DevOps Intern Assignment: Dockerize and Deploy a Web App on AWS EC2

Overview

This repository contains the complete implementation of a Node.js web application that has been containerized with Docker and deployed on AWS EC2 with automation features.

Project Summary

This project involved deploying a simple Node.js and Express.js web application. The core focus was not on the application's complexity, but on the surrounding DevOps practices to ensure a robust, secure, and automated deployment pipeline.

To guarantee **persistent storage** and prevent data loss upon container restarts, the application's data directory (`/app/data`) was mapped using Docker's **bind mounts**. This was implemented both in the local development environment—linking to a folder on the host machine—and on the EC2 instance, where it was mapped to `/home/ubuntu/thelonelybag_appdata`. This ensures that the application's data resides on the host filesystem and persists independently of the container's lifecycle.

Significant emphasis was placed on **enhancing security**. The default SSH port 22 was changed to a non-standard port (1234) to protect against automated bots and brute-force attacks that commonly target the default port. Furthermore, to address the security risks associated with my ISP's use of Carrier-Grade NAT (CGNAT)—where multiple users share a single public IPv4 address—SSH access to the EC2 instance was restricted to my unique **IPv6 address only**. This measure prevents unauthorized access attempts from other users on the same shared IPv4 network.

While management access was secured via IPv6, the web application itself was made accessible over both **IPv4 and IPv6**. This dual-stack configuration ensures the application is reachable by all users, regardless of their network protocol, providing maximum accessibility and a modern network architecture.

Finally, the entire deployment process was automated. A `cloud-init` script was developed to provision a new EC2 instance by installing Docker and deploying the application on boot. Additionally, a `deploy.sh` script was created to automate the setup process on an already running instance, streamlining any future deployments or updates.

Table of Contents

- Deployment Script

Step 1: GitHub Repository Setup

Commands Used

```
git init
git add Flow.md
git branch -M main
git commit -m "Initial commit"
git remote add origin https://github.com/Siarhii/theLonelyBag_Assignment.git
git push -u origin main
```

Deliverable: GitHub repository created and initialized

GitHub Repository Link

Deliverable: The complete source code and documentation are available at the following GitHub repository: https://github.com/Siarhii/theLonelyBag_Assignment

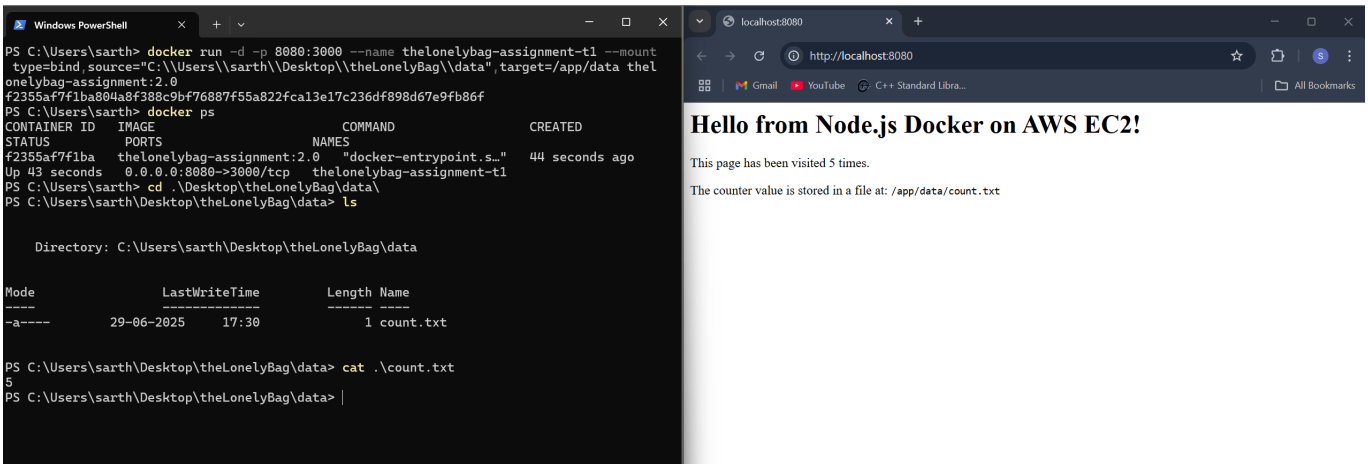
---

Step 2: Node.js Application Development

Setup Commands

```
npm init -y
npm install express
node index.js
```

Created a basic Express.js web application with proper error handling and logging. The application runs on port 3000 and includes a data mountpoint at `/app/data`.



Deliverable: Basic Node.js application created and tested locally

---

Step 3: Docker Containerization

3.1 Docker Configuration

- Created `Dockerfile` with multi-stage build optimization
- Created `.dockerignore` file to exclude unnecessary files

3.2 Building Docker Image

```
docker build -t thelonelybag_assignment:1.0 .
```

```
PS C:\Users\sarth\Desktop\theLonelyBag> docker build -t sarthak69/thelonelybag-assignment:2.0 .
[+] Building 2.4s (10/10) FINISHED
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 221B
 => [internal] load metadata for docker.io/library/node:20-alpine
 => [internal] load .dockerignore
 => => transferring context: 762B
 => [1/5] FROM docker.io/library/node:20-alpine@sha256:674181320f4f94582c6182eaa151bf92c6744d478be0f1d12db804b7d59b2d11
 => [internal] load build context
 => => transferring context: 4.18kB
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY package*.json ./
 => CACHED [4/5] RUN npm install --production
 => [5/5] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:2c8627aaf55117db4699d3b59d5ca74d0f70e3093719b18fce49006e5a04f54a
 => => naming to docker.io/sarthak69/thelonelybag-assignment:2.0
```

