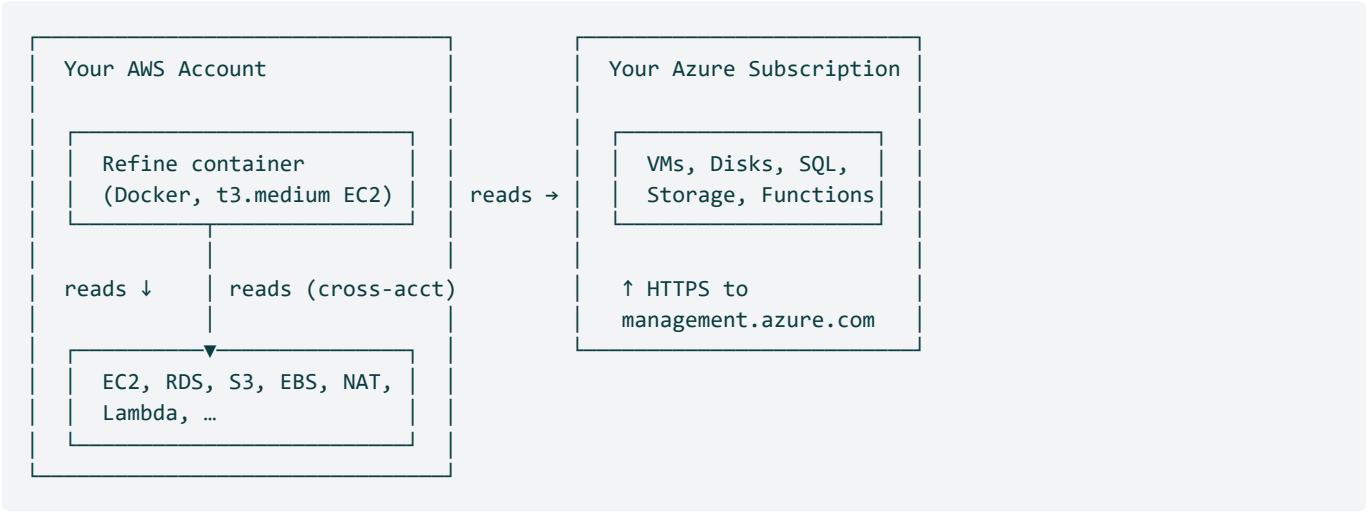
Refine — Customer Quick-Start

Single end-to-end guide: set up the Refine server on AWS, connect your AWS and Azure accounts, learn the daily-use flow, and troubleshoot common issues.

If you only read one page, read this one. Everything else in the `docs/` folder is reference material.

What you're about to set up



You'll deploy:

1. **The Refine server** — a single Docker container on an AWS EC2 instance that you own. Uses a SQLite database; no cloud database to manage.
2. **A CloudFormation stack in your AWS account** — gives Refine read-only permissions to your AWS resources. Refine never gets write access.
3. **A Service Principal in your Azure tenant** — gives Refine read-only access to Azure resources. Same model: read-only.

Total time, end to end: **~60 minutes**.

How did you get Refine? (two purchase paths)

Refine ships through two channels with slightly different setup flows. Pick the one that matches how your subscription was created:

Channel	What it looks like	Where to go
AWS Marketplace (BYOL)	You subscribed to *Blacktip Refine — Multi-Cloud Cost Intelligence* on AWS Marketplace, accepted the Standard Contract, and launched a CloudFormation stack from the listing. You received your license file via email from `AMelcher@blacktip-ops.com` (not as part of a ZIP).	Stop here. Follow docs/marketplace/DEPLOYMENT_GUIDE.md instead — the Marketplace flow handles EC2 provisioning, image pull, and per-account onboarding through a simpler path. Then come back here for **Part 2** (connecting Azure / GCP) and **Part 3** (daily-use flow).
Direct sales (delivery ZIP)	Blacktip Solutions sent you a delivery ZIP (~1.5 GB) via email, S3 link, or encrypted USB containing `refine.license`, `infra/refine-server-auto.yaml`, the Docker image tarball (for GovCloud / air-gap), and day-2 management scripts.	Continue with this guide.

