

Project Report: Word Count on AWS with PySpark, Node.js Deployment

by-Vivek Sai Chinna Burada

Objective

This project demonstrates two main tasks:

1. **Word Count on AWS EC2/LightSail using PySpark:** Setting up an EC2/LightSail instance to count words in a text file stored in an S3 bucket using PySpark.
 2. **Node.js Deployment with Docker:** Deploying a simple Node.js web server in a Docker container and hosting it on an EC2 instance.
-

Project 1: Word Count on AWS EC2/LightSail using PySpark

1. Prerequisites

Before starting, ensure that you have the following:

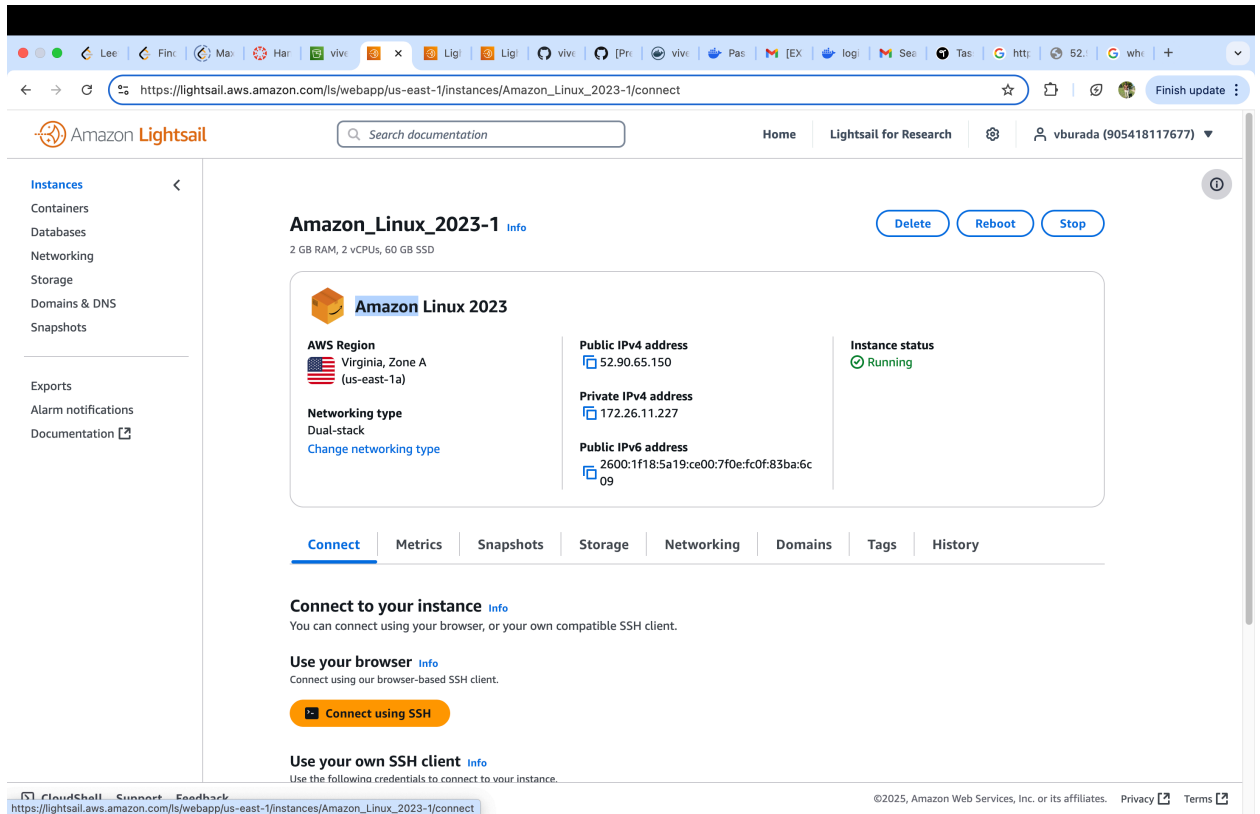
- An **AWS account** with access to **EC2/LightSail**.
- An **S3 bucket** containing the input text file.
- **SSH access** to your EC2/LightSail instance.

