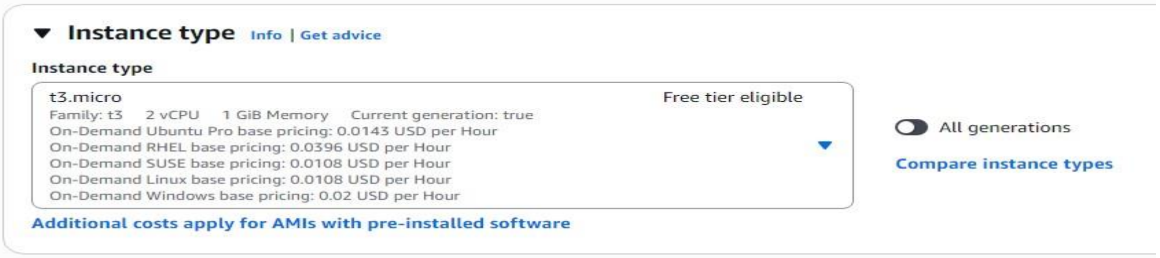

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Linux
De
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

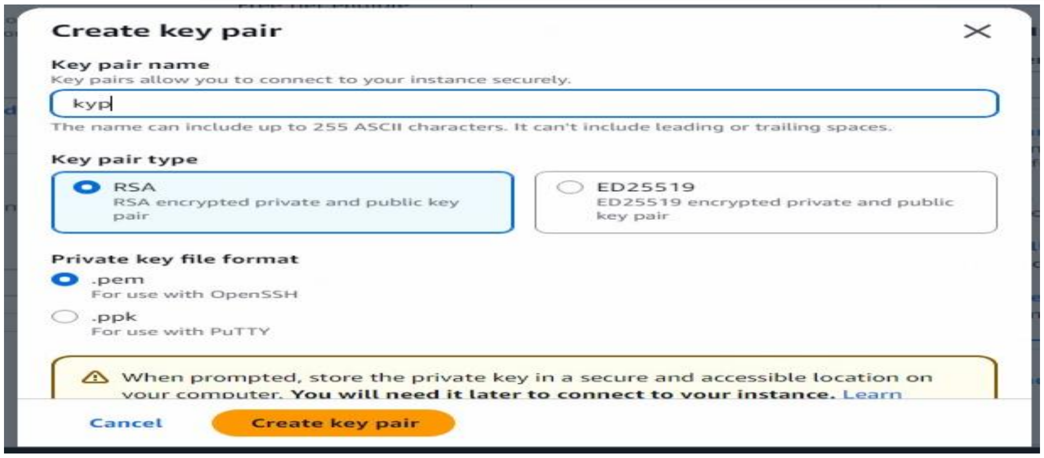
Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI
ami-0c4fc5dcabc9df21d (64-bit (x86), uefi-preferred) / ami-07ca8bbd22d87429f (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs


Instance type Info | Get advice

Instance type

t3.micro Free tier eligible
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Ubuntu Pro base pricing: 0.0143 USD per Hour
On-Demand RHEL base pricing: 0.0396 USD per Hour
On-Demand SUSE base pricing: 0.0108 USD per Hour
On-Demand Linux base pricing: 0.0108 USD per Hour
On-Demand Windows base pricing: 0.02 USD per Hour


Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

kyp

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

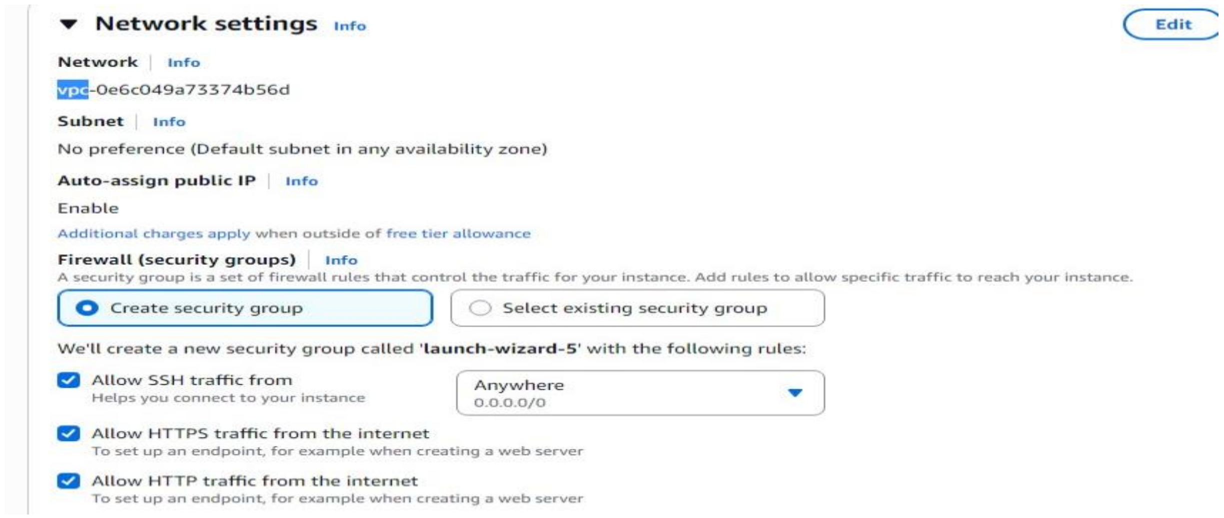
Key pair type

[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair

Private key file format

[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn


Network settings Info

Edit

Network | Info

vpc-0e6c049a73374b56d

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

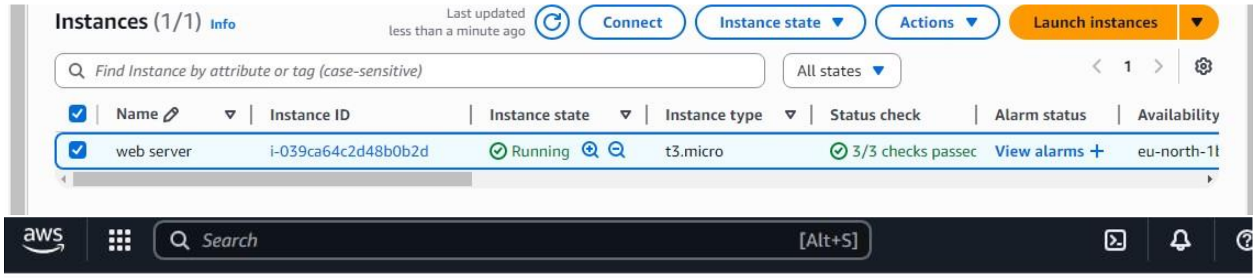
[✓ Create security group] Select existing security group

We'll create a new security group called 'launch-wizard-5' with the following rules:

✓ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

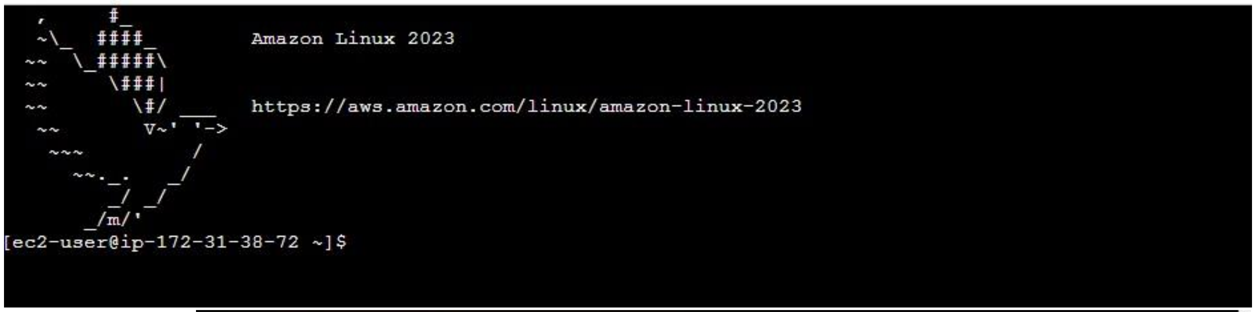
✓ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

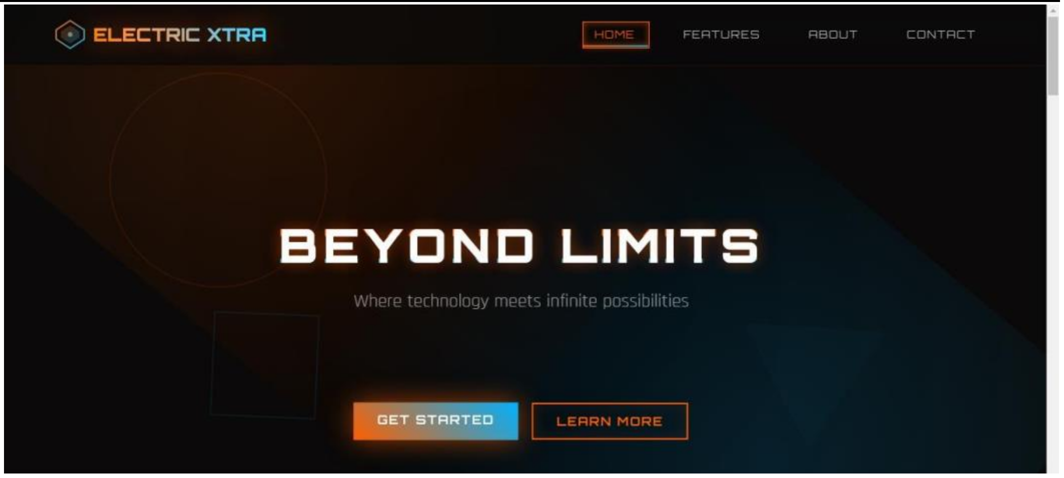
✓ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server


