AWS Labs – S3, Sagemaker, Devops.

- Amazon S3:
  - Scalable object storage service used for storing files, data backups, and hosting static websites.
- Amazon SageMaker:
  - A cloud service for building, training, and deploying machine learning models at scale.
- DevOps in AWS:
  - Tools like AWS CodePipeline, CodeBuild, and CodeDeploy streamline CI/CD processes and infrastructure automation.

Demo:
Amazon S3 and CloudFront: Static Website Deployment

Task 1: Create an S3 Bucket for Static Website Hosting

Step 1: Log into the AWS Management Console

- Go to AWS Console.
- Type "S3" in the search bar, and open Amazon S3.

Step 2: Create a New S3 Bucket

- Click Create bucket.
- Enter a unique Bucket name (e.g., `user10-static-website-demo`).
- Select a Region closest to your target audience, such as Mumbai (ap-south-1).
- Under Block Public Access settings, uncheck Block all public access to allow public website access. Confirm the acknowledgment.
- Click Create bucket to finish.

Step 3: Upload Website Files

- Download the HTML and CSS files from this source link.
- Extract the files on your computer.
- In your S3 bucket, click Upload and Add files.
- Select the HTML and other files and Upload them.
- Rename the main HTML file to index.html (Actions > Rename Object).

Step 4: Enable Static Website Hosting

- Go to the Properties tab of the bucket.
- Scroll to Static website hosting and click Edit.
- Select Enable.
- Set Index Document as `index.html`.
- (Optional) Enter an error document (e.g., `404.html`) if you have one.
- Click Save changes.

Step 5: Make Files Public

- Go to the Permissions tab.
- Scroll to Bucket Policy and click Edit.
- Paste the following bucket policy, replacing `<YOUR-BUCKET-NAME-HERE>` with your bucket name:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "granting-access-to-s3-objects",
      "Principal": "*",
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": "arn:aws:s3:::<YOUR-BUCKET-NAME-HERE>/*"
    }
  ]
}
```

Click Save changes.

Step 6: Test the Static Website

- Copy the Endpoint URL from the Static website hosting section.
- Open the URL in a browser to confirm that your website is accessible.

Task 2: Create a CloudFront Distribution for the S3 Bucket

Step 1: Navigate to CloudFront

- In the AWS Console, search for CloudFront and open it.
- Click Create Distribution.

Step 2: Configure the Origin Settings

- For Origin Domain, select your S3 bucket.
- Leave Origin Path blank unless files are stored in a subfolder.
- Under Origin access, select Create new OAC to create Origin Access Control, and update the S3 bucket policy to allow CloudFront access.
- Under Default cache behavior, set CachingOptimized for the cache policy.
- In Settings, select Use all edge locations for maximum performance.
- Set Default root object to `index.html`.
- Click Create Distribution. It may take a few minutes to deploy.

Task 3: Update S3 Bucket Permissions for CloudFront Access Only

- In CloudFront, observe the yellow notification to update the bucket policy for OAC.
- Copy the suggested policy and open S3.
- Go to your bucket's Permissions > Bucket Policy > Edit.
- Replace the existing policy with the copied OAC policy and Save changes.

Task 4: Test the CloudFront Distribution

- Wait until the CloudFront distribution status is Deployed.
- Copy the CloudFront domain name (e.g., `dxxxxxxxxxx.cloudfront.net`).
- Open this URL in a browser to check if the website is accessible via CloudFront.
- Refresh to see the caching effect from CloudFront.

Task 5: Delete Resources to Avoid Costs

- Delete CloudFront Distribution:
  - Disable the CloudFront distribution.
  - Wait until the status changes to Disabled, then delete it.
- Delete the S3 Bucket:
  - Empty the bucket by deleting all files, then delete the bucket itself.

Amazon SageMaker: Iris Dataset Classification

Step 1: Prepare the Dataset
Use Python to load, split, and save the dataset:

```
from sklearn.datasets import load_iris
import pandas as pd
from sklearn.model_selection import train_test_split

iris = load_iris()
data                =              pd.DataFrame(data=iris['data'],
columns=iris['feature_names'])
data['target'] = iris['target']

train,    test    =    train_test_split(data,    test_size=0.2,
random_state=42)

train.to_csv('train.csv', index=False)
test.to_csv('test.csv', index=False)
```

Step 2: Upload Data to S3
Open the AWS CLI or AWS Console and create an S3 bucket if you don't already have one:

```
aws s3 mb s3://your-bucket-name
```

Upload the files:

```
aws s3 cp train.csv s3://your-bucket-name/train/train.csv
aws s3 cp test.csv s3://your-bucket-name/test/test.csv
```

Step 3: Create a Model in SageMaker
In SageMaker, create an XGBoost Estimator for model training:

```
from sagemaker.estimator import Estimator
import boto3
import sagemaker

role = 'your-sagemaker-role'
xgboost_container    =    sagemaker.image_uris.retrieve("xgboost",
boto3.Session().region_name, "1.2-1")

estimator = Estimator(
    image_uri=xgboost_container,
    role=role,
    instance_count=1,
    instance_type='ml.m5.large',
    output_path='s3://your-bucket/output/',
    hyperparameters={
        'max_depth': 3,
        'eta': 0.2,
        'objective': 'multi:softmax',
        'num_class': 3,
        'num_round': 100,
    }
)
```

Step 4: Train the Model

```
s3_input_train = 's3://your-bucket-name/train/train.csv'
estimator.fit({'train': s3_input_train})
```

Step 5: Deploy the Model

```
predictor = estimator.deploy(
   initial_instance_count=1,
   instance_type='ml.m5.large'
)
```

Step 6: Test the Model

Send sample data to the deployed model:

```
iris_sample = [5.1, 3.5, 1.4, 0.2]  # Example data
result = predictor.predict(iris_sample)
print(result)  # Predicted class
```

DevOps: Deploy Web Application Using AWS DevOps Tools

Step 1: Fork the Repository

Log into GitHub, navigate to the aws-elastic-beanstalk-express-js-sample repository, and select Fork.

Step 2: Deploy the Web App to Elastic Beanstalk

- In the AWS Elastic Beanstalk Console, select Create Application.
- Choose a Web server environment and enter an Application Name (e.g., `DevOpsGettingStarted`).
- Set the Platform to Node.js, and confirm other settings as default.
- Review the application and environment configuration, and then submit to create.

Step 3: Create a Build Project in CodeBuild

- In the AWS CodeBuild Console, select Create project.
- Name the project (e.g., `Build-DevOpsGettingStarted`) and select GitHub as the Source Provider.
- Authenticate with GitHub and choose the repository you forked.
- Configure the environment with Amazon Linux 2 and Runtime: Standard. Use aws/codebuild/amazonlinux2-x86_64-standard:3.0 as the image.

Use the following `buildspec.yml` file:

```
version: 0.2
phases:
    build:
        commands:
            - npm i --save
artifacts:
    files:
        - '**/*'
```

Step 4: Set Up a Delivery Pipeline in CodePipeline

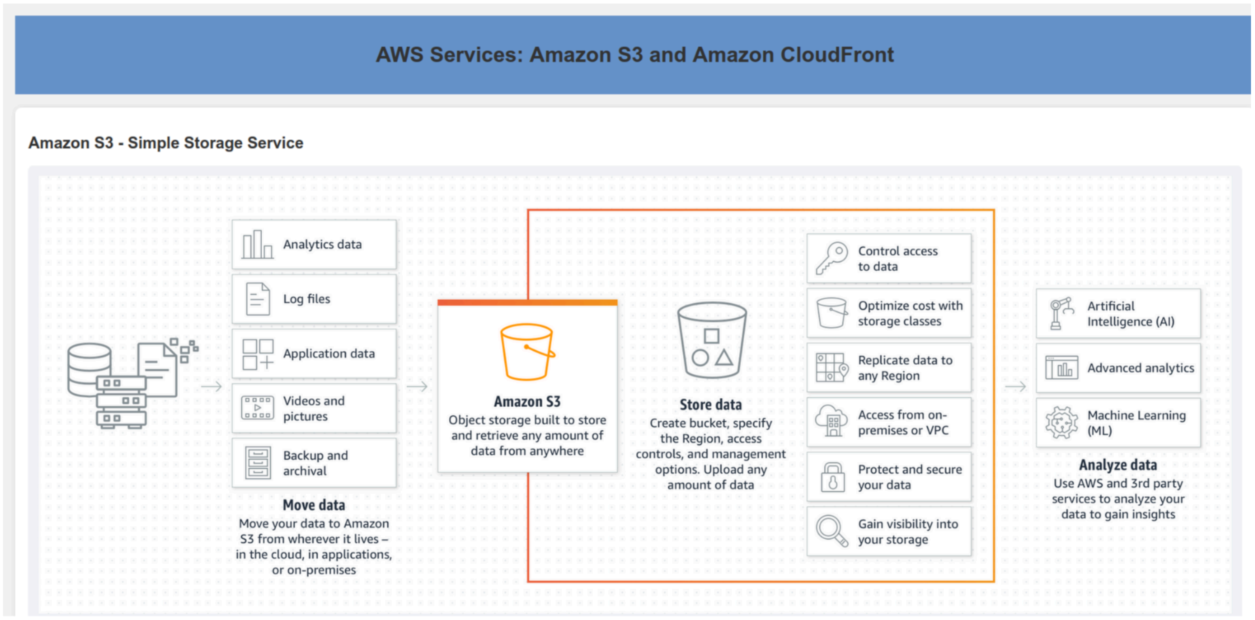
- In the AWS CodePipeline Console, choose Create pipeline.
- Select GitHub as the Source provider and authenticate to access the forked repository.
- Configure AWS CodeBuild as the Build provider and choose your `Build-DevOpsGettingStarted` project.
- For the Deploy provider, select AWS Elastic Beanstalk, and use `DevOpsGettingStarted-env` for the environment.
- Review and create the pipeline.

Step 5: Test the Pipeline

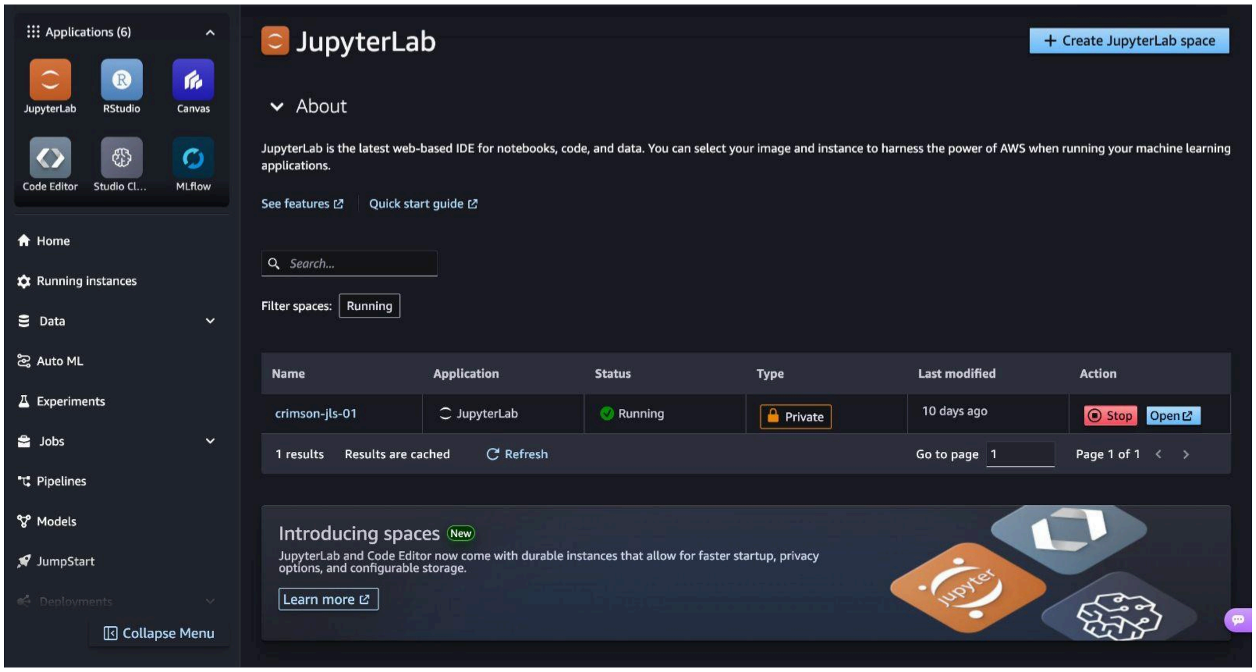
- After creation, watch the pipeline run and confirm each step completes successfully.
- Once deployed, open AWS Elastic Beanstalk, go to your environment, and click on the URL to see the deployed web app.

RESULTS - Screenshots

S3:


AWS Services: Amazon S3 and Amazon CloudFront
Amazon S3 - Simple Storage Service
Analytics data
Log files
Application data
Videos and pictures
Backup and archival
Move data
Move your data to Amazon S3 from wherever it lives – in the cloud, in applications, or on-premises
Amazon S3
