

The IT Map

Reading time: ~10 min · Prereqs: none

Why this matters to you

Over the next eleven lessons you'll meet IT one neighborhood at a time — the internet, the cloud, how software is built, how it runs, what AI is doing to all of it. Before any of that, you need a flyover. A map.

Most people who aren't in IT have a fuzzy picture: "there are programmers, and they make apps, and somehow there's a cloud." That's not wrong, but it's not enough to follow a real conversation at work, or to figure out which corner of this world you'd actually want to learn more about. By the end of this lesson you'll have the vocabulary to follow the rest of the course, and a mental map of where each topic fits. No code, no jargon without translation, no pretending it's simpler than it is.

1. IT as a city

The single best analogy for the modern IT landscape is a city. Imagine you walk into a busy downtown for the first time. You see **storefronts** — that's the *frontend*, the part the public touches. Behind each storefront sits a **warehouse** full of staff, inventory, and paperwork — that's the *backend*, where the real work of fulfilling a request happens. Goods move between buildings on **roads** — that's the *network*. Records live in **libraries** — *databases*. **Police and locks** keep all of it honest — *security*. **City services** — water, power, garbage — quietly run underneath — *DevOps*. And the whole city sits on a **utility grid** someone else owns and bills you for — the *cloud*.

Every real business runs on this whole city. But almost no engineer works on all of it. Most are specialists in one or two neighborhoods, with a passing familiarity with the others.

2. The seven neighborhoods

Here's the city, neighborhood by neighborhood. We'll spend lessons on the most important ones; this is your one-paragraph orientation to each.

Frontend is everything the user touches: web pages, mobile apps, the buttons and forms and animations. The languages of this neighborhood are HTML, CSS, and JavaScript (plus mobile-specific ones like Swift for iOS and Kotlin for Android). Frontend gets the most public attention because it's literally on screen — if a website looks ugly or glitches, everybody notices.

Backend and APIs are the brain behind the storefront. When you click "Buy", the frontend sends a request to a backend somewhere; the backend checks your account, talks to a payment service, updates the database, and sends back an answer. Backend engineers spend their days in languages like Python, Go, TypeScript, Java, and C#.

Databases and data are where state lives — the customers, the orders, the messages, the analytics. There are two big families: *SQL* databases like Postgres and MySQL (structured, predictable) and *NoSQL* like MongoDB and DynamoDB (flexible, scale-friendly). The cost of getting this layer wrong is usually the highest in the whole stack.

Networking and the internet is how anything talks to anything else: DNS, IP addresses, HTTPS, certificates, load balancers. Lesson 02 deep-dives this neighborhood — it's the one whose pieces are most likely to appear in the news as outages.

DevOps and cloud is how code goes from a laptop to running for real users at 3 a.m. on a Tuesday. Pipelines, containers, infrastructure as code, monitoring. Lessons 03 and 08 deep-dive this.

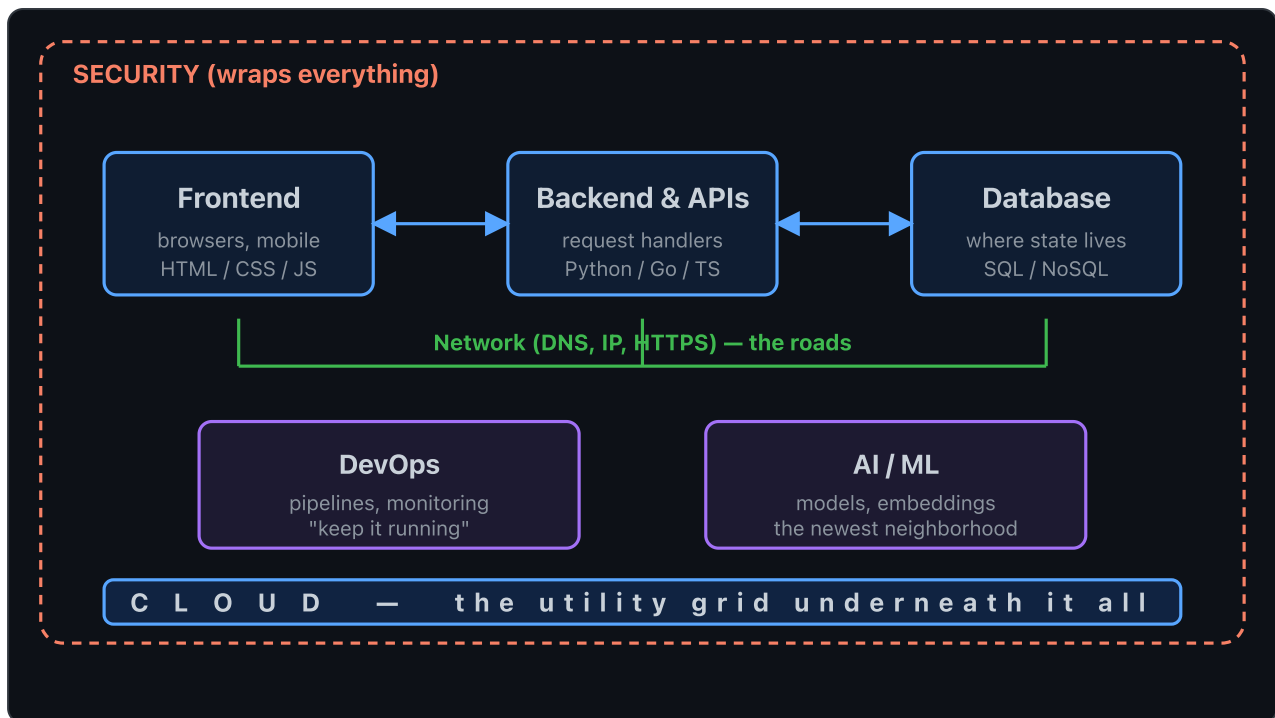
Security is the neighborhood whose job is to keep everyone else's mistakes from becoming front-page news. Cryptography, authentication, vulnerability scanning, incident response. Lesson 11.