Instances (1/1) Info

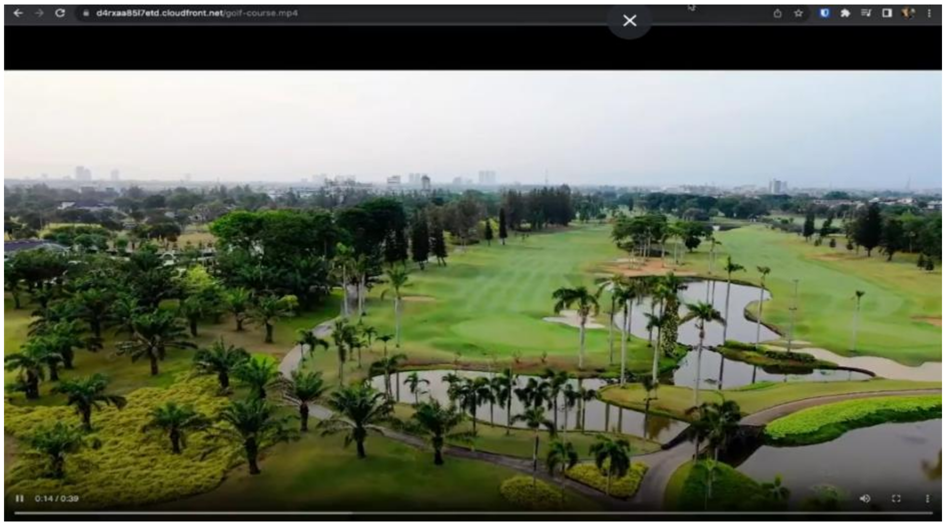
Last updated less than a minute ago

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
web server	i-039ca64c2d48b0b2d	Running	t3.micro	3/3 checks passec	View alarms +	eu-north-1t

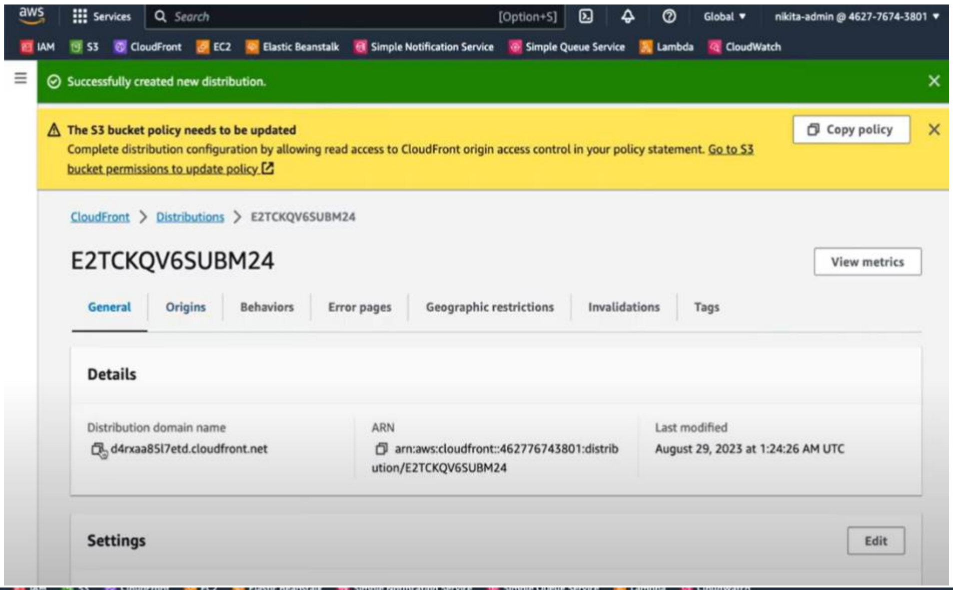

```
  ,     #_
  ~\_  ####_        Amazon Linux 2023
 ~~  \_#####\
 ~~     \###|
 ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
  ~~       V~' '->
   ~~~         /
     ~~._.   _/
        _/ _/
      _/m/'
[ec2-user@ip-172-31-38-72 ~]$
```



Exp 4



:



Successfully created new distribution.

The S3 bucket policy needs to be updated
Complete distribution configuration by allowing read access to CloudFront origin access control in your policy statement. Go to S3 bucket permissions to update policy

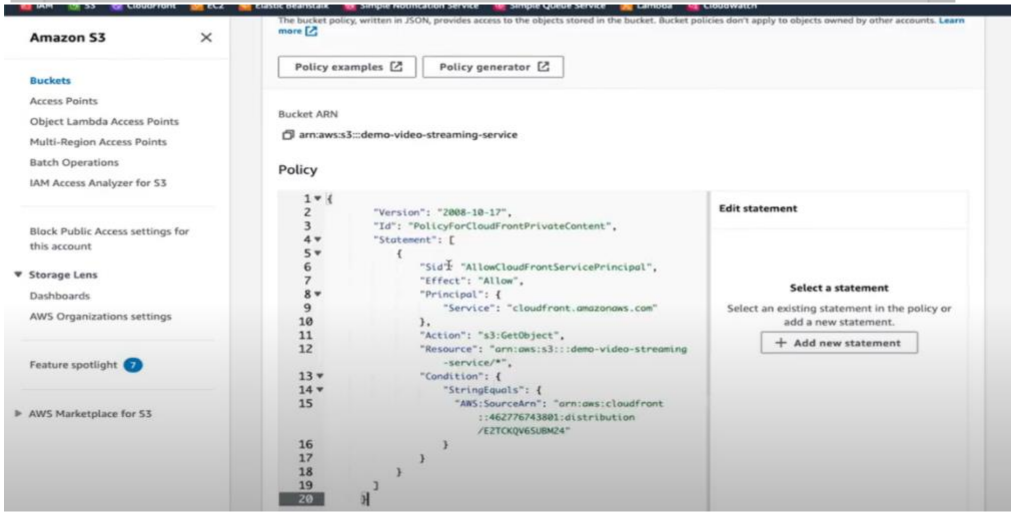
CloudFront > Distributions > E2TCKQV6SUBM24

E2TCKQV6SUBM24

Details

Distribution domain name	ARN	Last modified
d4rxaa85l7etd.cloudfront.net	arn:aws:cloudfront::462776743801:distribution/E2TCKQV6SUBM24	August 29, 2023 at 1:24:26 AM UTC

Settings



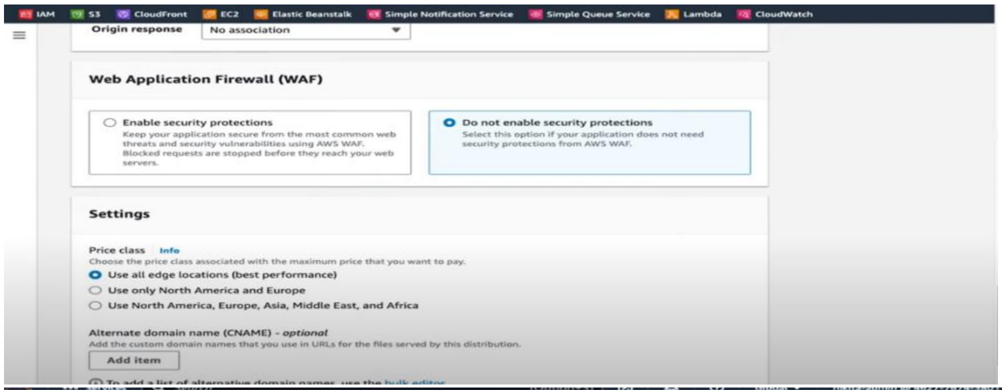
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Bucket ARN
arn:aws:s3:::demo-video-streaming-service

Policy

