# Study Plan — *NGINX Cookbook* (Derek DeJonghe)

User Story Template and Examples

October 28, 2025

## Contents


**1 Basics** **3**

**2 High-Performance Load Balancing** **4**

**3 Traffic Management** **5**

**4 Content Caching** **6**

**5 Programmability and Automation** **7**

**6 Authentication** **8**

**7 Security Controls** **9**

**8 HTTP/2 and HTTP/3** **10**

**9 Media Streaming** **11**

**10 Cloud Deployments** **12**

**11 Containers and Microservices** **13**

**12 High Availability** **14**

**13 Monitoring** **15**

**14 Debugging and Troubleshooting** **16**

**15 Performance Tuning** **17**

**A Blank Story Card (Copy/Paste)** **18**

## How to Use This Template

This document provides a **ready-to-use user story system** for mastering the chapters of *NGINX Cookbook*. Each chapter is mapped to a story card including: intent/value, persona, risks, BDD-style acceptance criteria, and a checklist of practical tasks.

**Structure.** For every chapter, fill out: Epic/Feature, Business Value, Priority/Estimate, Persona, Dependencies, Assumptions, Risks, the "As a ... I want ... so that ..." story, Non-Functional tags, AC (Given/When/Then), and a checklist of tasks.

**Files.** Save your lab artifacts under `/nginx-labs/<chapter>/`. Commit config snapshots and test notes to keep a traceable learning record.

## Writing Effective User Stories (Quick Guide)

**Pattern.** *As a <persona>, I want <capability> so that <business value/outcome>.*

**Tips.**

- Keep capability concrete and demonstrable (one deployable slice).
- State value in terms of risk, speed, reliability, or cost.
- Add BDD Acceptance Criteria that an observer could verify.
- Prefer 15–60 minute tasks; longer goals should be split.
- Capture *Non-Functional* tags (e.g., **Performance**, **Security**).

# 1 Basics

## NGX-1 — Getting Started with NGINX

| | |
|---|---|
| **Epic / Feature** | Foundations |
| **Business Value** | Stand up a reproducible NGINX instance and understand its layout to serve static content safely. |
| **Priority / Estimate** | Must SP: 3 |
| **Persona** | Developer on a new host |
| **Dependencies** | Linux VM; package manager access |
| **Assumptions** | Ports 80/443 open; DNS optional |
| **Risks** | Local vs server toolchain drift; firewall surprises |

**Story** As a developer, I want to install and run NGINX so that I can serve content and understand the configuration layout.

**Non-Functional** Performance Reliability Security

### Acceptance Criteria (BDD)

**Scenario** Happy path
**Given** a clean Linux VM
**When** NGINX is installed, enabled, and a minimal server block is configured
**Then** `curl -i` returns HTTP 200 for the site, and logs are written as expected.

### Tasks

- ☐ Install via `apt` or `yum`; verify `nginx -V`.
- ☐ Map the layout: `nginx.conf`, `sites-available`, `sites-enabled`, logs.
- ☐ Serve `/var/www/html` with a minimal server block; validate with `curl -i`.
- ☐ Split config using `include`; reload via `nginx -t` and `systemctl reload nginx`.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

## 2 High-Performance Load Balancing

### NGX-2 — Balance HTTP/TCP/UDP Traffic

| | |
|---|---|
| **Epic / Feature** | Traffic Distribution |
| **Business Value** | Increase availability and throughput by spreading load and handling node failures gracefully. |
| **Priority / Estimate** | **Must** **SP: 5** |
| **Persona** | Platform engineer |
| **Dependencies** | Two upstream demo apps |
| **Assumptions** | Health checks reachable |
| **Risks** | Uneven distribution; sticky sessions not honored |

**Story** As a platform engineer, I want to load balance across upstreams so that the service remains responsive during failures.

**Non-Functional** **Performance** **Reliability**

#### Acceptance Criteria (BDD)

**Scenario** Algorithm comparison
**Given** two upstream backends
**When** I configure `least_conn`, `ip_hash`, and `hash` strategies
**Then** requests distribute per algorithm and passive/active health checks remove bad nodes.

#### Tasks

- ☐ Define an `upstream` with two containers; add a server block that proxies to it.
- ☐ Switch algorithms (`least_conn`, `ip_hash`, `hash`); observe with repeated `curl`.
- ☐ Add passive and active health checks; simulate a failing node and verify removal/recovery.
- ☐ Enable slow-start and, if relevant, cookie stickiness notes.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 3 Traffic Management

**NGX-3 — Shape and Steer Requests**

| | |
|---|---|
| **Epic / Feature** | Policy and Control |
| **Business Value** | Protect origins and experiment safely under load via rate, conn, and geo controls. |
| **Priority / Estimate** | **Should** **SP: 5** |
| **Persona** | SRE |
| **Dependencies** | Ch. 2 config |
| **Assumptions** | Client IP visibility |
| **Risks** | Over-throttling legitimate users |

**Story** As an SRE, I want to manage request rates and routes so that services remain stable during bursts.

**Non-Functional** **Reliability** **Security**

Acceptance Criteria (BDD)

**Scenario** Rate limiting
**Given** `limit_req` and `limit_conn` zones
**When** I drive bursts with a load tool
**Then** 429s occur as configured and backends remain healthy.

