# Deployment Guide — From Tests to Production

This guide explains step-by-step how to move Hacienda API from local/tests/staging into production safely and repeatably. It covers pre-release checks, secrets, environment setup, CI/CD, deployment, monitoring, rollback and post-release validation.

**Audience:** Developers, DevOps, Release Engineers

---

## Checklist (high level)

- [ ] All tests pass locally and in CI (unit, integration, e2e)
- [ ] Coverage thresholds met (project requires 80% global)
- [ ] Linting & formatting clean
- [ ] Postman / Newman collections pass (smoke tests)
- [ ] Secrets (certificates, ATV credentials) stored in secret manager
- [ ] Real ATV SDK enabled and tested in staging
- [ ] Backups & storage permissions in place for `INVOICES_DIR`
- [ ] Monitoring, logging and alerting configured
- [ ] Rollback plan documented and tested

---

## 1) Pre-release (local & CI)

1.1 Run tests and coverage locally (PowerShell):

```
cd "c:\Users\Christofer Brenes\Documents\PRACTICA\Hacienda_API"
npm ci
npm test
npm run test:coverage
```

- Fix any failing tests before proceeding.
- Ensure coverage meets the required thresholds reported by `jest`.

1.2 Lint & format checks:

```
npm run lint
npm run format:check
```

1.3 Run integration / smoke tests (Supertest or Newman):

- If you have a Postman collection, run with Newman:

```
npx newman run postman/EnglishInvoices.postman_collection.json -e postman/env-prod.json
```

- Or run Supertest integration suites:

```
npx jest tests/integration --runInBand
```

1.4 Verify environment parity: staging should mimic production (node version, environment vars, storage path, sim vs real ATV). If `SIMULATE_IF_NO_KEYS=true` is set in staging, switch to REAL mode only after certs are in place.

---

## 2) Secrets and Certificates

2.1 Production secrets list (do NOT store in repo):

- `ATV_KEY_PATH` (path to private key / .p12)
- `ATV_CERT_PATH` (path to certificate / .p12)
- `ATV_CLIENT_ID` (client ID)
- `ATV_USERNAME` , `ATV_PIN`
- `DATABASE_URL` or other external service credentials
- `NODE_ENV=production`

2.2 Choose a secrets storage solution:

- GitHub Secrets (for Actions), AWS Secrets Manager, Azure Key Vault, HashiCorp Vault, or environment variables injected by the orchestrator.

2.3 Place certificate files securely on the host or use secret mounts. Example (Linux host):

- Store `/etc/secrets/hacienda/certificate.p12` and set `ATV_CERT_PATH` accordingly.
- Ensure file ownership and permissions (root or app user only):

```
chown root:appuser /etc/secrets/hacienda/certificate.p12
chmod 640 /etc/secrets/hacienda/certificate.p12
```

2.4 Validate cert and SDK in staging before enabling REAL mode.

---

## 3) Prepare Application for Production

3.1 Environment variables (example `.env` for production - but use secrets manager in real deployments):

```
PORT=3000
NODE_ENV=production
SIMULATE_IF_NO_KEYS=false
ATV_KEY_PATH=/etc/secrets/hacienda/private.key
ATV_CERT_PATH=/etc/secrets/hacienda/certificate.p12
ATV_CLIENT_ID=your-client-id
ATV_USERNAME=your-username
ATV_PIN=your-pin
LOG_LEVEL=warn
INVOICES_DIR=/var/lib/hacienda_api/invoices
MAX_FILE_SIZE=10MB
ALLOWED_MIME_TYPES=application/json,application/xml,application/pdf
```

3.2 File system: invoices storage

- Ensure `INVOICES_DIR` exists and is writable by the service user.

```
mkdir -p /var/lib/hacienda_api/invoices
chown appuser:appgroup /var/lib/hacienda_api/invoices
chmod 750 /var/lib/hacienda_api/invoices
```

3.3 Logging

- Ensure `logs/` is rotated (logrotate) or configure centralized logging (CloudWatch, ELK, etc.).

3.4 Enable REAL mode in `atvAdapter.js`

- Install the real SDK dependency in `package.json` if not already present: `npm install @facturacr/atv-sdk` .
- In `src/services/atvAdapter.js` uncomment or implement `_initRealMode()` logic to create the SDK instance using `ATV_KEY_PATH` , `ATV_CERT_PATH` , and credentials. Do this in a secure environment only.

3.5 Security hardening

- Run with least privilege user
- Keep secrets out of logs
- Use TLS for all external communications

---

## 4) CI/CD — Build, Test, Publish

4.1 Example GitHub Actions job (minimal) — `/.github/workflows/ci.yml` :

```
name: CI
on: [push, pull_request]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '18'
      - run: npm ci
      - run: npm test
      - run: npm run test:coverage
  build-and-push:
    runs-on: ubuntu-latest
    needs: test
    if: github.ref == 'refs/heads/main'
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: '18'
      - run: npm ci --production
      - run: docker build -t ghcr.io/${{ github.repository }}:v${{ github.run_number }} .
      - uses: docker/login-action@v2
        with:
          registry: ghcr.io
```

```
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}
      - run: docker push ghcr.io/${{ github.repository }}:v${{ github.run_number }}
```

4.2 Tagging & releases

- Use semantic version tags and create GitHub Releases for production deploys.

4.3 Build artifacts

- Push built Docker images to a registry (DockerHub, GHCR, ECR).

---

## 5) Deploy to Staging

5.1 Deploy the same artifact/image to a staging environment first.

- Use same config as production but with test certs or `SIMULATE_IF_NO_KEYS=true` if you want simulated behavior.