```
{
        "Version": "2008-10-17",
        "Id": "PolicyForCloudFrontPrivateContent",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipal",
                "Effect": "Allow",
                "Principal": {
                    "Service": "cloudfront.amazonaws.com"
                },
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::demo-video-streaming
                    -service/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn": "arn:aws:cloudfront
                          ::462776743801:distribution
                          /E2TCKQV6SUBM24"
                    }
                }
            }
        ]
    }
```

Select a statement
Select an existing statement in the policy or add a new statement.



Origin response: No association

Web Application Firewall (WAF)

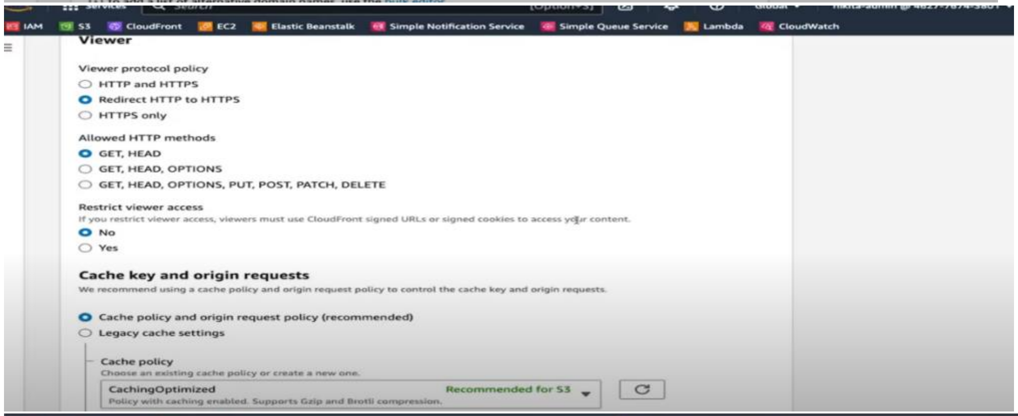
○ Enable security protections
Keep your application secure from the most common web threats and security vulnerabilities using AWS WAF. Blocked requests are stopped before they reach your web servers.

[✓ Do not enable security protections]
Select this option if your application does not need security protections from AWS WAF.

Settings

Price class Info
Choose the price class associated with the maximum price that you want to pay.
[✓ Use all edge locations (best performance)]
○ Use only North America and Europe
○ Use North America, Europe, Asia, Middle East, and Africa

Alternate domain name (CNAME) - *optional*
Add the custom domain names that you use in URLs for the files served by this distribution.
Add item



Viewer

Viewer protocol policy
○ HTTP and HTTPS
[✓ Redirect HTTP to HTTPS]
○ HTTPS only

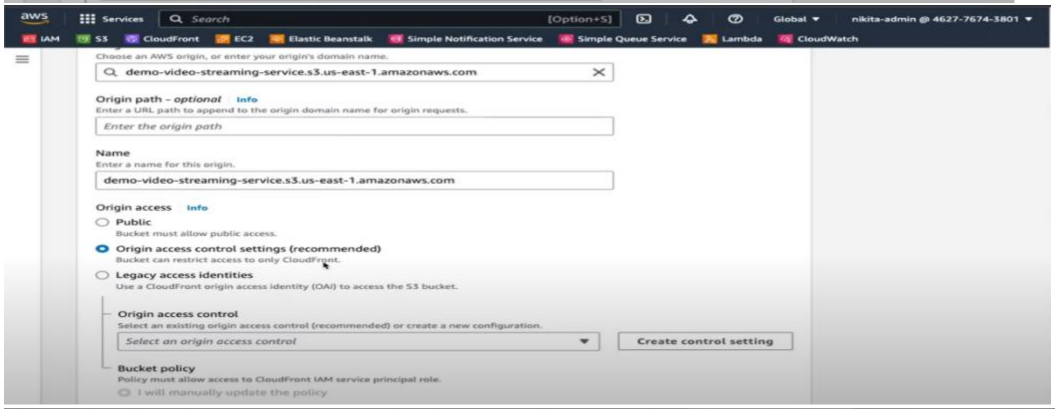
Allowed HTTP methods
[✓ GET, HEAD]
○ GET, HEAD, OPTIONS
○ GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Restrict viewer access
If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.
[✓ No]
○ Yes

Cache key and origin requests
We recommend using a cache policy and origin request policy to control the cache key and origin requests.

[✓ Cache policy and origin request policy (recommended)]
○ Legacy cache settings

Cache policy
Choose an existing cache policy or create a new one.
CachingOptimized — Recommended for S3
Policy with caching enabled. Supports Gzip and Brotli compression.



Choose an AWS origin, or enter your origin's domain name.
demo-video-streaming-service.s3.us-east-1.amazonaws.com

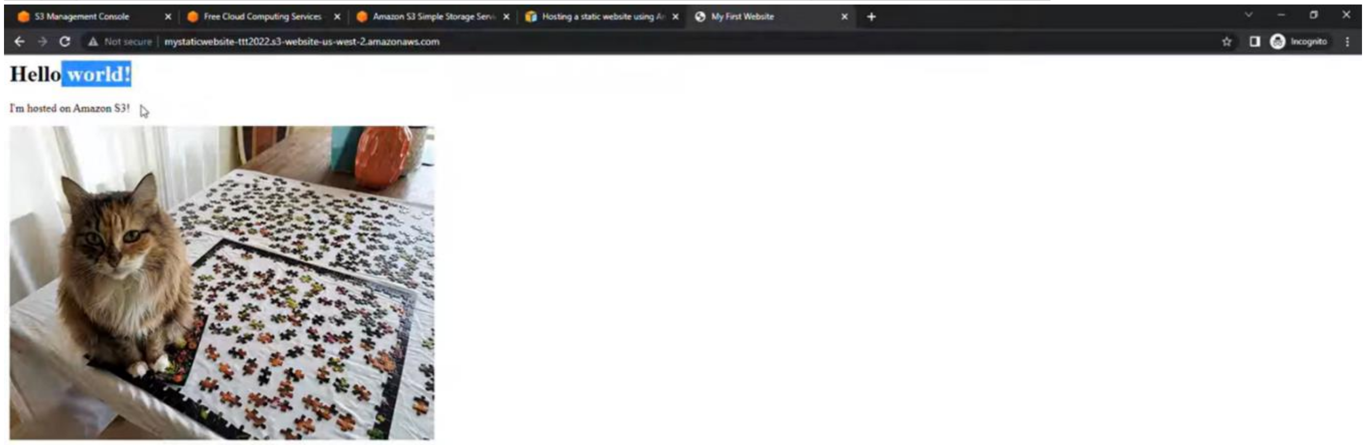
Origin path - *optional* Info
Enter a URL path to append to the origin domain name for origin requests.
Enter the origin path

Name
Enter a name for this origin.
demo-video-streaming-service.s3.us-east-1.amazonaws.com

Origin access Info
○ Public
Bucket must allow public access.
[✓ Origin access control settings (recommended)]
Bucket can restrict access to only CloudFront.
○ Legacy access identities
Use a CloudFront origin access identity (OAI) to access the S3 bucket.

Origin access control
Select an existing origin access control (recommended) or create a new configuration.
Select an origin access control
Create control setting

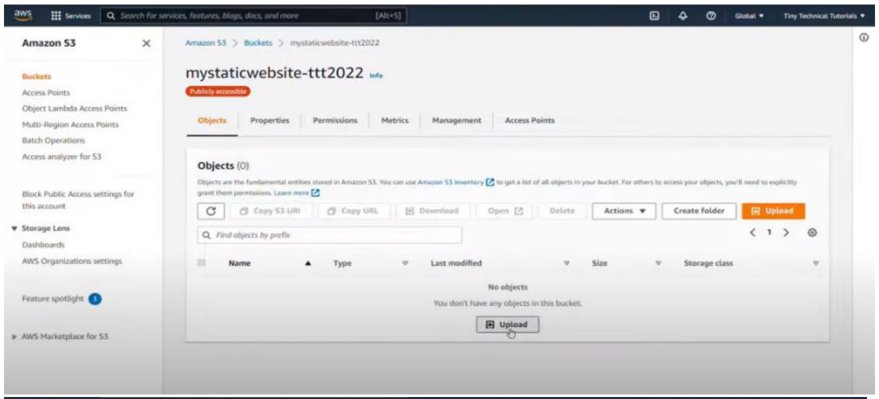
Bucket policy
Policy must allow access to CloudFront IAM service principal role.
○ I will manually update the policy


S3 Management Console
Free Cloud Computing Services
Amazon S3 Simple Storage Serv
Hosting a static website using A
My First Website
Not secure
Incognito

mystaticwebsite-ttt2022.s3-website-us-west-2.amazonaws.com

Hello world!

I'm hosted on Amazon S3!



Amazon S3 > Buckets > mystaticwebsite-ttt2022

mystaticwebsite-ttt2022 Info

Publicly accessible

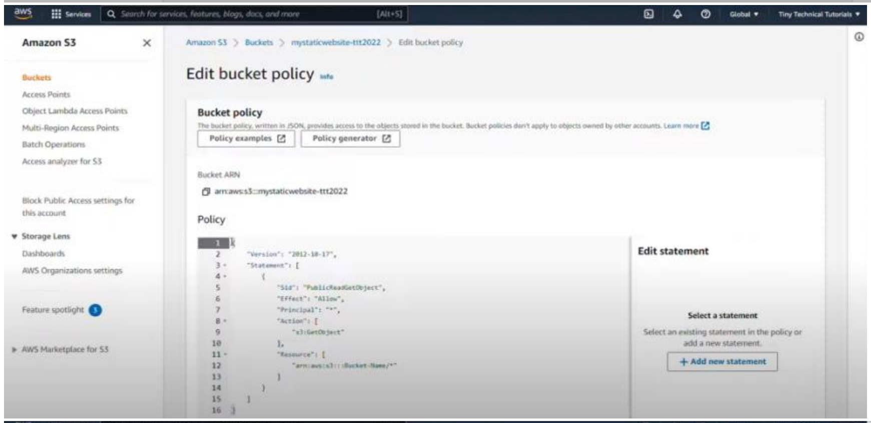
Objects (0)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this bucket.



Amazon S3 > Buckets > mystaticwebsite-ttt2022 > Edit bucket policy

Edit bucket policy Info

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

Bucket ARN

arn:aws:s3:::mystaticwebsite-ttt2022

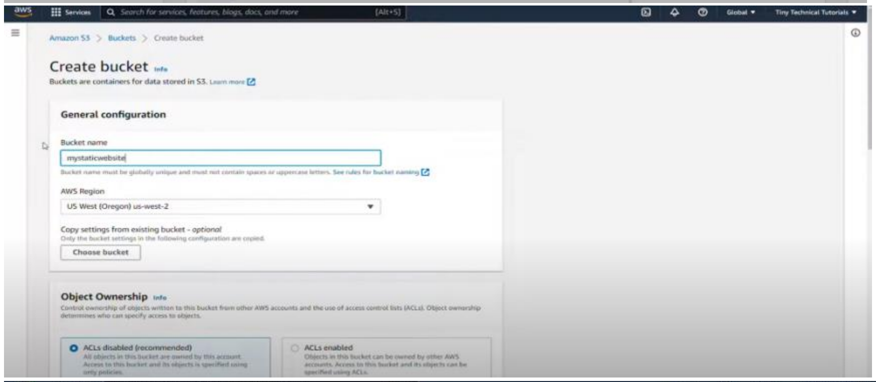
Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.


Services
Search for services, features, blogs, docs, and more [Alt+S]
Global
Tiny Technical Tutorials
Choose bucket

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name

mystaticwebsite

Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming

AWS Region

US West (Oregon) us-west-2

Copy settings from existing bucket - *optional*

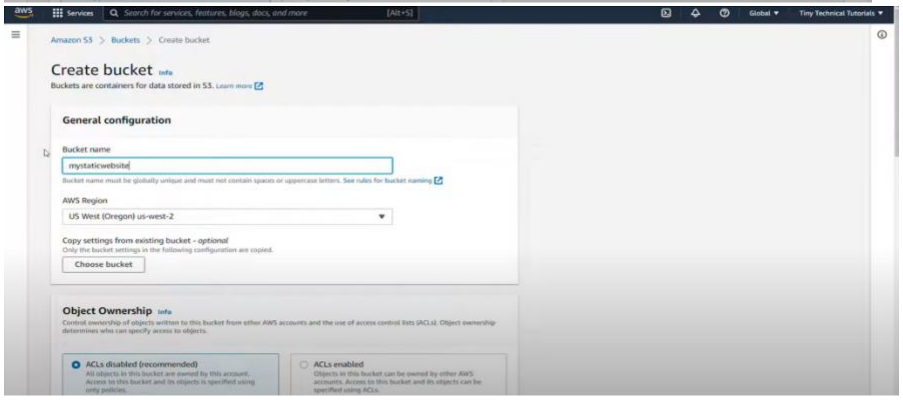
Only the bucket settings in the following configuration are copied.

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


Services
Search for services, features, blogs, docs, and more [Alt+S]
Global
Tiny Technical Tutorials
Choose bucket

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name

mystaticwebsite

Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming

AWS Region

US West (Oregon) us-west-2

Copy settings from existing bucket - *optional*

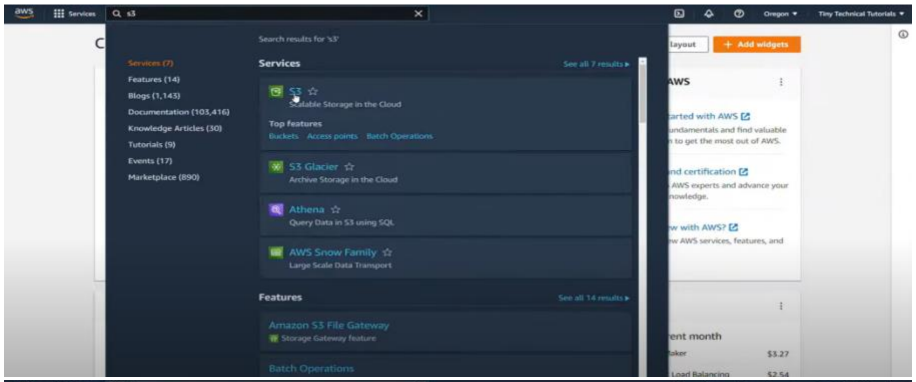
Only the bucket settings in the following configuration are copied.

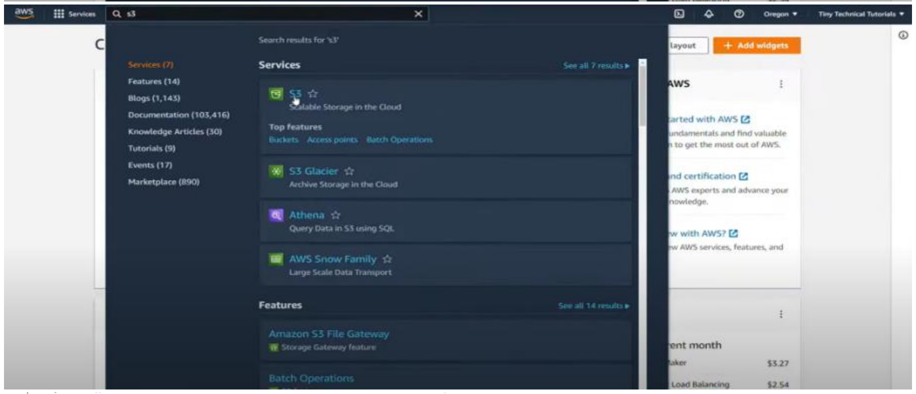
Object Ownership Info

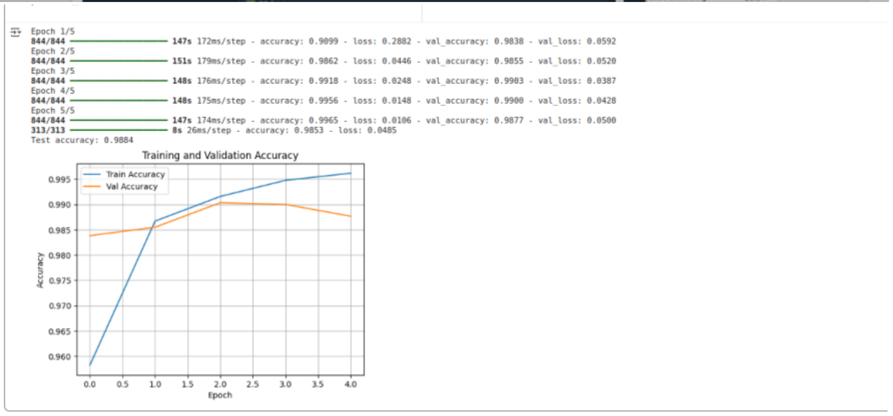
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

[✓ 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.

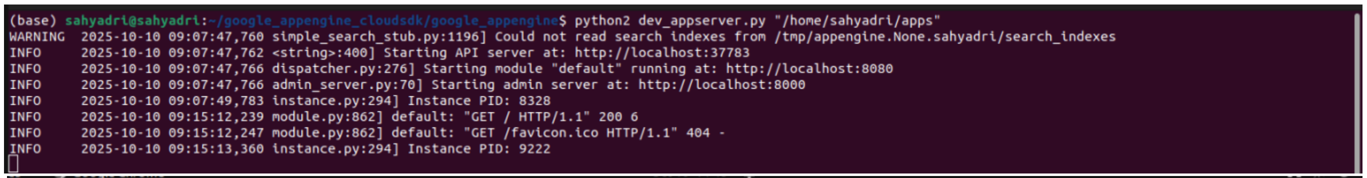
ACLs enabled Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.



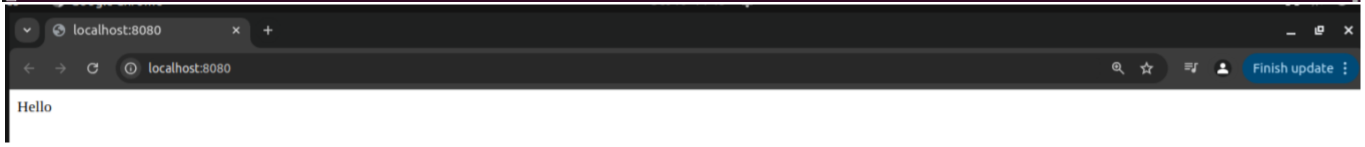




