

Admin/Operations Guide

This guide is for curators, developers, and operations staff.

Runbook – Common Operations

Task 1: Start the Application

Prerequisites: Ollama running, Node.js installed, `.env.local` configured

Full Startup Sequence

1. SSH into server

ssh ubuntu@museum-server.local

2. Navigate to project

cd /opt/harry-clarke/app/harry-clarke-art-assistant/frontend

3. Verify Ollama is running

ollama list

Expected: llama3.2, mxbai-embed-large listed

4. Check if models are available (if not, pull them)

ollama pull llama3.2

ollama pull mxbai-embed-large

5. Start Node.js app (development)

npm run dev

Output: Ready in X.XXs

Local: http://localhost:3000

OR start with PM2 (production)

pm2 start "npm run dev" --name "harry-clarke-bot" --logfile /opt/harry-clarke/logs/pm2.log

6. Verify app is running

curl http://localhost:3000

Expected: 200 response with HTML

7. Test API endpoint

**curl -X POST http://localhost:3000/api **

**-H "Content-Type: application/json" **

-d '{"messages": [{"role": "user", "content": "Hello"}]}'

Expected: { "reply": "...", "usedRag": false }

Troubleshooting:

Issue	Solution
"Port 3000 already in use"	`lsof -i :3000` then `kill -9 <PID>`
"Cannot find module"	Run `npm install` in frontend directory
"Ollama connection refused"	Check `ollama list` or restart Ollama
"Model not found"	Run `ollama pull llama3.2`

Quick Start (If Already Running)

Check status

```
pm2 status
```

Restart

```
pm2 restart harry-clarke-bot
```

View logs

```
pm2 logs harry-clarke-bot --lines 50
```

Task 2: Stop the Application

When: During maintenance, debugging, or graceful shutdown

Graceful stop (development)

Press Ctrl+C in terminal

Or if running with PM2

```
pm2 stop harry-clarke-bot
```

```
pm2 delete harry-clarke-bot
```

Or kill by port

```
lsof -i :3000 | grep LISTEN | awk '{print $2}' | xargs kill -9
```

Task 3: View Application Logs

Purpose: Diagnose issues, track errors, monitor performance

Real-time logs (PM2)

```
pm2 logs harry-clarke-bot --lines 100 --stream
```

Or if running in foreground

```
npm run dev
```

Logs output directly to terminal

Log locations

tail -f /opt/harry-clarke/logs/pm2.log PM2 process logs
tail -f /opt/harry-clarke/logs/next.log Next.js server logs
tail -f /opt/harry-clarke/ollama/ollama.log Ollama LLM logs

Search for errors

grep "ERROR" /opt/harry-clarke/logs/*.log
grep "Connection refused" /opt/harry-clarke/logs/*.log
grep "ECONNREFUSED" /opt/harry-clarke/logs/*.log

Common Error Messages:

Error	Meaning	Fix
`ECONNREFUSED 127.0.0.1:11434`	Ollama not running	Start Ollama: `ollama serve`
`Cannot find model llama3.2`	Model not downloaded	`ollama pull llama3.2`
`PINECONE_API_KEY not set`	Missing env var	Update `.env.local` with API key
`Port 3000 already in use`	App already running	Kill existing process: `lsof -i :3000`
`Module not found`	Dependency missing	`npm install`

Task 4: Upgrade Ollama Model

When: New version available, switching models, performance tuning

1. Stop application

pm2 stop harry-clarke-bot

2. List available models

ollama list

3. Pull new model

ollama pull llama2 *Alternative model*
ollama pull neural-chat *Faster model*
ollama pull mistral *Efficient model*

4. Update .env.local

```
nano /opt/harry-clarke/app/harry-clarke-art-assistant/frontend/.env.local
```

Change: OLLAMA_MODEL=llama2

5. Restart application

```
pm2 start harry-clarke-bot --name "harry-clarke-bot"
```

6. Test

```
curl -X POST http://localhost:3000/api \
```

```
-H "Content-Type: application/json" \
```

```
-d '{"messages": [{"role": "user", "content": "Test"}]}'
```

7. Verify model in logs

```
pm2 logs harry-clarke-bot | grep "llama2"
```

Task 5: Update Dependencies

When: Security patches, feature updates, bug fixes

1. Check for outdated packages

```
cd /opt/harry-clarke/app/harry-clarke-art-assistant/frontend
```

```
npm outdated
```

