Multi-Tenant SaaS Architecture Deep Dive for QNAP NAS Deployment

1. Overview

This document provides a step-by-step deep dive on deploying the Etsy Seller Automater as a multi-tenant SaaS platform on your QNAP TBS-464 NAS using Docker. This architecture aims to be cost-effective while still scalable, with an emphasis on local development and potential for future cloud migration.

2. QNAP NAS Capabilities

Your QNAP TBS-464 comes with: - Intel Celeron N5105/N5095 (quad-core) CPU - 8 GB DDR4 RAM - M.2 NVMe SSD bays for high-speed storage - 2.5GbE network interface These specs are sufficient for running Docker-based microservices, a reverse proxy, and a lightweight database stack for local SaaS testing and limited production use.

3. Recommended Architecture Components

- Reverse Proxy: Traefik or Nginx to handle routing, SSL, and multi-tenant domain mapping. - Application Services: Containerized microservices (Python/FastAPI or Node.js) for Etsy automation logic. - Database: PostgreSQL with schema-based multi-tenancy or separate databases per tenant. - Message Queue: Redis or RabbitMQ for background job processing. - Authentication: Keycloak or Auth0 (self-hosted Keycloak recommended for local). - Storage: MinIO (S3-compatible) running in a container for file storage.

4. Local Development on QNAP

- Install Container Station on QNAP to manage Docker containers. - Create a `docker-compose.yml` defining all services. - Use Docker networks to isolate services. - Configure Traefik for dynamic routing to `tenant1.localhost`, `tenant2.localhost`, etc. - Enable PostgreSQL with either `public` schema for shared tenants or `schema per tenant` approach.

5. Example Docker Compose Setup

version: '3.9' services: reverse-proxy: image: traefik:v2.10 command: - --providers.docker=true - --entrypoints.web.address=:80 ports: - "80:80" - "443:443" volumes: - /var/run/docker.sock:/var/run/docker.sock app-service: build: ./app environment: - DATABASE_URL=postgres://user:pass@db:5432/appdb depends_on: - db db: image: postgres:15 environment: - POSTGRES_USER=user - POSTGRES_PASSWORD=pass - POSTGRES_DB=appdb volumes: - db_data:/var/lib/postgresql/data volumes: db_data:

6. Data Model Considerations

Option 1: Single database, tenant_id column in every table. Option 2: Single database, schema per tenant. Option 3: Separate databases per tenant. For local QNAP deployment, Option 2 (schema per tenant) balances isolation and simplicity.

7. Cost Optimization Strategies

- Reuse QNAP hardware to avoid cloud costs for initial deployment. - Use open-source components (PostgreSQL, Traefik, MinIO, Keycloak). - Only spin up containers when needed (development mode). - Store logs locally with rotation to avoid excessive disk usage.

8. Future Cloud Migration Path

The Docker Compose architecture can be easily adapted to Kubernetes (K3s for lightweight clusters) if scaling is needed. You could start on QNAP, test locally, and then deploy the same containers to AWS, GCP, or Azure.