3.3 Running Container Locally

```
docker run -d -p 8080:3000 --name thelonelybag-assignment-t1 \
  --mount
type=bind,source="C:\Users\sarth\Desktop\theLonelyBag\data",target=/app/data \
  thelonelybag-assignment:2.0
```

```
PS C:\Users\sarth> docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS              PORTS
           NAMES
f2355af7f1ba   thelonelybag-assignment:2.0    "docker-entrypoint.s…"   7 minutes ago   Up About a minute   0.0.0.0:8080->
3000/tcp   thelonelybag-assignment-t1
PS C:\Users\sarth> docker exec -it f /bin/sh
/app # ls
Flow.md             dockerfile          node_modules        package.json
data                index.js            package-lock.json
/app # cd ..
/ # ls
app    dev    home   media  opt    root   sbin   sys    usr
bin    etc    lib    mnt    proc   run    srv    tmp    var
/ # ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 npm start
   17 root      0:00 node index.js
   24 root      0:00 /bin/sh
   32 root      0:00 ps aux
/ #
```

Deliverable: Application successfully containerized and running locally

Data Persistence Configuration

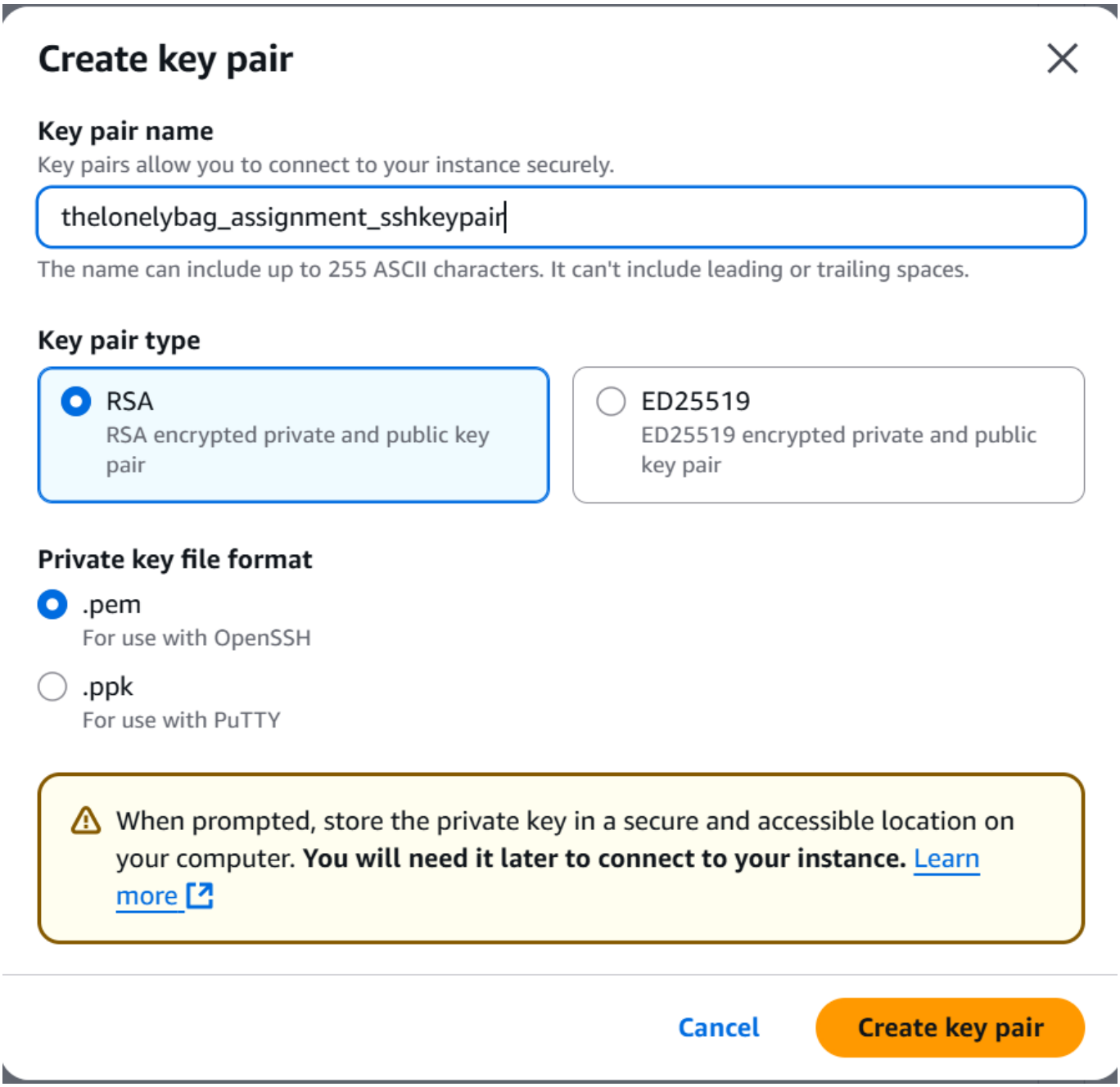
- **Bind Mount Implementation**: Configured persistent storage using bind mounts for the `/app/data` directory
- **Local Environment**: Data persists to `C:\Users\sarth\Desktop\theLonelyBag\data` on Windows host
- **Benefits**: Application data survives container restarts and updates
- **Mount Type**: Used bind mounts over volumes for direct host filesystem access

