Refine Deployment Guide

Step-by-step instructions for deploying Refine on AWS EC2 instances. The setup process is designed to be as simple as possible — most steps are handled by automated scripts.

> ***AWS Marketplace customers:*** *This guide covers the* ***direct-sales delivery-ZIP*** *path. If you subscribed to Refine through AWS Marketplace, your deployment is mostly automated by the Marketplace CloudFormation product — see docs/marketplace/DEPLOYMENT_GUIDE.md instead. The two flows differ in how the Refine container and license arrive (Marketplace: container pulled from AWS Public ECR + license emailed to you + you `aws s3 cp` it into your CFN's delivery bucket; direct-sales: a single ZIP delivered by Blacktip containing everything). Once Refine is running, both paths use the same UI, the same per-account onboarding, and the same daily-use flow described later in this guide.*

Table of Contents

---

1. Deployment Overview

What You're Deploying

Refine is an AWS cost optimization platform that analyzes 14 AWS services and provides actionable recommendations:

- **14 service inventory** — EC2, RDS, S3, EBS Volumes, EBS Snapshots, NAT Gateways, Load Balancers, Lambda, CloudWatch Logs, Fargate, ElastiCache, Data Transfer, Batch Jobs, Lightsail
- **Optimization recommendations** with exact AWS CLI commands for every action
- **Savings Plans tracking** — monitor utilization of purchased Compute SPs, EC2 Instance SPs, Database SPs, and RDS Reserved Instances with guardrails that prevent underutilization
- **Savings Report** — baseline vs actual cost tracking with 12-month trend analysis
- **Exemptions** — exclude resources from recommendations with documented reasons (contractual, compliance, performance, etc.)

- **Cost Cleanup** — find and remove resources by tag with downloadable cleanup scripts
- **System Events** — monitor sync health, errors, and warnings

See the User Guide for a walkthrough of every page.

What's in the Package

Refine ships as a delivery package (ZIP file) that contains everything you need:

File	Purpose
`setup.bat`	**Windows** — double-click to run the setup wizard
`setup.sh`	**Linux / Mac** — run with `./setup.sh`
`setup.py`	Setup wizard (called by the scripts above)
`docker-compose.yml`	Container configuration
`refine.license`	Your signed license file
`.env.example`	Configuration template

The setup process is fully automated. The wizard:

1. Checks that Docker and Python are installed (gives install instructions if not)
2. Prompts for an activation code (provided by Blacktip Solutions)
3. Prompts for an admin email and password (this becomes the ROOT account)
4. Asks which port to use (default: 8000)
5. Generates a JWT secret
6. Pulls (or loads) the Docker image
7. Starts the container and waits for it to be healthy
8. Opens your browser to the Refine dashboard

No manual configuration files to edit. The wizard handles everything.

Scale	Where to Run	Database	Setup Time
1-5 accounts	**Your laptop / desktop**	SQLite (built-in)	5 minutes
1-10 accounts	EC2 `t3.small` (2 vCPU, 2 GB)	SQLite (built-in)	10 minutes
10-50 accounts	EC2 `t3.medium` (2 vCPU, 4 GB)	SQLite (built-in)	10 minutes
50-200 accounts	EC2 `m6i.xlarge` (4 vCPU, 16 GB)	PostgreSQL (add later)	15 minutes
200+ accounts	EC2 `m6i.2xlarge` (8 vCPU, 32 GB)	PostgreSQL required	20 minutes

See SYSTEM_REQUIREMENTS.md for detailed sizing guidance.

2. Run Locally on Your Laptop or Desktop

You can run Refine directly on your Windows, Mac, or Linux machine. This is the fastest way to get started and works well for small environments (1-5 AWS accounts).