2. Update specific package

```
npm update @langchain/core
```

```
npm update next
```

3. Or update all (careful!)

```
npm update
```

4. Check for security vulnerabilities

```
npm audit
```

5. Fix vulnerabilities automatically

```
npm audit fix
```

6. Rebuild and test

```
npm run build
```

```
npm run dev
```

7. Commit changes (if in git repo)

```
git add package.json package-lock.json
```

```
git commit -m "Update dependencies - security patches"
```

Task 6: Restart Ollama

When: Service hangs, high memory, model unresponsive

Check Ollama status (Mac)

```
ps aux | grep ollama
```

Kill Ollama process

```
pkill -f ollama
```

Or restart service (Linux)

```
systemctl restart ollama
```

Start Ollama server

```
ollama serve
```

Verify models loaded

```
ollama list
```

Test inference

```
curl -X POST http://localhost:11434/api/generate \
-d '{"model": "llama3.2", "prompt": "Hello", "stream": false}'
```

Ollama Performance Tuning:

- Set concurrent request limit
 - `export OLLAMA_NUM_PARALLEL=2`
- Limit GPU usage
 - `export CUDA_VISIBLE_DEVICES=0`
- Increase timeout for slow models
 - `export REQUEST_TIMEOUT=300`
- Then start
 - `ollama serve`

Task 7: Enable RAG (Pinecone Integration)

Prerequisites: Pinecone account, API key, indexed documents

1. Create Pinecone index (via Pinecone UI)

- Index name: harry-clarke-docs
- Dimension: 1024 (for mxbai-embed-large)
- Metric: cosine
- Pod type: starter (free)

2. Get API key from Pinecone dashboard

3. Update .env.local

```
cat > /opt/harry-clarke/app/harry-clarke-art-assistant/frontend/.env.local << 'EOF'
OLLAMA_BASE_URL=http://127.0.0.1:11434
OLLAMA_MODEL=llama3.2
OLLAMA_EMBEDDING_MODEL=mxbai-embed-large
OLLAMA_TEMPERATURE=0.2
ENABLE_RAG=true
RAG_TOP_K=4
PINECONE_API_KEY=your_api_key_here
PINECONE_INDEX=harry-clarke-docs
PINECONE_NAMESPACE=prod
EOF
```

4. Pull embedding model

```
ollama pull mxbai-embed-large
```

5. Restart application

```
pm2 restart harry-clarke-bot
```

6. Verify RAG working

```
curl -X POST http://localhost:3000/api \
-H "Content-Type: application/json" \
-d '{"messages": [{"role": "user", "content": "Tell me about Harry Clarke"}]}'
```

Response should include "usedRag": true

7. Check logs for Pinecone connection

```
pm2 logs harry-clarke-bot | grep -i pinecone
```

Disable RAG (*if Pinecone down*):

- **Quick disable**

- `sed -i 's/ENABLE_RAG=true/ENABLE_RAG=false/' \`
`/opt/harry-clarke/app/harry-clarke-art-assistant/frontend/.env.local`

- **Restart**

- `pm2 restart harry-clarke-bot`

App will continue without RAG (fallback mode)

Task 8: Clear Cache & Restart Fresh

When: Strange behavior, memory leaks, stale data

1. Stop application

```
pm2 stop harry-clarke-bot
```

2. Clear Next.js cache

```
rm -rf /opt/harry-clarke/app/harry-clarke-art-assistant/frontend/.next
```

3. Clear npm cache (optional)

```
npm cache clean --force
```

4. Restart

```
pm2 start harry-clarke-bot --name "harry-clarke-bot"
```

5. Rebuild

```
cd /opt/harry-clarke/app/harry-clarke-art-assistant/frontend  
npm run build
```

6. Verify

```
curl http://localhost:3000
```

On-Call Procedures:

Alert Types & Severity-

CRITICAL

- App not responding (HTTP 502/503)
- Ollama crashes repeatedly
- Data loss detected
- Security breach suspected