This guide covers the **direct-sales path** end-to-end. If you're on the Marketplace path, the deployment steps in Part 1 are handled for you by the Marketplace CloudFormation; skip to **Part 2** below once your Refine UI loads.

Before you begin — what you'll need

Item	Notes
AWS account with admin permissions	To launch the EC2 host stack and the per-account onboarding stack
Azure subscription with Owner or User Access Administrator	To create the Service Principal
Your Refine delivery ZIP from Blacktip	Contains license, setup wizard, and config
Your `refine.license` file	Inside the ZIP. Bound to your installation.
An email address that gets used for Refine login	The first user you create is the root admin
30 GB free disk space on the EC2 host	For container + database + cost data
Outbound HTTPS access from the EC2 host	To `management.azure.com`, `login.microsoftonline.com`, and S3 endpoints

If you're in a regulated industry (GovCloud, FedRAMP), stop here and read `AIR_GAPPED_SETUP_GUIDE.md` instead.

Part 1 — Deploy the Refine server on AWS EC2

The Refine server runs as a single Docker container. The fastest path is to launch it from a CloudFormation template that creates the EC2, IAM role, security group, and Elastic IP for you.

1.1 — Upload the delivery ZIP to S3

The CloudFormation template downloads the Refine delivery ZIP from an S3 bucket you control. So first:

```
# Create a private S3 bucket (or reuse one)
aws s3 mb s3://my-refine-delivery-bucket --region us-east-1

# Upload the ZIP Blacktip sent you
aws s3 cp refine-CUSTOMER-vX.Y.Z.zip s3://my-refine-delivery-bucket/refine-delivery.zip
```

The S3 object name **must be exactly** `refine-delivery.zip` — the CloudFormation template hardcodes that name.

1.2 — Launch the host CloudFormation stack

In the AWS Console:

1. **CloudFormation** → **Create stack** → **With new resources**
2. **Upload a template file** → pick `infra/refine-server-auto.yaml` from your delivery ZIP
3. Click **Next** and fill in parameters:

Parameter	Value	Notes
StackName	`refine-server`	Anything you like
InstanceType	`t3.medium`	4 GB RAM. `t3.small` (2 GB) works for ≤ 100 resources; `t3.micro` (1 GB) will OOM
DiskSizeGB	`30`	30 is plenty for a year of data
AllowedAppCidr	`1.2.3.4/32`	Your office / VPN egress IP. See "How to find your CIDR" below. Use `0.0.0.0/0` only if you'll put Refine behind an ALB / Cloudflare.
EnableHttps	`false` for now	Flip to `true` once you have a domain + ACM cert
UseExistingBucket	`true`	You created and pre-loaded one in step 1.1
DeliveryBucketName	`my-refine-delivery-bucket`	The bucket from 1.1
KeyPairName	(your EC2 key pair)	For SSH if needed; otherwise leave blank to use SSM Session Manager