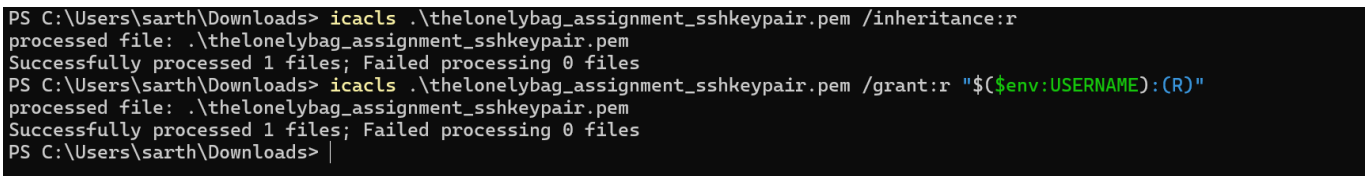
---

Step 4: AWS EC2 Deployment

4.1 SSH Key Configuration (Windows)

```
# Set proper permissions for SSH key
icacls .\thelonelybag_assignment_sshkeypair.pem /inheritance:r
icacls .\thelonelybag_assignment_sshkeypair.pem /grant:r "$($env:USERNAME):(R)"
```





4.2 Initial SSH Connection

```
ssh -i .\thelonelybag_assignment_sshkeypair.pem ubuntu@54.210.213.67
```

```
PS C:\Users\sarth\Downloads> ssh -i .\thelonelybag_assignment_sshkeypair.pem ubuntu@54.210.213.67
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Jun 29 12:59:08 UTC 2025

  System load:           0.04
  Usage of /:            25.4% of 6.71GB
  Memory usage:          20%
  Swap usage:            0%
  Processes:             106
  Users logged in:       0
  IPv4 address for enX0: 10.0.8.59
  IPv6 address for enX0: 2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-8-59:~$
```

4.3 Basic Security Hardening

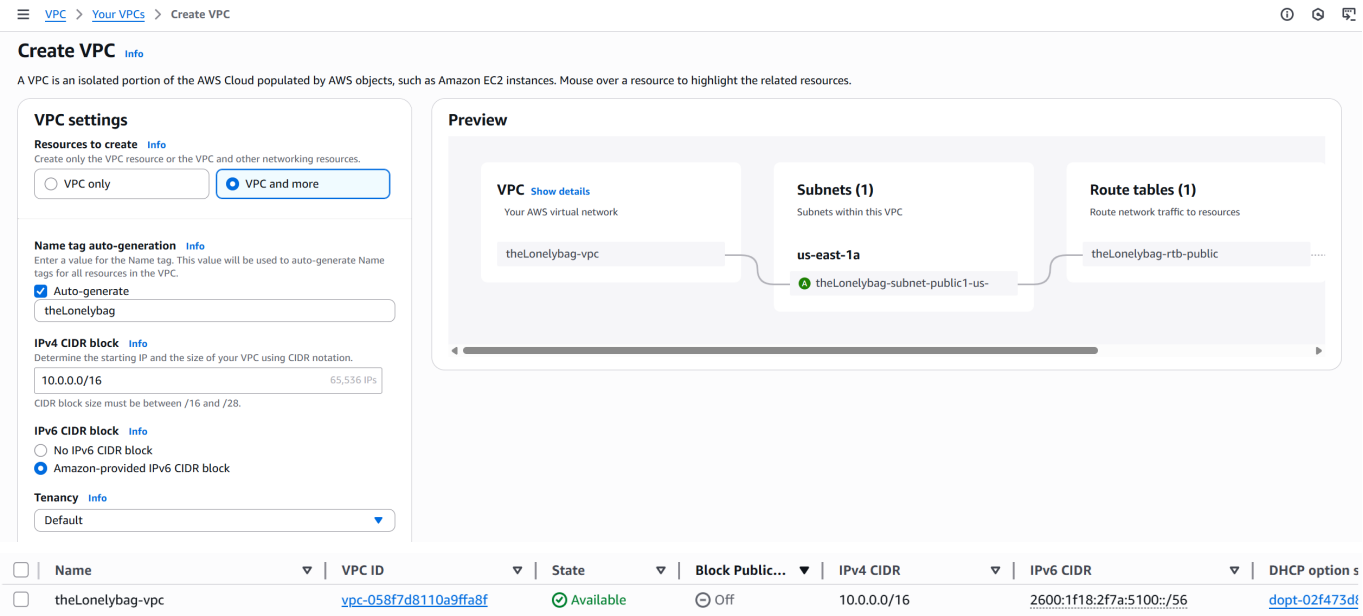
```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Change SSH port for security
sudo nano /etc/ssh/sshd_config
sudo systemctl restart ssh
```

```
#
#    systemctl daemon-reload
#    systemctl restart ssh.socket
#
Port 1234
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

4.4 IPv6 SSH Configuration

Configured SSH to use IPv6 for enhanced security due to ISP CGNAT limitations:

```
ssh -i .\thelonelybag_assignment_sshkeypair.pem -p 1234
ubuntu@2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7
```



Edit subnet settings Info

Subnet

Subnet ID
subnet-09d795893bbffa3e0

Name
theLonelybag-subnet-public1-us-east-1a

Auto-assign IP settings Info

Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

- [x] Enable auto-assign IPv6 address Info
- [x] Enable auto-assign public IPv4 address Info
- [ ] Enable auto-assign customer-owned IPv4 address Info
  Option disabled because no customer owned pools found.

Resource-based name (RBN) settings Info

Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

- [ ] Enable resource name DNS A record on launch Info
- [ ] Enable resource name DNS AAAA record on launch Info

Hostname type Info
- ( ) Resource name
- (•) IP name

DNS64 settings

Enable DNS64 to allow IPv6-only services in Amazon VPC to communicate with IPv4-only services and networks.

- [ ] Enable DNS64 Info

i-019e3f15bc67a8565 (theLonelyBag_Assignment)

Instance ID i-019e3f15bc67a8565	**Public IPv4 address** 54.210.213.67 | open address	**Private IPv4 addresses** 10.0.8.59
IPv6 address 2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7	**Instance state** Running	**Public DNS** ec2-54-210-213-67.compute-1.amazonaws.com | open address
Hostname type IP name: ip-10-0-8-59.ec2.internal	**Private IP DNS name (IPv4 only)** ip-10-0-8-59.ec2.internal	**Private resource DNS name** –
Answer private resource DNS name –	**Instance type** t2.micro	**Elastic IP addresses** –
Auto-assigned IP address 54.210.213.67 [Public IP]	**VPC ID** vpc-058f7d8110a9ffa8f (theLonelybag-vpc)	**AWS Compute Optimizer finding** Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Security group rule 1 (TCP, 22, 223.233.86.20/32, ssh ipv4)

Remove

Type Info	Protocol Info	Port range Info
ssh	TCP	22

Source type Info	Name Info	Description - *optional* Info
My IP	Add CIDR, prefix list or security group 223.233.86.20/32	ssh ipv4

Security group rule 2 (TCP, 22, 2401:4900:1c42:bea:4b8a:b83c:4f74:be6f/128, ssh ipv6)

Remove

Type Info	Protocol Info	Port range Info
ssh	TCP	22

Source type Info	Source Info	Description - *optional* Info
Custom	Add CIDR, prefix list or security group 2401:4900:1c42:bea:4b8a:b83c:4f74:be6f/128	ssh ipv6

Security group rule 3 (TCP, 8080, Multiple sources, webapp ipv4)

Remove

Type Info	Protocol Info	Port range Info
Custom TCP	TCP	8080

Source type Info	Source Info	Description - *optional* Info
Anywhere	Add CIDR, prefix list or security group 0.0.0.0/0 ::/0	webapp ipv4

Inbound rules (4)

Manage tags | Edit inbound rules

Search

Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
sgr-01aec0c8680eb5c5e	IPv4	SSH	TCP	22	223.233.86.20/32	ssh ipv4
sgr-0ba7b2de81c44e996	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	webapp ipv4
sgr-0db61a61fce91a2ef	IPv6	Custom TCP	TCP	8080	::/0	webapp ipv4
sgr-00779afab523c10af	IPv6	Custom TCP	TCP	1234	2401:4900:1c42:bea:4b...	ssh changedport ipv6

```
PS C:\Users\sarth\Downloads> ssh -i .\thelonelybag_assignment_sshkeypair.pem -p 1234 ubuntu@2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7
The authenticity of host '[2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7]:1234 ([2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7]:1234)' can't be established.
ED25519 key fingerprint is SHA256:V4bSWAXRBhQP5kz7IqHVistWN4f5J6QwGIujI6QECeE.
This host key is known by the following other names/addresses:
    C:\Users\sarth/.ssh/known_hosts:38: 54.210.213.67
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7]:1234' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Jun 29 13:33:49 UTC 2025

  System load:           0.0
  Usage of /:            29.1% of 6.71GB
  Memory usage:          23%
  Swap usage:            0%
  Processes:             107
  Users logged in:       0
  IPv4 address for enX0: 10.0.8.59
  IPv6 address for enX0: 2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7
```

Deliverable: EC2 instance launched and configured with security hardening

---

Security Configurations & Network Setup

Port Configuration Changes

- **SSH Port Change**: Modified default SSH port from 22 to 1234 to deter automated botnet attacks
- **Rationale**: Default ports are commonly targeted by automated scanners and bots

IPv4 vs IPv6 Access Strategy

- **IPv4 Limitation**: ISP provides shared IPv4 (CGNAT) which means multiple users share the same public IPv4 address
- **Security Risk**: Allowing IPv4 SSH access would potentially allow other users on the same shared IP to attempt connections
- **Solution Implemented**: Disabled IPv4 SSH access and configured SSH exclusively over IPv6
- **IPv6 Benefits**: Each device gets a unique global IPv6 address, providing better security isolation

Application Accessibility

- **Dual Stack Support**: Web application remains accessible via both IPv4 and IPv6
  - IPv4: `http://54.210.213.67:8080` (for general web access)
  - IPv6: `http://[2600:1f18:2f7a:5100:b86a:5f4e:36eb:30a7]:8080`
- **Management Access**: SSH restricted to IPv6 only for security
- **Public Access**: HTTP service available on both protocols for maximum compatibility

Documentation Evidence

- Screenshots available showing:
  - EC2 dashboard with running instances
  - SSH connections over IPv6

- Application running on both IPv4 and IPv6 addresses
- Security group configurations
- Docker container status on both local and EC2 environments

Step 5: Docker Installation on EC2

Installation Steps