Object storage built to store and retrieve any amount of data from anywhere
Store data
Create bucket, specify the Region, access controls, and management options. Upload any amount of data
Control access to data
Optimize cost with storage classes
Replicate data to any Region
Access from on-premises or VPC
Protect and secure your data
Gain visibility into your storage
Artificial Intelligence (AI)
Advanced analytics
Machine Learning (ML)
Analyze data
Use AWS and 3rd party services to analyze your data to gain insights

AWS Sagemaker:



JupyterLab

About

JupyterLab is the latest web-based IDE for notebooks, code, and data. You can select your image and instance to harness the power of AWS when running your machine learning applications.

See features Quick start guide

Search...

Filter spaces: Running

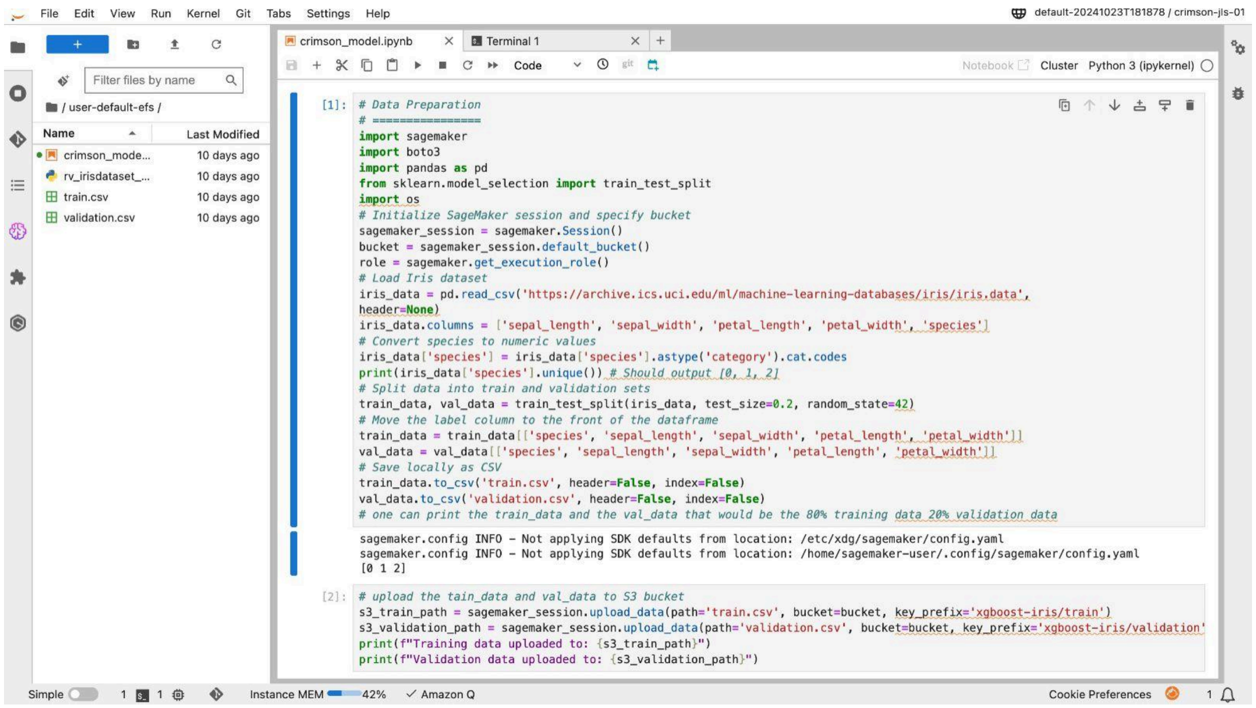
Name	Application	Status	Type	Last modified	Action
crimson-jls-01	JupyterLab	Running	Private	10 days ago	Stop Open

1 results Results are cached Refresh Go to page 1 Page 1 of 1

Introducing spaces New

JupyterLab and Code Editor now come with durable instances that allow for faster startup, privacy options, and configurable storage.

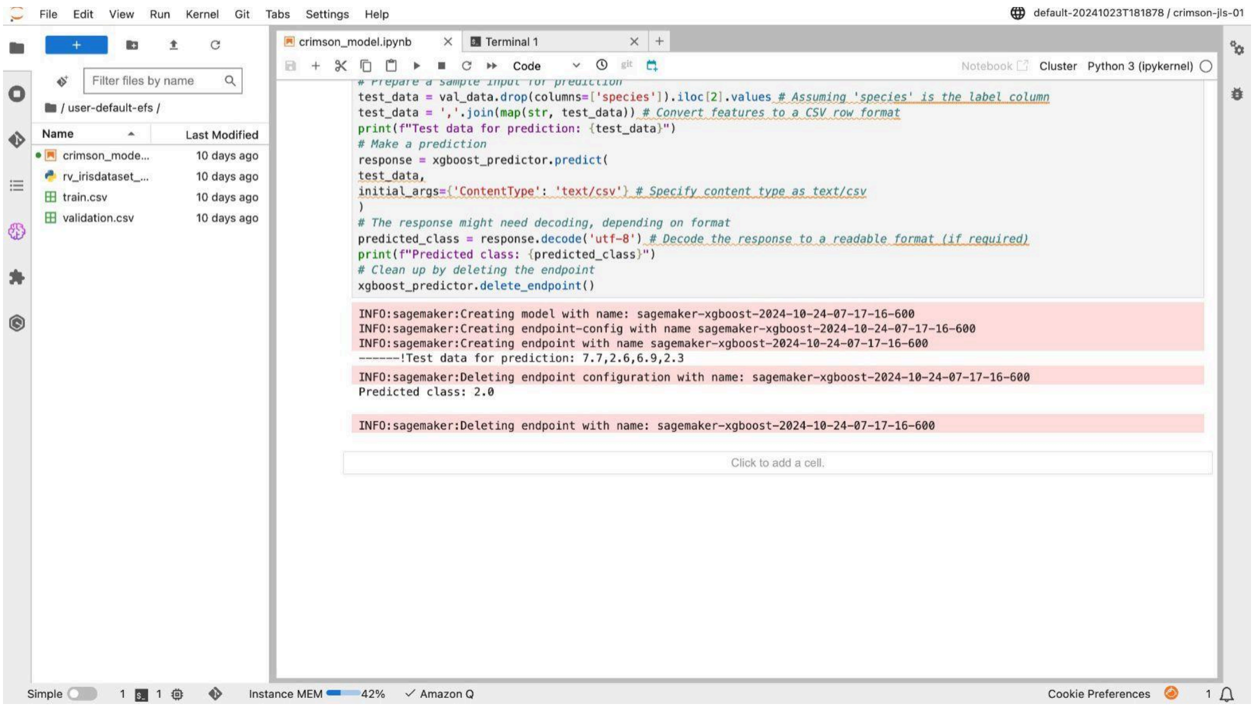
Learn more