> ***How to find your `AllowedAppCidr`*** *— this is the public IP address you'll connect to Refine from. Pick the path that matches where you'll use Refine:*
>
> - ***From a single laptop / home office:*** *open a new browser tab and go to https://whatismyip.com or https://ifconfig.me. Copy the IPv4 address shown (e.g. `203.0.113.42`) and add `/32` to the end → `203.0.113.42/32`. The `/32` means "this exact IP, nothing else."*
> - ***From an office network (multiple people, one egress):*** *ask your IT team for the office's* ***public egress IP range*** *in CIDR notation. It usually looks like `203.0.113.0/24` (256 IPs) or `203.0.113.0/29` (8 IPs). If they're not sure, every person on the network sees the same IP at whatismyip.com — use that with `/32` to start, then expand if needed.*
> - ***From a corporate VPN:*** *find your VPN's egress IP. While connected to the VPN, go to whatismyip.com. That's the address you need. Add `/32`. If your team uses multiple VPN exit nodes, ask IT for the full CIDR range.*
> - ***From AWS (e.g. a bastion host):*** *use the bastion's Elastic IP `/32`.*
> - ***You'll connect from multiple places (home + office + on the road):*** *you can list multiple CIDRs separated by commas in some CF templates, but the simplest path is to put Refine behind an Application Load Balancer with `0.0.0.0/0` inbound on the EC2 SG and lock down the ALB with your SSO. That's a Day-2 upgrade — for the first install, just use one `/32` and adjust later.*
>
> ***Your IP can change.*** *Home ISPs and small offices often get a new public IP every few weeks. If you suddenly can't reach Refine, re-check whatismyip.com and update the security group rule. See the Troubleshooting matrix in Part 6 for the exact AWS CLI command.*

4. Acknowledge IAM creation → **Create stack**.
5. Wait ~5 minutes for `CREATE_COMPLETE`. Then **Outputs** → copy the **Elastic IP**.

1.3 — Connect to the EC2 (SSM or SSH)

The CloudFormation template launched the EC2 and ran a user-data script that installed Docker and extracted your delivery ZIP to `/opt/refine`. **You now need to log into the EC2 to finish the install** (run the setup wizard, which creates `.env`, logs Docker into the registry, pulls the image, and starts the container).

Two scripts, two different jobs — read this once:

Script	When to run	What it does
`setup.sh`	**First time only.** First thing you run after logging in.	Checks prerequisites, copies `.env.example → .env`, logs Docker into the GHCR registry with your bundled token, pulls the image, starts the container.
`refine.sh`	**Day-2 ops, after `setup.sh` finished.**	Status checks, start/stop, view logs, pull updates, take backups. **Don't run this first** — it expects the install to already be done.

Option A — SSM Session Manager (recommended, no SSH key needed)

- **AWS Console:** EC2 → Instances → select `refine-server` → **Connect** → **Session Manager** tab → **Connect**
- **CLI:** `aws ssm start-session --target i-XXXXXXXXXXXXX` (instance ID is in the CloudFormation stack **Outputs** tab, key `InstanceId`)

You'll land in a shell as `ssm-user` in `/`. Continue to step 1.4 below.

Option B — SSH (if you provided a KeyPairName at stack creation)

```
ssh -i ~/.ssh/your-key.pem ec2-user@<your-elastic-ip>
# (Use 'ubuntu@' instead of 'ec2-user@' if the AMI is Ubuntu)
```

You'll land in a shell as `ec2-user` (or `ubuntu`). Continue to step 1.4 below.

1.4 — Run the setup wizard (`setup.sh`)

You're now logged in. Run these commands **in order, exactly as shown**:

```
# 1. Become root (needed for Docker + writing into /opt/refine)
sudo su -

# 2. Move to the install directory
cd /opt/refine

# 3. Confirm the install files are in place
ls
#   You should see at minimum:
#     setup.sh   setup.py   refine.sh   docker-compose.yml
#     .env.example   refine.license   registry-login.sh   scripts/

# 4. Run the install wizard
./setup.sh
```

`setup.sh` runs for ~1–2 minutes (mostly pulling the Docker image). When it finishes you'll see a success banner with the Refine URL. The container is now running.

