Overview and Install of Incus Commander

https://pt.opensourceisawesome.com/videos/embed/aiuBBsMGR9x92aqEY5baRC

This guide provides a step-by-step walkthrough for deploying Incus-Commander using Docker, connecting your Incus infrastructure, and completing the essential administrative configurations (OIDC, SMTP, and Multi-tenancy).

Repository is available here: https://forge.routemehome.org/brian/incus-commander

---

Phase 1: Deployment (Docker)

1. Prepare the Environment

Ensure you have **Docker** and **Docker Compose** installed. Create a directory for your deployment and prepare your environment file.

bash

```
mkdir -p docker/incus-commander && cd docker/incus-commander
```

```
nano .env
```

2. Configure Environment Variables

Edit the `.env` file by copying the code block below, and pasting it into the file we just created.

```
POSTGRES_USER=incus_commander
POSTGRES_PASSWORD=< a-L0n6_S7ronG-P4ssw0RD-needs-to-G0-here >
POSTGRES_DB=incus_commander
FERRETDB_DB=incus_logs
SECRET_KEY=< use the command -> openssl rand --base64 32 >
```

```
FRONTEND_URL=http://incus.mygreatdomain.org # <- Where you intend to reach your incus-commander
SERVER_URL=https://incus.mygreatdomain.org # <- same as above in most cases
VITE_API_URL=/api
#ALLOWED_ORIGINS=incus.mygreatdomain.org # <- comma separated list of ways a user may access your server
```

Replace the values listed inside of the less than “<“ and greater than “>” signs with your actual values.

For the SECRET_KEY, you can run the command in a terminal window as follows:

`openssl rand -base64 32`

This will give you a strong value for the SECRET_KEY. Copy and paste the output value into the .env file.

Once you’ve replaced all the values as needed, save with CTRL + O, then press Enter to confirm, and exit the editor with CTRL + X.

Variable	Recommendation
`POSTGRES_PASSWORD`	Generate a strong random password.
`SECRET_KEY`	Use a long, random string (crucial for JWT security).
`FRONTEND_URL`	Your production domain (e.g., `https://commander.example.com`).
`SERVER_URL`	Your production domain (used for OIDC callbacks).
`ALLOWED_ORIGINS`	Your production domain (to prevent CORS errors).

3. Setup Docker Compose Configuration File

Now we’ll create the docker compose yaml file that tells the system how to run Incus Commander:

```
---
services:
  postgres:
    image: postgres:16-alpine
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    volumes:
```

```
      - ./postgres_data:/var/lib/postgresql/data
    restart: unless-stopped
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER}"]
      interval: 10s
      timeout: 5s
      retries: 5

  incus-commander:
    image: bmcgonag/incus-commander:latest
    ports:
      - "80:80"
    environment:
      - POSTGRES_HOST=postgres
      - POSTGRES_PORT=5432
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_DB=${POSTGRES_DB}
      - SECRET_KEY=${SECRET_KEY}
      - FRONTEND_URL=${FRONTEND_URL}
      - SERVER_URL=${SERVER_URL}
      - ALLOWED_ORIGINS
    depends_on:
      postgres:
        condition: service_healthy
    restart: unless-stopped
    volumes:
      - ./app/certs:/app/certs
    cap_add:
      - CAP_NET_BIND_SERVICE
```

There should be no modifications needed for this file. Simply save with CTRL + O, then press Enter to confirm, and exit the editor with CTRL + X.

4. Pull Images and Launch Services

Pull the images with

```
docker compose pull
```

Next, start the stack in detached mode:

bash

```
docker compose up -d
```

*Note: The first user to register via the web interface will be automatically promoted to **System Administrator**.*

Phase 2: Connecting Incus Hosts

To allow Incus-Commander to manage your servers, you must establish a secure connection using client certificates.

1. Generate Certificates via UI

1. Log in to Incus-Commander as an Admin.
2. Navigate to the **Systems** view and click **Add System**.
3. Enter a name and the Incus endpoint (e.g., `https://10.0.0.5:8443`). *NOTE: Port 8443 is the default port for the Incus API. If you haven't changed it, it should be 8443. You must access the API over https.*
4. Click **Generate Client Certificate**.
5. **Important:** Copy the displayed PEM certificate and Private Key and save them to a secure file on your local machine or server.

2. Trust the Certificate on the Incus Host

First, copy the certificate generated by Incus Commander to your Incus Server, and save it to a file in an path you have write access to.

Save the file with a name like 'incus-cmdr.crt' or 'cert.crt'. You'll need this path and filename in the next step, which will be run on your server, not the incus-commander host.

On the **actual Incus server you are trying to add,** run the following command to allow the certificate to communicate with the Incus API:

bash

```
# Assuming you saved the cert as incus-commander.crt
incus config trust add-certificate /path/to/incus-commander.crt
```

3. Finalize Connection

1. Go back to the **Edit System** form in Incus-Commander.
2. Provide the paths to the certificate and key files you generated in Step 1.
3. Save. If successful, the system status will change to **Connected** and show live resource usage.

Phase 3: Admin Level Setup

Once the systems are connected, perform these three critical setup steps to make the platform production-ready.

1. SMTP Configuration (Email)

Email is required for sending invitation tokens to new users and alerting admins of backup failures.

1. Navigate to **System Settings**.
2. Find the **Email Setup** section.
3. Enter your SMTP details:
   - `SMTP Host`, `Port`, `Username`, and `Password`.
   - `From Email` (the address users will see).
   - Set `Use TLS` if required by your provider.
4. **Test it:** Use the **Send Test Email** button to ensure your credentials are correct.

2. OIDC / SSO Configuration

To allow users to log in via providers like Authentik, Keycloak, or Google:

1. Navigate to **System Settings > OIDC Providers**.
2. Click **Add Provider**.
3. Enter the **Issuer URL**, **Client ID**, and **Client Secret** provided by your OIDC service.
4. **Configure the Redirect URI:** In your OIDC provider's admin panel, you **must** whitelist the following callback URL:
   `https://<your-server-url>/api/auth/callback/<provider_name>`
   (Example: `https://commander.example.com/api/auth/callback/Authentik`)
5. Save and test the login button on the main login page.

3. Multi-Tenancy & Project Isolation

If you are running this for different departments or clients, set up Tenants to ensure resource isolation.

1. **Create a Tenant:**
   - Go to **System Settings > Tenants** and click **Add Tenant**.
   - Enter a name and, crucially, the **Incus Project** name. This ensures that all instances created by this tenant are isolated within that Incus project.
2. **Assign Users:**
   - Go to **System Settings > Users**.
   - Edit a user and assign them to the newly created Tenant.
3. **Tenant Admin (Optional):**
   - If you want a user to manage their own templates and users, promote them to `tenant_admin` in the Users menu.

Summary Checklist for Production

☐
`SECRET_KEY` is long and random.

☐
Reverse proxy (Nginx/Caddy) is configured with TLS.

☐
WebSocket support is enabled in your proxy (required for the Root Console).

☐
SMTP test email was successful.

☐
OIDC callback URL is correctly white-listed in your provider.

☐
Incus hosts are "Connected" and showing live data.

Support this Channel and Content

Become a Patron at Patreon

Check me out on Patreon

Be me a Coffee or Beer at Paypal

Paypal Support for Awesome Open Source

Get the Awesome Open Source Merch

Great Merch for Open Source Advocates!

Revision #3
Created 2026-10-17 12:44:17 UTC by Brian McGonagill
Updated 2026-10-17 13:07:52 UTC by Brian McGonagill