# Launch Docker using Docker Compose on EC2 (with Optional HTTPS)

## ✅ Step 1: Prerequisites for EC2 Instance Setup

To deploy and run Docker Compose services for CloudPi, ensure the following EC2 instance configuration:

* **Instance Name**: CloudPiDockerHost
* **AMI**: Amazon Linux 2023 AMI
* **Instance Type**: t2.2xlarge
* **Key Pair**: Existing or new key pair (download and save the .pem file)
* **Storage Configuration**:
  + Volume Size: **512 GB**
  + Volume Type: **gp3**
* **Networking Configuration**:
  + Auto-assign **Public IP**: Enabled
  + Open the following **inbound ports** in the security group:
    - TCP: 22 (SSH)
    - TCP: 80, 443 (HTTP/HTTPS)
    - TCP: 3000, 5001, 5005, 6379, 8088, 3306 (Application-specific ports)

## ✅ Step 2: Connect to EC2 Instance

ssh -i cloudpi\_private\_key.pem ec2-user@<EC2\_PUBLIC\_IP>

## ✅ Step 3: Install Docker and Docker Compose and git

sudo yum update -y

sudo amazon-linux-extras enable docker

sudo yum install -y docker

sudo service docker start

sudo usermod -a -G docker ec2-user

🔁 Log out and log back in to apply Docker group permissions.

### ➕ Install Docker Compose v2:

mkdir -p ~/.docker/cli-plugins

curl -SL https://github.com/docker/compose/releases/download/v2.24.2/docker-compose-linux-x86\_64 -o ~/.docker/cli-plugins/docker-compose

chmod +x ~/.docker/cli-plugins/docker-compose

docker compose version  
  
**Install Git**  
sudo yum update -y

# Install Git

sudo yum install git -y

✅ **Step 4: Upload or Clone Your Docker Compose Project**

🔹 **Clone from GitHub**

# Clone the CloudPi repository

git clone https://github.com/PurpleDataInc-TX/cloudpi.git

cd cloudpi

✅ **After cloning, make sure the required files are present:**

ls

You should see both of the following files in the directory:

* docker-compose.yml
* .env

### 🔹 Set Up the .env File

CloudPi requires environment variables to be set at runtime. Make sure you have a .env file in the same directory as docker-compose.yml and add the necessary values. Make sure you have correct .env if any issue contact cloudpi.  
  
**Note:** Replace placeholders with correct values.

### ✅ Step 5: Upload SSL Certificates (If Using HTTPS)

If HTTPS is required for your CloudPi setup, follow these steps to securely upload the SSL certificate files to your EC2 instance.

#### 🔼 Upload Certificate Files to EC2

Run the following command from your local system (where your certificates are stored):

scp -i cloudpi\_private\_key.pem -r ./certs ec2-user@<EC2\_PUBLIC\_IP>:/home/ec2-user/certs

Replace <EC2\_PUBLIC\_IP> with your actual EC2 instance's public IP address.

#### 🔐 Ensure the following files exist in /home/ec2-user/certs on your EC2 instance:

/home/ec2-user/certs/

├── ca\_bundle.crt

├── cloudpi\_certificate.crt

└── cloudpi\_private.key

These files should correspond to:

* **cloudpi\_certificate.crt** – Your domain's main certificate
* **cloudpi\_private.key** – The private key associated with your certificate
* **ca\_bundle.crt** – The intermediate CA chain file (sometimes called fullchain)

Make sure the folder and files are readable by Nginx and have the correct permissions (e.g., chmod 600 on the key file).

## ✅ Step 6: Run Docker Compose

### cd ~/cloudpi

docker compose up –d  
  
🔁 **After Docker Compose is up and running:**

We need to **copy Cloudrestartnew.sh into the running container's /app directory**, set executable permissions, and then run it inside the container.

### 📝 **Note:** The script Cloudrestartnew.sh is located in the cloned repo **outside the container**. It must be placed inside the container to be executed properly. 🛠️ Steps to move Cloudrestartnew.sh into the container and run it

#### 1. ****Check container status****

docker ps

Look for the container with the name cloudpi-app.

#### 2. ****Copy the script into the container’s**** /app ****directory****

docker cp Cloudrestartnew.sh cloudpi-app:/app/

#### 3. ****Connect to the container****

docker exec -it cloudpi-app /bin/bash

#### 4. ****Inside the container: Set permissions & run the script****

cd /app

chmod +x Cloudrestartnew.sh

You can now proceed to the next step (HTTPS configuration or service validation).

## ✅ Step 7: Configure HTTPS Option via Cloudrestartnew.sh

You can enable/disable HTTPS and set a subdomain by editing the cloud\_restart.sh script inside the container.

### 1. (Optional) Install nano Editor

If nano is not installed:

apt-get update && apt-get install nano

### 2. Modify the Script

cd /app

nano Cloudrestartnew.sh

#### To Enable HTTPS:

use\_https=true

SUBDOMAIN=cloudpie.ai

#### To Disable HTTPS:

use\_https=false

💾 Save with Ctrl + O, Enter  
❌ Exit with Ctrl + X

### 5. Run the Script

./ Cloudrestartnew.sh

## ✅ Step 8: Verify Services

Open the following in your browser:

* http://<EC2\_PUBLIC\_IP>:3000
* Or, if HTTPS is enabled and DNS is configured:  
  https://<SUBDOMAIN> (e.g., https://cloudpie.ai)

## 🔁 Common Docker Commands

### List Running Containers

docker ps

### Connect to a Container

docker exec -it <container\_name> /bin/bash

### Stop All Containers

## docker compose down 🧹 Clean-Up Commands to Delete All (Docker Resources)

### ✅ 1. ****Stop and Remove All Docker Containers, Volumes, Networks, and Images****

Run these **inside your EC2 instance**:

# Step into your project folder (if needed)

cd ~/cloudpi

# Stop and remove all Docker Compose services

docker compose down

# Remove unused containers, networks, images, build cache

docker system prune -a --volumes -f