> ***Troubleshooting `setup.sh`:***
>
> - *`setup.sh: command not found` → run `bash setup.sh` instead (file may have lost +x on extract)*
> - *`Permission denied` → confirm you ran `sudo su -` first*
> - *`Compose file not found` → confirm you're in `/opt/refine` (run `pwd`). Also confirm the file is named `docker-compose.yml` and not `docker-compose.customer.yml` (older deliveries; if so, rename it: `mv docker-compose.customer.yml docker-compose.yml`)*
> - ***Fully-manual fallback** if `setup.sh` keeps erroring:*
>
>   ```
>   cp .env.example .env
>   bash registry-login.sh        # logs Docker into GHCR with bundled token
>   docker compose pull
>   docker compose up -d
>   docker compose ps             # should show "refine" Up (healthy)
>   ```

1.5 — Open Refine in your browser

Back on your laptop, open:

```
http://<your-elastic-ip>:8000
```

You should see the **Welcome to Refine** screen. Click through:

1. **Create your first user** — email + password. This is the root admin.
2. **License check** — Refine reads `/app/refine.license` automatically. Confirm it shows your customer name and expiration date.
3. **Done.** You're at the Dashboard. It's empty — that's expected; we haven't connected any cloud accounts yet.

1.6 — Verify health (and day-2 ops cheat sheet)

Back in the SSM/SSH shell, the `refine.sh` script is your day-2 toolkit:

```
./refine.sh status      # should show refine "Up (healthy)"
./refine.sh health      # prints {"status":"ok",...} from /health
./refine.sh url         # prints the URL to open
./refine.sh version     # prints running Refine version
./refine.sh logs -f     # tail logs live (Ctrl+C to stop)
./refine.sh restart     # restart the container
./refine.sh update      # pull latest image + recreate
./refine.sh backup      # snapshot data dir to backups/
```

If `./refine.sh status` shows nothing running, the container has stopped:

```
tail -80 /var/log/cloud-init-output.log   # first-boot user-data log
./refine.sh start
./refine.sh logs -f
```

If any step in 1.4 / 1.5 / 1.6 still fails, jump to Part 6 — Troubleshooting.

Part 2 — Connect your AWS account

This deploys a CloudFormation stack in **the AWS account you want Refine to monitor**. That can be the same account that hosts Refine, a different account, or both. Repeat for every account you want monitored.

2.1 — Start the Add Account flow

In Refine: **Cloud Accounts** → **Add Account** → **AWS**.

Fill in:

Field	Value
AWS Account ID	The 12-digit ID of the account to monitor
Deployment Region	The region where you'll create the CloudFormation stack. Pick somewhere you have most of your resources (e.g. `us-east-1`)
Account Name	Optional — friendly label like "Production"

Click **Add Account**. Refine generates an **External ID** unique to this account link and presents the **Account Setup** modal.

2.2 — Download and deploy the CloudFormation template

In the Account Setup modal:

1. Pick the same region you chose in 2.1
2. Click **Download CloudFormation Template** — the External ID is baked in; don't edit the file
3. In the AWS Console for the target account: **CloudFormation → Create stack → Upload template**
4. Accept all defaults — the template is parameterless (it reads the External ID from inside)
5. Acknowledge IAM creation → **Create stack**
6. Wait for `CREATE_COMPLETE` (~3 min)

2.3 — Paste the stack outputs back into Refine

Open the **Outputs** tab of the new stack. Copy these four values to the Account Setup modal in Refine:

CF Output	Refine field
`RoleArn`	Role ARN
`CostDataBucketName`	Cost data bucket name
`RefineServiceAccessKeyId`	Service Access Key ID
`RefineServiceSecretAccessKey`	Service Secret Access Key (shown ONCE in the CF Outputs — copy now)

The Stack Name field auto-fills if you used the default.

Click **Save Outputs**, then **Validate**.

If validation returns a green ✓, the account is live. The first nightly Lambda collection happens automatically; or click **Sync Now** to trigger one now.

> ***First-day note:*** *cost data takes up to 24 hours for the first AWS Cost & Usage Report to drop into the bucket. Inventory data (EC2, RDS, S3, etc.) appears within minutes of the first sync. The cost number on the dashboard will be* `$0.00` *until the first CUR drops — that's normal.*

Repeat 2.1–2.3 for every AWS account you want monitored.

Part 3 — Connect your Azure subscription

Azure onboarding is faster than AWS — one Cloud Shell command provisions everything.