Requirements

- **Docker Desktop** installed and running (download)
- **Python 3.8+** installed
- At least **2 GB of free RAM** (Docker Desktop allocates this from your system)

Setup

Extract the delivery ZIP to any folder and run the setup wizard:

Windows: Double-click `setup.bat`

Mac / Linux:

```
cd ~/Downloads/refine-CUST-XXX   # wherever you extracted the ZIP
chmod +x setup.sh
./setup.sh
```

The wizard handles everything — Docker check, activation code, configuration, image pull, and startup.

Limitations of Running Locally

Concern	Impact	Recommendation
Laptop sleeps / shuts down	Nightly sync won't run if the machine is off. Data goes stale until you restart.	Fine for evaluation. For always-on monitoring, use an EC2 instance.
Shared CPU/RAM	Other applications compete for resources. With 5+ accounts, the dashboard may be slower.	Close heavy apps (Slack, Chrome tabs) during use, or allocate more RAM to Docker Desktop.
Network dependency	Refine calls AWS APIs (STS, Lambda, S3). If your laptop is on a VPN or restrictive network, connections may fail.	Ensure outbound HTTPS (443) to `*.amazonaws.com` is allowed.
Not accessible to teammates	Only you can access `http://localhost:8000`.	For team access, deploy on EC2 or share your local network IP.
5+ AWS accounts	Performance degrades as data volume grows. Dashboard loads slow, sync takes longer.	Migrate to EC2 when you exceed 5 accounts or need always-on availability.

> ***Bottom line:*** *Local deployment is great for evaluation, demos, and small teams with 1-5 AWS accounts. For production use or 10+ accounts, deploy on an EC2 instance for reliability and performance.*

Migrating from Local to EC2 Later

When you're ready to move to a dedicated server:

1. Launch an EC2 instance (see next section)

2. Copy your delivery folder to the EC2 instance
3. Run `./setup.sh` on the EC2 instance — it's a fresh install
4. Re-connect your AWS accounts from the Refine web UI (account configuration is stored in the database, which doesn't transfer automatically)

---

3. Prepare Your EC2 Instance

> ***Want a one-click deploy?*** *Use the* ***AWS Linux Deployment Guide*** *— it includes a CloudFormation template that creates the EC2, installs Docker, downloads your delivery ZIP automatically, and* ***optionally provisions an Application Load Balancer with HTTPS*** *if you provide an ACM cert. The manual steps below are for customers who prefer to launch their own EC2.*

Step 1: Launch the Instance

1. Open the EC2 Console
2. Click **Launch Instance**
3. Configure:

Setting	Value
Name	`refine-server`
AMI	Amazon Linux 2023 **or** Ubuntu 22.04 LTS
Instance type	See table above for your scale
Key pair	Select or create an SSH key pair
Storage	20 GB gp3 (increase to 50 GB for 50+ accounts)

4. Under **Network settings**, create a security group with:

Type	Port	Source
SSH	22	Your IP
Custom TCP	8000	Your IP (or `0.0.0.0/0` for open access)

5. Click **Launch Instance**

Step 2: Install Docker and Python

SSH into your instance:

```
ssh -i your-key.pem ec2-user@your-ec2-ip    # Amazon Linux
# or
ssh -i your-key.pem ubuntu@your-ec2-ip       # Ubuntu
```

Amazon Linux 2023:

```
# Install Docker + Python (one command)
sudo dnf update -y && sudo dnf install -y docker python3 unzip

# Start Docker and enable on boot
sudo systemctl enable --now docker
sudo usermod -aG docker $USER

# Install Docker Compose
sudo mkdir -p /usr/local/lib/docker/cli-plugins
sudo curl -SL https://github.com/docker/compose/releases/latest/download/docker-compose-linux-x86_64 \
  -o /usr/local/lib/docker/cli-plugins/docker-compose
sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose

# IMPORTANT: Log out and back in for Docker group to take effect
exit
```

Ubuntu 22.04:

```
# Install Docker + Python (one command)
sudo apt update && sudo apt install -y python3 unzip
curl -fsSL https://get.docker.com | sh
sudo usermod -aG docker $USER

# IMPORTANT: Log out and back in for Docker group to take effect
exit
```