2. Set Up EC2/LightSail Instance

Follow these steps to set up the EC2/LightSail instance:

1. **Launch an Amazon Linux 2 Instance:**
Launch the instance from the AWS Management Console.
2. **Connect to the Instance via SSH:**
Use the following command to connect to the EC2 instance:

```
bash
CopyEdit
ssh -i "your-key.pem" ec2-user@your-instance-ip
```



3. Install Java 11:

Java is required for running Spark. Install Java 11:

```
bash
CopyEdit
sudo yum install java-11 -y
export JAVA_HOME=$(dirname $(dirname $(readlink -f $(which java))))
java -version
```

4. Increase /tmp Size:

Increase the /tmp directory size to prevent Spark errors:

```
bash
CopyEdit
sudo mount -o remount,size=2G /tmp
```

5. Install Python and PySpark:

Install Python and PySpark:

```
bash
CopyEdit
sudo yum install python3-pip -y
pip install pyspark
spark-submit --version
```

3. Word Count Script

Create a Python script `word_count.py` to perform the word count operation.

```
python
CopyEdit
from pyspark.sql import SparkSession

# AWS Credentials
AWS_ACCESS_KEY_ID = 'YOUR_ACCESS_KEY'
AWS_SECRET_ACCESS_KEY = 'YOUR_SECRET_KEY'

# S3 paths
S3_INPUT = 's3a://your-bucket-name/input_file.txt'
S3_OUTPUT = 's3a://your-bucket-name/output_folder/'

# Spark Session
spark = SparkSession.builder \
    .appName("WordCount") \
    .config("spark.jars.packages", "org.apache.hadoop:hadoop-aws:3.3.1,com.amazonaws:aws-java-sdk-bundle:1.11.901") \
    .getOrCreate()

# Hadoop S3 Configuration
hadoop_conf = spark.sparkContext._jsc.hadoopConfiguration()
hadoop_conf.set("fs.s3a.access.key", AWS_ACCESS_KEY_ID)
hadoop_conf.set("fs.s3a.secret.key", AWS_SECRET_ACCESS_KEY)

# Word Count Logic
text_file = spark.sparkContext.textFile(S3_INPUT)
counts = text_file.flatMap(lambda line: line.split()) \
    .map(lambda word: (word, 1)) \
    .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile(S3_OUTPUT)
spark.stop()
```

4. Running the Script

Run the word count script using `spark-submit`:

```
bash
CopyEdit
spark-submit word_count.py
```

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight 11

AWS Marketplace for S3

vivek-bucket-word

Info

ObjectsMetadataPropertiesPermissionsMetricsManagementAccess Points

Objects (2)

Copy S3 URICopy URLLownloadOpenDeleteActionsCreate folderUpload

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](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
hello.txt	txt	April 15, 2025, 15:33:47 (UTC-04:00)	39.0 B	Standard
output_folder/	Folder	-	-	-

CloudShellFeedback

© 2025, Amazon Web Services, Inc. or its affiliates. PrivacyTermsCookie preferences

Amazon S3

General purpose buckets

Directory buckets

Table buckets

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight 11

AWS Marketplace for S3

output_folder/

Copy S3 URI

ObjectsProperties

Objects (3)

Copy S3 URICopy URLLownloadOpenDeleteActionsCreate folderUpload

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](#)

Find objects by prefix

Name	Type	Last modified	Size	Storage class
_SUCCESS	-	April 15, 2025, 15:37:22 (UTC-04:00)	0 B	Standard
part-00000	-	April 15, 2025, 15:37:21 (UTC-04:00)	47.0 B	Standard
part-00001	-	April 15, 2025, 15:37:22 (UTC-04:00)	41.0 B	Standard

CloudShellFeedback

© 2025, Amazon Web Services, Inc. or its affiliates. PrivacyTermsCookie preferences