5.2 Run smoke tests (Postman/Newman) against staging

```
npx newman run postman/EnglishInvoices.postman_collection.json -e postman/env-staging.json --bail
```

5.3 Run basic load test (Artillery or ApacheBench):

```
artillery quick --count 20 -n 100 http://staging.example.com/health
```

5.4 Validate logs, storage, and ATV SDK connectivity (if REAL).

---

## 6) Production Release Strategy

6.1 Deployment model options

- Single host (systemd): good for small setups
- Docker container: recommended
- Kubernetes: recommended for scale and rolling updates
- Blue/Green or Canary deployments recommended for minimal risk

6.2 Rolling deploy (example Kubernetes strategy)

- Use `Deployment` with `maxUnavailable: 1` and `maxSurge: 1` to safely rollout.
- Liveness and readiness probes: ensure readiness probe checks `/health` and that `atvAdapter` is initialized when in REAL mode.

6.3 Example systemd unit (simple):

```
[Unit]
Description=Hacienda API
After=network.target

[Service]
```

```
User=appuser
Group=appgroup
WorkingDirectory=/opt/hacienda_api
ExecStart=/usr/bin/node src/server.js
Restart=on-failure
Environment=NODE_ENV=production
Environment=ATV_CERT_PATH=/etc/secrets/hacienda/certificate.p12

[Install]
WantedBy=multi-user.target
```

6.4 Steps to release to production (minimal):

1. Tag the release in Git (e.g., `v1.2.0`) and push
2. CI builds and pushes Docker image
3. Pull image in production orchestrator or deploy using your pipeline
4. Run smoke tests immediately after deploy
5. Monitor logs and metrics for 10-30 minutes
6. If issues, rollback to previous image tag

---

## 7) Post-deploy Validation & Smoke Tests

7.1 Smoke tests after deploy (PowerShell):

```
# Basic health and info
Invoke-RestMethod -Uri "https://api.example.com/health"
Invoke-RestMethod -Uri "https://api.example.com/info"

# Issue a lightweight invoice (simulated or staging data) and validate
$payload = Get-Content -Raw .\docs\COMPLETE_PAYLOADS.md | ConvertFrom-Json # or construct a
small payload
Invoke-RestMethod -Uri "https://api.example.com/api/english-invoices/emit" -Method POST -
Body ($payload | ConvertTo-Json) -ContentType 'application/json'
```

7.2 Verify storage: ensure invoice files are saved and permissions correct. 7.3 Verify logs: no repeated errors in `logs/error.log`. 7.4 Monitor metrics: CPU, memory, disk usage, request latency, error rate.

---

## 8) Monitoring & Alerts

8.1 Important metrics to monitor

- Request rate (RPS)
- Error rate (5xx, 4xx)
- Average latency and p95/p99
- Disk usage of `INVOICES_DIR`
- ATV connectivity failures (timeouts, auth errors)
- Application logs for uncaught exceptions

8.2 Alerts to configure

- Error rate > 1% for 5m

- Average latency > 1s for 5m (adjust per endpoint)
- ATV connectivity failures repeated N times
- Disk usage > 80%

8.3 Logging

- Aggregate logs to a central system (ELK, CloudWatch, Datadog)
- Keep `error.log` and `combined.log` rotated

---

## 9) Backup & Retention

- Back up `./src/data/consecutivo.json` periodically (daily) and before releases.
- Back up `INVOICES_DIR` to object storage (S3, Azure Blob) with lifecycle rules.
- Retain critical logs and invoices per regulatory requirements.

---

## 10) Rollback Plan

10.1 Quick rollback options:

- Re-deploy previous Docker image tag
- If running on a single host, start previous systemd unit with older binary

10.2 Steps to rollback:

1. Pause traffic (if behind load balancer)
2. Deploy previous artifact (image or code)
3. Run smoke tests
4. Verify invoice storage integrity
5. Notify stakeholders

10.3 Post-rollback: open incident, collect logs, perform root cause analysis.

---

## 11) Security & Compliance Notes

- Keep ATV credentials offline and rotate regularly.
- Ensure TLS for all external endpoints.
- Encrypt backups if they contain sensitive data.
- Avoid logging PII or certificate contents.
- Review regulatory retention policies for invoices.

---

## 12) Example Quick-Start Commands (PowerShell)

```
# Build and run locally in production mode
npm ci
$env:NODE_ENV='production'
$env:ATV_CERT_PATH='C:\secrets\hacienda\certificate.p12'
$env:ATV_KEY_PATH='C:\secrets\hacienda\private.key'
$env:ATV_CLIENT_ID='CLIENT_ID'
$env:SIMULATE_IF_NO_KEYS='false'
node src/server.js
```

```
# Run smoke tests (one-liner example)
Invoke-RestMethod -Uri 'http://localhost:3000/health'
```

---

## 13) Troubleshooting

- `ATV adapter fails to init` : check certificate path, permissions and SDK version. Enable debug logs temporarily.
- `Invoices not saved` : check `INVOICES_DIR` path and write permissions.
- `High error rate after deploy` : revert immediately and investigate logs.

---

## 14) Release Sign-off Template

- Tests: ✅ unit ✅ integration ✅ e2e
- Certs: ✅ available in secret manager
- Backups: ✅ verified
- Monitoring: ✅ dashboards + alerts
- Security review: ✅ secrets/permissions checked
- Sign-off: [Name], [Date], [Notes]

---

## Appendix: Useful Links

- `docs/TESTING.md` — test instructions
- `docs/COMPLETE_PAYLOADS.md` — payload examples
- `docs/MANUAL_TECNICO.md` — technical manual

**End of guide**