```
Epoch 1/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 147s 172ms/step - accuracy: 0.9099 - loss: 0.2882 - val_accuracy: 0.9838 - val_loss: 0.0592
Epoch 2/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 151s 179ms/step - accuracy: 0.9862 - loss: 0.0446 - val_accuracy: 0.9855 - val_loss: 0.0520
Epoch 3/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 148s 176ms/step - accuracy: 0.9918 - loss: 0.0248 - val_accuracy: 0.9903 - val_loss: 0.0387
Epoch 4/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 148s 175ms/step - accuracy: 0.9956 - loss: 0.0148 - val_accuracy: 0.9900 - val_loss: 0.0428
Epoch 5/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 147s 174ms/step - accuracy: 0.9965 - loss: 0.0106 - val_accuracy: 0.9877 - val_loss: 0.0500
313/313 ━━━━━━━━━━━━━━━━━━━━ 8s 26ms/step - accuracy: 0.9853 - loss: 0.0485
Test accuracy: 0.9884
```



```
(base) sahyadri@sahyadri:~/google_appengine_cloudsdk/google_appengine$ python2 dev_appserver.py "/home/sahyadri/apps"
WARNING  2025-10-10 09:07:47,760 simple_search_stub.py:1196] Could not read search indexes from /tmp/appengine.None.sahyadri/search_indexes
INFO     2025-10-10 09:07:47,762 <string>:400] Starting API server at: http://localhost:37783
INFO     2025-10-10 09:07:47,766 dispatcher.py:276] Starting module "default" running at: http://localhost:8080
INFO     2025-10-10 09:07:47,766 admin_server.py:70] Starting admin server at: http://localhost:8000
INFO     2025-10-10 09:07:49,783 instance.py:294] Instance PID: 8328
INFO     2025-10-10 09:15:12,239 module.py:862] default: "GET / HTTP/1.1" 200 6
INFO     2025-10-10 09:15:12,247 module.py:862] default: "GET /favicon.ico HTTP/1.1" 404 -
INFO     2025-10-10 09:15:13,360 instance.py:294] Instance PID: 9222
```



Hello

Develop asingle applcn using salesforce.com

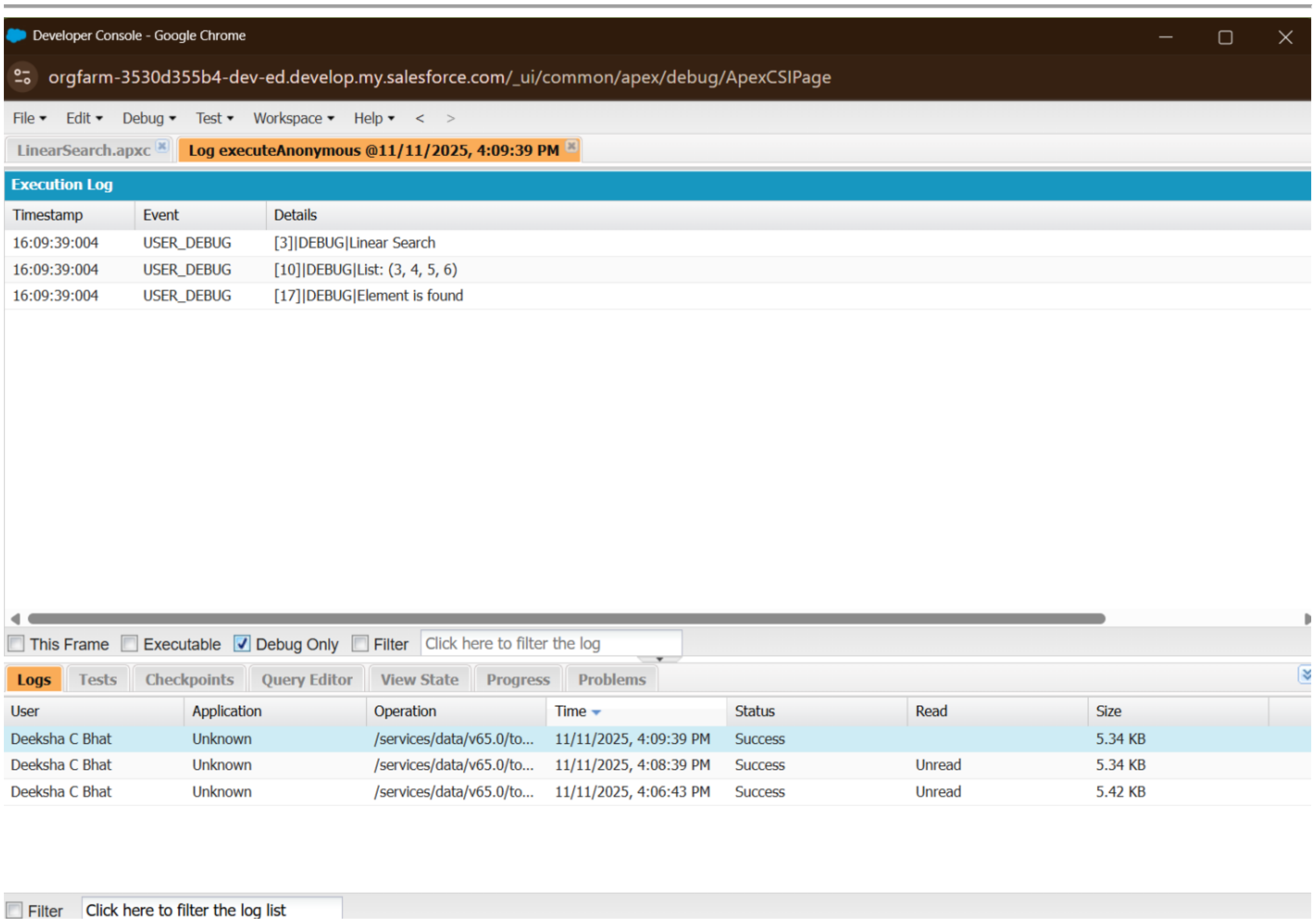
---

1. Go to the website **developer.salesforce.com** and log in using your developer account.
2. Click on **Setup** and then go to **Developer Console**.
3. In the Developer Console, go to **File → New → Apex Class**.
4. Give the class name as **LinearSearch** and click **OK**.
5. In the editor window, type the following program:

```
public class LinearSearch {
   public static void demo(Integer key){
      System.debug('Linear Search');
      integer s = -1;
      List<Integer> lon = new List<Integer>();
      lon.add(3);
      lon.add(4);
      lon.add(5);
      lon.add(6);
      System.debug('List: '+lon);
      for(integer i = 0; i < lon.size(); i++) {
         if (key == lon[i]) {
            s = 1;
         }
      }
      if (s == 1)
         System.debug('Element is found');
      else
         System.debug('Element not found');
   }
}
```

6. Save the file using **Ctrl + S**.
7. Click on **Debug → Open Execute Anonymous Window**.
8. In the window, type the following statement to execute the program:
9. LinearSearch.demo(3);
10. Click on **Execute**.
11. Open the log and check the output.
12. The output will show:
    - Linear Search
    - List: (3, 4, 5, 6)
    - Element is found
13. Again open **Execute Anonymous Window** and type:
14. LinearSearch.demo(11);
15. Click on **Execute**.
16. Open the log and check the output.

17. The output will show:
    - Linear Search
    - List: (3, 4, 5, 6)
    - Element not found
18. Hence, the program for **creating an application in salesforce.com using Apex programming language** is successfully executed and verified.


Developer Console - Google Chrome
orgfarm-3530d355b4-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File
Edit
Debug
Test
Workspace
Help
LinearSearch.apxc
Log executeAnonymous @11/11/2025, 4:09:39 PM
Execution Log
This Frame
Executable
Debug Only
Filter
Click here to filter the log
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Filter
Click here to filter the log list

Timestamp	Event	Details
16:09:39:004	USER_DEBUG	[3]|DEBUG|Linear Search
16:09:39:004	USER_DEBUG	[10]|DEBUG|List: (3, 4, 5, 6)
16:09:39:004	USER_DEBUG	[17]|DEBUG|Element is found

User	Application	Operation	Time	Status	Read	Size
Deeksha C Bhat	Unknown	/services/data/v65.0/to...	11/11/2025, 4:09:39 PM	Success		5.34 KB
Deeksha C Bhat	Unknown	/services/data/v65.0/to...	11/11/2025, 4:08:39 PM	Success	Unread	5.34 KB
Deeksha C Bhat	Unknown	/services/data/v65.0/to...	11/11/2025, 4:06:43 PM	Success	Unread	5.42 KB

Email salesforce

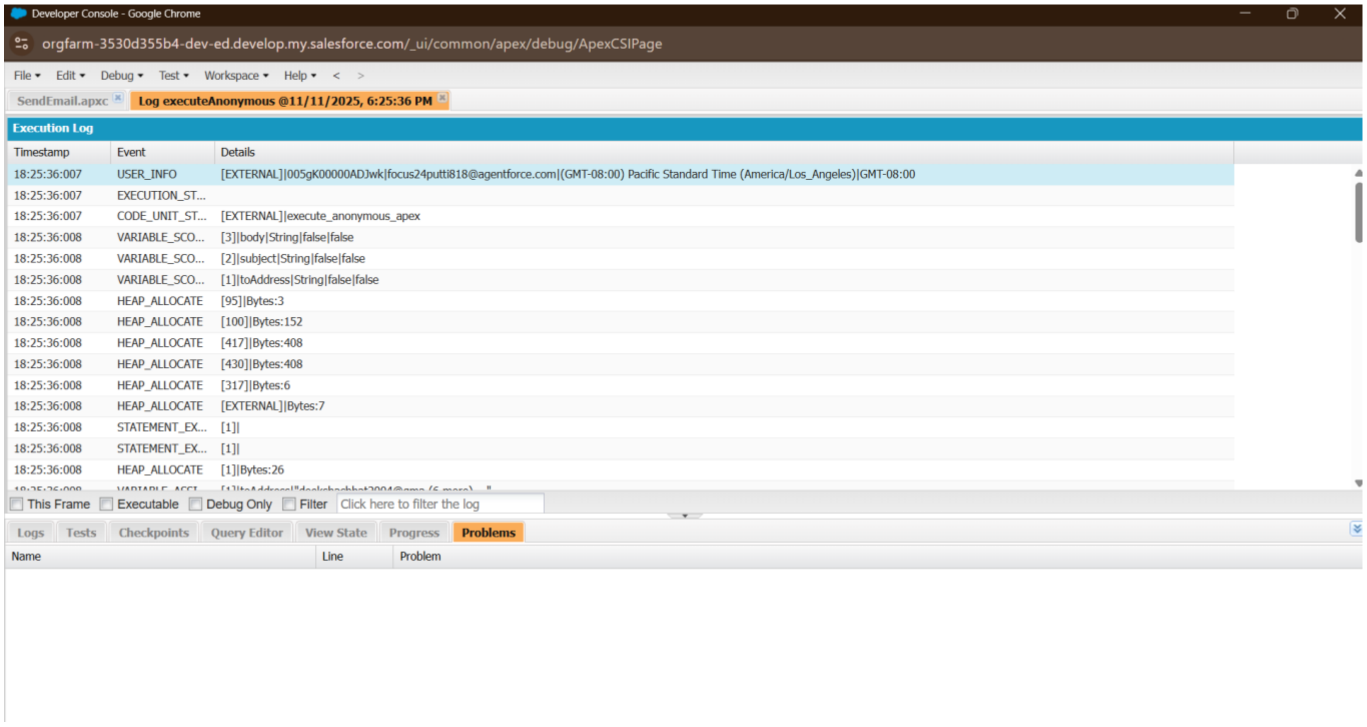
1. Open Salesforce Developer Console.
2. Go to **File → New → Apex Class**.
3. Name the class SendEmail.
4. Copy and paste this code:

```
public class SendEmail {
   public static void sendEmailMethod(String toAddress, String subject, String body) {
      Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
      email.setToAddresses(new String[] {toAddress});
      email.setSubject(subject);
      email.setPlainTextBody(body);
      Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email});
   }
}
```

5. Save the file.
6. Go to **Debug → Open Execute Anonymous Window**.
7. Paste the following code there:

```
String toAddress='deekshacbhat2004@gmail.com';
String subject='Salesforce_Email_CCLab';
String body='This is a lab program for 7th sem subject Cloud Computing and Security';
SendEmail.sendEmailMethod(toAddress, subject, body);
```

8. Check the box **Open Log** if you want to view logs.
9. Click **Execute**.
10. If you get an error saying *no single mail permission*, go to **Setup → Deliverability**.
11. Change **Access Level** from *System email only* to *All email*.
12. Save changes and execute the anonymous block again.
13. Check the recipient email inbox — you should receive the message.

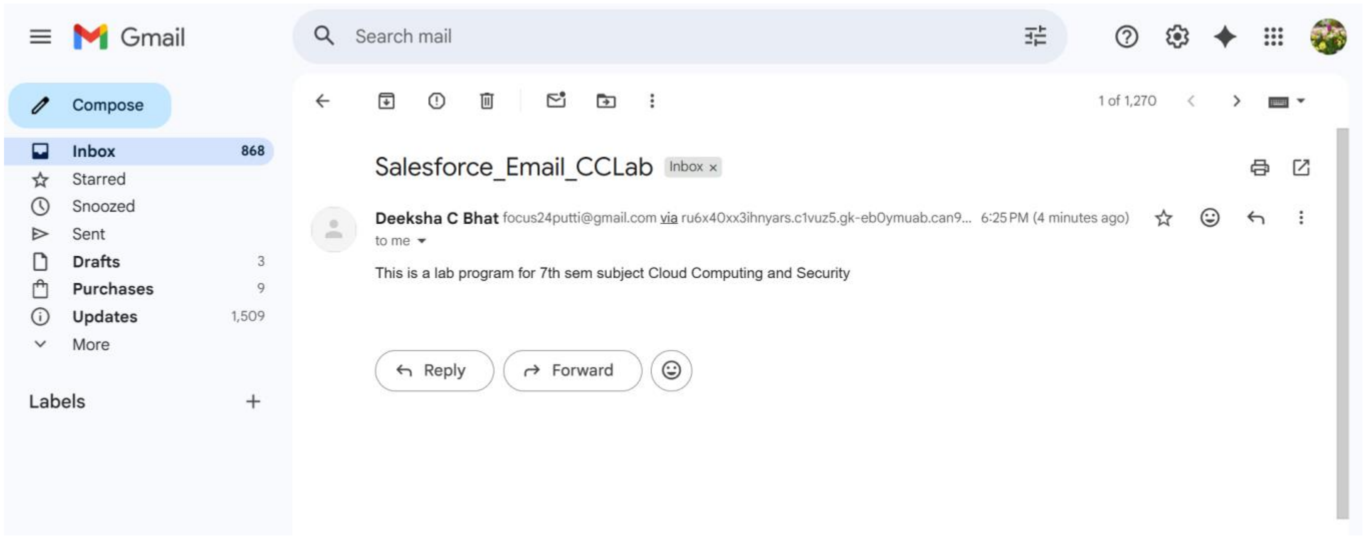

Developer Console - Google Chrome

orgfarm-3530d355b4-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help
SendEmail.apxc
Log executeAnonymous @11/11/2025, 6:25:36 PM
Execution Log

