DevOps GDD

Version: v1.0

Author: Solo Developer

Date: September 15, 2025

Purpose

This Game-ready DevOps GDD defines a small but production-like service that demonstrates the core capabilities expected of a DevOps/Platform Engineer in game development: CI/CD, observability, SLOs, IaC, security-by-default, and operational excellence. The scope is intentionally minimal so a single developer can deliver it before summer internships begin in June 2026.

High-level Goal

Build a telemetry-enabled microservice with a tiny combat simulation endpoint and a write-optimized event ingest API. Ship with dashboards, alerts-as-code, IaC for cloud bootstrapping, secure pipelines, test strategy, and runnable demo.

Functional Scope (Minimal, Demo-friendly)

- POST /events: idempotent event ingestion for game telemetry (item_collected, login, error).
- POST /combat/simulate: deterministic small-scale combat simulator with RNG seeding for reproducibility.
- GET /metrics, /healthz, /readyz: operational endpoints for Prometheus scraping and K8s probes.
- Admin: simple read-only list of last 50 events (for demo).

Non-Functional Requirements (NFR)

- Latency: p95 < 500 ms on /events and /combat/simulate under 100 RPS.
- Availability: 99.5% monthly SLO.
- Throughput: sustain 1k events/min with < 1% drops.
- Security: no plaintext secrets in repo; images and deps scanned on every PR; SBOM generated.
- Cost: dev+demo infra under $30/month; teardown documented.

Architecture Overview

Stack: ASP.NET Core (.NET 8), EF Core, PostgreSQL, Docker/Compose for local, optional K8s+Helm for demo cloud. Prometheus + Grafana for metrics, Loki (optional) for logs, Alertmanager for paging. Infrastructure via Terraform (VPC/network, managed Postgres, container runtime), container registry, and minimal compute.

Key Components

- API Service: stateless; exposes /events, /combat/simulate, /metrics, /healthz, /readyz.

- Database: PostgreSQL with append-only events table; daily snapshots + PITR if supported.
- Observability: Prometheus scrape, custom business metrics, Grafana dashboard JSON in repo.
- CI/CD: GitHub Actions (or GitLab CI) with build, test, scan, SBOM, image push, deploy.
- IaC: Terraform root module + env workspaces; Helm chart with values for dev/demo.

Data Model (Minimal)

```
events(id uuid, occurred_at timestamptz, type text, player_id text, session_id text, idempotency_key text unique, payload jsonb, ingested_at timestamptz default now()). Index on (type, occurred_at).
```

API Endpoints

- POST /events: { type, player_id, session_id, idempotency_key, occurred_at, payload } → 202 Accepted.
- POST /combat/simulate: { seed, attacker, defender } → { result, rounds, seed_used }.
- GET /metrics: Prometheus exposition. GET /healthz (dependencies optional), GET /readyz (DB required).

Idempotency & Reproducibility

All /events writes must be idempotent on idempotency_key (conflict = 200 OK, no duplicate). Combat sim uses provided seed; responses include seed_used for replay. Log correlation via correlation_id and stack_id fields.

Observability: Metrics, SLI/SLO, Dashboards

Service-level Indicators (SLI) & Objectives (SLO)

- Availability (A): successful request fraction over 30 days; SLO A ≥ 99.5%.
- Latency (L): p95 request duration on /events and /combat/simulate; SLO L_p95 ≤ 500 ms.
- Error Rate (E): 5xx fraction; SLO E ≤ 0.5%.

Prometheus Counters/Gauges/Histograms

- http_requests_total{path,method,code}
- http_request_duration_seconds_bucket{path,method} (histogram)
- events_ingested_total{type}
- combat_simulations_total{result}

Alerting Rules (as code)

Examples (PromQL):

```
ALERT HighErrorRate expr: sum(rate(http_requests_total{code=~"5.."}[5m])) / sum(rate(http_requests_total[5m])) > 0.005 for: 10m labels: { severity="page" } annotations: { summary="5xx > 0.5% for 10m" }
```

```
ALERT HighLatencyP95 expr: histogram_quantile(0.95, sum(rate(http_request_duration_seconds_bucket[5m])) by (le, path)) > 0.5 for: 10m labels: { severity="page" } annotations: { summary="p95 latency > 500ms" }
```

Logging & Tracing (Optional Enhancements)

Structured logs (JSON) with correlation_id and stack_id. Optional Loki for aggregation. Lightweight tracing via W3C Trace Context headers.

Security-by-Default

- Secrets: GitHub OIDC to cloud; no long-lived keys. Local dev uses .env (excluded) + sample.env.
- SAST: CodeQL on PRs; Dependency scanning (dotnet list package --vulnerable).
- Container scan: Trivy on built image; fail on HIGH/CRITICAL severities; attach report artifact.
- SBOM: Syft (SPDX) generated per release.
- gitleaks pre-commit hook; branch protection; required reviews for privileged changes.
- Threat model (mini): STRIDE summary in /docs/security/threat-model.md.

CI/CD Pipeline (GitHub Actions example)

- Build: restore, build, test (.NET 8), integration tests with Testcontainers for Postgres.
- Quality Gates: Code coverage gate; CodeQL; Trivy; SBOM.
- Contract & Load: Pact-like contract tests; k6 smoke/load (100 RPS, pass if p95 < 500ms, err < 0.5%).
- Image: build + tag (semver + git sha), push to registry.
- Deploy: Helm upgrade --install to demo environment; run post-deploy smoke checks.
- Release Notes: auto-generate from Conventional Commits.

