Magento 2 Container Stack – Full Project Guide

1. Executive Summary

This repository provisions a fully containerized Magento 2.4.8-p2 lab that mirrors a production-ready reference implementation. It orchestrates PHP-FPM 8.3, NGINX (edge and origin tiers), Varnish 7, Redis 7, MySQL 8.0, and Elasticsearch 8.15 into a cohesive stack with HTTPS termination, Redis-backed cache/session storage, and sample-data-enabled Magento. The solution is designed for repeatable local builds, reliable onboarding, and inspection of a modern Magento architecture.

2. Architecture Overview

2.1 High-Level Flow

```
Browser --HTTPS--> nginx-ssl (edge TLS proxy)
  |__ varnish (6081) --HTTP--> nginx-backend (origin)
        |__ php-fpm (FastCGI 9000) --> Magento application
             |__ mysql (3306)        --> transactional data
             |__ redis (6379)        --> cache db0 + sessions db1/2
             |__ elasticsearch (9200) --> catalog search
```

phpMyAdmin traffic follows the same TLS proxy but is reverse-proxied directly to `nginx-backend:8081` instead of Varnish.

2.2 Component Responsibilities

Service	Image Build Context	Purpose
`nginx-ssl`	`nginx-ssl/`	TLS termination, host-based routing (`test.mgt.com` to Varnish, `pma.mgt.com` to phpMyAdmin).
`varnish`	`varnish/`	Full page cache accelerator, fronting Magento storefront responses.

Service	Image Build Context	Purpose
`nginx-backend`	`nginx-backend/`	Origin NGINX serving static assets and proxying PHP requests to FPM.
`php-fpm`	`php-fpm/`	Runs PHP 8.3, Magento application, cron, Composer, and sample data.
`mysql`	`mysql/`	MySQL 8.0 server with initialization scripts for Magento schema.
`redis`	`redis/`	Shared cache store for Magento cache frontend (db0) and sessions (db1/db2).
`elasticsearch`	`elasticsearch/`	Elasticsearch 8.15 search engine with security disabled for lab use.

3. File & Directory Reference

- `docker-compose.yml` – Service topology, volumes, health checks, port bindings, dependency graph.
- `nginx-ssl/`
  - `Dockerfile` – Installs nginx, creates `test-ssh:clp` runtime, drops privileges.
  - `default.conf` – HTTPS redirect, upstream maps, proxy buffer tuning, SAN certificate

references.
  - `certs/selfsigned.*` – Regenerated SAN certificate covering `test.mgt.com` and `pma.mgt.com`.
- `varnish/`
  - `Dockerfile` – Installs varnish 7, copies generated VCL.
  - `default.vcl` – Magento-generated VCL targeting `nginx-backend:8080` with health probes.
- `nginx-backend/`
  - `Dockerfile` – Installs nginx, creates app user, installs curl for health checks.
  - `magento.conf` – Origin server block for Magento (`8080`, FastCGI, static/media handlers).
  - `phpmyadmin.conf` – Server block for phpMyAdmin on `8081`.
- `php-fpm/`
  - `Dockerfile` – PHP 8.3 stack, Composer, phpMyAdmin files, `test-ssh` user, cron registration.
  - `php.ini` – Magento-friendly PHP overrides (memory, opcache, redis sessions).
  - `crontab-magento` – Magento cron definitions running as `test-ssh`.
  - `test-ssh.conf` – PHP-FPM pool for `test-ssh:clp` (port 9000 listener).
  - `entrypoint.sh` – Ensures `pub/health-check.php` exists, enforces ownership, starts cron.
- `mysql/`
  - `Dockerfile` – Installs MySQL 8.0, ensures data dirs, wires init scripts.
  - `docker-entrypoint.sh` – Initializes data dir, runs MySQL bootstrap, executes init SQL.
  - `init-db.sh` – Creates Magento schema, grants user permissions.
- `redis/`
  - `Dockerfile` – Installs redis-server.
  - `redis.conf` – Binds to all interfaces, enables append-only persistence, DB layout.
- `elasticsearch/`
  - `Dockerfile` – Installs Elasticsearch 8.15 with APT repo, adjusts ownership.
  - `elasticsearch.yml` – Single-node cluster config with xpack security disabled.
- `docs/full-guide.md` – (this document) comprehensive project guide.
- `README.md` – Task-focused entry point for quick start and maintenance.

