# Refine System Requirements

Minimum hardware and configuration requirements for running Refine at different scales.

## Quick Reference

| | Local (1-5 accounts) | Starter (1-10 accounts) | Business (10-50 accounts) | Enterprise (50-200 accounts) | Large Enterprise (200+) |
|---|---|---|---|---|---|
| **Where** | Your laptop/desktop | EC2 `t3.small` | EC2 `t3.medium` | EC2 `m6i.xlarge` | EC2 `m6i.2xlarge` |
| **CPU** | Any modern CPU | 1 vCPU | 2 vCPUs | 4 vCPUs | 8+ vCPUs |
| **RAM** | 2 GB (Docker Desktop) | 512 MB | 1-2 GB | 2-4 GB | 4-8 GB |
| **Storage** | 1 GB | 1 GB | 5 GB | 20 GB | 50+ GB |
| **Database** | SQLite (included) | SQLite (included) | SQLite (WAL mode) | PostgreSQL recommended | PostgreSQL required |
| **Dashboard Load** | < 1 second | < 1 second | 1-2 seconds | 2-4 seconds | Requires pagination |
| **Nightly Sync** | < 3 minutes | < 5 minutes | 5-15 minutes | 15-45 minutes | Requires parallelization |
| **Always-on** | No (sleeps with laptop) | Yes | Yes | Yes | Yes |

## Multi-User Overhead

Refine uses a shared server model — all users connect to the same Refine instance. Each additional concurrent user adds minimal overhead (approximately 10-20 MB of RAM per active session). The resource requirements below are sufficient for typical multi-user workloads. No additional scaling is needed for user count alone; the primary scaling factor remains the number of AWS accounts.

## Running Locally (1-5 AWS Accounts)

**For:** Evaluation, demos, small teams, personal use.

Refine runs on any Windows, Mac, or Linux machine with Docker Desktop installed. This is the fastest way to get started — extract the ZIP, run the setup script, and you're live in under 5 minutes.

**Requirements:** Docker Desktop (2 GB RAM allocated) + Python 3.8+

**Limitations:**

- Nightly sync only runs when your machine is on — data goes stale if the laptop sleeps overnight
- Other applications compete for CPU/RAM — close heavy apps during use
- Only accessible at `localhost:8000` (teammates can't access it unless you share your local network)
- Performance degrades above 5 AWS accounts as data volume grows

**When to move to EC2:** When you need always-on monitoring, team access, or connect more than 5 AWS accounts.

See the Deployment Guide for setup instructions.

---

## Tier 1: Starter (1-10 AWS Accounts)

**Typical customer:** Small startup or single-team environment.

### Minimum Requirements

| Resource | Minimum | Recommended |
|---|---|---|
| CPU | 1 vCPU | 2 vCPUs |
| RAM | 512 MB | 1 GB |
| Disk | 1 GB | 2 GB |
| OS | Any Linux (Docker-capable) or Windows with WSL2 | Ubuntu 22.04 LTS |
| Docker | Docker Engine 20.10+ | Docker Engine 24+ |
| Network | Outbound HTTPS to AWS APIs | Same |

### Database

SQLite (default, included) is sufficient. No additional setup required.

### Expected Performance

- **Dashboard load:** < 500ms
- **API response time:** < 200ms average
- **Nightly sync duration:** 2-5 minutes
- **Data volume:** ~600 KB/account/month of inventory data
- **Database size:** 20-30 MB after 12 months

### Suitable Cloud Instances

| Provider | Instance Type | Monthly Cost |
|---|---|---|
| AWS | t3.small (2 vCPU, 2 GB) | ~$15/mo |
| Azure | B1ms (1 vCPU, 2 GB) | ~$15/mo |
| GCP | e2-small (2 vCPU, 2 GB) | ~$14/mo |
| On-premises | Any machine with Docker | $0 (hardware cost only) |