Tasks

- ☐ Add A/B routing by header/path for canaries.
- ☐ Configure `limit_req` and `limit_conn`; test with `ab` or `wrk`.
- ☐ Configure `set_real_ip_from` and `real_ip_header` if behind another proxy.
- ☐ Add GeoIP-based allow/deny and verify logic.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 4 Content Caching

**NGX-4 — Reverse-Proxy Cache Acceleration**

| | |
|---|---|
| **Epic / Feature** | Caching |
| **Business Value** | Reduce latency and origin load with safe cache keys, locking, and stale serving. |
| **Priority / Estimate** | **Must** **SP: 5** |
| **Persona** | Backend engineer |
| **Dependencies** | Ch. 2 upstream |
| **Assumptions** | Cacheable responses |
| **Risks** | Stale data exposure; incorrect keys |

**Story** As a backend engineer, I want a tuned proxy cache so that users see faster responses with consistent behavior.

**Non-Functional** **Performance** **Reliability**

Acceptance Criteria (BDD)

**Scenario** Cache hit/miss
**Given** a configured `proxy_cache_path` and key
**When** I request the same resource multiple times
**Then** hit ratios rise and stale-on-error works when backends fail.

Tasks

- ☐ Define cache path/zone; add cache key and cache lock.
- ☐ Enable `proxy_cache_revalidate`, `stale` on errors; expose an `X-Cache` header.
- ☐ Demonstrate bypass and, if available, purge.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 5 Programmability and Automation

**NGX-5 — API, njs, and Templating**

| | |
|---|---|
| **Epic / Feature** | Automation |
| **Business Value** | Reduce manual config drift and enable dynamic routing via scripts and templates. |
| **Priority / Estimate** | **Should** **SP: 3** |
| **Persona** | Automation engineer |
| **Dependencies** | Ch. 1–4 |
| **Assumptions** | Shell access; optional NGINX Plus features |
| **Risks** | Script errors affecting prod |

**Story** As an automation engineer, I want scriptable configuration so that changes are safe and repeatable.

**Non-Functional** **Reliability** **Security**

Acceptance Criteria (BDD)

**Scenario** Template render
**Given** a template with environment variables
**When** I render and reload NGINX
**Then** the new upstream endpoints take effect without syntax errors.

Tasks

- ☐ Create an env-subst or Consul-Template to render upstreams.
- ☐ Write a tiny njs function to rewrite a header.
- ☐ (Optional) Explore NGINX Plus key-value store/API.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 6 Authentication

## NGX-6 — Access Control and Tokens

| | |
|---|---|
| **Epic / Feature** | Identity |
| **Business Value** | Protect endpoints and centralize access policies to reduce risk. |
| **Priority / Estimate** | Must SP: 3 |
| **Persona** | Security engineer |
| **Dependencies** | Auth service/JWT if available |
| **Assumptions** | TLS in place or planned |
| **Risks** | Locking out legitimate users |

**Story** As a security engineer, I want to enforce auth at the edge so that only authorized users reach backends.

**Non-Functional** Security Privacy

### Acceptance Criteria (BDD)

**Scenario** Basic and subrequest auth
**Given** a protected location
**When** I present valid credentials or token
**Then** the request succeeds; otherwise it fails with the proper code.

### Tasks

- ☐ Enable Basic auth for a path; verify 401/200 behavior.
- ☐ Configure an auth subrequest to a small demo service.
- ☐ (Optional) Validate JWTs if features are available.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 7 Security Controls

**NGX-7 — Transport and Origin Security**

| | |
|---|---|
| **Epic / Feature** | Edge Security |
| **Business Value** | Enforce encryption and safe origin access to lower exploit and abuse risk. |
| **Priority / Estimate** | Must SP: 3 |
| **Persona** | Security engineer |
| **Dependencies** | Valid certs |
| **Assumptions** | Browser clients |
| **Risks** | Misconfigured redirects or HSTS |

**Story** As a security engineer, I want hardened TLS and strict origin policy so that data in transit is protected.

**Non-Functional** Security Reliability

Acceptance Criteria (BDD)

**Scenario** TLS hygiene
**Given** a TLS-enabled vhost
**When** a client connects over HTTPS
**Then** modern ciphers are used, HSTS is set, and HTTP redirects appropriately.