Reconnect and verify:

```
docker --version          # Should show Docker version
docker compose version    # Should show Compose version
python3 --version         # Should show Python 3.8+
```

Step 3: Upload Your Delivery Package

From your local machine, upload the ZIP:

```
scp -i your-key.pem refine-CUST-XXX.zip ec2-user@your-ec2-ip:~/
```

On the EC2 instance:

```
mkdir -p ~/refine && cd ~/refine
unzip ~/refine-CUST-XXX.zip
chmod +x setup.sh
```

---

4. Run the Setup Wizard

Once Docker, Python, and the delivery package are in place, the setup is a single command:

Linux / Mac:

```
cd ~/refine
./setup.sh
```

Windows (if running on Windows with Docker Desktop): Double-click `setup.bat`

What Happens

The wizard runs through these steps automatically:

```
 ============================================================
   Refine Setup Wizard
 ============================================================

   Checking system requirements...

   [✓] Python 3.11.9
   [✓] Docker 27.3.1 (engine running)

 ============================================================
   All checks passed -- starting Refine setup wizard...
 ============================================================

=========================================================
  Refine Setup
  YourCompany
  License: CUST-001  |  Expires: 2027-05-17
=========================================================

  This installation requires an activation code.
  Contact Blacktip Solutions if you did not receive one.

  Activation code [1/10]: RF-XXXX-XXXX-XXXX-XXXX
  Activation code accepted.

[1/4] Checking Docker...
  Docker version 27.3.1, build ...

  Admin email: admin@yourcompany.com
  Admin password: ********
  Confirm password: ********

  Port [8000]: 8000

[2/4] Writing configuration...
  .env written.

[3/4] Getting Refine image...
  Authenticating with container registry...
  Pulling image — this may take a few minutes...
  ✓ Image pulled

[4/4] Starting container...
  Waiting for Refine..... ready!

=========================================================
  Refine is running!

  URL      : http://localhost:8000
  Admin    : admin@yourcompany.com
  Role     : ROOT

  Log in with the password you chose during setup.

  License  : CUST-001  |  Expires: 2027-05-17
=========================================================
```

That's it. Open the URL in your browser and you're ready to connect your AWS accounts.

If Something Goes Wrong

The wizard provides specific instructions for any failure:

- **Docker not installed** — shows the exact install command for your OS
- **Docker not running** — tells you how to start it
- **Python not found** — shows download links for your platform
- **Image pull failed** — prompts for registry login or bundle path
- **Port in use** — asks you to choose a different port
- **Activation code wrong** — allows 10 retries before locking

Every error message includes a "fix this, then re-run `./setup.sh`" instruction.

Multi-User Architecture

Refine uses a **one server, multiple users** model. All customer AWS accounts are connected to a single Refine server, and access is controlled through user roles and account groups.

How It Works

- The person who runs the setup wizard becomes the first user with the **ROOT** role. There is one Root per server.
- Root creates additional users from the **User Management** page (`/user-management`) and assigns each user a role (ROOT, ADMIN, or USER).
- Root creates **account groups** — named collections of AWS accounts (e.g., "Production Team", "Dev Team"). Users are assigned to groups and see only the accounts in their groups.
- The ROOT user sees all accounts regardless of group assignment. ADMIN and USER roles see only their group-assigned accounts.

Roles at a Glance

Role	See All Accounts	Take Actions	Manage Users & Groups
ROOT	Yes	Yes	Yes (all)
ADMIN	No (group only)	Yes (within group)	Yes (within own groups)
USER	No (group only)	Yes (within group)	No

See the User Guide — User Management for detailed instructions.

Directory Integration (LDAP / Azure AD)

Refine can integrate with your existing Active Directory or Azure AD so employees sign in with corporate credentials instead of separate Refine passwords.