Action: *Immediate response required*

1. Acknowledge alert (send message to team)

2. Check system status

```
systemctl status ollama
```

```
pm2 status
```

3. Check logs for errors

```
pm2 logs harry-clarke-bot --lines 100
```

4. Restart services if needed

```
pm2 restart harry-clarke-bot
```

```
systemctl restart ollama
```

5. Verify app is back

```
curl http://localhost:3000
```

6. Document incident

Create GitHub issue with: timestamp, error, action taken, resolution time

HIGH

- Response times > 30 seconds
- 50%+ requests failing
- Memory usage > 80%
- Pinecone API errors

Action: *Respond within 30 minutes*

1. Investigate

`top -b -n 1 | head -20` (Check CPU/memory)
`netstat -s | grep -i dropped` (Check packet loss)

2. Check specific service

`pm2 logs harry-clarke-bot | grep ERROR`

3. Restart if needed

`pm2 restart harry-clarke-bot`

4. Monitor for 15 minutes

`watch -n 5 'pm2 status'`

MEDIUM

- High memory usage (60-80%)
- Slow responses (10-30 seconds)
- Occasional Ollama timeouts
- Minor API errors (< 10% failure rate)

Action: *Respond within 2 hours*

LOW

- Non-critical warnings in logs
- Documentation updates needed
- Minor performance optimizations
- Feature requests

Action: *Normal business hours, can defer*

On-Call Checklist – First Response

- ☐ 1. Acknowledge alert to team
- ☐ 2. Check app status: `curl http://localhost:3000`
- ☐ 3. Check Ollama: `ollama list`
- ☐ 4. Check PM2: `pm2 status`
- ☐ 5. View recent logs: `pm2 logs harry-clarke-bot`
- ☐ 6. Check system resources: `top`
- ☐ 7. Check disk space: `df -h`
- ☐ 8. Document findings in incident log
- ☐ 9. Decide: Fix or Escalate?
- ☐ 10. Update team with status

Common On-Call Scenarios

Scenario 1: App Returns 502 (Bad Gateway)

Step 1: Verify Next.js server is running

`pm2 status`

If status shows "stopped" or "errored"...

Step 2: Check error in logs

pm2 logs harry-clarke-bot --lines 50

Step 3: Most likely causes:

- Ollama not responding
- Environment variables missing
- Node.js crash

Step 4: Verify Ollama

curl http://127.0.0.1:11434/api/tags

If connection refused: **systemctl restart ollama**

Step 5: Restart app

pm2 restart harry-clarke-bot

Step 6: Verify fixed

sleep 5

curl http://localhost:3000

Should return 200

Scenario 2: Slow Responses (> 30 seconds)

Step 1: Check system load

top

Look for high CPU or memory usage

Step 2: Check Ollama performance

ps aux | grep ollama

Is Ollama using > 90% CPU?

Step 3: Check if Pinecone is slow

grep "Pinecone" /opt/harry-clarke/logs/*.log | tail -20

Is retrieval taking > 10 seconds?

Step 4: Reduce load

- Disable RAG temporarily: **ENABLE_RAG=false**
- Kill other processes using CPU/memory
- Reduce Ollama parallelization: **export OLLAMA_NUM_PARALLEL=1**

Step 5: Restart Ollama

systemctl restart ollama

Step 6: Monitor

Send test request and time it

time curl -X POST http://localhost:3000/api \

-H "Content-Type: application/json" \

-d '{"messages": [{"role": "user", "content": "hi"}]}'

Scenario 3: Ollama Crashes Repeatedly

Step 1: Check error logs

journalctl -u ollama -n 50 --no-pager

Look for segmentation faults, OOM errors

Step 2: Check memory available

free -h

If < 2GB free: system is out of memory

Step 3: Clear cache and restart

ollama stop

```
rm -rf ~/.ollama/models/manifests/cache
```

 Clear manifest cache

ollama serve

Step 4: If still crashing, try different model

Update .env.local to use smaller model

```
sed -i 's/OLLAMA_MODEL=llama3.2/OLLAMA_MODEL=neural-chat/' \
/opt/harry-clarke/app/harry-clarke-art-assistant/frontend/.env.local
```

```
pm2 restart harry-clarke-bot
```

Step 5: If crashes persist

Create incident: <https://github.com/.../issues>

Tags: [critical] [ollama-crash]

Scenario 4: Pinecone Connection Error

Step 1: Verify PINECONE_API_KEY is set

```
grep PINECONE_API_KEY
/opt/harry-clarke/app/harry-clarke-art-assistant/frontend/.env.local
```