## Tier 2: Business (10-50 AWS Accounts)

**Typical customer:** Mid-size company with separate accounts for production, staging, and development per team.

### Minimum Requirements

| Resource | Minimum | Recommended |
|---|---|---|
| CPU | 2 vCPUs | 4 vCPUs |
| RAM | 1 GB | 2 GB |
| Disk | 5 GB | 10 GB |
| OS | Any Linux (Docker-capable) | Ubuntu 22.04 LTS |
| Docker | Docker Engine 20.10+ | Docker Engine 24+ |
| Network | Outbound HTTPS to AWS APIs | Same |

### Database

SQLite is still workable at this tier but may show occasional slowness during nightly sync when many accounts are updating simultaneously.

**Recommended:** Enable SQLite WAL (Write-Ahead Logging) mode for better concurrent read/write performance. Set this in your `.env`:

```
DATABASE_URL=sqlite:////app/data/refine.db?mode=wal
```

**Optional upgrade:** PostgreSQL provides better concurrent access and query performance at this scale. If you choose PostgreSQL:

```
DATABASE_URL=postgresql://refine:password@localhost:5432/refine
```

### Expected Performance

- **Dashboard load:** 1-2 seconds

- **API response time:** 200-500ms average
- **Nightly sync duration:** 5-15 minutes
- **Data volume:** ~30 MB/month total across all accounts
- **Database size:** 75-150 MB after 12 months
- **Container memory usage:** 400 MB - 1 GB during peak API loads

### Suitable Cloud Instances

| Provider | Instance Type | Monthly Cost |
|---|---|---|
| AWS | t3.medium (2 vCPU, 4 GB) | ~$30/mo |
| Azure | B2s (2 vCPU, 4 GB) | ~$30/mo |
| GCP | e2-medium (2 vCPU, 4 GB) | ~$27/mo |

## Tier 3: Enterprise (50-200 AWS Accounts)

**Typical customer:** Large organization with multi-account strategy (AWS Organizations), multiple business units, and shared services accounts.

### Minimum Requirements

| Resource | Minimum | Recommended |
|---|---|---|
| CPU | 4 vCPUs | 8 vCPUs |
| RAM | 2 GB | 4 GB |
| Disk | 20 GB SSD | 50 GB SSD |
| OS | Linux (Ubuntu 22.04 LTS recommended) | Same |
| Docker | Docker Engine 24+ | Same |
| Database | PostgreSQL 14+ (separate) | PostgreSQL 16 with connection pooling |
| Network | Outbound HTTPS + internal DB access | Same |

### Database

**PostgreSQL is strongly recommended at this tier.** SQLite's single-writer limitation causes sync bottlenecks and API timeouts when more than 50 accounts are actively writing data.

```
# docker-compose.yml addition for PostgreSQL
services:
  postgres:
    image: postgres:16
    environment:
      POSTGRES_DB: refine
      POSTGRES_USER: refine
      POSTGRES_PASSWORD: your-secure-password
    volumes:
      - postgres_data:/var/lib/postgresql/data
    deploy:
      resources:
        limits:
          memory: 1G

  refine:
    image: ghcr.io/eichenbergerb/refine:latest
    environment:
      DATABASE_URL: postgresql://refine:your-secure-password@postgres:5432/refine
    depends_on:
      - postgres
    deploy:
      resources:
        limits:
          memory: 4G
          cpus: "4"
```

### Expected Performance

- **Dashboard load:** 2-4 seconds (large datasets)
- **API response time:** 500ms - 2 seconds for inventory endpoints
- **Nightly sync duration:** 15-45 minutes (sequential account processing)
- **Data volume:** ~100-120 MB/month total
- **Database size:** 200-400 MB after 12 months
- **Container memory usage:** 1-3 GB during peak loads (dashboard + savings recalculation)

### Performance Considerations

At this tier, you may notice:

1. **Dashboard load times increase** — The dashboard loads all 14 service inventories in parallel. With 200 accounts and ~60 EC2 instances each, that's 12,000+ instances loaded into memory per API call.
2. **Nightly sync takes longer** — Each account is synced sequentially. With 200 accounts averaging 10 minutes each, the full sync cycle takes 30-45 minutes.
3. **Savings recalculation is heavier** — The savings engine iterates through every EC2 and RDS instance to calculate rightsizing recommendations. At 12,000+ instances, this takes 1-2 seconds per call.