3.1 — Open Azure Cloud Shell

Go to https://shell.azure.com in the **tenant that owns the subscription** you want to monitor. Pick **Bash** mode.

Confirm you're in the right subscription:

```
az account show --query '[name, id]' -o tsv
az account set --subscription <your-subscription-id>     # if you need to switch
```

3.2 — Upload the onboarding script

In Cloud Shell, click **Manage files** → **Upload** (the icon in the top toolbar). Pick the file from your delivery ZIP at:

```
infra/azure/refine-onboarding.sh
```

After upload completes, in the shell:

```
chmod +x refine-onboarding.sh
./refine-onboarding.sh
```

Type `y` at the confirmation prompt.

The script takes ~60 seconds. It creates:

- An Entra ID **App Registration** + **Service Principal** (named `refine-readonly-<random>`)
- A 1-year **Client Secret** for that Service Principal
- A **Reader** role assignment at subscription scope
- A **Storage Account** + Blob container (`refine-cost-export`)
- A daily **Cost Management Export** dropping CSV into that container

> *If Cost Management Export creation is skipped (Pay-As-You-Go subscriptions require EA / MCA for Cost Management), don't worry — inventory still works fine. You can wire up the export manually later in Portal → Cost Management → Exports.*

3.3 — Copy the seven values

At the end, the script prints:

```
Azure Subscription ID : ed59e273-…
Azure Tenant ID       : c636ad17-…
Azure Client ID       : a4d93cb1-…
Azure Client Secret   : ABc8Q~xxxx...
Azure Environment     : Public
Storage Account       : refinecost4b0c5c55
Storage Container     : refine-cost-export
```

Copy the Client Secret immediately — Azure never shows it again. If you lose it, just re-run the script; it generates a fresh App Registration each time (old ones can be cleaned up later in Entra ID → App Registrations).

3.4 — Paste into Refine

In Refine: **Cloud Accounts** → **Add Account** → **Azure**. Fill in all five required fields (Subscription ID, Tenant ID, Client ID, Client Secret, Environment) plus the optional Account Name. Click **Add Account**.

Validation runs automatically; you should see a green ✓ within a few seconds.

Click **Sync Now** to pull inventory right away. Inventory lands in seconds; cost data lands after the first daily export drop (≤ 6 hours).

Part 4 — Daily use

The first time inventory lands you'll have data on the following pages:

4.1 — Dashboard (`/`)

Top of the page: **Spend by Cloud** widget showing the AWS / Azure split. Below: nine resource category tiles (Compute, Databases, Storage, Block Storage, Serverless, Containers, Network, Caching, Logs). Each tile shows per-cloud counts (e.g. "AWS 8 · Azure 3"). Click any tile to drill into the unified inventory page for that category.

4.2 — Cost Cleanup (`/cost-cleanup`)

The "low-hanging fruit" page. Items here are 1-click wins — orphaned NAT Gateways, idle Public IPs, gp2 EBS volumes that should be gp3, S3 buckets without lifecycle policies, oversized Azure VMs. Sorted by savings descending. Each row has a Copy CLI button that gives you the exact AWS CLI / Azure CLI / gcloud command to execute the change.

This is the page to check weekly.

4.3 — Recommendations (`/recommendations`)

The fuller, more nuanced list. Rightsizing recommendations based on 1-month, 3-month, and 12-month utilization metrics. Use the period toggle at the top of the page to switch between windows — sizing based on 1-year data is typically the safest.

4.4 — Inventory pages (`/inventory/{category}`)

One page per resource category, unified across clouds. Filter by cloud at the top (All / AWS / Azure). Filter by region, recommendation type, or free-text search. Click **Export CSV** to dump the current filtered view for offline analysis or reporting.

4.5 — Savings Tracking (`/savings-report`)

Once you start acting on recommendations, this page shows actual savings realized vs the baseline (when each resource was first observed). The Refine attribution engine matches "instance X was downsized from m5.xlarge to m5.large on date Y" to a delta in your billed cost.

