

Nexgate — New Service & Repository Onboarding Guide

Author: Josh Sakweli — DevOps Lead **Version:** 1.0

Last Updated: April 2026

Classification: Internal — DevOps Lead Only

Overview

This guide explains exactly what to do when adding a new service or repository to the Nexgate platform. Every new service follows the same pattern — this document is the checklist to follow every time.

There are three types of services we add:

Type	Example	Needs CI/CD?	Needs Vault?	Needs Compose?
Backend microservice	notification-server, payment-service	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes	<input type="checkbox"/> Yes
Frontend app	nexgate-web, nexgate-admin	<input type="checkbox"/> Yes	<input type="checkbox"/> No	<input type="checkbox"/> Yes
Mobile/Android app	nexgate-check-in-app	<input type="checkbox"/> Yes	<input type="checkbox"/> No	<input type="checkbox"/> No

Table of Contents

- [1. Adding a New Backend Microservice](#)
 - [2. Adding a New Frontend Service](#)
 - [3. Adding a New Android App](#)
 - [4. Checklist Summary](#)
-

1. Adding a New Backend Microservice

Example: Adding nexgate-payment-service

A backend microservice is a Spring Boot application that:

- Has its own database
- Communicates with other services via RabbitMQ
- Connects to Vault for secrets
- Gets deployed to the VPS via Jenkins

Step 1 — Create GitHub Repo

GitHub → nexgate-hq org → New repository

Name: nexgate-payment-service

Visibility: Private

Initialize: Add README

Step 2 — Setup Spring Boot Project

In IntelliJ, create the project with these dependencies:

```
<!-- pom.xml -->
<groupId>org.qbithub</groupId>
<artifactId>nexgate-payment-service</artifactId>
<version>1.0.0</version>

<dependencies>
  <!-- Spring Boot starters -->
  <dependency>spring-boot-starter-web</dependency>
  <dependency>spring-boot-starter-data-jpa</dependency>

  <!-- Vault integration (required!) -->
  <dependency>spring-cloud-starter-vault-config</dependency>
  <dependency>spring-cloud-starter-bootstrap</dependency>

  <!-- RabbitMQ (if needed) -->
  <dependency>spring-boot-starter-amqp</dependency>

  <!-- Database -->
```

```
<dependency>postgresql</dependency>
</dependencies>
```

Step 3 — Create bootstrap.properties

Create `src/main/resources/bootstrap.properties` and add to `.gitignore`:

```
# bootstrap.properties – NEVER COMMIT!
spring.application.name=nexgate-payment-service

spring.cloud.vault.enabled=true
spring.cloud.vault.host=vault.qbitspark.com
spring.cloud.vault.port=443
spring.cloud.vault.scheme=https
spring.cloud.vault.authentication=TOKEN
spring.cloud.vault.token=YOUR_LOCAL_VAULT_TOKEN

spring.cloud.vault.kv.enabled=true
spring.cloud.vault.kv.backend=secret
spring.cloud.vault.kv.default-context=nexgate/payment_service/payment_local

spring.cloud.vault.fail-fast=true
spring.cloud.vault.connection-timeout=5000
spring.cloud.vault.read-timeout=15000
```

Add to `.gitignore`:

```
src/main/resources/bootstrap.properties
```

Step 4 — Create application.properties

```
# application.properties – SAFE TO COMMIT (no secrets!)
spring.application.name=nexgate-payment-service
server.port=8767

# Database (values come from Vault)
spring.datasource.url=${database.url}
spring.datasource.username=${database.username}
spring.datasource.password=${database.password}
```

```
# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

# RabbitMQ (values come from Vault)
spring.rabbitmq.host=${rabbitmq.host}
spring.rabbitmq.port=${rabbitmq.port}
spring.rabbitmq.username=${rabbitmq.username}
spring.rabbitmq.password=${rabbitmq.password}
spring.rabbitmq.virtual-host=${rabbitmq.virtual-host}
```