### Suitable Cloud Instances

| Provider | Instance Type | Monthly Cost |
|---|---|---|
| AWS | m6i.xlarge (4 vCPU, 16 GB) | ~$140/mo |
| Azure | D4s v5 (4 vCPU, 16 GB) | ~$140/mo |
| GCP | e2-standard-4 (4 vCPU, 16 GB) | ~$120/mo |

> ***Note:*** *Memory is the primary scaling constraint, not CPU. Choose instances with more RAM over more cores.*

---

## Tier 4: Large Enterprise (200+ AWS Accounts)

**Typical customer:** Large enterprise with AWS Organizations, hundreds of accounts across multiple regions and business units.

### Minimum Requirements

| Resource | Minimum | Recommended |
|---|---|---|
| CPU | 8 vCPUs | 16 vCPUs (across replicas) |
| RAM | 4 GB | 8 GB per replica |
| Disk | 50 GB SSD | 100 GB SSD |
| OS | Linux (Ubuntu 22.04 LTS) | Same |
| Docker | Docker Engine 24+ or Kubernetes | Kubernetes recommended |
| Database | PostgreSQL 16 with connection pooling | PostgreSQL 16 + pgBouncer |
| Cache | Redis 7+ (optional but recommended) | Redis 7+ |
| Network | Outbound HTTPS + internal DB/cache access | Same |

### Architecture Recommendations

At 200+ accounts, a single Refine container is no longer sufficient. We recommend:

1. **Multiple Refine replicas** behind a load balancer (2-3 containers)
2. **PostgreSQL** as the primary database with connection pooling
3. **Redis** for rate limiting and API response caching
4. **Separate sync worker** — run the nightly sync in a dedicated container so it doesn't compete with API traffic

```
# docker-compose.yml for large-scale deployment
services:
  postgres:
    image: postgres:16
    environment:
      POSTGRES_DB: refine
      POSTGRES_USER: refine
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    deploy:
      resources:
        limits:
          memory: 2G

  redis:
    image: redis:7-alpine
    deploy:
      resources:
        limits:
          memory: 256M

  refine-api:
    image: ghcr.io/eichenbergerb/refine:latest
    environment:
      DATABASE_URL: postgresql://refine:${DB_PASSWORD}@postgres:5432/refine
      RATE_LIMIT_STORAGE_URI: redis://redis:6379
      DISABLE_SCHEDULER: "true"  # API replicas don't run sync
    deploy:
      replicas: 2
      resources:
        limits:
          memory: 4G
          cpus: "4"

  refine-sync:
    image: ghcr.io/eichenbergerb/refine:latest
    environment:
      DATABASE_URL: postgresql://refine:${DB_PASSWORD}@postgres:5432/refine
      SYNC_ONLY: "true"  # Only runs scheduler, no API
    deploy:
      replicas: 1
      resources:
        limits:
          memory: 4G
          cpus: "2"
```

### Expected Performance

- **Dashboard load:** 3-6 seconds (consider using account filtering)
- **API response time:** 1-3 seconds for large inventory queries
- **Nightly sync duration:** 45+ minutes (sequential); can be reduced with parallel sync
- **Data volume:** 200+ MB/month
- **Database size:** 500 MB - 1 GB after 12 months

### Scaling Strategies

1. **Use account filtering** — Instead of loading all 200+ accounts on the dashboard, select specific accounts or account groups. This dramatically reduces payload size and response times.
2. **Stagger nightly syncs** — Configure different sync times for different account groups to spread the load.
3. **Monitor container memory** — Set Docker memory limits and monitor usage. Memory spikes during dashboard loads and savings recalculations are the primary concern.