Step 2: Check if Pinecone service is up

```
curl -H "Api-Key: $PINECONE_API_KEY" \
https://your_index.svc.pinecone.io/describe_index_stats
```

Step 3: If 401 (unauthorized), API key is wrong/expired

- Get new key from Pinecone dashboard
- Update .env.local
- Restart app

Step 4: If 503 (service unavailable), Pinecone is down

Disable RAG temporarily:

```
sed -i 's/ENABLE_RAG=true/ENABLE_RAG=false/' .env.local
```

```
pm2 restart harry-clarke-bot
```

Step 5: Notify team via Slack

"Pinecone down - running in fallback mode"

Step 6: Monitor Pinecone status page

<https://status.pinecone.io>

Backup & Restore

What to Backup:

Item	Location	Size	Frequency	Retention
Code	Git repo	Small	Per commit	Forever
Secrets (.env.local)	Vault	Tiny	Manual	Keep current + 1 old
Ollama Models	~/.ollama/models	~8-15GB	After model update	Keep current
Pinecone Data	Cloud (Pinecone)	Managed by Pinecone	Continuous	30 days
Application Logs	/opt/harry-clarke/ logs	~100MB/ month	Daily	90 days

Backup Strategy

Daily Backup (Automated)

- Git commits
- Application logs
- .env.local (encrypted)

Time: Once a day.

Weekly Backup (Manual)

- Ollama models
- Full /opt/harry-clarke/

Time: Once a week.

Monthly Backup (Archive)

- S3/Cloud storage

Time: 1st of month or Once a Month.

Backup Application Code

Frequency: After each deployment, daily

Manual backup (before major changes)

```
cd /opt/harry-clarke/app/harry-clarke-art-assistant/frontend
```

1. Commit to git

```
git add .
```

```
git commit -m "Pre-backup commit - $(date +%Y-%m-%d_%H:%M:%S)"
```

2. Push to remote

```
git push origin main
```

3. Tag for recovery

```
git tag -a backup-$(date +%Y%m%d-%H%M%S) -m "Backup before deployment"
```

```
git push origin --tags
```

Full System Backup

Frequency: Weekly, keep 4 weeks of backups

1. Create full backup

```
BACKUP_DATE=$(date +%Y%m%d-%H%M%S)
```

```
BACKUP_DIR="/opt/harry-clarke/backups/full_backup_${BACKUP_DATE}"
```

```
mkdir -p $BACKUP_DIR
```

2. Backup code

```
cp -r /opt/harry-clarke/app $BACKUP_DIR/
```

3. Backup configs

```
cp /opt/harry-clarke/app/harry-clarke-art-assistant/frontend/.env.local.gpg
```

```
$BACKUP_DIR/
```

4. Backup logs

```
tar -czf $BACKUP_DIR/logs_$(date +%Y%m%d).tar.gz /opt/harry-clarke/logs/
```

5. Backup Ollama models (if space available)

```
tar -czf $BACKUP_DIR/ollama_models_$(date +%Y%m%d).tar.gz
```

```
~/.ollama/models/
```

6. Create backup manifest

```
cat > $BACKUP_DIR/MANIFEST.txt << EOF
```

```
Backup Date: $BACKUP_DATE
```

```
Server: $(hostname)
```

```
System: $(uname -a)
```

```
App Version: $(cd /opt/harry-clarke && git describe --tags)
```

```
Node Version: $(node --version)
```

```
Ollama Version: $(ollama --version)
```

Backup Contents:

- app/ (source code)
 - .env.local.gpg (encrypted secrets)
 - logs/ (compressed)
 - ollama_models/ (if applicable)
- EOF

7. Upload to cloud

```
aws s3 sync $BACKUP_DIR/ s3://your-bucket/full-backups/$BACKUP_DATE/
```

8. Log backup

```
echo "Full backup completed: $BACKUP_DATE - Size: $(du -sh  
$BACKUP_DIR)" \
```

```
>> /opt/harry-clarke/logs/backup.log
```

9. Cleanup old backups (keep 4 weeks)

```
find /opt/harry-clarke/backups/full_backup_* -type d -mtime +28 -exec rm -rf {} \;
```

Verify backup exists in cloud

```
aws s3 ls s3://your-bucket/full-backups/$BACKUP_DATE/
```