Step 5 — Create Dockerfile

Create `Dockerfile` in project root:

```
# Stage 1: Build
FROM eclipse-temurin:21-jdk-alpine AS builder
WORKDIR /app
COPY .mvn/ .mvn/
COPY mvnw pom.xml ./
COPY src/ src/
RUN ./mvnw package -DskipTests

# Stage 2: Run
FROM eclipse-temurin:21-jre-alpine
WORKDIR /app
COPY --from=builder /app/target/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Step 6 — Create CHANGELOG.md

Create `CHANGELOG.md` in project root:

```
# Changelog
All notable changes to Nexgate Payment Service will be documented here.

---
```

```
## [Unreleased]
<!-- Developers: add your changes here before creating PR to staging -->

---

## [1.0.0] - 2026-04-06

### Added
- Initial release of Nexgate Payment Service
- Selcom payment integration
- M-Pesa integration
```

Step 7 — Create PR Template

Create `.github/pull_request_template.md`:

```
##  Description
<!-- What does this PR do? -->

##  Required Checklist

### For feature → staging:
- [ ] Code tested locally
- [ ] No hardcoded credentials
- [ ] CHANGELOG.md updated under [Unreleased]
- [ ] API endpoints tested

### For staging → master (Production):
- [ ] All features tested on staging
- [ ] CHANGELOG.md [Unreleased] renamed to version e.g [1.0.1]
- [ ] pom.xml version updated to match changelog
- [ ] No broken endpoints
- [ ] Database migrations tested
- [ ] Leadership notified of release

##  Optional
### Testing Evidence
### Notes for Reviewer
```

Step 8 — Create GitHub Actions CI Workflow

Create `.github/workflows/ci.yml`:

```
name: CI Pipeline

on:
  push:
    branches:
      - staging
      - master

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Java
        uses: actions/setup-java@v4
        with:
          java-version: '21'
          distribution: 'temurin'

      - name: Get version from pom.xml
        run: |
          VERSION=$(mvn help:evaluate -Dexpression=project.version -q -DforceStdout)
          echo "VERSION=$VERSION" >> $GITHUB_ENV

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Log in to ghcr.io
        uses: docker/login-action@v3
        with:
          registry: ghcr.io
          username: ${ github.actor }
          password: ${ secrets.GITHUB_TOKEN }
```

```
- name: Build and push Docker image
  uses: docker/build-push-action@v5
  with:
    context: .
    push: true
    tags: |
      ghcr.io/nexgate-hq/nexgate-payment-service:${{ env.VERSION }}
      ghcr.io/nexgate-hq/nexgate-payment-service:latest

- name: Trigger Jenkins
  if: success()
  run: |
    curl -X POST "${{ secrets.JENKINS_URL }}/job/nexgate-payment-
deploy/buildWithParameters" \
      --user "${{ secrets.JENKINS_USER }}:${{ secrets.JENKINS_TOKEN }}" \
      --data-urlencode "IMAGE_TAG=${{ env.VERSION }}" \
      --data-urlencode "BRANCH=${{ github.ref_name }}"
```

“ **Note:** Change `nexgate-payment-service` and `nexgate-payment-deploy` to match your service name!

Step 9 — Add GitHub Secrets to Repo

```
GitHub → nexgate-hq → nexgate-payment-service
  → Settings → Secrets and variables → Actions
  → New repository secret
```

Add:

```
JENKINS_URL    = https://jenkins.nexgate.co
JENKINS_USER   = admin_devops
JENKINS_TOKEN  = (get from Jenkins → admin_devops → Configure → API Token)
```

Step 10 — Setup Vault Paths

Create these paths in Vault (`vault.qbithub.com`):

```
secret/nexgate/payment_service/
```

```
├─ payment_local           ← local dev secrets
├─ payment_staging        ← staging app secrets
├─ payment_staging_infra  ← staging DB credentials
├─ payment_prod           ← production app secrets
└─ payment_prod_infra     ← production DB credentials
```

`payment_staging_infra` example:

```
{
  "POSTGRES_DB": "nexgate_payment_stg",
  "POSTGRES_USER": "nxg_payment_stg_user",
  "POSTGRES_PASSWORD": "StrongPasswordNoSpecialChars"
}
```

`payment_staging` example:

```
{
  "database": {
    "url": "jdbc:postgresql://payment-
postgres:5432/nexgate_payment_stg?stringtype=unspecified",
    "username": "nxg_payment_stg_user",
    "password": "StrongPasswordNoSpecialChars"
  },
  "rabbitmq": {
    "host": "rabbitmq",
    "port": "5672",
    "username": "nexgate_admin",
    "password": "RabbitMQPassword",
    "virtual-host": "/staging"
  }
}
```

“ **Important:** Passwords in `*_infra` and app secrets must match!

Step 11 — Update nexgate-infra Compose Files

Add the new service to both `staging/docker-compose.yml` and `prod/docker-compose.yml` in `nexgate-infra` repo:

staging/docker-compose.yml — add:

```
# New service: payment-postgres
payment-postgres:
  image: postgres:17
  container_name: nexgate_staging_payment_postgres
  restart: unless-stopped
  environment:
    POSTGRES_DB: ${PAYMENT_POSTGRES_DB}
    POSTGRES_USER: ${PAYMENT_POSTGRES_USER}
    POSTGRES_PASSWORD: ${PAYMENT_POSTGRES_PASSWORD}
  volumes:
    - nexgate_staging_payment_postgres_data:/var/lib/postgresql/data
  networks:
    - nexgate-staging

# New service: payment-service
payment-service:
  image: ghcr.io/nexgate-hq/nexgate-payment-service:${PAYMENT_TAG:-latest}
  container_name: nexgate_staging_payment_service
  restart: unless-stopped
  environment:
    SPRING_PROFILES_ACTIVE: staging
    SPRING_CLOUD_VAULT_ENABLED: true
    SPRING_CLOUD_VAULT_HOST: vault.qbitspark.com
    SPRING_CLOUD_VAULT_PORT: 443
    SPRING_CLOUD_VAULT_SCHEME: https
    SPRING_CLOUD_VAULT_AUTHENTICATION: TOKEN
    SPRING_CLOUD_VAULT_TOKEN: ${VAULT_TOKEN}
    SPRING_CLOUD_VAULT_KV_ENABLED: true
    SPRING_CLOUD_VAULT_KV_BACKEND: secret
    SPRING_CLOUD_VAULT_KV_DEFAULT_CONTEXT: nexgate/payment_service/payment_staging
    SPRING_CLOUD_VAULT_FAIL_FAST: true
    SPRING_CLOUD_VAULT_CONNECTION_TIMEOUT: 5000
    SPRING_CLOUD_VAULT_READ_TIMEOUT: 15000
  depends_on:
    - payment-postgres
    - rabbitmq
  networks:
    - nexgate-staging
```

Also add volume:

```
volumes:  
  nexgate_staging_payment_postgres_data:
```

Step 12 — Update Jenkins Pipeline

Update `nexgate-backend-deploy` pipeline to also fetch payment infra secrets:

In staging deploy stage, add:

```
# Fetch payment infra secrets  
PAYMENT_INFRA=$(curl -s -H "X-Vault-Token: $VAULT_TOKEN" \  
  https://vault.qbitspark.com/v1/secret/data/nexgate/payment_service/payment_staging_infra)  
  
PAYMENT_POSTGRES_DB=$(echo $PAYMENT_INFRA | jq -r '.data.data.POSTGRES_DB')  
PAYMENT_POSTGRES_USER=$(echo $PAYMENT_INFRA | jq -r '.data.data.POSTGRES_USER')  
PAYMENT_POSTGRES_PASSWORD=$(echo $PAYMENT_INFRA | jq -r '.data.data.POSTGRES_PASSWORD')
```

Add to `.env` generation:

```
echo "PAYMENT_TAG=latest"  
echo "PAYMENT_POSTGRES_DB=$PAYMENT_POSTGRES_DB"  
echo "PAYMENT_POSTGRES_USER=$PAYMENT_POSTGRES_USER"  
echo "PAYMENT_POSTGRES_PASSWORD=$PAYMENT_POSTGRES_PASSWORD"
```

Step 13 — Create Jenkins Job

```
jenkins.nexgate.co  
→ New Item  
→ Name: nexgate-payment-deploy  
→ Type: Pipeline  
→ OK  
  
→ This project is parameterized:  
  - IMAGE_TAG (String, default: latest)  
  - BRANCH (String, default: staging)  
  
→ Pipeline script: (copy from nexgate-notification-deploy and adapt)
```

- Change repo URL
- Change Vault paths
- Change service name in docker compose commands

Step 14 — Add DNS Record (if exposed via Traefik)

If the service needs a public URL:

```
API URL: payment.nexgate.co → 161.97.163.158
```

Add DNS A record in your domain provider.

Add Traefik labels in compose:

```
labels:
  - "traefik.enable=true"
  - "traefik.http.routers.payment-staging.rule=Host(`dev.payment.nexgate.co`)"
  - "traefik.http.routers.payment-staging.entrypoints=websecure"
  - "traefik.http.routers.payment-staging.tls.certresolver=letsencrypt"
  - "traefik.http.services.payment-staging.loadbalancer.server.port=8767"
networks:
  - nexgate-staging
  - proxy ← must join proxy network for Traefik!
```

Step 15 — Create staging branch and test

```
# In the new repo
git checkout -b staging
git push origin staging

# Push some code to trigger the pipeline
git add .
git commit -m "ci: initial service setup"
git push origin staging
```

Watch GitHub Actions → Jenkins → staging deployment!

2. Adding a New Frontend Service

Example: Adding `nexgate-web` (Next.js/React frontend)

A frontend service is different from backend:

- No Vault needed (secrets handled differently)
- No database
- Nginx serves static files or Next.js runs as Node server
- Still needs CI/CD and Docker

Step 1 — Create GitHub Repo

```
Name:      nexgate-web
Visibility: Private
```

Step 2 — Create Dockerfile

For Next.js:

```
# Stage 1: Build
FROM node:20-alpine AS builder
WORKDIR /app
COPY package*.json ./
RUN npm ci
COPY . .
RUN npm run build

# Stage 2: Run
FROM node:20-alpine
WORKDIR /app
COPY --from=builder /app/.next/standalone ./
COPY --from=builder /app/.next/static ./next/static
COPY --from=builder /app/public ./public
EXPOSE 3000
CMD ["node", "server.js"]
```

For React (static):

```
# Stage 1: Build
FROM node:20-alpine AS builder
WORKDIR /app
COPY package*.json ./
RUN npm ci
COPY . .
RUN npm run build

# Stage 2: Serve with Nginx
FROM nginx:alpine
COPY --from=builder /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
```

Step 3 — Handle Environment Variables

Frontend apps need API URLs. These are NOT secrets — they are public URLs. Use build args:

```
ARG NEXT_PUBLIC_API_URL=https://api.nexgate.co
ENV NEXT_PUBLIC_API_URL=$NEXT_PUBLIC_API_URL
```

In compose file:

```
nexgate-web:
  image: ghcr.io/nexgate-hq/nexgate-web:${WEB_TAG:-latest}
  container_name: nexgate_staging_web
  environment:
    NEXT_PUBLIC_API_URL: https://dev.api.nexgate.co ← staging
  networks:
    - nexgate-staging
    - proxy
  labels:
    - "traefik.enable=true"
    - "traefik.http.routers.nexgate-web-staging.rule=Host(`dev.nexgate.co`)"
    - "traefik.http.routers.nexgate-web-staging.entrypoints=websecure"
    - "traefik.http.routers.nexgate-web-staging.tls.certresolver=letsencrypt"
    - "traefik.http.services.nexgate-web-staging.loadbalancer.server.port=3000"
```

Step 4 — Create GitHub Actions CI Workflow