AI and ML is the newest neighborhood and the one re-shaping every other one. Module 4 (lessons 10 and 11) covers it in detail.

3. The map, drawn

Here's the same idea as a picture. The seven neighborhoods sit inside a perimeter labelled "Security" (because security wraps around everything, not under it), with the

cloud as the substrate underneath:



4. Roles, briefly

Once you can see the city, you can place the people. **Software Engineer** is the generic term — it can mean any neighborhood, or several. **Frontend Engineer** lives in the storefront, **Backend Engineer** in the warehouse plus the library. **DevOps Engineer** or **SRE** (Site Reliability Engineer) keeps the city services running — pipelines, deployments, monitoring, on-call.

Data Engineer builds the plumbing that moves data into the libraries; **Data Scientist** reads from the libraries to find patterns; **ML Engineer** ships those patterns as actual services that run in production. **Security Engineer** watches the perimeter (and gets called when something gets through it). **Product Manager** talks to every neighborhood, builds none of it, and is responsible for deciding what gets built in what order. **UX Designer** shapes how the storefronts feel.

In real life the lines blur. A "backend engineer" at a 10-person startup may also write some frontend, set up the CI pipeline, and design the database schema. A "frontend engineer" at a 10,000-person company may spend half a year on a single performance optimisation. Lesson 14 revisits this in detail.

5. What “writing code” really looks like in 2026

Here's the part that surprises non-engineers most: *engineers do not spend most of their time typing new code*. A typical day looks more like this. You open the laptop, read three Slack messages, skim a pull request someone left for review overnight, leave a few comments. You pick up a bug ticket, spend forty minutes reading the existing code to figure out where the bug lives. You write maybe twenty lines to fix it. You spend another hour writing a test, running it, watching it fail, fixing the fix, watching it pass. You attend two meetings — one design discussion, one standup. You read someone else's design document and write a paragraph of feedback. You go home.

In 2026, AI assistants — Claude, Copilot, Cursor, Windsurf — write a large slice of the lines that actually get committed. The engineer's job has shifted toward *directing, reviewing, integrating, and owning* the result. The myth of the lone genius typing furiously into a black terminal is dead. The reality is closer to a senior editor at a newsroom, except the staff reporters are very fast and need careful supervision.

What people argue about

Three debates you'll hear in any IT room this year:

- **Will AI replace engineers?** Some say within five years; others say never. Reality so far: AI assistants are making existing engineers measurably faster but have not, in most teams, reduced headcount. Junior-engineer hiring is the part most under pressure.
- **Microservices vs monolith.** One camp argues every new system should be split into many small services from day one. The other camp argues you should start with one well-organised codebase and split only when scale forces you to. Lesson 06 returns to this.
- **Full-stack vs specialist.** The pendulum is swinging back toward full-stack — AI assistants close knowledge gaps, so one engineer can plausibly own frontend, backend, and infrastructure for a small product. The deep-specialist career is far from over, but the generalist is having a moment.

Reading on

- **The Pragmatic Programmer** — Hunt & Thomas (20th anniversary edition). The classic introduction to software as a craft. Reads well even if you never plan to write code.
- **Roadmap.sh** — community-maintained role-by-role learning roadmaps. A great "what does a Frontend / Backend / DevOps engineer actually study?" reference.
- **Lesson 14 of this course** — revisits roles and how to enter the field, with current 2026 hiring patterns.

LAST-REVIEWED: 2026-05-22