Tasks

- ☐ Add IP allow/deny and CORS rules for an API.
- ☐ Configure HSTS and HTTP to HTTPS redirect.
- ☐ (Optional) Use secure-link for expiring signed URLs.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 8 HTTP/2 and HTTP/3

**NGX-8 — Modern HTTP and gRPC**

| | |
|---|---|
| **Epic / Feature** | Protocols |
| **Business Value** | Lower latency and better multiplexing for modern clients and services. |
| **Priority / Estimate** | Should SP: 3 |
| **Persona** | Platform engineer |
| **Dependencies** | TLS configured |
| **Assumptions** | gRPC sample available |
| **Risks** | Client fallback issues |

**Story** As a platform engineer, I want HTTP/2 and HTTP/3 enabled so that clients benefit from multiplexing and QUIC.

**Non-Functional** Performance

Acceptance Criteria (BDD)

**Scenario** Protocol negotiation
**Given** a TLS site with HTTP/2 and HTTP/3 enabled
**When** I connect with tools that prefer each protocol
**Then** negotiation succeeds and responses are valid.

Tasks

- ☐ Enable HTTP/2 on TLS vhost; verify with `curl -http2 -I`.
- ☐ Enable HTTP/3 (QUIC); verify with `curl -http3 -I`.
- ☐ Proxy to a gRPC backend and confirm streaming.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 9 Media Streaming

**NGX-9 — Static and Adaptive Streaming**

| | |
|---|---|
| **Epic / Feature** | Media Delivery |
| **Business Value** | Deliver video efficiently and fairly under bandwidth constraints. |
| **Priority / Estimate** | Could SP: 3 |
| **Persona** | Media engineer |
| **Dependencies** | Sample media files |
| **Assumptions** | Players available |
| **Risks** | Bandwidth exhaustion |

**Story** As a media engineer, I want to serve MP4/HLS with throttling so that users get smooth playback.

**Non-Functional** Performance

Acceptance Criteria (BDD)

**Scenario** Throttled streaming
**Given** a media location with limits
**When** clients request content
**Then** transfers honor bandwidth caps without 5xx errors.

Tasks

- ☐ Serve MP4; configure byte-range support.
- ☐ Enable HLS/HDS (as available); test playback.
- ☐ Apply bandwidth limits and observe client behavior.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 10 Cloud Deployments

**NGX-10 — Images and Instances**

| | |
|---|---|
| **Epic / Feature** | Cloud Ops |
| **Business Value** | Enable rapid, consistent provisioning across environments. |
| **Priority / Estimate** | **Should** **SP: 5** |
| **Persona** | Cloud engineer |
| **Dependencies** | Cloud account |
| **Assumptions** | User-data/bootstrap allowed |
| **Risks** | Misconfigured security groups |

**Story** As a cloud engineer, I want a golden image and bootstrap so that I can provision NGINX nodes quickly.

**Non-Functional** **Reliability** **Security**

Acceptance Criteria (BDD)

**Scenario** Reproducible node
**Given** an image and bootstrap script
**When** I launch a new instance
**Then** the node serves traffic with the expected config and tags.

Tasks

- ☐ Bake a cloud image with NGINX preinstalled.
- ☐ Write user-data to fetch config and register with LB.
- ☐ Compare LB-in-front vs direct routing (pros/cons).

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 11 Containers and Microservices

**NGX-11 — Containerized Edge and Ingress**

| | |
|---|---|
| **Epic / Feature** | Container Ops |
| **Business Value** | Ship consistent NGINX builds and route to dynamic services. |
| **Priority / Estimate** | Must SP: 5 |
| **Persona** | Platform/SRE |
| **Dependencies** | Docker; optional Kubernetes |
| **Assumptions** | Cluster access if testing Ingress |
| **Risks** | Stale endpoints; DNS caching |

**Story** As a platform engineer, I want containerized NGINX and ingress patterns so that microservices are exposed safely.

**Non-Functional** Reliability Security

Acceptance Criteria (BDD)

**Scenario** Dynamic backend
**Given** containerized NGINX
**When** I deploy a new service with a new endpoint
**Then** routing updates without downtime.

Tasks

- ☐ Use the official image; add custom config via Dockerfile.
- ☐ Template upstreams from env/labels; reload hot.
- ☐ Explore NGINX Ingress Controller path rules in a cluster.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 12 High Availability

**NGX-12 — Remove Single Points of Failure**

| | |
|---|---|
| **Epic / Feature** | Resilience |
| **Business Value** | Maintain service continuity during node failures and maintenance. |
| **Priority / Estimate** | **Must** **SP: 5** |
| **Persona** | SRE |
| **Dependencies** | Multiple nodes |
| **Assumptions** | Shared network or VIP |
| **Risks** | Split-brain; config divergence |

**Story** As an SRE, I want HA deployment modes so that traffic continues during failures.
**Non-Functional** **Reliability**

Acceptance Criteria (BDD)

**Scenario** Failover drill
**Given** two NGINX nodes
**When** I simulate a failure
**Then** traffic continues via VIP/DNS without client-visible outage.