Project 2: Deploy a Node.js Web Server with Docker

1. Node.js Server

1. Create a Directory and Initialize the Project:

Create a directory and initialize a Node.js project:

```
bash
CopyEdit
mkdir node-webserver && cd node-webserver
npm init -y
npm install express
```

2. Create the `server.js` File:

The `server.js` file contains a simple Express server:

```
javascript
CopyEdit
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello, World! Running in Docker.');
```

```
});

app.listen(port, () => console.log(`Server running at
http://localhost:${port}`));
```

A screenshot of a web browser window displaying a terminal session. The browser's address bar shows the URL: https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/Amazon_Linux_2023-1/terminal?protocol=ssh. The terminal window is titled 'server.js' and shows the GNU nano 8.3 editor. The code in the editor is a simple Express.js server that listens on port 3000 and responds with 'Hello, World! Running in Docker.' when accessed at the root path. The terminal output shows the server running at http://localhost:3000. The browser's status bar at the bottom indicates the connection is to Amazon_Linux_2023-1 with IP 52.90.65.150.

2. Dockerize the Application

1. Create the Dockerfile:

Create a Dockerfile to define how the application will be containerized:

```
dockerfile
CopyEdit
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

2. Build and Push the Docker Image to Docker Hub: Build the Docker image:

```
bash
CopyEdit
docker build -t your-dockerhub-username/webserver:latest .
```

Push the image to Docker Hub:

```
bash
CopyEdit
docker push your-dockerhub-username/webserver:latest
```

The screenshot shows a terminal window with a browser address bar at the top displaying `https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/Amazon_Linux_2023-1/terminal?protocol=ssh`. The terminal output shows the following sequence of commands and results:

```
=> [5/5] COPY . . 0.2s
=> exporting to image 0.1s
=> exporting layers 0.1s
=> writing image sha256:d08a960fcec8700228a3649b8ed1c6a9192f823214740812cf84a6000562dcc5 0.0s
=> naming to docker.io/viveksaichinna/webserver:latest 0.0s
[ec2-user@ip-172-26-11-227 node-webserver]$ docker build -t viveksaichinna/webserver:latest .
[+] Building 0.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 212B
=> [internal] load metadata for docker.io/library/node:14
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
FROM node:14
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

Below the terminal window, there is a small icon of an Amazon Linux instance and the text `Amazon_Linux_2023-1` and `52.90.65.150`.

3. Deploy on EC2

1. Install Docker on EC2: Update and install Docker:

```
bash
CopyEdit
sudo yum update -y
sudo yum install docker -y
sudo service docker start
sudo usermod -aG docker ec2-user
```

2. Run the Container:

Pull the image and run the container:

```
bash
CopyEdit
docker pull your-dockerhub-username/webserver:latest
docker run -d -p 80:3000 your-dockerhub-username/webserver:latest
```

Lee

Finc

Ma

Har

vive

Ligi

Ligi

vive

Pr

vive

Pas

EX

logi

See

Tas

htt

52

whi

https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/Amazon_Linux_2023-1/terminal?protocol=ssh