Option	Best For	How It Works
On-Prem AD (LDAP)	Government, air-gapped, traditional enterprise	Direct LDAP bind from Docker container to AD server
Azure AD / Entra ID	Cloud-first, Microsoft 365 organizations	OAuth SSO — "Sign in with Microsoft" button on login page

Directory integration is **optional** — you can set it up during the setup wizard or later from the **Directory Settings** page in the sidebar. The ROOT user always uses local email/password as an escape hatch.

See LDAP_SETUP_GUIDE.md or AZURE_AD_SETUP_GUIDE.md for step-by-step instructions.

---

5. Production: Add PostgreSQL

For 50+ accounts, PostgreSQL provides better performance than the built-in SQLite database. You can add it at any time — no data loss.

Option A: Amazon RDS (Recommended)

1. Open the RDS Console
2. Click **Create database**

Setting	Value
Engine	PostgreSQL 16
Template	Production
DB instance identifier	`refine-db`
Master username	`refine`
Master password	Generate and save a strong password
Instance class	`db.t3.medium` (50-100 accounts) or `db.t3.large` (100-200)
Storage	20 GB gp3, enable autoscaling to 100 GB
VPC	Same VPC as your EC2 instance
Public access	No
Initial database	`refine`

3. Under **Connectivity**, create a security group that allows port 5432 from your EC2's security group
4. Click **Create database** and wait for **Available** status (~5 minutes)
5. Copy the **Endpoint** from the Connectivity tab

Option B: PostgreSQL on the Same EC2

For simpler setups without a separate RDS instance:

```
# Add PostgreSQL to your docker-compose.yml
cat >> ~/refine/docker-compose.override.yml << 'EOF'
version: "3.9"
services:
  postgres:
    image: postgres:16-alpine
    environment:
      POSTGRES_DB: refine
      POSTGRES_USER: refine
      POSTGRES_PASSWORD: change-this-password
    volumes:
      - postgres_data:/var/lib/postgresql/data
    restart: unless-stopped

volumes:
  postgres_data:
EOF
```

Switch Refine to PostgreSQL

1. Stop Refine:

```
cd ~/refine
docker compose down
```

2. Edit `.env` — change the `DATABASE_URL` line:

```
# For RDS:
DATABASE_URL=postgresql://refine:YOUR_PASSWORD@refine-db.xxxx.us-east-1.rds.amazonaws.com:5432/refine

# For local PostgreSQL (same EC2):
DATABASE_URL=postgresql://refine:change-this-password@postgres:5432/refine
```

3. Restart:

```
docker compose up -d
```

Refine creates all tables automatically on startup. Your existing data (if any) stays in the old SQLite file at `data/refine.db` — it is not migrated automatically.

---

6. HTTPS with an Application Load Balancer

For production deployments with a custom domain and HTTPS.

Step 1: Request an SSL Certificate

1. Open AWS Certificate Manager
2. Click **Request certificate** > **Request a public certificate**

3. Enter your domain: `refine.yourdomain.com`
4. Choose **DNS validation**
5. Add the CNAME record to your DNS provider
6. Wait for status: **Issued**

Step 2: Create a Load Balancer

1. Open EC2 > Load Balancers
2. Click **Create Load Balancer** > **Application Load Balancer**

Setting	Value
Name	`refine-alb`
Scheme	Internet-facing
Subnets	Select at least 2 availability zones

3. **Listeners:**
   - HTTPS (443) — forward to target group, select your ACM certificate
   - HTTP (80) — redirect to HTTPS
4. **Target group:** HTTP on port 8000, health check path `/health`
5. Register your EC2 instance, click **Create**

Step 3: Update DNS and Config

1. Create a DNS record pointing `refine.yourdomain.com` to the ALB
2. Update `.env`:

```
APP_BASE_URL=https://refine.yourdomain.com
ENV=production
```

3. Restart: `docker compose down && docker compose up -d`
4. Lock down EC2 security group — only allow port 8000 from the ALB security group (remove public access)

---

7. Hosting Behind IIS (Windows Server)

For organizations that require IIS as the public-facing web server — common in Microsoft-heavy shops, government, and enterprises with a strict "all traffic through IIS" security posture.