4. Prerequisites

- Host OS: Linux/macOS/WSL2 with Docker Engine >= 24, docker-compose plugin >= 2.0.
- Hardware: >= 4 vCPU, >= 8 GB RAM, >= 30 GB free disk.
- Optional but recommended: `make`, `curl`, `jq` for workflow scripts.
- DNS/hosts: map `127.0.0.1 test.mgt.com pma.mgt.com` in `/etc/hosts` (see instructions below).

5. Bootstrapping Workflow

5.1 One-Time Setup

1. Clone repository: `git clone <repo> && cd gertest`.
2. Generate SAN certificate (only needed once or when expiring):

```
docker run --rm -v $(pwd)/nginx-ssl/certs:/certs \
  -v $(pwd)/gencert.sh:/gencert.sh alpine sh /gencert.sh
```

3. Add hosts to `/etc/hosts` with sudo privileges:

```
printf '127.0.0.1 test.mgt.com\n127.0.0.1 pma.mgt.com\n' | sudo tee -a /etc/hosts
```

5.2 Build Images

```
docker compose build
```

This builds custom images for MySQL, Redis, Elasticsearch, PHP-FPM, Varnish, and both NGINX tiers.

5.3 Launch Stack

```
docker compose up -d
```

Health checks will delay dependent services until backing services are healthy.

5.4 Install Magento Sample Data (already baked)

The provided repository already contains Magento + sample data modules copied into the shared volume. If rebuilding from scratch, inside the `php-fpm` container:

```
docker compose exec php-fpm bash -lc 'cd /var/www/magento && \
  php bin/magento module:enable Magento_SampleData Magento_CatalogSampleData ... && \
  php bin/magento setup:upgrade && \
  php bin/magento setup:di:compile && \
  php bin/magento setup:static-content:deploy -f en_US'
```

5.5 Verify

1. `docker compose ps` – ensure all services are `Up`, `healthy` where applicable.
2. `docker compose exec php-fpm php -v` – confirm PHP 8.3.
3. `docker compose exec elasticsearch curl -s http://localhost:9200/_cluster/health?pretty` – expect `"status" : "green"`.
4. HTTPS endpoints:

- Storefront: `https://test.mgt.com/` (accept self-signed cert). `curl -kI ...` should return 200.
- phpMyAdmin: `https://pma.mgt.com/` – 200 OK with phpMyAdmin headers.

5. Redis: `docker compose exec redis redis-cli info keyspace` – view DB0/1 usage.
6. Magento CLI: verify base URLs, Elasticsearch engine, FPC config:

```
docker compose exec php-fpm bash -lc 'cd /var/www/magento && \
  php bin/magento config:show web/secure/base_url && \
  php bin/magento config:show catalog/search/engine && \
  php bin/magento config:show system/full_page_cache/caching_application'
```

6. Operational Runbook

6.1 Common Commands

Task	Command
Start stack	`docker compose up -d`
Stop stack	`docker compose down`
Tail nginx logs	`docker compose logs -f nginx-backend`
Execute Magento CLI	`docker compose exec php-fpm bash -lc 'cd /var/www/magento && php bin/magento ...'`
Rebuild static content	`... setup:static-content:deploy -f en_US`
Reindex	`... php bin/magento indexer:reindex`
Cache flush	`... php bin/magento cache:flush`

6.2 Cron Jobs

- Defined in `/etc/cron.d/magento`, run as `test-ssh` every 5 minutes.
- Logs: `/var/www/magento/var/log/magento.cron.log`, `setup.cron.log`, `update.cron.log`.