```
Timestamp     Event              Details
18:25:36:007  USER_INFO          [EXTERNAL]|005gK00000ADJwk|focus24putti818@agentforce.com|(GMT-08:00) Pacific Standard Time (America/Los_Angeles)|GMT-08:00
18:25:36:007  EXECUTION_ST...
18:25:36:007  CODE_UNIT_ST...    [EXTERNAL]|execute_anonymous_apex
18:25:36:008  VARIABLE_SCO...    [3]|body|String|false|false
18:25:36:008  VARIABLE_SCO...    [2]|subject|String|false|false
18:25:36:008  VARIABLE_SCO...    [1]|toAddress|String|false|false
18:25:36:008  HEAP_ALLOCATE      [95]|Bytes:3
18:25:36:008  HEAP_ALLOCATE      [100]|Bytes:152
18:25:36:008  HEAP_ALLOCATE      [417]|Bytes:408
18:25:36:008  HEAP_ALLOCATE      [430]|Bytes:408
18:25:36:008  HEAP_ALLOCATE      [317]|Bytes:6
18:25:36:008  HEAP_ALLOCATE      [EXTERNAL]|Bytes:7
18:25:36:008  STATEMENT_EX...    [1]|
18:25:36:008  STATEMENT_EX...    [1]|
18:25:36:008  HEAP_ALLOCATE      [1]|Bytes:26
```

This Frame Executable Debug Only Filter Click here to filter the log
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem



Salesforce_Email_CCLab

Deeksha C Bhat focus24putti@gmail.com via ru6x40xx3ihnyars.c1vuz5.gk-eb0ymuab.can9... 6:25 PM (4 minutes ago)
to me

This is a lab program for 7th sem subject Cloud Computing and Security

Custom vision-azure

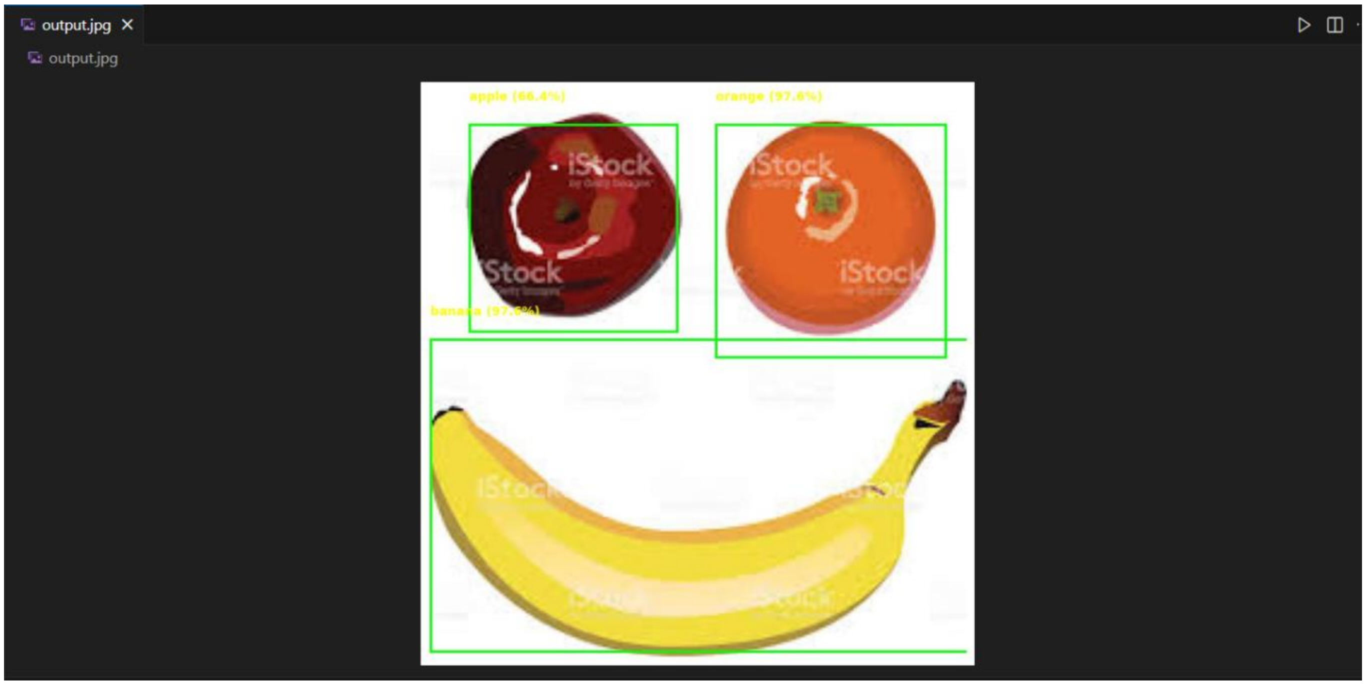
1. Go to the **Azure portal** and create a **Custom Vision Service**.
2. Select your **subscription** and **resource group**.
3. Choose the **region** and give the name **CustomVisionDemoService**.
4. Select **Pricing Tier** as *Free (F0)*.
5. Select **Prediction Tier** as *Standard (S0)*.
6. Two resources will be created — one for **training** and one for **prediction**.
7. Once the resources are created, open them and copy the **Key** and **Endpoint** from the *Keys and Endpoint* section.

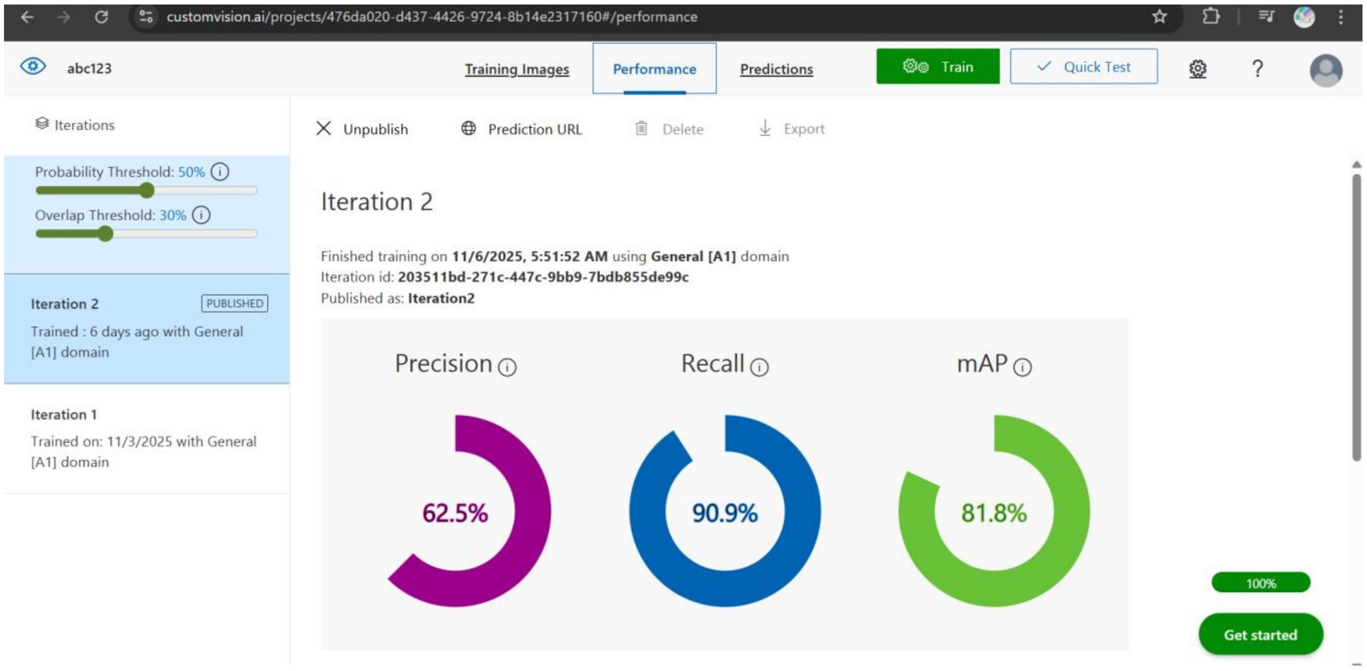
8. Go to the **Custom Vision Portal** (https://www.customvision.ai/).
9. Sign in using your Azure account.
10. Click on **New Project** and give the project name **Detect Fruit Project**.
11. Enter a short description for the project.
12. Select the **Custom Vision service** that was created.
13. Choose **Project Type** as *Object Detection*.
14. Select **Domain** as *General* and click **Create Project**.
15. In the project, click **Add Images** and upload the **Fruit Dataset**.
16. Select each image, draw the box around the object, and label it (e.g., *Orange*, *Banana*).
17. Label all images manually.
18. Click on the **Tagged** section to view labeled data and **Untagged** to see unlabeled ones.
19. After labeling, click **Train** to train the model.
20. Select **Quick Training** so that the model trains quickly.
21. Wait till the model is trained.
22. After training, note down the **Precision**, **Recall**, and **mAP** scores shown.
23. When the accuracy looks good, click the **Publish** icon to publish the model.
24. Give the model name as **Iteration2**.
25. Select the **Prediction Resource** and click **Publish**.
26. After publishing, click on the **Settings** icon to get the **Project ID**, **Keys**, and **Endpoints**.
27. Copy the **Endpoint**, **Prediction Key**, **Project ID**, and **Iteration Name**.
28. Open **Visual Studio Code**.
29. Paste these details (Endpoint, Key, Project ID, Model Name) in the code.
30. Copy and paste the following code:

```
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient
from msrest.authentication import ApiKeyCredentials
from matplotlib import pyplot as plt
from PIL import Image, ImageDraw

ENDPOINT = "https://<your-endpoint>.cognitiveservices.azure.com/"
PREDICTION_KEY = "<your-prediction-key>"
PROJECT_ID = "<your-project-id>"
MODEL_NAME = "Iteration2"

credentials = ApiKeyCredentials(in_headers={"Prediction-key": PREDICTION_KEY})
predictor = CustomVisionPredictionClient(endpoint=ENDPOINT, credentials=credentials)