Example Workflow Snippet

```
jobs: build_test: runs-on: ubuntu-latest steps: - uses: actions/checkout@v4 - uses: actions/setup-dotnet@v4 with: { dotnet-version: "8.0.x" } - name: Unit & Integration Tests run: dotnet test --collect:"XPlat Code Coverage" - name: Build Image run: docker build -t ghcr.io/org/app:${{ github.sha }} . - name: Trivy Scan run: trivy image --exit-code 1 --severity HIGH,CRITICAL ghcr.io/org/app:${{ github.sha }} - name: SBOM run: syft ghcr.io/org/app:${{ github.sha }} -o spdx-json > sbom.json
```

Infrastructure as Code (Terraform + Helm)

- Terraform: root module with variables for project, region, instance sizes; separate workspaces for dev/demo.
- Resources: VPC/network, managed Postgres (e.g., AWS RDS), container runtime (ECS/Fargate or small K8s), ECR/GAR.
- Outputs: DB URL (rotated), Prometheus endpoint, Grafana admin secret (randomized).
- Helm: chart under /deploy/helm/app with values-dev.yaml and values-demo.yaml; rolling updates (`maxSurge: 1`, `maxUnavailable: 0`).

Testing Strategy

- Unit tests for domain logic (combat).
- Integration tests with ephemeral Postgres (Testcontainers).

- Contract tests for /events and /combat/simulate (schema + behavior).
- Load tests with k6: smoke (10 RPS, 1m) and stress (100 RPS, 10m).
- Chaos-lite: random DB unavailability during tests to validate retries and /readyz.

Release & Deployment Strategy

- Semantic versioning (MAJOR.MINOR.PATCH).
- Blue/Green or Canary (1 pod → 2 pods) via Helm values; automatic rollback on failed health checks.
- Migrations: EF Core with safe, incremental migrations; pre-deploy validation; rollback plan documented.
- Artifacts: container image, SBOM, k6 and scan reports attached to releases.

Backups, Restore & Runbooks

- Backups: daily snapshot + 7-day retention; PITR if available; encrypt at rest.
- Restore tests: monthly restore into disposable DB and run data integrity checks.
- Runbooks: incident templates for HighErrorRate, HighLatencyP95, DB connection saturation, and failed deploys.
- On-call (solo): notifications to email/Slack; quiet hours policy; auto-teardown for demo.

Perforce (p4) Mirror Proof-of-Concept

Provide a tiny script to mirror selected Git content into a Perforce depot (read-only) to demonstrate familiarity with game studio workflows. Document prerequisites and a dry-run.

Repo Evidence Index (What to Review in 5 Minutes)

Capability	Artifact Path(s)
CI/CD	.github/workflows/ci.yml, .github/workflows/release.yml
Observability	ops/grafana/dashboard.json, ops/alerts/alertmanager.yml
Metrics	src/App/Metrics/*.cs, GET /metrics
Security	codeql.yml, ops/security/trivy-report.md, sbom.json, .pre-commit-config.y
IaC	infra/terraform/*, deploy/helm/app/*
Testing	tests/unit/*, tests/integration/*, tests/contracts/*, tests/k6/*
Runbooks	docs/runbooks/*.md
Perforce PoC	tools/p4/mirror.sh, tools/p4/README.md

Roadmap (Solo Dev → Summer Internships 2026)

Dates assume start on September 15, 2025. Aim to finish by end of May 2026.

Month / Milestone	Deliverables
Sep 2025 (M0)	Project scaffold, Docker/Compose, healthz/readyz, /events & /combat M
Oct 2025 (M1)	Business metrics + Prometheus + /metrics; Grafana dashboard JSON; bas
Nov 2025 (M2)	CI basics (build/test/coverage); integration tests (Testcontainers);
Dec 2025 (M3)	SLO/SLI formalized; alerts-as-code; on-call notifications; README rev
Jan 2026 (M4)	Security gates: CodeQL, Trivy, gitleaks, SBOM; threat model doc; bran
Feb 2026 (M5)	Terraform minimal cloud (VPC + managed Postgres + registry); demo dep
Mar 2026 (M6)	Helm chart + rollout strategy; blue/green or canary; post-deploy smok
Apr 2026 (M7)	k6 load/stress with thresholds; capacity notes; cost controls & teard
May 2026 (M8)	Hardening, docs polish, Perforce mirror PoC, mock incident drill, fin

Acceptance Criteria

- Run locally with docker compose up; /healthz green, demo data visible, Grafana dashboard loads.
- CI green on main: unit+integration+contract tests, coverage ≥ 70%, scans pass, SBOM attached.
- One-click deploy to demo with Helm; alerts loaded; k6 smoke passes; SLO graphs present.
- Disaster drill: restore latest backup to fresh DB and run integrity checks successfully.
- Repo contains Evidence Index paths and Reviewer Guide section in README.

Appendix: Example Config Paths

- Prometheus scrape: ops/prometheus/prometheus.yml
- Alertmanager routing: ops/alerts/alertmanager.yml (email/Slack/Webhook)
- Grafana dashboard JSON: ops/grafana/dashboard.json
- Helm chart: deploy/helm/app/ (values-dev.yaml, values-demo.yaml)
- Terraform root: infra/terraform/ (variables.tf, outputs.tf, main.tf)
- k6 tests: tests/k6/smoke.js, tests/k6/stress.js
- Runbooks: docs/runbooks/*.md

Note on fonts: The document uses Courier (monospace) as a Consolas-like style available by default for portability.