4.6 — Sharing an account with a teammate

Each account row has a **Share** button. Click it to download a JSON config file containing everything the teammate needs to connect to the same account from their Refine — except secrets.

You'll need to share the secret (AWS Secret Access Key / Azure Client Secret) via a **secure channel** (1Password, encrypted email, in-person). The JSON file alone is safe to email.

The teammate then uses **Cloud Accounts** → **Import Account** in their Refine to paste the JSON + the secret you sent them separately.

Part 5 — Day-2 operations: the `refine.sh` script

Every customer delivery ZIP ships a `refine.sh` script at the top level for common operational tasks. Run it from `/opt/refine`:

Command	What it does
`./refine.sh status`	Show container + image status
`./refine.sh start`	Start the stack
`./refine.sh stop`	Stop the stack
`./refine.sh restart`	Restart
`./refine.sh update`	Pull the latest image and recreate the container
`./refine.sh logs`	Last 200 lines of logs
`./refine.sh logs -f`	Follow logs live (Ctrl+C to stop)
`./refine.sh shell`	Open a shell inside the container
`./refine.sh backup`	Snapshot the data dir to `backups/`
`./refine.sh version`	Print the running Refine version
`./refine.sh health`	Hit `/health` locally
`./refine.sh url`	Print the URL to open in a browser

Auto-updates: the container has a Watchtower sidecar that polls GHCR once per hour for new versions. When one is available, a banner appears in the Refine UI with an **Update Now** button. Clicking it triggers Watchtower to pull and restart immediately — no command line required.

Manual updates: `./refine.sh update` from the shell does the same thing on demand. Equivalent to `docker compose pull && docker compose up -d`.

Backups before risky changes:

```
./refine.sh backup            # before
# … do the thing …
# if you need to roll back:
./refine.sh stop
tar -xzf backups/refine-data-<timestamp>.tar.gz
./refine.sh start
```

The `data/` directory holds your SQLite DB and license file. Backing it up is sufficient to recover the entire installation.

Part 6 — Troubleshooting

Symptom	Diagnosis	Fix
`./refine.sh status` says container is "Restarting" or "Exited"	Most common cause is OOM on `t3.micro`	Bump instance to `t3.small` or `t3.medium`. Stop EC2 → Change instance type → Start.
`./refine.sh logs` shows "license file not found"	The CloudFormation template couldn't download the ZIP, or `refine.license` is missing from it	Verify the ZIP is at `s3://<bucket>/refine-delivery.zip` (exact name), and that the ZIP has a `refine.license` at its root
Login page loads but Add Account → AWS validation fails with "Access Denied"	The Refine service IAM user can't assume the role	Re-check that `RefineServiceAccessKeyId` + `RefineServiceSecretAccessKey` from the CF Outputs are pasted correctly. They're shown only once in CF Outputs — if you missed them, **re-deploy the stack** (or use the CF console to view the new secret in the IAM user's `AccessKey1` if your account allows it)
Add Account → Azure validation fails: "Azure SDK not installed"	You're on a Refine image older than v2.26.2	`./refine.sh update` to pull the latest image
Add Account → Azure validation fails: "AAD authentication failed"	Tenant ID / Client ID / Client Secret mismatch, or you're in the wrong Azure environment	Re-check all three against the Cloud Shell script output. Confirm `Environment = Public` (not USGov / China)
Validation passes for Azure but no resources appear in inventory	First sync hasn't run yet	Click **Sync Now** on the Cloud Accounts page. Wait 30–60 seconds, then refresh inventory page
Dashboard shows `$0.00` total cost on day 1	Normal — AWS Cost & Usage Reports take 24 h to first drop, Azure Cost Management Export takes ≤ 6 h	Wait. Or for AWS, click **Refresh Costs** on the Cost Overview page after the first CUR drops
Browser shows "Connection refused" hitting `http://ip:8000`	Either container is down OR your IP is not in `AllowedAppCidr`	Run `./refine.sh status` to check the container. Then check your current IP at `curl ifconfig.me` and compare to `AllowedAppCidr` in the CloudFormation parameters. Update the SG inbound rule if your IP changed.
Sidebar says "Licensed until …" but license validation fails on login	License file moved or permissions wrong	`ls -la /opt/refine/refine.license` should show it owned by the user running Docker. Re-extract from the delivery ZIP if needed.
You see Azure resources you don't recognize	Cost Management Export may include sub-resources from other deployments	Filter by Account in the inventory page to see only the subscription you expect. The unrecognized rows are likely from a hidden Resource Group; check Azure Portal to confirm
Watchtower auto-update happened but the new version "looks the same"	Browser cache	Hard refresh: **Ctrl+Shift+R** (Win/Linux) or **Cmd+Shift+R** (Mac). Or open in an incognito window