### Suitable Cloud Instances

| Provider | Instance Type | Monthly Cost (2 replicas) |
|---|---|---|
| AWS | m6i.2xlarge (8 vCPU, 32 GB) x2 | ~$560/mo |
| Azure | D8s v5 (8 vCPU, 32 GB) x2 | ~$560/mo |
| GCP | e2-standard-8 (8 vCPU, 32 GB) x2 | ~$480/mo |
| AWS ECS/Fargate | 4 vCPU, 8 GB x2 | ~$400/mo |

> *Add ~$50-100/mo for managed PostgreSQL (RDS) and ~$15/mo for ElastiCache Redis.*

---

## Storage Sizing Guide

### Disk Space by Scale

| Component | 10 Accounts | 50 Accounts | 200 Accounts |
|---|---|---|---|
| Docker image | 500 MB | 500 MB | 500 MB |
| SQLite/PostgreSQL | 30 MB | 150 MB | 400 MB |
| Inventory JSON (12 months) | 70 MB | 350 MB | 1.4 GB |
| Cost summaries (12 months) | 2 MB | 10 MB | 40 MB |
| Savings report data | 5 MB | 25 MB | 100 MB |
| Application logs | 50 MB | 200 MB | 500 MB |
| **Total** | **~660 MB** | **~1.2 GB** | **~3 GB** |

**Recommendation:** Provision at least 3x the expected total to allow for growth and temporary files.

---

## Network Requirements

### Outbound Access

Refine needs outbound HTTPS (port 443) access to:

| Endpoint | Purpose |
|---|---|
| `sts.amazonaws.com` | Cross-account role assumption |
| `lambda.{region}.amazonaws.com` | Invoke data collection Lambda |
| `s3.{region}.amazonaws.com` | Read inventory data from customer S3 buckets |
| `ghcr.io` | Pull Docker image updates (commercial deployments) |

### Inbound Access

| Port | Purpose |
|---|---|
| 8000 (configurable) | Refine web interface |

For air-gapped/government deployments, see AIR_GAPPED_SETUP_GUIDE.md.

## Monitoring Recommendations

### What to Monitor

| Metric | Warning Threshold | Critical Threshold |
|---|---|---|
| Container memory usage | > 70% of limit | > 90% of limit |
| API response time (p95) | > 2 seconds | > 5 seconds |
| Nightly sync duration | > 30 minutes | > 60 minutes |
| Database size | > 500 MB (SQLite) | > 1 GB (SQLite) |
| Disk usage | > 70% | > 90% |

### Health Check

Refine exposes a health endpoint at `GET /health` that returns `{"status": "ok"}`. Use this for Docker health checks, load balancer probes, or monitoring tools.

## FAQ

**Q: Can I run Refine on a Raspberry Pi?** A: Only for 1-3 accounts. The ARM-based Pi 4 with 4 GB RAM can run the Docker image, but performance will be limited.

**Q: Do I need a GPU?** A: No. Refine is CPU and memory bound, not GPU bound.

**Q: Can I use an existing PostgreSQL instance?** A: Yes. Set `DATABASE_URL` in your `.env` file to point to any PostgreSQL 14+ server. Refine creates its own tables on first startup.

**Q: What happens if the container runs out of memory?** A: Docker will kill the container (OOMKilled). It will auto-restart if `restart: unless-stopped` is set. Consider increasing the memory limit or upgrading to the next tier.

**Q: Can I run multiple Refine instances against the same database?** A: Yes, with PostgreSQL. Do not use SQLite with multiple instances. Ensure only one instance runs the nightly sync scheduler (set `DISABLE_SCHEDULER=true` on API-only replicas).

**Q: How do I migrate from SQLite to PostgreSQL?** A: Export your data with `sqlite3 data/refine.db .dump > backup.sql`, create a PostgreSQL database, update `DATABASE_URL` in `.env`, and restart Refine. Tables are auto-created. Historical data requires manual migration.