```
[1]: # Data Preparation
     # ================
     import sagemaker
     import boto3
     import pandas as pd
     from sklearn.model_selection import train_test_split
     import os
     # Initialize SageMaker session and specify bucket
     sagemaker_session = sagemaker.Session()
     bucket = sagemaker_session.default_bucket()
     role = sagemaker.get_execution_role()
     # Load Iris dataset
     iris_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',
     header=None)
     iris_data.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
     # Convert species to numeric values
     iris_data['species'] = iris_data['species'].astype('category').cat.codes
     print(iris_data['species'].unique()) # Should output [0, 1, 2]
     # Split data into train and validation sets
     train_data, val_data = train_test_split(iris_data, test_size=0.2, random_state=42)
     # Move the label column to the front of the dataframe
     train_data = train_data[['species', 'sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
     val_data = val_data[['species', 'sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
     # Save locally as CSV
     train_data.to_csv('train.csv', header=False, index=False)
     val_data.to_csv('validation.csv', header=False, index=False)
     # one can print the train_data and the val_data that would be the 80% training data 20% validation data
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/sagemaker-user/.config/sagemaker/config.yaml
[0 1 2]
```

```
[2]: # upload the tain_data and val_data to S3 bucket
     s3_train_path = sagemaker_session.upload_data(path='train.csv', bucket=bucket, key_prefix='xgboost-iris/train')
     s3_validation_path = sagemaker_session.upload_data(path='validation.csv', bucket=bucket, key_prefix='xgboost-iris/validation'
     print(f"Training data uploaded to: {s3_train_path}")
     print(f"Validation data uploaded to: {s3_validation_path}")
```



