# **CloudPi Platform Deployment Guide**

## **1. Introduction**

Welcome to the CloudPi Platform! This guide provides comprehensive, step-by-step instructions for deploying the entire CloudPi application stack using Docker.

The containerized deployment ensures a consistent, reliable, and isolated environment, simplifying the setup process and allowing you to get started quickly. This guide is intended for system administrators and developers responsible for setting up and managing the CloudPi instance.

## **2. Prerequisites**

Before you begin, please ensure the target server meets the following requirements:

* **Operating System:** A compatible Linux distribution (Ubuntu 24.02 LTS is recommended).
* **Docker Engine:** Version 20.10 or newer.
* **Docker Compose:** Version 2.5 or newer (often included with Docker Desktop or installed as a plugin).
* Minimum **8 GB RAM** and **20 GB** free disk space.
* Make sure **3000 ,5001 ,5005 ,3306 ,6379 ,8088 ,80 ,443** ports are available.
* **Administrative Privileges:** You must have sudo access on the server.
* **Project Files:** You should have the CloudPi project directory, which includes the docker-compose.yml file.

**Installation Help:** If Docker is not installed, please follow the official guide for your operating system: [Install Docker Engine](https://docs.docker.com/engine/install/ubuntu/).

## **3. Step 1: Obtain Your Configuration File**

Clone the GitHub repository containing docker-compose.yml, .env:

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

Before starting the deployment, log in to Docker using the credentials provided by CloudPi:

**docker login –u <username>**

**When Prompted:**

* **Username**: contact cloudpi (refer to the email shared).
* **Password or Token**: contact cloudpi (refer to the email shared).

Once you receive the file, place it in the root of your CloudPi project directory, alongside the docker-compose.yml file.

## **4. Step 2: Environment Configuration**

The .env file controls how your application behaves. Before launching the application, you must review and configure the networking settings to match your environment.

Open the .env file in a text editor to make the following adjustments.

### **4.1. Network Host Configuration (Required)**

The HOST variable determines the IP address or domain name the application will be accessible from.

* **For Local Access Only:** If you only need to access the application on the server itself, you can leave the default setting:

HOST=localhost

* **For Network Access:** If you need to access the application from other computers on the same network (e.g., your laptop), you must set this to the server's private IP address.

**To find your server's IP address on Ubuntu, run the command ip addr show.** Look for the address under an interface like eth0 or ens18.

*Example:*

HOST=192.168.1.50

### **4.2. (Optional) Enabling HTTPS/SSL**

For a secure deployment using HTTPS, you must enable the feature and provide valid SSL certificates.

1. **Create a certs directory** inside your main project folder.
2. **Place your certificate files** inside this new certs directory. The files must be named as follows:
   1. certificate.crt (your SSL certificate)
   2. private.key (your private key)
   3. ca\_bundle.crt (your CA bundle/intermediate certificate, if applicable)
3. **Update the .env file** to enable HTTPS and, if applicable, set your subdomain:# Set to true to enable HTTPS.  
   HTTPS=true  
     
   # The public-facing subdomain (e.g., myapp.yourdomain.com)  
   SUBDOMAIN=cloudpi.yourcompany.com  
    The certificate paths in the .env file are pre-configured to point to the certs folder and should not be changed.

## **5. Step 3: Deploy the Application**

With the configuration complete, you can now launch the entire application stack.

1. Navigate to the root of your project directory in the terminal.
2. Run the following command. The sudo prefix is required as Docker needs administrative privileges to manage networks and containers.

**sudo docker compose up -d**

* **What this command does:**
  + sudo: Executes the command with root privileges.
  + docker compose up: Reads your docker-compose.yml and .env files to build and start the application stack.
  + -d: Runs the containers in "detached" mode, meaning they will run in the background.

The first time you run this command, Docker will download all the necessary images, which may take several minutes depending on your internet connection. Subsequent launches will be much faster.

## **6. Step 4: Verify the Deployment**

After running the up command, it's important to verify that all services have started correctly.

1. **Check Container Status:**

**sudo docker compose ps**

This will display a list of all running containers. Ensure that all services (e.g., cloudpi-db, cloudpi-app) have a STATUS of **running** or **healthy**.

1. **View Logs (for troubleshooting):** If a container is not running correctly, you can view its logs to diagnose the issue.

**sudo docker compose logs -f**

## **7. Step 5: Accessing the Application**

Once all containers are running, you can access the various components of the CloudPi platform through your web browser.

|  |  |
| --- | --- |
| **Service** | **URL** |
| **Cloudpi App (If HTTP)** | http://<your-host-ip> |
| **Cloudpi App (If HTTPS)** | https://<your-host-ip> |

**Note:** Replace <your-host-ip> with the value you configured for the HOST variable in your .env file (or localhost if you are accessing it from the server itself).

## **8. Step 6: Managing the Application**

You can manage the lifecycle of your CloudPi deployment with the following commands. Always run them from your project's root directory.

|  |  |  |
| --- | --- | --- |
| **Action** | **Command** | **Description** |
| **Stop Application** (Keeps data) | sudo docker compose stop | Pauses all running containers. You can restart them quickly. Your database data is preserved. |
| **Start Application** | sudo docker compose start | Restarts containers that have been stopped. |
| **Tear Down** (Deletes containers) | sudo docker compose down | Stops and removes the containers and network. Your database data in the volume is **preserved**. |
| **Full Cleanup** (Deletes everything) | sudo docker compose down -v | **DANGER:** Stops and removes containers, network, AND **deletes all database and Redis data volumes.** |

## **9. Support**

If you encounter any issues during the deployment process, please do not hesitate to contact our support team for assistance at [**contact@cloudpi.ai**](mailto:contact@cloudpi.ai).

© 2025 **CloudPi**

All rights reserved. Unauthorized redistribution or commercial use is prohibited.