How It Works

Refine still runs in Docker (on the same Windows Server or a separate Linux box). IIS sits in front as a reverse proxy, terminating SSL and forwarding requests to `http://localhost:8000`:

```
Internet ──► IIS (https://refine.company.com)
             │  Application Request Routing
             ▼
            Docker container (http://localhost:8000)
```

Benefits:

- SSL certificates managed in the Windows certificate store (not in Refine)
- Windows Authentication available (AD-integrated SSO)
- IIS access logs, request filtering, and WAF modules apply
- Meets "all public traffic must flow through IIS" compliance policies

Prerequisites

- Windows Server 2019 or later with IIS installed
- Docker Desktop installed (which runs the Refine container under Linux containers)
- Administrator access to both IIS and Docker

Step 1: Install IIS Prerequisites

Open **PowerShell as Administrator** and run:

```
# Install IIS + features needed for reverse proxy
Install-WindowsFeature -name Web-Server -IncludeManagementTools
Install-WindowsFeature Web-WebSockets

# Download and install URL Rewrite module
$urlRewrite = "https://download.microsoft.com/download/1/2/8/128E2E22-C1B9-44A4-BE2A-5859ED1D4592/rewrite_
Invoke-WebRequest -Uri $urlRewrite -OutFile "$env:TEMP\rewrite_amd64.msi"
Start-Process msiexec.exe -Wait -ArgumentList "/i $env:TEMP\rewrite_amd64.msi /quiet"

# Download and install Application Request Routing (ARR)
$arr = "https://download.microsoft.com/download/E/9/8/E9849D6A-020E-47E4-9FD0-A023E99B54EB/requestRouter_a
Invoke-WebRequest -Uri $arr -OutFile "$env:TEMP\arr.msi"
Start-Process msiexec.exe -Wait -ArgumentList "/i $env:TEMP\arr.msi /quiet"

# Restart IIS to pick up the new modules
iisreset
```

Step 2: Enable ARR Proxy

1. Open **IIS Manager** (`inetmgr`)
2. Click the **server node** in the left tree (not a site)
3. Open **Application Request Routing Cache**
4. In the right Actions panel, click **Server Proxy Settings...**
5. Check **Enable proxy**
6. Click **Apply**

Step 3: Install Your SSL Certificate

1. In IIS Manager, click the server node
2. Open **Server Certificates**

3. **Import** your certificate (`.pfx` file) or request a new one via **Create Certificate Request** → CA → **Complete Certificate Request**

If you're testing with a self-signed cert, use **Create Self-Signed Certificate** for non-production only.

Step 4: Create the IIS Site

1. In IIS Manager, right-click **Sites** → **Add Website...**
2. Configure:

Setting	Value
Site name	`Refine`
Physical path	`C:\inetpub\refine` (create this folder — it stays empty, IIS just needs a root)
Binding type	`https`
IP address	All Unassigned
Port	`443`
Host name	`refine.company.com` (your domain)
SSL certificate	Select the cert you imported