IMAGE_PATH = "fruit.jpg"
image_data = open(IMAGE_PATH, "rb")

results = predictor.detect_image(PROJECT_ID, MODEL_NAME, image_data)

image = Image.open(IMAGE_PATH)
draw = ImageDraw.Draw(image)
```

```
for prediction in results.predictions:
    left = prediction.bounding_box.left * image.width
    top = prediction.bounding_box.top * image.height
    width = prediction.bounding_box.width * image.width
    height = prediction.bounding_box.height * image.height

    draw.rectangle([left, top, left + width, top + height], outline="red", width=3)
    draw.text((left, top), f"{prediction.tag_name}: {round(prediction.probability * 100, 2)}%",
fill="yellow")

plt.imshow(image)
plt.axis("off")
plt.show()
image.save("output.jpg")
```

31. Save the file and run it in Visual Studio Code.
32. The model connects to Azure Custom Vision Service and detects the fruits in the image.
33. The detected fruits are displayed with bounding boxes and confidence scores.
34. The output is stored in **output.jpg**.
35. The program successfully identifies fruits such as orange, banana, and apple with confidence scores around 99%.





Probability Threshold: 50%

Overlap Threshold: 30%

Iteration 2 PUBLISHED

Trained : 6 days ago with General [A1] domain

Iteration 1

Trained on: 11/3/2025 with General [A1] domain

Iteration 2

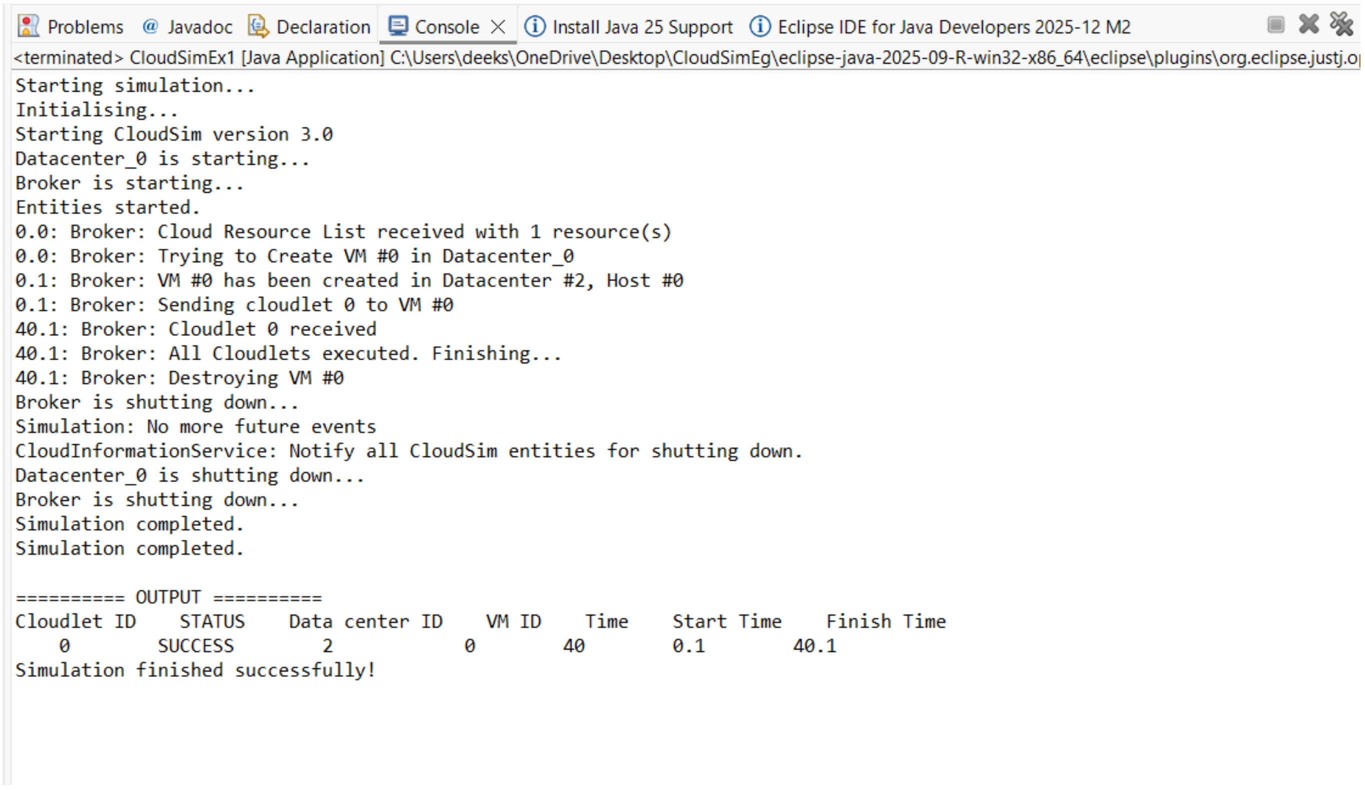
Finished training on **11/6/2025, 5:51:52 AM** using **General [A1]** domain
Iteration id: **203511bd-271c-447c-9bb9-7bdb855de99c**
Published as: **Iteration2**

Precision	Recall	mAP
62.5%	90.9%	81.8%

Cloudsim

1. Install JDK and Eclipse IDE for Java Developers.
2. Download the CloudSim 3.0.3 zip file and extract it.
3. Download the commons-math3-3.6.1-bin.zip file, extract it, and locate the file named commons-math3-3.6.1.jar.
4. Open Eclipse and create a new Java project.
5. Uncheck “Use default location” and browse to the extracted CloudSim 3.0.3 folder.
6. Click “Finish” to create the project.
7. In the Eclipse Project Explorer, verify that the folders examples, sources, and jars are visible.
8. Create a folder named lib inside the project directory and copy commons-math3-3.6.1.jar into it.
9. In Eclipse, right-click the project → Properties → Java Build Path → Libraries.
10. Remove any missing entries and click **Add JARs...** to add:
    - cloudsim-3.0.3.jar
    - cloudsim-examples-3.0.3.jar
    - commons-math3-3.6.1.jar
11. Ensure all jars are listed under **Classpath** (not under Modulepath).
12. In the **Order and Export** tab, tick the checkboxes for the above jar files.
13. Delete module-info.java if present in the src folder.
14. Clean the project using **Project → Clean...** and rebuild it.
15. Right-click the examples folder → New → Package → name it mycloudsim.
16. Right-click the mycloudsim package → New → Class → name it CloudSimEx1, and check “public static void main(String[] args)”.
17. Paste the CloudSim example code into the newly created class file.
18. If a package name is specified, ensure the first line of code matches it (e.g., package mycloudsim;).

19. Save the file and open **Window → Show View → Console**.
20. Right-click the file → Run As → Java Application.
21. If execution does not start, open **Run → Run Configurations → Java Application → New Configuration**.
22. Set the project name and main class (for example, mycloudsim.CloudSimEx1).
23. On the **Classpath** tab, confirm that all three jar files are included.
24. Run the configuration.
25. The expected output should display CloudSim initialization messages followed by a table with cloudlet details such as Cloudlet ID, Status, Datacenter ID, VM ID, Time, Start Time, and Finish Time.
26. If errors such as "class cannot be resolved" appear, recheck the Build Path and ensure all jars are correctly added under Classpath.
27. Modify parameters like cloudlet length, VM MIPS, or the number of VMs and rerun to observe changes in simulation output.