```
# Update package lists
sudo apt update

# Install prerequisites
sudo apt install -y apt-transport-https ca-certificates curl software-properties-
common

# Add Docker GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg

# Add Docker repository
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null

# Update package lists with Docker repo
sudo apt update

# Install Docker
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin

# Add user to docker group
sudo usermod -aG docker ubuntu
newgrp docker

# Verify installation
docker run hello-world
```

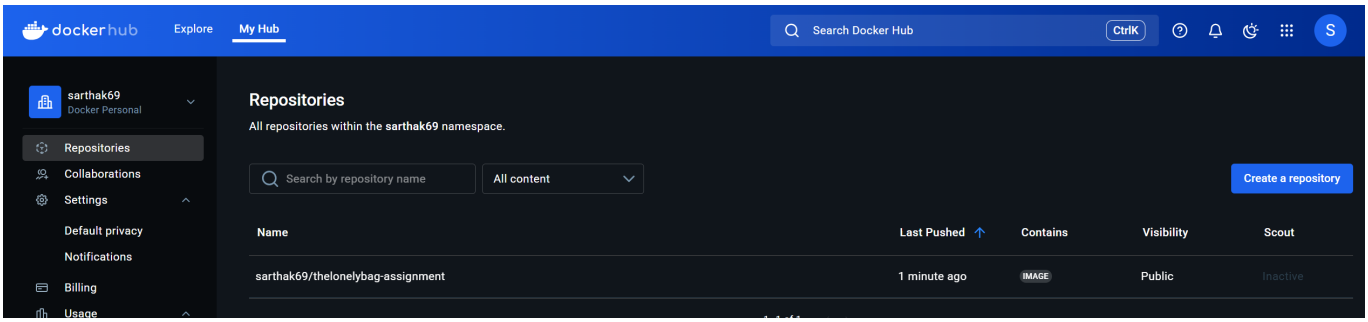
Deliverable: Docker successfully installed and verified on EC2

Step 6: Application Deployment on EC2

6.1 Pull Docker Image

```
docker pull sarthak69/thelonelybag-assignment:2.0
```

```
PS C:\Users\sarth\Desktop\theLonelyBag> docker push sarthak69/thelonelybag-assignment:2.0
The push refers to repository [docker.io/sarthak69/thelonelybag-assignment]
942aef24dce9: Pushed
60a1566be5a5: Pushed
ec231bfca5b0: Pushed
d0522da8c75f: Pushed
842a9b22af3c: Mounted from library/node
7db44a23820d: Mounted from library/node
5cef6a9646ba: Mounted from library/node
fd2758d7a50e: Mounted from library/node
2.0: digest: sha256:a5e2a0ab274278960c74b233810665978b8da5a9de46c6cb5d301d7a87b270c5 size: 1992
```



6.2 Run Application Container

```
docker run -d -p 8080:3000 --name thelonelybag-assignment-ec2 \
  --mount type=bind,source=/home/ubuntu/thelonelybag_appdata,target=/app/data \
  sarthak69/thelonelybag-assignment:2.0
```

Persistent Storage on EC2

- **Bind Mount Configuration**: Data directory mounted to `/home/ubuntu/thelonelybag_appdata` on EC2 host
- **Data Persistence**: Application data survives container recreation and system reboots
- **Storage Benefits**: Direct filesystem access allows for easy backup and data management

```
ubuntu@ip-10-0-8-59:~$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORTS                                                   NAMES
0651b421384d   sarthak69/thelonelybag-assignment:2.0   "docker-entrypoint.s…"   23 seconds ago   Up 22 seconds   0.0.0.0:8080->3000/tcp, [::]:8080->3000/tcp   thelonelybag-assignm
ent-ec2
ubuntu@ip-10-0-8-59:~$
```

```
ubuntu@ip-10-0-8-59:~$ docker pull sarthak69/thelonelybag-assignment:2.0
2.0: Pulling from sarthak69/thelonelybag-assignment
fe07684b16b8: Already exists
5432aa916e08: Already exists
2506673f5536: Already exists
98c4889b578e: Already exists
b618eb049860: Pull complete
76693731eb94: Pull complete
ce5c502127ee: Pull complete
4c43a25147ab: Pull complete
Digest: sha256:a5e2a0ab274278960c74b233810665978b8da5a9de46c6cb5d301d7a87b270c5
Status: Downloaded newer image for sarthak69/thelonelybag-assignment:2.0
docker.io/sarthak69/thelonelybag-assignment:2.0
ubuntu@ip-10-0-8-59:~$ ls
ubuntu@ip-10-0-8-59:~$ cd ..
ubuntu@ip-10-0-8-59:/home$ ls
ubuntu
ubuntu@ip-10-0-8-59:/home$ cd ubuntu/
ubuntu@ip-10-0-8-59:~$ mkdir thelonelybag_appdata
ubuntu@ip-10-0-8-59:~$ ls
thelonelybag_appdata
ubuntu@ip-10-0-8-59:~$ docker run -d -p 8080:3000 --name thelonelybag-assignment-ec2 --mount type=bind,source=/home/ubuntu/thelonelybad_appdata,target=/app/data thelonelybag-assig
nment:2.0
docker: Error response from daemon: invalid mount config for type "bind": bind source path does not exist: /home/ubuntu/thelonelybad_appdata