3. Click **OK**

Step 5: Configure the Reverse Proxy Rule

Create a `web.config` file at `C:\inetpub\refine\web.config` with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <rewrite>
            <rules>
                <rule name="Proxy to Refine Docker" stopProcessing="true">
                    <match url="(.*)" />
                    <action type="Rewrite" url="http://localhost:8000/{R:1}" />
                    <serverVariables>
                        <set name="HTTP_X_FORWARDED_PROTO" value="https" />
                        <set name="HTTP_X_FORWARDED_HOST" value="{HTTP_HOST}" />
                    </serverVariables>
                </rule>
            </rules>
        </rewrite>
        <security>
            <requestFiltering>
                <requestLimits maxAllowedContentLength="10485760" />
            </requestFiltering>
        </security>
    </system.webServer>
</configuration>
```

The `X-Forwarded-Proto` and `X-Forwarded-Host` headers tell Refine that the original request was HTTPS (so it generates correct URLs in emails and redirects).

Step 6: Allow IIS to Set Server Variables

IIS by default blocks custom server variables. Allow the two we use:

1. In IIS Manager, click the server node
2. Open **Configuration Editor**
3. In the **Section** dropdown, choose `system.webServer/rewrite/allowedServerVariables`
4. Click **Edit Items** in the right panel, then **Add**
5. Add `HTTP_X_FORWARDED_PROTO` → Apply
6. Add `HTTP_X_FORWARDED_HOST` → Apply

Step 7: Update Refine Configuration

```
# In your .env file
APP_BASE_URL=https://refine.company.com
ENV=production
```

Restart the container:

```
docker compose down && docker compose up -d
```

Step 8: Lock Down Port 8000

Now that IIS fronts all traffic, firewall-off the Docker port from external access:

```
New-NetFirewallRule -DisplayName "Block Refine Docker Port" -Direction Inbound -LocalPort 8000 -Protocol T
```

This way the only public entry point is IIS on 443.

Optional: Windows Authentication (AD SSO)

To require Windows Auth (users must be domain-joined) before reaching Refine:

1. In IIS Manager, select your Refine site
2. Open **Authentication**
3. Disable **Anonymous Authentication**
4. Enable **Windows Authentication**

Note: This puts an additional auth layer in front of Refine's own login. Useful for air-gapped corporate intranets where you want to tie access to AD group membership before Refine's own licensing kicks in.

Verification

- Open `https://refine.company.com` in a browser — should load the Refine dashboard
- Check IIS logs at `C:\inetpub\logs\LogFiles\W3SVC1\` to confirm traffic is flowing through
- In Docker: `docker compose logs --tail 50` should show requests with `X-Forwarded-For` headers

Troubleshooting

502 Bad Gateway: Docker container isn't running. Check `docker compose ps` and start if needed.

ARR error "could not find module": Re-run the Install-WindowsFeature + MSI commands from Step 1, then `iisreset`.

Mixed content warnings: `APP_BASE_URL` is still `http://`. Update to `https://` in `.env` and restart.

Login redirects loop: `X-Forwarded-Proto` header isn't being set. Confirm the `allowedServerVariables` step was completed.

---

8. Air-Gapped / Offline Deployment

For environments with no internet access (government, military, restricted networks).

The delivery ZIP includes a bundled Docker image (~500 MB `.tar` file). No internet access is needed after the ZIP is transferred.

Steps

1. Transfer the ZIP to the target machine via approved method (USB, SFTP, etc.)
2. Extract and run the setup wizard:

```
mkdir -p ~/refine && cd ~/refine
unzip refine-GOV-XXX.zip
chmod +x setup.sh
./setup.sh
```

The wizard automatically detects the bundled image and loads it with `docker load` instead of pulling from a registry. No registry login is needed.

See AIR_GAPPED_SETUP_GUIDE.md for detailed instructions.

---

9. Updating Refine

Method 1: Re-run the Setup Script (Recommended)

The same script you used for installation handles updates:

```
cd ~/refine
./setup.sh
```

The wizard detects the existing installation and runs in **update mode**:

```
  Existing Refine installation detected.
  Running in UPDATE mode — your configuration and data are preserved.

[1/2] Pulling latest image...
  Pulling image — this may take a few minutes...
  ✓ Image pulled
  Updating: v2.11.0 -> v2.12.0

[2/2] Restarting Refine...
  Starting container...
  Waiting for Refine...... ready!