Finish update

```
=> => transferring dockerfile: 212B 0.0s
=> [internal] load metadata for docker.io/library/node:16 0.3s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 83.37kB 0.0s
=> [1/5] FROM docker.io/library/node:16@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c6389705c0b 21.0s
=> => resolve docker.io/library/node:16@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c6389705c0b 0.0s
=> => sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c6389705c0b 776B / 776B 0.0s
=> => sha256:311da6c465ea1576925360eba391bcd32dece9be95960a0bc9ffcb25fe712017 50.50MB / 50.50MB 0.8s
=> => sha256:ffd9397e94b74abcb54e514f1430e00f604328d1f895eadbd482f08cc02444e5 51.89MB / 51.89MB 0.8s
=> => sha256:c94b82f9827cab6e421b350965a9ef11b25b13ffbd1030536203d541f55dcbe2 2.00kB / 2.00kB 0.0s
=> => sha256:1ddc7e4055fdb6f6bf31063b593befda814294f9f904b6ddfc21ab1513bafa8e 7.23kB / 7.23kB 0.0s
=> => sha256:7e9bf114588c05b2df612b083b96582f3b8dbf51647aa6138a50d09d42df2454 17.58MB / 17.58MB 0.4s
=> => sha256:513d779256048c961239af5f500589330546b072775217272e19ffae1635e98e 191.90MB / 191.90MB 2.5s
=> => sha256:ae3b95bbaa61ce24cefd89e7c74d6fbd7713b2bcae93af47063d06bd7e02172 4.20kB / 4.20kB 0.8s
=> => extracting sha256:311da6c465ea1576925360eba391bcd32dece9be95960a0bc9ffcb25fe712017 3.2s
=> => sha256:0e421f66aff42bb069dffc26af6d132194b22a1082b08c5ef7cd69c627783c04 34.79MB / 34.79MB 1.6s
=> => sha256:ca266fd6192108b67fb57b74753a8c4ca5d8bd458baae3d4df7ce9f42dedcc1d 2.27MB / 2.27MB 1.0s
=> => sha256:ee7d78be1eb92caf6ae84fc3af736b23eca018d5dedc967ae5bdee6d7082403b 450B / 450B 1.0s
=> => extracting sha256:7e9bf114588c05b2df612b083b96582f3b8dbf51647aa6138a50d09d42df2454 0.6s
=> => extracting sha256:ffd9397e94b74abcb54e514f1430e00f604328d1f895eadbd482f08cc02444e5 3.1s
=> => extracting sha256:513d779256048c961239af5f500589330546b072775217272e19ffae1635e98e 10.5s
=> => extracting sha256:ae3b95bbaa61ce24cefd89e7c74d6fbd7713b2bcae93af47063d06bd7e02172 0.0s
=> => extracting sha256:0e421f66aff42bb069dffc26af6d132194b22a1082b08c5ef7cd69c627783c04 1.8s
=> => extracting sha256:ca266fd6192108b67fb57b74753a8c4ca5d8bd458baae3d4df7ce9f42dedcc1d 0.1s
=> => extracting sha256:ee7d78be1eb92caf6ae84fc3af736b23eca018d5dedc967ae5bdee6d7082403b 0.0s
=> [2/5] WORKDIR /app 0.3s
=> [3/5] COPY package*.json ./ 0.1s
=> [4/5] RUN npm install 2.2s
=> [5/5] COPY . . 0.6s
=> exporting to image 0.2s
=> exporting layers 0.2s
=> writing image sha256:36354a994f0ed47266a4618be14c703306802389f1857031598332a9119d4891 0.0s
=> naming to docker.io/viveksaichinna/webserver:latest 0.0s
[ec2-user@ip-172-26-11-227 node-webserver]$ docker run -d -p 80:3000 viveksaichinna/webserver:latest
ae7e444645caa6e3660cc383fd745fb0fd6ce692ec73e68060f1fa253cd35997
[ec2-user@ip-172-26-11-227 node-webserver]$
```

Amazon_Linux_2023-1

52.90.65.150

Deploy Node.js with Docker

http://c-your-lightsail-public-1

52.90.65.150

Not Secure

http://52.90.65.150

Finish update

Hello, World! Running in Docker.

URL of we app hosted:

<http://52.90.65.150/>

Conclusion

In this project, we successfully:

- Set up an EC2/LightSail instance to run PySpark for counting words in a text file stored in an S3 bucket.
 - Deployed a Node.js server in a Docker container and hosted it on an EC2 instance.
-

GitHub Repository Link

<https://github.com/viveksaichinna/word-count-spark>

Word Count:

The total word count for this report, excluding the code snippets and instructions, is

('name', 1)

('is', 1)

('sai', 1)

('chinna', 1)

('hello', 1)

('vivek', 1)

('burada.', 1)