Tasks

- ☐ Compare VRRP/VIP vs DNS-based HA.
- ☐ Sync configs/state; test failover.
- ☐ Document recovery and rollback steps.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 13 Monitoring

**NGX-13 — Observe and Measure**

| | |
|---|---|
| **Epic / Feature** | Observability |
| **Business Value** | Spot regressions quickly; prove SLIs/SLOs. |
| **Priority / Estimate** | **Must** **SP: 3** |
| **Persona** | SRE |
| **Dependencies** | Metrics stack |
| **Assumptions** | Exporter or Plus API |
| **Risks** | Sampling gaps |

**Story** As an SRE, I want metrics and dashboards so that I can track errors, latency, and capacity.

**Non-Functional** **Reliability**

Acceptance Criteria (BDD)

**Scenario** Dashboard review
**Given** scraped metrics
**When** I deploy a configuration change
**Then** p95 latency and 4xx/5xx panels reflect the change.

Tasks

- ☐ Enable `stub_status` or Plus metrics; scrape with Prometheus.
- ☐ Build a Grafana dashboard (QPS, errors, p95).
- ☐ (Optional) Export traces with OpenTelemetry.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 14 Debugging and Troubleshooting

**NGX-14 — Logs and Request Tracing**

| | |
|---|---|
| **Epic / Feature** | Diagnostics |
| **Business Value** | Shorten MTTR by making failures easy to isolate. |
| **Priority / Estimate** | Must SP: 3 |
| **Persona** | SRE/Developer |
| **Dependencies** | Log collection |
| **Assumptions** | JSON logs acceptable |
| **Risks** | Excessive debug logging |

**Story** As an engineer, I want structured logs and correlation IDs so that I can trace requests end to end.

**Non-Functional** Reliability

Acceptance Criteria (BDD)

**Scenario** Correlated request
**Given** a unique request ID header
**When** a request crosses multiple services
**Then** the ID appears in all logs for that flow.

Tasks

- ☐ Configure JSON access logs and structured error logs.
- ☐ Forward logs to syslog/collector; add request ID propagation.
- ☐ Perform a bad-deploy drill and roll back guided by logs.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# 15 Performance Tuning

## NGX-15 — Tune for Throughput and Latency

| | |
|---|---|
| **Epic / Feature** | Optimization |
| **Business Value** | Meet SLOs under peak load with efficient resources. |
| **Priority / Estimate** | Should SP: 5 |
| **Persona** | Performance engineer |
| **Dependencies** | Load tool |
| **Assumptions** | Baseline captured |
| **Risks** | Premature tuning; kernel mis-tuning |

**Story** As a performance engineer, I want a tuning checklist so that I can improve p95/p99 latencies without regressions.

**Non-Functional** Performance

### Acceptance Criteria (BDD)

**Scenario** Before/after benchmark
**Given** a baseline test
**When** I apply buffer/keepalive/kernel tweaks
**Then** throughput increases or latency decreases within error bounds.

### Tasks

- ☐ Capture baseline with `wrk` or `k6`.
- ☐ Adjust `keepalive_requests`, proxy buffers, log buffering, and gzip; retest.
- ☐ Evaluate kernel params (`somaxconn`, ephemeral ports, TCP tuning); retest and record deltas.

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.

# A Blank Story Card (Copy/Paste)

**<ID> — <Short Title>**

| | |
|---|---|
| **Epic / Feature** | <Epic or feature> |
| **Business Value** | <Outcome/value statement> |
| **Priority / Estimate** | Must/Should/Could SP: n |
| **Persona** | <Primary user> |
| **Dependencies** | <Systems/teams> |
| **Assumptions** | <What must be true> |
| **Risks** | <What could go wrong> |

**Story** As a <persona>, I want <capability> so that <value>.

**Non-Functional** Performance Security Reliability

Acceptance Criteria (BDD)

**Scenario** Happy path
**Given** <preconditions>
**When** <action>
**Then** <observable outcome>.

Tasks

- ☐ <Task 1 (15–60 min)>
- ☐ <Task 2>
- ☐ <Task 3>

*Definition of Ready:* Persona clear; AC drafted; Dependencies known; Estimate set. *Definition of Done:* All ACs pass; tests green; security/accessibility checks; docs updated; deployed or feature-flagged.