============================================================
  Refine updated! (v2.11.0 -> v2.12.0)

  URL      : http://localhost:8000
  License  : CUST-001  |  Expires: 2027-05-17
============================================================
```

What update mode preserves:

- All user accounts and passwords (not regenerated)
- JWT secret (sessions stay valid)
- Port selection
- Database and all data
- Connected AWS accounts
- All configuration in `.env`

Method 2: Manual Update

```
cd ~/refine
docker compose pull          # Pull latest image
docker compose down          # Stop current container
docker compose up -d         # Start with new image
```

Pin a Specific Version

Edit `docker-compose.yml` and change `image: ghcr.io/eichenbergerb/refine:latest` to a specific version tag (e.g., `ghcr.io/eichenbergerb/refine:v2.12.0`).

---

10. Configuration Reference

The setup wizard creates the `.env` file automatically. You should not need to edit it manually unless adding PostgreSQL or customizing email settings.

Core Settings (set by wizard)

Variable	Description
`JWT_SECRET_KEY`	Authentication secret (auto-generated, 64 hex chars)
`SELF_HOSTED`	Always `true` for self-hosted deployments
`REFINE_LICENSE_PATH`	`/app/refine.license` (container path)
`DATABASE_URL`	Database connection string
`PORT`	Web interface port (default: 8000)
`CONTAINER_NAME`	Docker container name

Database Options

Type	Connection String
SQLite (default)	`sqlite:////app/data/refine.db`
PostgreSQL (RDS)	`postgresql://user:pass@hostname:5432/refine`
PostgreSQL (local)	`postgresql://user:pass@postgres:5432/refine`
PostgreSQL (SSL)	`postgresql://user:pass@hostname:5432/refine?sslmode=require`

Optional Settings

Variable	Default	Description
`ENV`	`production`	Set to `local` to disable HSTS and secure cookies (dev only)
`APP_BASE_URL`	`http://localhost:8000`	Your public URL (for email links)
`LOG_LEVEL`	`INFO`	Logging verbosity (DEBUG, INFO, WARNING, ERROR)
`EMAIL_PROVIDER`	`console`	Email delivery: `console` (log only), `smtp`, or `ses`
`SMTP_HOST`	—	SMTP server (if EMAIL_PROVIDER=smtp)
`SMTP_PORT`	`587`	SMTP port
`SMTP_USER`	—	SMTP username
`SMTP_PASSWORD`	—	SMTP password

Docker Resource Limits

For 50+ accounts, add resource limits to `docker-compose.yml`:

```
services:
  refine:
    image: ghcr.io/eichenbergerb/refine:latest
    deploy:
      resources:
        limits:
          memory: 2G    # 2 GB for 50-200 accounts, 4 GB for 200+
          cpus: "2"
```

Volume Mounts

Mount	Purpose
`./data:/app/data`	Persistent storage (database, sync data)
`./refine.license:/app/refine.license:ro`	License file (read-only)

11. Monitoring and Maintenance

Health Check

```
curl -s http://localhost:8000/health
# {"status":"ok"}
```

Container Status

```
docker compose ps              # Is it running?
docker stats refine --no-stream # CPU and memory usage
```

Logs

```
docker compose logs --tail 50     # Recent logs
docker compose logs -f             # Follow in real-time
docker compose logs 2>&1 | grep -i error  # Find errors
```

Backups

SQLite:

```
docker compose stop
cp data/refine.db data/refine.db.backup.$(date +%Y%m%d)
docker compose start
```

PostgreSQL (RDS): RDS automated backups are enabled by default (7-day retention). For manual backups:

```
pg_dump "postgresql://refine:PASSWORD@HOST:5432/refine" \
  > refine_backup_$(date +%Y%m%d).sql
```

Disk Space

```
du -sh data/                # Data directory size
df -h /                     # Disk usage
docker system prune -f      # Clean unused Docker resources
```

---

12. Troubleshooting

Setup wizard says "Docker not found"

Install Docker first (see Step 2), then re-run `./setup.sh`.

Setup wizard says "Docker not running"

```
sudo systemctl start docker     # Start Docker
./setup.sh                       # Re-run wizard
```

Image pull fails with "denied" or "unauthorized"

Run the registry login script:

```
./registry-login.sh              # Authenticates with container registry
./setup.sh                       # Re-run wizard
```

Cannot connect to PostgreSQL

```
# Test connectivity from EC2
nc -zv your-rds-endpoint 5432
# If "Connection refused": check RDS security group allows port 5432 from EC2

# Test authentication
psql "postgresql://refine:PASSWORD@HOST:5432/refine" -c "SELECT 1;"
```

Dashboard shows no data

1. Check **AWS Accounts** page — account should show **Validated**
2. Click **Sync Now** on the dashboard to trigger a manual data refresh
3. Wait 2-5 minutes for the first sync to complete
4. Check **System Events** for any sync errors

High memory or slow performance

```
# Check memory usage
docker stats refine --no-stream

# If consistently above 80%:
# 1. Add memory limits to docker-compose.yml (see Configuration Reference)
# 2. Upgrade to a larger EC2 instance
# 3. Use account filtering on the dashboard
# 4. Switch to PostgreSQL for 50+ accounts
```

Container keeps restarting

```
# Check if killed by out-of-memory
docker inspect refine | grep -i oom

# Check error logs
docker compose logs --tail 100 | grep -i "error\|fatal\|killed"

# Fix: increase memory limit in docker-compose.yml
```

License issues

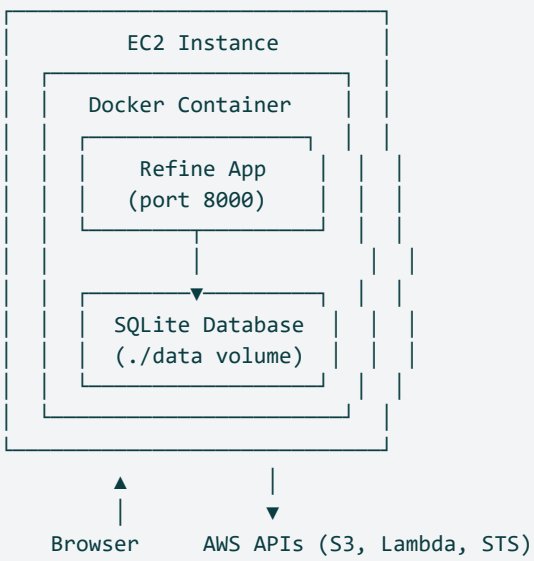
```
curl -s http://localhost:8000/auth/license-status | python3 -m json.tool
```

- **"License not found"** — verify `refine.license` is in the same directory as `docker-compose.yml`
- **"License expired"** — contact Blacktip Solutions for renewal
- **"Installation ID mismatch"** — the license is bound to a specific delivery; do not copy between installations

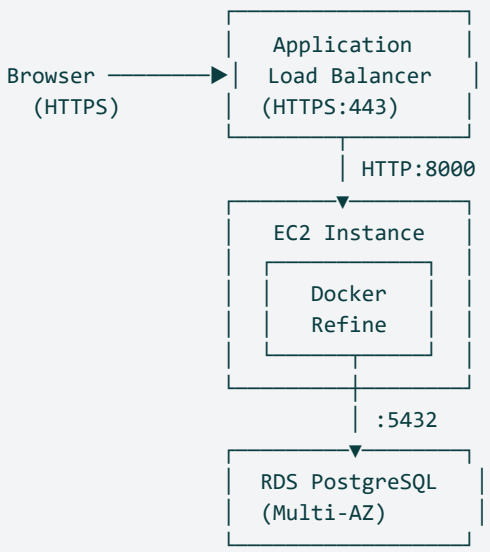
---

Architecture Diagrams

Starter (Single EC2 + SQLite)



Production (EC2 + RDS + ALB)



Need Help?

Contact Blacktip Solutions at **support@blacktipsolutions.com** for:

- Activation codes and license renewals
- Deployment assistance
- Scaling guidance
- Registry access credentials