Symptom	Diagnosis	Fix
In-app banner says "Update available" but `./refine.sh update` says "already up to date"	The banner shows a banner for any tag newer than `/app/VERSION`; sometimes a release is pushed but not yet pulled	`./refine.sh restart` to recycle the container so it re-checks
CloudFormation stack rolls back with `"The maximum number of addresses has been reached"`	Your AWS account has hit the default Elastic IP cap (5 per region)	See **"Cannot launch the host stack — Elastic IP limit reached"** below

Cannot launch the host stack — Elastic IP limit reached

If the CloudFormation deploy in Part 1.2 rolls back with:

```
Resource handler returned message: 'The maximum number of addresses has
been reached.' (Service: Ec2, Status Code: 400, ...)
```

...it means your AWS account is at the default cap of **5 Elastic IPs per region**. Try these solutions in order — the first usually fixes it within a minute.

Solution 1 — Release unused EIPs (most common cause)

Orphaned EIPs (allocated but not associated with a running instance) still count against the cap and each cost ~$3.60/month. List them:

```
aws ec2 describe-addresses --region us-east-1 \
  --query 'Addresses[?AssociationId==null].[PublicIp,AllocationId,Tags]' \
  --output table
```

For each EIP you no longer need, release it:

```
aws ec2 release-address --region us-east-1 --allocation-id eipalloc-XXXX
```

Then **retry the CloudFormation deploy.** This resolves the issue ~80% of the time.

Solution 2 — Request a quota increase

If all 5 EIPs are legitimately in use, raise the cap:

1. AWS Console → **Service Quotas** → AWS services → **Amazon EC2**
2. Search for **"EC2-VPC Elastic IPs"**
3. Click **Request quota increase** and ask for `10` (or `20`)
4. Approval is typically automatic and arrives in 5–30 minutes

Then retry the deploy.

Solution 3 — Deploy without an Elastic IP

If you don't need a stable public IP (you'll only access Refine via VPN / SSM / private DNS), edit `infra/refine-server-auto.yaml` in your delivery ZIP and **comment out** the `RefineEIP` and `RefineEIPAssociation` resources plus any `!Ref RefineEIP` references in the Outputs section. Re-upload the edited template and retry the stack.

You'll connect to Refine via the EC2's auto-assigned public IPv4 (which changes on stop/start) or, better, via SSM Session Manager + port forwarding:

```
aws ssm start-session --target i-xxxxxxxx \
  --document-name AWS-StartPortForwardingSession \
  --parameters '{"portNumber":["8000"],"localPortNumber":["8000"]}'
# then open http://localhost:8000
```

Solution 4 — Use a different AWS region

The EIP cap is **per region**. If `us-east-1` is full but `us-west-2` is empty, deploy the Refine host stack in `us-west-2` and use cross-region calls for cost/inventory collection. Latency from a `us-west-2` host to `us-east-1` monitored resources is fine (~70 ms per API call; sync runs nightly).