```
name: CI Pipeline

on:
  push:
    branches:
      - staging
      - master

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Get version
        run: |
          VERSION=$(cat package.json | jq -r '.version')
          echo "VERSION=$VERSION" >> $GITHUB_ENV

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Log in to ghcr.io
        uses: docker/login-action@v3
        with:
          registry: ghcr.io
          username: ${GITHUB_ACTOR}
          password: ${GITHUB_TOKEN}

      - name: Build and push
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          build-args: |
```

```
    NEXT_PUBLIC_API_URL=${{ github.ref_name == 'master' && 'https://api.nexgate.co' ||
'https://dev.api.nexgate.co' }}
    tags: |
      ghcr.io/nexgate-hq/nexgate-web:${{ env.VERSION }}
      ghcr.io/nexgate-hq/nexgate-web:latest

- name: Trigger Jenkins
  if: success()
  run: |
    curl -X POST "${{ secrets.JENKINS_URL }}/job/nexgate-web-deploy/buildWithParameters"
\
    --user "${{ secrets.JENKINS_USER }}:${{ secrets.JENKINS_TOKEN }}" \
    --data-urlencode "IMAGE_TAG=${{ env.VERSION }}" \
    --data-urlencode "BRANCH=${{ github.ref_name }}"
```

“ **Key difference from backend:** Version comes from `package.json` not `pom.xml`

Step 5 — Add GitHub Secrets

Same as backend:

```
JENKINS_URL
JENKINS_USER
JENKINS_TOKEN
```

Step 6 — Create Jenkins Job

```
jenkins.nexgate.co
→ New Item: nexgate-web-deploy
→ Pipeline
```

Pipeline script (simpler than backend — no Vault needed):

```
pipeline {
  agent any

  parameters {
```

```

    string(name: 'IMAGE_TAG', defaultValue: 'latest')
    string(name: 'BRANCH', defaultValue: 'staging')
}

stages {
  stage('Pull Latest Infra') {
    steps {
      sh 'cd /opt/nexgate && git pull origin main'
    }
  }

  stage('Deploy Staging') {
    when { expression { params.BRANCH == 'staging' } }
    steps {
      withCredentials([string(credentialsId: 'VAULT_TOKEN', variable:
'VAULT_TOKEN')]) {
        sh '''
          {
            echo "WEB_TAG=$IMAGE_TAG"
            echo "VAULT_TOKEN=$VAULT_TOKEN"
          } >> /opt/nexgate/staging/.env

          cd /opt/nexgate/staging
          docker compose pull nexgate-web
          docker compose up -d nexgate-web
          ...
        '''
      }
    }
  }

  stage('Approval for Production') {
    when { expression { params.BRANCH == 'master' } }
    steps {
      script {
        mail to: 'joshuasimon656@gmail.com',
            subject: "☐☐Nexgate Web v${params.IMAGE_TAG} - Production Deploy
Approval",
            body: "Approve at: ${BUILD_URL}input"
            input message: 'Deploy Web to Production?', ok: 'Approve'
      }
    }
  }
}

```

```

    }
  }
}

stage('Deploy Production') {
  when { expression { params.BRANCH == 'master' } }
  steps {
    withCredentials([string(credentialsId: 'VAULT_TOKEN', variable:
'VAULT_TOKEN')]) {
      sh '''
        {
          echo "WEB_TAG=$IMAGE_TAG"
          echo "VAULT_TOKEN=$VAULT_TOKEN"
        } >> /opt/nexgate/prod/.env

        cd /opt/nexgate/prod
        docker compose pull nexgate-web
        docker compose up -d nexgate-web
        ...
      '''
    }
  }
}

post {
  success { echo "☑ Web deployment successful!" }
  failure { echo "☒ Web deployment failed!" }
}
}

```

Step 7 — Add DNS Records

nexgate.co (or www.nexgate.co)	→ 161.97.163.158	(production)
dev.nexgate.co	→ 161.97.163.158	(staging)

Step 8 — Update nexgate-infra

Add to staging and prod compose files:

```
nexgate-web:
  image: ghcr.io/nexgate-hq/nexgate-web:${WEB_TAG:-latest}
  container_name: nexgate_staging_web
  restart: unless-stopped
  networks:
    - nexgate-staging
    - proxy
  labels:
    - "traefik.enable=true"
    - "traefik.http.routers.nexgate-web-staging.rule=Host(`dev.nexgate.co`)"
    - "traefik.http.routers.nexgate-web-staging.entrypoints=websecure"
    - "traefik.http.routers.nexgate-web-staging.tls.certresolver=letsencrypt"
    - "traefik.http.services.nexgate-web-staging.loadbalancer.server.port=3000"
```

No Vault needed — no volumes needed — no database. Simple! ☐

3. Adding a New Android App

Example: Adding `nexgate-merchant-app`

Android apps are different:

- No Docker
- No VPS deployment
- Build produces an APK or AAB
- Distribution via Play Store or direct download

Step 1 — Create GitHub Repo

```
Name:      nexgate-merchant-app
Visibility: Private
Language:  Kotlin
```

Step 2 — Create GitHub Actions CI Workflow

```
name: Android CI
```

```
on:
  push:
    branches:
      - staging
      - master

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Setup JDK 21
        uses: actions/setup-java@v4
        with:
          java-version: '21'
          distribution: 'temurin'

      - name: Setup Android SDK
        uses: android-actions/setup-android@v3

      - name: Make gradlew executable
        run: chmod +x ./gradlew

      - name: Build Debug APK (staging)
        if: github.ref_name == 'staging'
        run: ./gradlew assembleDebug

      - name: Build Release APK (production)
        if: github.ref_name == 'master'
        run: ./gradlew assembleRelease
        env:
          KEYSTORE_PASSWORD: ${ secrets.KEYSTORE_PASSWORD }}
          KEY_ALIAS: ${ secrets.KEY_ALIAS }}
          KEY_PASSWORD: ${ secrets.KEY_PASSWORD }}

      - name: Upload APK as artifact
```

```
uses: actions/upload-artifact@v4
with:
  name: nexgate-merchant-app
  path: app/build/outputs/apk/**/*.*apk
```

Step 3 — GitHub Secrets for Android

GitHub → nexgate-merchant-app → Settings → Secrets

```
KEYSTORE_PASSWORD ← APK signing keystore password
KEY_ALIAS          ← Key alias
KEY_PASSWORD       ← Key password
```

“ Store the actual keystore file in the repo (encrypted) or use GitHub Secrets for base64 encoded keystore.

Step 4 — API URLs in Android

Android apps connect to our backend. Configure base URLs per build variant:

build.gradle:

```
buildTypes {
  debug {
    buildConfigField "String", "BASE_URL", '"https://dev.api.nexgate.co/'"
  }
  release {
    buildConfigField "String", "BASE_URL", '"https://api.nexgate.co/'"
    minifyEnabled true
  }
}
```

No Jenkins needed — no VPS deployment — just build artifacts! ☐

4. Checklist Summary

New Backend Microservice Checklist

GitHub:

- Create private repo in nexgate-hq org
- Add JENKINS_URL, JENKINS_USER, JENKINS_TOKEN secrets
- Create staging branch
- Add branch protection (when on paid plan)

Code:

- Spring Boot project with Vault dependency
- bootstrap.properties created (added to .gitignore)
- application.properties (no secrets, placeholders only)
- Dockerfile (multistage)
- CHANGELOG.md
- .github/pull_request_template.md
- .github/workflows/ci.yml

Vault:

- Create {service}_local path
- Create {service}_staging path
- Create {service}_staging_infra path
- Create {service}_prod path
- Create {service}_prod_infra path
- Fill all paths with correct values
- Ensure passwords match between infra and app paths

nexgate-infra:

- Add service postgres to staging compose
- Add service container to staging compose
- Add service postgres to prod compose
- Add service container to prod compose
- Add volumes for new postgres
- Add Traefik labels if public URL needed

Jenkins:

- Create new Jenkins job (Pipeline)
- Add IMAGE_TAG and BRANCH parameters
- Write pipeline script
- Update nexgate-backend-deploy to fetch new service infra secrets

- Update .env generation to include new service credentials

DNS (if needed):

- Add A record for staging URL
- Add A record for production URL

Test:

- Push to staging branch → verify GitHub Actions runs
- Verify Jenkins deploys to staging
- Verify service starts and connects to DB/Vault
- Test API endpoints

New Frontend Service Checklist

GitHub:

- Create private repo in nexgate-hq org
- Add JENKINS_URL, JENKINS_USER, JENKINS_TOKEN secrets
- Create staging branch

Code:

- Dockerfile (multistage with Node + Nginx or Node server)
- .github/workflows/ci.yml
- API URLs via build args (not hardcoded!)

nexgate-infra:

- Add service to staging compose (no DB, no Vault)
- Add service to prod compose
- Add Traefik labels with correct domain

Jenkins:

- Create new Jenkins job
- Simple pipeline (no Vault fetching needed)
- Add WEB_TAG to .env generation in main pipeline

DNS:

- Add staging domain A record
- Add production domain A record

Test:

- Push to staging → GitHub Actions → Jenkins → staging deploy
- Verify frontend loads at dev.nexgate.co

New Android App Checklist

GitHub:

- Create private repo in nexgate-hq org
- Add signing secrets (KEYSTORE_PASSWORD, KEY_ALIAS, KEY_PASSWORD)

Code:

- Kotlin project with correct build variants
- Staging URL → dev.api.nexgate.co
- Production URL → api.nexgate.co
- .github/workflows/android-ci.yml

No Jenkins needed □

No Vault needed □

No compose files needed □

Test:

- Push to staging → APK built as artifact
- Download and install APK
- Verify connects to dev.api.nexgate.co

Quick Reference — Port Numbers

When adding a new backend service, pick a unique port:

Service	Port
nexgate-backend	8765
nexgate-notification-server	8766
nexgate-payment-service	8767
nexgate-web (Next.js)	3000
<i>next new service</i>	8768

Keep a running list and never reuse ports!

Quick Reference — Naming Convention

Thing	Pattern	Example
GitHub repo	<code>nexgate-{service-name}</code>	<code>nexgate-payment-service</code>
Docker image	<code>ghcr.io/nexgate-hq/nexgate-{service}</code>	<code>ghcr.io/nexgate-hq/nexgate-payment-service</code>
Container (staging)	<code>nexgate_staging_{service}</code>	<code>nexgate_staging_payment_service</code>
Container (prod)	<code>nexgate_prod_{service}</code>	<code>nexgate_prod_payment_service</code>
Vault path	<code>nexgate/{service}/{env}</code>	<code>nexgate/payment_service/payment_staging</code>
Jenkins job	<code>nexgate-{service}-deploy</code>	<code>nexgate-payment-deploy</code>
Traefik router	<code>{service}-{env}</code>	<code>payment-staging</code>
Staging URL	<code>dev.{service}.nexgate.co</code>	<code>dev.payment.nexgate.co</code>
Production URL	<code>{service}.nexgate.co</code>	<code>payment.nexgate.co</code>
Docker volume	<code>nexgate_{env}_{service}_data</code>	<code>nexgate_staging_payment_postgres_data</code>

*Document maintained by Josh Sakweli — DevOps Lead,
Classification: Internal — DevOps Lead Only
For questions: office@qbitspark.com*

Revision #1

Created 6 April 2026 19:19:10 by Admin Qbit

Updated 6 April 2026 19:19:56 by Admin Qbit