Run 'docker run --help' for more information
ubuntu@ip-10-0-8-59:~$ docker run -d -p 8080:3000 --name thelonelybag-assignment-ec2 --mount type=bind,source=/home/ubuntu/thelonelybag_appdata,target=/app/data thelonelybag-assig
nment:2.0
Unable to find image 'thelonelybag-assignment:2.0' locally
docker: Error response from daemon: pull access denied for thelonelybag-assignment, repository does not exist or may require 'docker login': denied: requested access to the resour
ce is denied

Run 'docker run --help' for more information
ubuntu@ip-10-0-8-59:~$ docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
sarthak69/thelonelybag-assignment   2.0       2c8627aaf551   9 minutes ago   138MB
ubuntu@ip-10-0-8-59:~$ docker run -d -p 8080:3000 --name thelonelybag-assignment-ec2 --mount type=bind,source=/home/ubuntu/appdata,target=/app/data sarthak69/thelonelybag-assignme
nt:2.0
docker: Error response from daemon: invalid mount config for type "bind": bind source path does not exist: /home/ubuntu/appdata

Run 'docker run --help' for more information
ubuntu@ip-10-0-8-59:~$ docker run -d -p 8080:3000 --name thelonelybag-assignment-ec2 --mount type=bind,source=/home/ubuntu/thelonelybag_appdata,target=/app/data sarthak69/thelonel
ybag-assignment:2.0
0651b421384dd501318d9582e82f3f2e687389e29a7735ab7843ee704c74dfff
ubuntu@ip-10-0-8-59:~$
```

Hello from Node.js Docker on AWS EC2!

This page has been visited 2 times.

The counter value is stored in a file at: `/app/data/count.txt`

Hello from Node.js Docker on AWS EC2!

This page has been visited 1 times.

The counter value is stored in a file at: `/app/data/count.txt`

Deliverable: Application successfully running on EC2 via public IP with persistent storage

---

Bonus Tasks

IAM Role and S3 Access

Setup Steps:

1. Created S3 bucket via AWS Console
2. Created IAM role with S3 access permissions
3. Attached IAM role to EC2 instance

aws | Search [Alt+S] | United States (N. Virginia) | Sarthak%20Hinge

Amazon S3 > Buckets > Create bucket

General configuration

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

Bucket type Info

- General purpose
  Recommended 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
  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.

Bucket name Info

thelonelybag_assignment

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn More

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

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.

⚠ 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 preferred
  If 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 writer
  The object writer remains the object owner.

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

- [x] **Block *all* public access**
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - [x] **Block public access to buckets and objects granted through *new* access control lists (ACLs)**
    S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - [x] **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
    S3 will ignore all ACLs that grant public access to buckets and objects.
  - [x] **Block public access to buckets and objects granted through *new* public bucket or access point policies**
    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - [x] **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

Copy ARN | Empty | Delete | Create bucket

Find buckets by name

Name	AWS Region	IAM Access Analyzer	Creation date
thelonelybag-assignment	US East (N. Virginia) us-east-1	View analyzer for us-east-1	June 29, 2025, 19:50:00 (UTC+05:30)

readOnly-s3-theLonelyBagAssignment Info

Delete

Allows EC2 instances to have read-only access to S3 buckets.

Summary

Edit

Creation date
June 29, 2025, 19:52 (UTC+05:30)

Last activity
-

ARN
arn:aws:iam::477712042273:role/readOnly-s3-theLonelyBagAssignment

Maximum session duration
1 hour

Instance profile ARN
arn:aws:iam::477712042273:instance-profile/readOnly-s3-theLonelyBagAssignment

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (1) Info

You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Search | Filter by Type: All types

Policy name	Type	Attached entities
AmazonS3ReadOnlyAccess	AWS managed	1

✓ Successfully attached readOnly-s3-theLonelyBagAssignment to instance i-019e3f15bc67a8565

Instances (1/1) Info

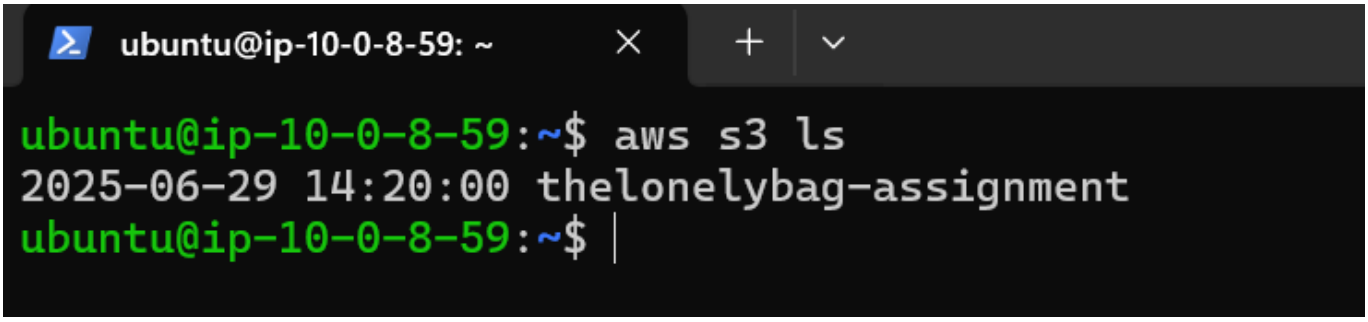
Last updated 1 minute ago | Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
onelyBag...	i-019e3f15bc67a8565	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-210-213-67.co...	54.210.213.67

AWS CLI Installation on EC2:

```
# Install prerequisites
sudo apt update && sudo apt install -y unzip

# Download and install AWS CLI v2
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install

# Verify installation
aws --version

# Test S3 access
aws s3 ls
```



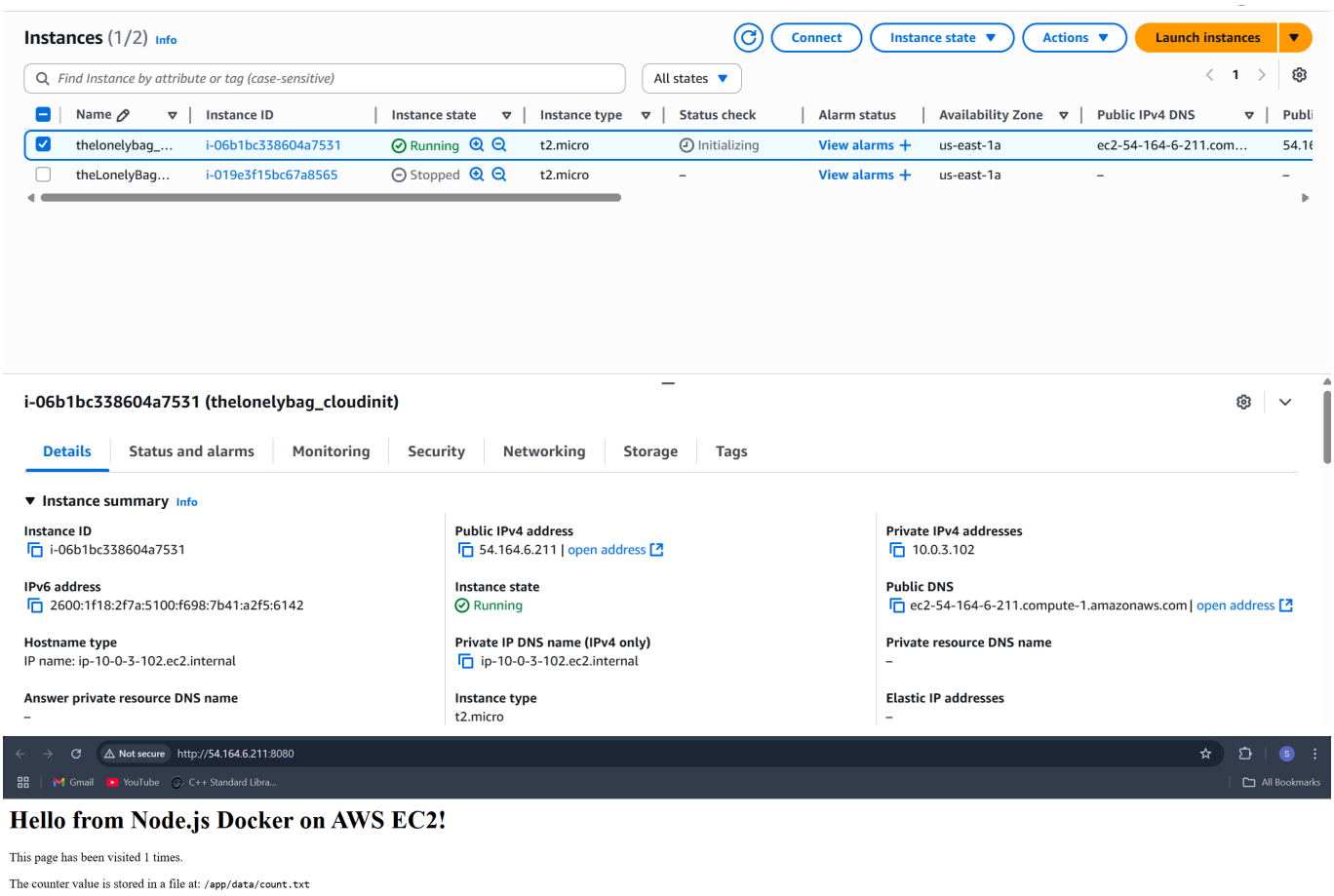
Deliverable: EC2 instance can access S3 using IAM role

Cloud-Init Automation

Created `cloud-init.sh` script to automate:

- Docker installation
- Application deployment
- System configuration

Launched new EC2 instance using cloud-init script in User Data section.



Deliverable: EC2 instance auto-configured using cloud-init

Deployment Script

Created `deploy.sh` automation script for:

- Repository cloning
- Application setup
- Container deployment

Usage:

```
# Clone repository
git clone https://github.com/Siarhii/theLonelyBag_Assignment.git