> ***Tip:*** *Solution 1 is almost always the right answer. Run the* `describe-addresses` *command first — most customers find at least one orphan EIP from an old experiment.*

Getting a fresh log dump for Blacktip support

If you contact us, please include:

```
cd /opt/refine
./refine.sh version
./refine.sh logs > /tmp/refine-logs.txt
sudo docker logs --tail 500 watchtower > /tmp/watchtower-logs.txt 2>&1
tar -czf /tmp/refine-diag.tar.gz /tmp/refine-logs.txt /tmp/watchtower-logs.txt
```

Attach `/tmp/refine-diag.tar.gz` to your email. **No license file, no credentials, no customer data — just diagnostic logs.**

Part 7 — Removing an account

Click **Remove** on a cloud account row. Confirm in the dialog.

As of v2.27, this is a **real hard delete**:

- The account row is physically removed from the database
- Every inventory + cost + recommendation row referencing it is also deleted
- The toast shows the cascade count (e.g. "1,247 associated inventory rows")

The CloudFormation stack / Azure Service Principal in the cloud is **not** touched. To clean those up:

- **AWS:** `aws cloudformation delete-stack --stack-name <name>` in the monitored account
- **Azure:** `az ad app delete --id <client-id>` in the Cloud Shell where you ran the onboarding script

If you change your mind, you can re-add the account by repeating Part 2 or 3.

Part 8 — Need help?

Email **support@blacktipsolutions.com** with:

- Your customer ID (top-left of the sidebar)
- Your Refine version (`./refine.sh version`)
- A short description of what you tried + what happened
- The diagnostic tar from the Troubleshooting section above

Response time: **24 hours for non-blocking, 2 hours for production-down.**

Appendix — Quick reference

URLs you'll use a lot

Page	URL on your server
Dashboard	`http://<ip>:8000/`
Cost Cleanup (start here weekly)	`http://<ip>:8000/cost-cleanup`
Recommendations	`http://<ip>:8000/recommendations`
Compute inventory (EC2 + Azure VMs)	`http://<ip>:8000/inventory/compute`
Databases	`http://<ip>:8000/inventory/databases`
Cloud Accounts (Add / Remove / Share)	`http://<ip>:8000/accounts`
Savings Tracking	`http://<ip>:8000/savings-report`
Usage Trends	`http://<ip>:8000/trends`

Where things live on the EC2

```
/opt/refine/
├── refine.license          ← your license file, do not delete
├── docker-compose.yml      ← stack definition
├── refine.sh               ← ops script
├── data/                   ← SQLite DB + collected data (BACK THIS UP)
│   └── refine.db
├── backups/                ← snapshots from ./refine.sh backup
└── .env                    ← environment overrides (port, etc.)
```

Common AWS CLI snippets

```
# What's my current public IP (for AllowedAppCidr updates)?
curl ifconfig.me

# Update Refine SG to allow a new /32
aws ec2 authorize-security-group-ingress \
  --group-id sg-XXXXXX \
  --protocol tcp --port 8000 --cidr 1.2.3.4/32

# Tail Refine logs via SSM (no SSH key needed)
aws ssm start-session --target i-xxxxxxxx
# then inside: cd /opt/refine && ./refine.sh logs -f
```

Common Azure CLI snippets

```
# What App Registrations do I have for Refine?
az ad app list --display-name "refine-readonly-*" --query "[].{name:displayName,id:appId}"

# Rotate the client secret
APP_ID=$(az ad app list --display-name "refine-readonly-XXXXXXXX" --query "[0].appId" -o tsv)
az ad app credential reset --id $APP_ID --years 1

# Then paste the new password into Refine: Cloud Accounts → Edit → Azure Client Secret
```

That's the whole loop. Most weeks you'll only open the Cost Cleanup page, glance at recommendations, and otherwise leave Refine alone. Reach out if anything in this guide is unclear or stale.