```
<terminated> CloudSimEx1 [Java Application] C:\Users\deeks\OneDrive\Desktop\CloudSimEg\eclipse-java-2025-09-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.o
Starting simulation...
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
40.1: Broker: Cloudlet 0 received
40.1: Broker: All Cloudlets executed. Finishing...
40.1: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0       40       0.1        40.1
Simulation finished successfully!
```

1.Distinguish between real word and non real word errors

Non-word Errors	**Real-word Errors**
Occur when a word does **not exist** in the dictionary or lexicon.	Occur when the incorrect word **exists** in the dictionary but is **used wrongly in context**.
These are **spelling mistakes** or **typographical errors** that result in invalid or unknown words.	These are **contextual errors** where a valid word is used in the wrong grammatical or semantic context.
Can be detected easily using **dictionary lookup** or **spell-checking algorithms**.	Detection is **difficult**, as it requires **contextual, syntactic, and semantic analysis**.
Correction methods involve **edit distance**, **phonetic similarity**, or **probabilistic spelling correction**.	Correction requires **context-based disambiguation** or **language modeling techniques**.
Do not form valid tokens in the language.	Form valid tokens, but are incorrect in the sentence context.
Example: *exampel → example*, *thier → their*	**Example:** *Their going to school* instead of *They're going to school*, *I red the book* instead of *I read the book*
Ease of Detection: Simple	**Ease of Detection:** Difficult
Nature of Error: Orthographic	**Nature of Error:** Contextual / Semantic

3. Comment on the validity of the following statements

(a) Rule-based taggers are non-deterministic.
Invalid.
Rule-based taggers are **deterministic systems** that apply a sequence of hand-crafted linguistic rules to assign part-of-speech (POS) tags to words. The process is based on **morphological, lexical, and contextual rules** defined by linguists.
For a given input and the same set of rules, the output of a rule-based tagger will always be the same, making it deterministic in nature. There is **no element of randomness** or probability involved.
Example: The ENGTWOL and constraint grammar taggers use deterministic rule application order to tag words.
Thus, the statement "Rule-based taggers are non-deterministic" is **false** because rule-based taggers follow a fixed, rule-governed tagging procedure that yields predictable results.

(b) Stochastic taggers are language independent.
Valid.
Stochastic taggers rely on **probabilistic models** such as Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs) that learn tagging probabilities from annotated corpora. These taggers are data-driven and do not depend on manually written linguistic rules specific to a language.
As a result, they can be trained on any language **as long as sufficient annotated training data** is available. For example, the same HMM algorithm can be trained for English, Hindi, or French without modification to the tagging logic.
Therefore, stochastic taggers are considered **language independent**, though their performance depends on the quality and quantity of the training corpus.

(c) Brill's tagger is a rule-based tagger.
Valid.
Brill's tagger, developed by Eric Brill, is a **transformation-based error-driven rule-based tagger**. It combines the strengths of both stochastic and rule-based approaches.
The tagger starts with an initial tagging (often based on the most frequent tag or stochastic method) and then **iteratively learns correction rules** to reduce tagging errors. These transformation rules are derived automatically from a tagged corpus by comparing the system's output with the correct tags.
Each rule specifies a condition under which a tag should be changed to another tag, based on the surrounding context.
Example Rule:
"Change tag from VB to VBD if the previous word is a past-tense indicator."
Hence, Brill's tagger is classified as a **rule-based tagger** because it ultimately relies on learned transformation rules rather than probability estimation.

4. How can unknown words be handled in the tagging process?

In Part-of-Speech (POS) tagging, an unknown word is one that does not appear in the training corpus or dictionary used by the tagger. Handling such words is an important task to Improve the accuracy and robustness of tagging systems.

Methods to handle unknown words:

1. Suffix and Prefix Analysis:

Many unknown words share common suffixes or prefixes with known words.
Example: Words ending with -ing are often present participles (VBG).
By examining suffix patterns, taggers can assign probable tags to unknown words.

2. Capitalization and Numeric Patterns:

If a word starts with a capital letter, it is often a proper noun (NNP).

If it contains digits, it can be a number or symbol (CD).

3. Morphological Analysis:

The word is broken down into its root form and affixes, and the POS of the root word is used to infer the tag.
Example: "Nationalization" → root "Nation" → likely noun.

4. Contextual Information (Surrounding Words):

The tag of neighboring words (previous and next words) helps predict the unknown word’s tag using contextual probabilities in stochastic taggers (like HMM).

5. Default Tagging Strategy:

If no clue is available, a default tag (often noun) is assigned since nouns are statistically the most frequent tags in text.

6. Statistical Estimation / Back-off Models:

Probabilistic taggers may use back-off strategies — relying on less specific probability distributions when the exact word is unseen.

7. Use of External Lexicons:

Additional lexical resources or domain-specific dictionaries can be used to expand the vocabulary and reduce the number of unknowns.

Example:
Sentence: She likes drishifying the design.
Here, “drishifying” is unknown.
By observing the suffix -ing, the tagger can infer it as a verb (VBG).

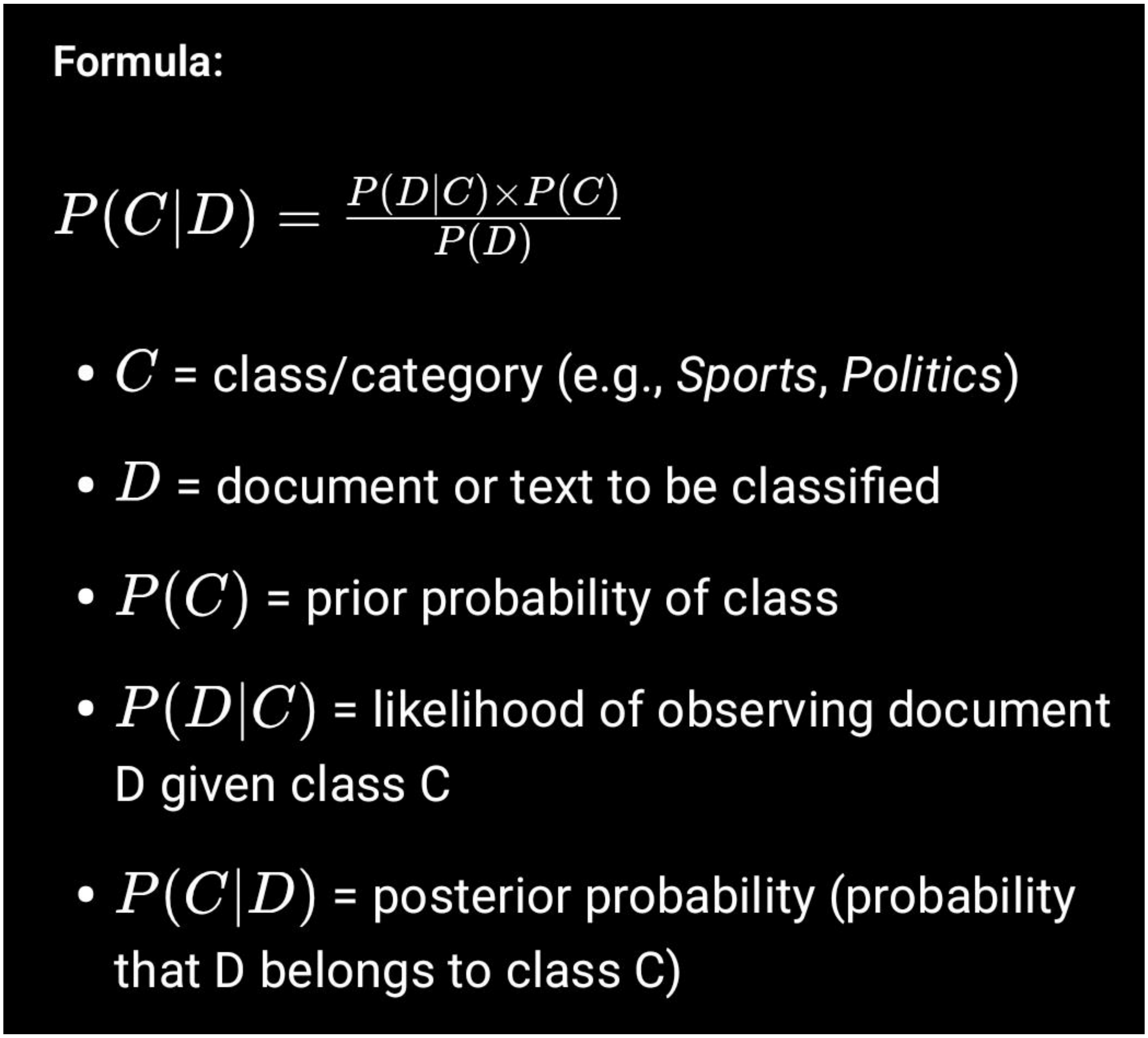
Hence, unknown word handling ensures the tagger remains accurate and generalizes well even for unseen vocabulary.

---

5. Explain how Naïve Bayes (NB) model can be used as a language model for text classification. Explain with an example.

The Naïve Bayes model is a probabilistic classifier based on Bayes’ theorem with the assumption of conditional independence among features (words). It is widely used for text classification tasks such as spam detection, topic identification, and sentiment analysis.

Formula:



Formula:

$$P(C|D) = \frac{P(D|C) \times P(C)}{P(D)}$$

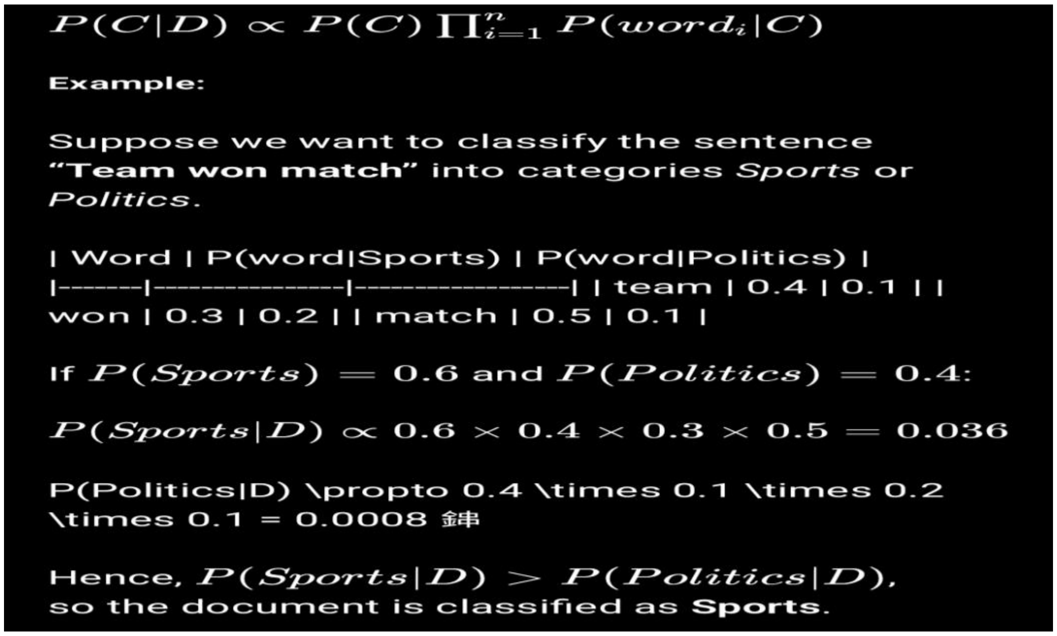
- C = class/category (e.g., *Sports*, *Politics*)
- D = document or text to be classified
- $P(C)$ = prior probability of class
- $P(D|C)$ = likelihood of observing document D given class C
- $P(C|D)$ = posterior probability (probability that D belongs to class C)

Working Steps:
1.Training Phase:
Collect labeled training data.
Compute prior probabilities ▯ for each class.
Compute likelihood probabilities ▯ using word frequencies in each class.

Compute likelihood probabilities using word frequencies in each class.

1. Classification Phase:

For a new document, compute posterior probability for each class:



$P(C|D) \propto P(C) \prod_{i=1}^{n} P(word_i|C)$

Example:

Suppose we want to classify the sentence **"Team won match"** into categories *Sports* or *Politics*.

Word	P(word|Sports)	P(word|Politics)
team	0.4	0.1
won	0.3	0.2
match	0.5	0.1

If $P(Sports) = 0.6$ and $P(Politics) = 0.4$:

$P(Sports|D) \propto 0.6 \times 0.4 \times 0.3 \times 0.5 = 0.036$

P(Politics|D) \propto 0.4 \times 0.1 \times 0.2 \times 0.1 = 0.0008 鉮

Hence, $P(Sports|D) > P(Politics|D)$, so the document is classified as **Sports**.

Thus, the NB model acts as a simple yet effective language model for text classification tasks.

---

6. Naïve Bayes model can be optimized for sentiment analysis of a text. Validate the above sentence with justification.

The statement is valid.

Naïve Bayes (NB) can indeed be optimized and effectively applied for sentiment analysis, where the goal is to classify text as positive, negative, or neutral based on sentiment-bearing words.

Justification:

1. Feature Selection / Weighting:

NB can be optimized by selecting only sentiment-related words (e.g., good, excellent, bad, poor) and removing neutral words using feature selection techniques like Chi-square or Mutual Information.

2. Use of TF-IDF Weights:

Instead of raw word counts, NB can use TF-IDF (Term Frequency–Inverse Document Frequency) to give more importance to sentiment-rich words.

3. Handling Negations:

Optimization involves incorporating negation handling (e.g., “not good” should be treated as negative). Preprocessing can create special tokens like “not_good” for better classification.

4. Laplace Smoothing:

Adding Laplace (add-one) smoothing prevents zero probabilities for unseen sentiment words, improving generalization on test data.

5. Domain-specific Training:

Training the NB classifier on domain-specific sentiment corpora (like movie or product reviews) helps it adapt to language patterns in that domain.

6. Example:

Sentence: “The movie was not interesting.”
After preprocessing → [not_interesting]
NB model learns that not_interesting has a higher probability under the negative class, leading to correct sentiment classification.

Hence, by optimizing feature selection, weighting schemes, and preprocessing, the Naïve Bayes model becomes highly effective for sentiment analysis tasks.