6.3 Backups

1. Stop stack: `docker compose down`.
2. Archive volumes:

```
for vol in magento-app phpmyadmin-app mysql-data redis-data elasticsearch-data varnish-cac
  docker run --rm -v gertest_${vol}:/volume -v $(pwd):/backup \
    debian:12.2 tar -czf /backup/${vol}.tar.gz -C /volume .
done
```

3. Save images (optional): `docker image ls | awk '/m2-/{print $1":"$2}' | xargs -n1 docker save -o`.

7. Magento Application Configuration

Key settings enforced via CLI/database: - Base URLs (secure & unsecure): `https://test.mgt.com/`. - Elasticsearch engine: `elasticsearch8`, host `elasticsearch`, port `9200`. - Redis session storage: DB1, cache storage: DB0, page cache: Redis DB2 (`env.php`). - Varnish host: `varnish:6081` (`http_cache_hosts` in `env.php`). - Cron & queue consumers disabled by default (manual start). Run queue consumer if needed: `bash docker compose exec php-fpm bash -lc 'cd /var/www/magento && php bin/magento queue:consumers:start <consumer-name>'`

8. Troubleshooting

Symptom	Diagnosis	Fix
`503` from TLS proxy	Check `docker compose ps`; confirm nginx-backend healthy, varnish logs.	Restart backend chain: `docker compose restart php-fpm nginx-backend varnish nginx-ssl`.
Elasticsearch `yellow` status	Replicas >0 on single node.	Run `PUT /_all/_settings {"index":{"number_of_replica`
Redis connection refused	Ensure redis container is running; check firewall; view logs via `docker compose logs redis`.	Restart redis.
Cron logs contain `www-data not found`	Cron user misconfigured.	Ensure `/etc/cron.d/magento` uses `test-ssh`, rebuild PHP image.
Magento CLI complains DOM extension missing	Wrong PHP version selected.	Confirm `/usr/bin/php` symlinks to PHP 8.3 (`update-alternatives --set php /usr/bin/php8.3`).

Symptom	Diagnosis	Fix
Varnish returns `503 Backend fetch failed`	Backend unreachable or `health-check.php` missing.	Regenerate `pub/health-check.php` (entrypoint auto-creates) and ensure php-fpm is running.

9. Extending the Stack

- **TLS Certificates**: Replace `nginx-ssl/certs` with CA-issued cert/key. Update `default.conf` if paths change.
- **Scaling**: Add more PHP-FPM replicas by scaling the service (`docker compose up -d --scale php-fpm=2`) and adjust NGINX upstream definitions.
- **Environment Overrides**: Use `.env` to change exposed ports, credentials, or resource limits. For production, consider secrets management and network hardening.
- **Monitoring**: Add services such as Prometheus exporters, ELK, or integrate `docker compose` with external observability stacks.

10. Appendix

10.1 Useful CURL Checks

```
# Storefront with host header
docker run --rm curlimages/curl -kI https://host.docker.internal/ -H "Host: test.mgt.com"

# phpMyAdmin
docker run --rm curlimages/curl -kI https://host.docker.internal/ -H "Host: pma.mgt.com"

# Redis keyspace
docker compose exec redis redis-cli info keyspace
```

10.2 Default Credentials & Ports

Component	Host Alias	Port Mapping	Notes
Magento Admin	`test.mgt.com`	`443 -> 443`	Admin URI from `php bin/magento info:adminuri`.

Component	Host Alias	Port Mapping	Notes
phpMyAdmin	`pma.mgt.com`	`443 -> 443`	Uses same TLS proxy; credentials managed via MySQL users.
MySQL	`localhost`	`3306:3306`	Root password `magento`, app user `magento/magento`.
Redis	`localhost`	`6379:6379`	No auth; limited to Docker network.
Elasticsearch	`localhost`	`9200:9200`	Security disabled; lab only.

10.3 Data Persistence

- Magento code and phpMyAdmin assets share the `magento-app`/`phpmyadmin-app` named volumes.
- MySQL, Redis, Elasticsearch, Varnish each mount dedicated named volumes for durability across restarts.

Document History - 2025-10-02: Initial full guide authored for project hand-off (Codex).