

Adoption Map

AI-Native DevOps - Appendix G, Adopt This in Your Own Project

Elvis Tafoh-Ngunjoh

G.1 The substitution map

The book uses fixed names so that every command is copyable and every chapter's evidence is comparable. None of those names is load-bearing. When you apply a chapter to your own project, substitute the left column with your value, record your values once in a `docs/adoption-map.md` file in your repository, and keep using the same values in every later chapter.

Book name	What it is	Your value
reference-app	The service under management	Your service or repository name
Port 8080	The service's listen port	Your service's port
/, /health, /ready	Root, liveness, and readiness endpoints	Your service's equivalent endpoints, added if absent
reference-staging	Staging namespace and Argo CD Application	Your staging namespace
reference-production	Production namespace and Argo CD Application	Your production namespace
ai-native-devops	Argo CD project	Your platform or team project name
ghcr.io/<owner>/reference-app	Image path in the registry	Your registry and image path
src/, tests/	Application code and contract tests	Your repository's source and test layout
.github/workflows/ci.yml, delivery.yml, rollback.yml	The delivery workflows	Your CI system's equivalent pipeline files
infrastructure/	Terraform module and environment roots	Your infrastructure repository or directory
docs/incidents/, docs/decisions/, docs/security/	Evidence and decision records	The same paths, created in your repository

Two rules keep the substitution honest. First, substitute names, never controls: if a chapter requires a human approval, a digest rather than a tag, or a recorded decision, that requirement transfers unchanged. Second, when your project lacks something the map names, such as a readiness endpoint, treat adding it as the first Apply It to Your Project step for the chapter that uses it, because the later chapters assume it exists.

G.2 If your stack differs

The chapters demonstrate one concrete stack: GitHub Actions, Docker, Terraform on AWS, Kubernetes, and Argo CD. The methods are contracts, and each contract has an equivalent on the stacks below. When an Apply It to Your Project section says “see the stack branches,” it means this list.

No Kubernetes. Chapters 1 through 5 do not require a cluster; apply them to your project as written. For the interlude and Chapters 6 through 13, either run a local disposable cluster with kind or k3d and treat it as your staging environment, or map each control to your actual runtime: a readiness gate becomes your load balancer health check, a namespace becomes your environment boundary, and an admission policy becomes whatever rejects a noncompliant deployment before traffic. The evidence rule is unchanged: prove the control rejects a bad input, not only that it accepts a good one.

GitLab CI, Jenkins, or another CI system. The `ci.yml` concepts map one to one: a job becomes a GitLab job or Jenkins stage, needs: becomes stage ordering, required status checks become protected-branch merge rules or a gated stage, and environment approvals become GitLab protected environments or a Jenkins input step with a named approver. Keep the chapter’s ordering rule: lint and test before build, build before scan, scan before any push, and a human approval before any production step.

Pulumi, CloudFormation, or OpenTofu instead of Terraform. Chapter 5’s sequence is format, validate, plan, review, and it transfers directly: `pulumi preview` or a change set is the plan artifact, and Checklist 5.1’s saved-plan review rows apply to it verbatim. The account assertion before a plan and the destroy-and-verify at the end are stack-independent and are the parts most worth carrying over.

No cloud account or no budget. Chapter 5 is a design-only plan review by intent; follow it as printed. Every other chapter runs against local tooling or a free-tier hosted repository, so the absence of a cloud account removes cost, not coverage.

A different AI assistant. The book never depends on a specific model or product. Any assistant that can read your prompt and return text fits every chapter, because the method evaluates the response with your tests and your review, not with trust in the tool.

G.4 The order of adoption

Apply the chapters to your project in book order the first time, because later chapters consume earlier chapters’ outputs in your repository exactly as they do in the reference repository: Chapter 4’s pipeline gates its own promotion step, Chapter 8’s telemetry feeds Chapter 10’s incident packet, and Chapter 10’s incident record is Chapter 11’s knowledge source. Where a chapter does not fit your environment yet, record the skip and its reason in `docs/adoption-map.md` rather than skipping silently, so the gap is a decision instead of an accident.

From AI-Native DevOps by Elvis Tafoh-Ngunjoh (Tafoh Publishing). Companion: <https://tafohpublishing.com/ai-native-devops>