

Lesson 10 — DevOps & Deployment

How code goes from a laptop to running for real users — CI/CD, containers, IaC.

IT Landscape

Module 3

The journey of one line of code

Type → commit → push → test → build → deploy → users.

- **20 years ago** — this took *weeks* of manual handoffs between teams.
 - **In 2026** — it can take **10 minutes**, end to end, fully automated.
 - The cluster of tools and habits that makes that possible is called **DevOps**.
-

What “DevOps” actually means

- Less a tool, more a **culture**.
- *“The people who write the code also own running it in production.”*
- Replaced the old **“developers throw code over a wall to ops”** model.
- The **toolchain** we’ll tour today is what grew around that culture.

SRE (Site Reliability Engineering, coined by Google) is the sibling discipline — for this lesson, treat them as the same job family.

Tool 1 — Version control as the trigger

- Every change starts with a `git push` (see Lesson 05).
- The remote — **GitHub, GitLab, Bitbucket** — **fires events** on every push and every pull request.
- Those events are what **kicks off everything else** in the pipeline.

No commit, no pipeline. The git history is the source of truth.

Tool 2 — CI (Continuous Integration)

Every change is automatically built and tested before it merges.

- On every push, an automated pipeline runs your **test suite**, lints the code, builds the artifacts.
- If anything fails, the change is **blocked**.

Popular CI hosts (as of 2026):

- **GitHub Actions** — dominant.
 - **GitLab CI**, **CircleCI**, **Buildkite**.
 - **Jenkins** — the old guard, still huge in enterprise.
-

What a CI run looks like

- A **green check** or a **red X** next to every pull request.
- Click in → see the log of every step: clone repo, install dependencies, compile, run tests.
- Each step shows its own duration; the whole run usually finishes in **5–20 minutes**.

The green check is the modern code review's best friend.

A real CI config (illustrative)

A minimal **GitHub Actions** workflow, lives at `.github/workflows/ci.yml`:

```
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
      - run: npm test
```

You don't need to run this — just notice it's plain text in your repo, versioned alongside the code.

Tool 3 — CD (Continuous Deployment / Delivery)

- Same machinery, **one step further**: after tests pass, automatically **deploy**.
- **Continuous Delivery** — every successful build is *ready* to deploy (human clicks the button).
- **Continuous Deployment** — every successful build *is* deployed, automatically.

The most mature teams (Amazon, GitHub, Shopify) deploy **thousands of times a day** — fully automated, with zero human pressing a button.

Tool 4 — Containers (Docker)

A container is your app plus everything it needs, sealed in a box that runs the same anywhere.

- Solved the **“works on my machine”** problem once and for all.
 - **Docker** is the brand most people learn first.
 - A container image is a frozen **filesystem + config + run-command**.
 - Often **<100 MB**, starts in **<1 second**, identical across laptop / CI runner / production server.
-

Tool 5 — Orchestrators (Kubernetes & friends, as of 2026)

- When you have **100 containers across 50 machines**, you need something to schedule them, restart them when they crash, and route traffic.
- **Kubernetes** (a.k.a. `k8s`) is the heavyweight standard.
- Lighter alternatives: **Nomad**, **AWS ECS**, **Fly.io**, **Render**, **Railway**, **Cloud Run** (Google).

Most teams in 2026 do NOT run Kubernetes themselves — they pay a cloud to run it for them (EKS / GKE / AKS).

Tool 6 — IaC (Infrastructure as Code)

Your servers, networks, and databases are declared in text files — not clicked into existence.

- **Terraform** (HashiCorp) / **OpenTofu** (the open fork) — cross-cloud standard.
- **Pulumi** — same idea, written in Python / TypeScript.
- **AWS CloudFormation** — the AWS-native one.
- **Ansible** — for configuring existing servers.

Your cloud setup becomes version-controlled, reviewed, and reproducible.

Tool 7 — Monitoring & observability

Once it's running, you need to know if it's **healthy**. Three telemetry families:

- **Logs** — lines of text. *Datadog, Loki, CloudWatch, Splunk.*
- **Metrics** — numbers over time. *Prometheus + Grafana, Datadog.*
- **Traces** — one request's path across services. *Jaeger, Honeycomb, Datadog APM.*

If a tree falls in production and nobody's monitoring — your users tell you on Twitter.

Recap

- The **pipeline**: commit → CI → container → orchestrator → monitoring.
- The **culture**: whoever writes it, owns it.
- The **goal**: deploy small changes, often, safely.

Coming up: Lesson 11 — keeping all of that from getting hacked.