```
# Prepare a sample input for prediction
test_data = val_data.drop(columns=['species']).iloc[2].values # Assuming 'species' is the label column
test_data = ','.join(map(str, test_data)) # Convert features to a CSV row format
print(f"Test data for prediction: {test_data}")
# Make a prediction
response = xgboost_predictor.predict(
test_data,
initial_args={'ContentType': 'text/csv'} # Specify content type as text/csv
)
# The response might need decoding, depending on format
predicted_class = response.decode('utf-8') # Decode the response to a readable format (if required)
print(f"Predicted class: {predicted_class}")
# Clean up by deleting the endpoint
xgboost_predictor.delete_endpoint()
```

```
INFO:sagemaker:Creating model with name: sagemaker-xgboost-2024-10-24-07-17-16-600
INFO:sagemaker:Creating endpoint-config with name sagemaker-xgboost-2024-10-24-07-17-16-600
INFO:sagemaker:Creating endpoint with name sagemaker-xgboost-2024-10-24-07-17-16-600
------!Test data for prediction: 7.7,2.6,6.9,2.3
INFO:sagemaker:Deleting endpoint configuration with name: sagemaker-xgboost-2024-10-24-07-17-16-600
Predicted class: 2.0

INFO:sagemaker:Deleting endpoint with name: sagemaker-xgboost-2024-10-24-07-17-16-600
```