# Make script executable and run
chmod +x deploy.sh
./deploy.sh
```

```
ubuntu@ip-10-0-8-59:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-10-0-8-59:~$ ls
aws  awscliv2.zip  thelonelybag_appdata
ubuntu@ip-10-0-8-59:~$ git clone https://github.com/Siarhii/theLonelyBag_Assignment.git
Cloning into 'theLonelyBag_Assignment'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 27 (delta 7), reused 24 (delta 4), pack-reused 0 (from 0)
Receiving objects: 100% (27/27), 14.46 KiB | 2.89 MiB/s, done.
Resolving deltas: 100% (7/7), done.
ubuntu@ip-10-0-8-59:~$ cd theLonelyBag_Assignment/
ubuntu@ip-10-0-8-59:~/theLonelyBag_Assignment$ ls
Flow.md  cloud-init.sh  data  deploy.sh  dockerfile  index.js  package-lock.json  package.json
ubuntu@ip-10-0-8-59:~/theLonelyBag_Assignment$ chmod +x deploy.sh
ubuntu@ip-10-0-8-59:~/theLonelyBag_Assignment$ ./deploy.sh
Starting Node.js app deployment (pulling from Docker Hub)...
Pulling latest changes from GitHub...
Already up to date.
Pulling Docker image sarthak69/thelonelybag-assignment:2.0 from Docker Hub...
2.0: Pulling from sarthak69/thelonelybag-assignment
Digest: sha256:a5e2a0ab274278960c74b233810665978b8da5a9de46c6cb5d301d7a87b270c5
Status: Image is up to date for sarthak69/thelonelybag-assignment:2.0
docker.io/sarthak69/thelonelybag-assignment:2.0
Creating host data directory for persistence...
Stopping any existing container named 'thelonelybag-assignment-ec2'...
Running the Docker container...
b31c9a1cfb2a0ef29f0f98bca4016fa01c4e973e63bdaafcd8e8c87e46afff66
Verifying container status...
b31c9a1cfb2a   sarthak69/thelonelybag-assignment:2.0   "docker-entrypoint.s…"   1 second ago   Up Less than a second   0.0.0.0:8080->3000/tcp, [::]:8080->3000/tcp   thelonelybag-a
ssignment-ec2
Deployment successful! App should be running on port 8080.
ubuntu@ip-10-0-8-59:~/theLonelyBag_Assignment$ cat deploy.sh
#!/bin/bash

# deploy.sh - Script to deploy the Node.js Docker application by pulling from Docker Hub.
# This script assumes Docker is already installed and the user is in the docker group.

set -e

echo "Starting Node.js app deployment (pulling from Docker Hub)..."

# 1. Define the application directory (where the repo is cloned)
# Based on your cloud-init, the repo is cloned directly into /home/ubuntu/theLonelyBag_Assignment
APP_DIR="/home/ubuntu/theLonelyBag_Assignment"
if [ ! -d "$APP_DIR" ]; then
  echo "Error: Application directory not found at $APP_DIR. Please ensure the repo is cloned correctly."
  exit 1
```

Deliverable: Automated deployment script created and tested

Project Structure

```
theLonelyBag_Assignment/
├── index.js              # Main application file
├── package.json          # Node.js dependencies
├── Dockerfile            # Container configuration
├── .dockerignore         # Docker ignore rules
├── cloud-init.sh         # EC2 automation script
├── deploy.sh             # Deployment automation
└── README.md             # This documentation
```

Screenshots and Documentation

- EC2 Dashboard showing running instances
- SSH terminal sessions
- Application running via public EC2 IP address
- Docker containers running locally and on EC2
- S3 access verification from EC2

Visual Documentation & Screenshots

Infrastructure Setup

- `newVPCforIPV6andIpv4.png` - VPC creation with dual-stack support
- `new created vpc.png` - VPC configuration completed
- `enabling ipv6 and ipv4 in new vpc subnet.png` - Subnet configuration for both protocols

- `creating-sshkeypair.png` - SSH key pair generation
- `first ec2 aws console ss.png` - EC2 instance dashboard
- `ec2 inbound security rules for ssh and webapp.png` - Security group configuration

Local Development & Docker

- `runningapp-localhost.png` - Application running locally on development machine
- `applicationRunnignInsideDockerContainerLocal.png` - Docker container running locally
- `dockerbuild image.png` - Docker image build process
- `pushing docker container on dockerhub.png` - Docker image push to registry
- `dockerhub home ss showing pushed container repo.png` - DockerHub repository confirmation

Security Configuration

- `using icacls for changing .pem permissions on windows.png` - Windows SSH key permissions
- `first ec2 ssh login.png` - Initial SSH connection to EC2
- `changing port ssh.png` - SSH port configuration change
- `after changing inbound rules (port change 22-1234).png` - Security group update for custom SSH port
- `ssh ipv6 login on port 1234 ec2.png` - IPv6 SSH connection verification

Application Deployment on EC2

- `docker ps running on ec2.png` - Docker container status on EC2
- `pulling and running that docker container with bindmound and portmapping on ec2.png` - Container deployment with persistent storage
- `webpage url ipv4 ec2.png` - Application accessible via IPv4
- `webpage url ipv6 from ec2.png` - Application accessible via IPv6

AWS IAM & S3 Integration

- `creating s3 bucket with acl enabled.png` - S3 bucket creation process
- `s3 bucked with disable public access.png` - S3 security configuration
- `running dashboard ss for s3.png` - S3 dashboard overview
- `s3readonlyaccess for s3 IAM main ss.png` - IAM role configuration for S3 access
- `attched iam role to ec2.png` - IAM role attachment to EC2 instance
- `checking s3 access inside ec2 via aws s3 ls.png` - S3 access verification from EC2

Automation & Cloud-Init

- `cloud-init ec2 instance running.png` - Cloud-init automated EC2 instance
- `cloud-init ec2 output url (ipv4 url).png` - Automated deployment verification
- `gitclone repo and running deploy.sh in ec2.png` - Deployment script execution