

Running the Companion Chapter by Chapter

AI-Native DevOps - Companion Website and Repository

Elvis Tafoh-Ngunjoh

The reference implementation and its supporting material live outside the printed book rather than in printed appendices, so they stay runnable and can be updated without a new edition.

The companion website is the page to start from: <https://tafohpublishing.com/ai-native-devops>

The companion repository is what you clone and run: <https://github.com/netbiz-ai/ai-native-devops-companion>

This edition is pinned to companion release v1.0.0. Later patches publish corrections and errata; they do not silently redefine what this edition claimed.

Material	Companion location
Environment setup	README.md, .env.example, Makefile, plus workspace/, which you create in Chapter 1
Prompt library	devops-prompt-library/, which you create in Chapter 1
Command and tool reference	Makefile, scripts/, docs/supported-versions.md
Every printed command, as a file to run or paste	labs/<subject>/, one folder per subject, one numbered file per printed block
AI output verification	scripts/validate_repo.py, scripts/validate-offline.sh
Cloud cost and cleanup	optimization/, scripts/
Portfolio and interview material	docs/capstone/, labs/capstone/capstone-verify.sh

docs/subject-map.md is the contract between this book and that repository. It maps every subject to the paths it uses, what it starts from and ends with, and the acceptance boundary it has to meet. Where the book and the repository disagree about a path or a name, the map is correct and the book is the defect.

Lab folders, make targets and handoff tags are named for their **subject**, not for a chapter number: labs/gitops/, make gitops-start, and the gitops-start tag are the same thing. A chapter number describes where work sits in the book; a subject describes what the work is, so a printed path survives a reordering. The map's table gives the chapter for each subject.

The repository also carries instructions for AI coding agents, in AGENTS.md at its root, with a reader guide in docs/running-labs-with-a-coding-agent.md. A coding agent can run each lab's files in order and finish with its validation command, while destructive and cost-bearing steps still stop for your confirmation.

Running the companion chapter by chapter

The repository is cumulative: each chapter starts from the state the previous one ended in, and the chapter's own validation command is what tells you the state is real.

Prerequisites arrive in three tiers, and no tier is needed before the chapter that uses it. Chapters 1, 2, 11 and 12 need only Python 3.11 or newer, Bash and Git; Chapters 3 and 4 add a container runtime and a repository of your own; Chapters 6 to 10 and 13 add a disposable cluster, for which a local kind cluster is enough. The companion's README.md carries the setup commands for each tier, including the virtual environment and the `pip install -r reference-app/requirements.lock` that the offline validator checks for before it runs anything.

Chapter	Start from	Confirm with
1	An empty workspace/ you create	The chapter's own success and failure runs
2	reference-app/src, reference-app/tests	<code>cd reference-app && python3 -m unittest discover -s tests</code>
3	The pinned release tree	The chapter's docker build and label assertions
4	<code>make delivery-start</code>	<code>actionlint .github/workflows/*.yaml</code>
5	The pinned release tree	<code>terraform validate</code>
Interlude	labs/cluster/, scripts/lab-environment/	The interlude's own cluster and teardown checks
6	The pinned release tree	<code>kubectrl kustomize deployment/kubernetes/base</code>
7	<code>make gitops-start</code>	<code>kubectrl kustomize deployment/gitops/overlays/staging</code>
8	<code>make observability-start</code>	<code>make observability-validate</code>
9	<code>make security-start</code>	<code>./scripts/validate-offline.sh,</code> <code>labs/security/verify-security-ruleset.sh</code>
10	<code>make incident-start</code> , then <code>make capacity-start</code>	<code>make incident-validate</code> , <code>make capacity-validate</code>
11	The pinned release tree	<code>make assistant-validate</code>
12	The pinned release tree	<code>cd operations-agent && python3 -m unittest discover -s tests</code>
13	<code>make capstone-start</code>	<code>make capstone-validate</code>

Where the table says "the pinned release tree", that subject has no handoff tag cut yet, so `git switch --detach v1.0.0` is the binding starting state. A `make <subject>-start` target whose tag does not exist says so and exits rather than failing with a bare git error; treat that lab as the design and review exercise its execution status describes. The paired `make <subject>-complete` puts you on the finished state, and the difference between the two is the work.

Four rows carry a condition the table cannot hold. Chapter 4's gates only bind once your own repository has the staging and production environments, with branches restricted to main and a

required reviewer on production who is not the person dispatching the run. Chapter 5's apply path stays opt-in unless you supply and approve a sandbox of your own, and Chapter 6 expects you to apply the base to your own cluster after it renders. Chapter 8 has to start from its tag rather than from main, because `reference-app/src/telemetry.py` is the exercise and it is already present on main. Chapter 11 reads the incident record Chapter 10 produces, so it has no input until Chapter 10 has been run.

Three things the labs need cannot ship in the repository, because they are yours, and a chapter that appears not to work from Chapter 6 onward is usually one of them left unset: a disposable cluster in your `kubectl` context; a registry the cluster can pull from, with the image digest set in `deployment/kubernetes/base/kustomization.yaml`; and a Git source Argo CD can reach and you can push to, set in `deployment/gitops/argocd/`. The published digest is an all-zero placeholder that cannot pull, which is deliberate: an unreplaced value fails at admission rather than deploying something you never reviewed. For readers with no Git host the cluster can reach, `deployment/gitops/lab-source/` supplies a disposable in-cluster Git server and states its trade-offs.

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