FREE GUIDE · BY QAIS

The DevOps Roadmap.

The exact order I'd learn DevOps in if I was starting from zero today. Nine tools, twelve weeks, one project that gets you hired.

Failed A-levels · dropped out of uni in my first term · now 22 and leading a DevOps team in Dubai · Kubestronaut

BEFORE YOU START

I'm not smarter than you.

I failed my A-levels. I started uni and dropped out in my first term. On paper, I was the last person who should be leading a DevOps team at 22.

What changed wasn't talent. It was having a clear order to learn things in, building real stuff instead of collecting tutorials, and having the right people around me when I got stuck.

That's what this roadmap is. It isn't a list of every tool in DevOps. It's the sequence that actually works, what "good enough" looks like at each stage, and the mistakes I see people make every single week.

Three rules before you start:

01
One stage at a time
Switching between five tools is why none of them stick.

02
Build, don't watch
If you can't rebuild it without the video, you haven't learned it yet.

03
Explain everything
Interviews don't test what you've done. They test whether you can explain it.

Read it once to see the route. Then come back to the stage you're on.

Qais
Lead DevOps Engineer · Head of Community & Growth, CoderCo

THE WHOLE PICTURE

The whole roadmap on one page.

Start at the top. Do them in order. Don't skip. That's literally it.

▼ START HERE

LEVEL 1 · LEARN HOW COMPUTERS WORK weeks 1-4

1. **Linux** wk 1-2
   The system every server runs on. Learn to drive it.
2. **Bash** wk 3
   Write tiny scripts so the computer does boring jobs for you.
3. **Git** wk 3
   Save every change and share code with a team.
4. **Networking** wk 4
   How computers find and talk to each other.

LEVEL 2 · LEARN THE TOOLS COMPANIES PAY FOR weeks 5-11

5. **Docker** wk 5
   Put your app in a box that runs the same everywhere.
6. **AWS** wk 6-7
   Rent computers on the internet to run it.
7. **Terraform** wk 8
   Build that cloud setup with code, not clicks.
8. **CI/CD** wk 9
   Push code, it tests and goes live by itself.
9. **Kubernetes** wk 10-11
   Run loads of those boxes and keep them alive.

LEVEL 3 · PROVE IT week 12

10. **Build one real project**
    Use all 9 tools on one small app.
11. **Show it off**
    Put it on GitHub with a simple write-up.
12. **Explain it out loud**
    Practise until you can talk anyone through it.

✓ GET HIRED

SUGGESTED PACE

The 12-week plan.

About 10 hours a week. Go slower if you need to. Don't go wider.

WEEK	FOCUS	YOU'RE DONE WHEN…
1-2	**Linux**	You can explain why a service won't start using its logs.
3	**Bash**	You've automated one task you were doing by hand.
3	**Git**	You've taken a change from a branch to a reviewed pull request.
4	**Networking**	You can sketch a web request from browser to server.
5	**Docker**	Your app runs as a container and keeps its data.
6-7	**AWS**	You can draw your app on AWS and explain who can access what.
8	**Terraform**	That AWS setup is now code, and you can read a plan.
9	**CI/CD**	A push to main builds, tests and deploys automatically.
10-11	**Kubernetes**	You can debug a pod that keeps restarting.
12	**Proof**	Write-up done, CV updated, explanation practised out loud.

REAL TALK

The weeks are a guide, not a race. People who rush weeks 1-4 always come back to them. The foundation is what makes Kubernetes make sense.

Stuck already? Book a free 1:1 call with the team **Book →**

01 · WEEKS 1-2

Linux and the command line

Every server you'll ever touch runs on this. Get comfortable here and everything later has somewhere to land.

- Find your way around the filesystem: where config, data and logs live
- Users, groups, permissions, and SSH into a remote machine
- Processes, services and logs: explain a failure before you fix it

MILESTONE

You can explain a permissions problem and use logs to say why a service won't start.

QAIS'S NOTE

Don't memorise commands. Before you run one, say out loud what you expect to happen.

FREE START

linuxjourney.com

02 · WEEK 3

Bash, and small useful scripts

Once you can do something by hand, Bash makes it repeatable. Keep scripts small and boring.

- Variables, arguments, environment, and why quoting matters
- Conditions, loops and functions, one job per script
- Exit codes and clear messages, so failure is obvious

MILESTONE

You can explain a short script that automates a real task, including what happens when a step fails.

QAIS'S NOTE

Automating something you don't understand just breaks it faster.

FREE START

Your own boring tasks. Automate one this week.

03 · WEEK 3

Git and the team workflow

Git isn't a place to upload finished files. It's how teams review and understand every change.

- Working files vs staged changes vs local history vs the shared repo
- Branches, focused commits, and pull requests that explain why
- Reading diffs, resolving conflicts, undoing safely

MILESTONE

You can take a small change from a branch to a reviewed PR, and explain what changed and how you checked it.

QAIS'S NOTE

Hiring managers read your commit history. "fix" ×40 tells them a lot.

FREE START

learngitbranching.js.org

04 · WEEK 4

Networking, and how things connect

Most "it's broken" problems are really "it can't reach it" problems. Networking is how you find out where.

- DNS, IPs and ports: finding something vs connecting to it
- TCP, TLS, HTTP, and status codes as evidence
- Subnets, routing, firewalls, load balancers

MILESTONE

You can sketch a web request and tell a DNS problem from a blocked port from an app error.

QAIS'S NOTE

Change one thing at a time. Changing five settings at once hides the cause.

FREE START

roadmap.sh/devops (networking section)

05 · WEEK 5

Docker, and the container idea

Package your app with everything it needs, so it runs the same everywhere.

- Images vs containers, and containers vs virtual machines
- Dockerfiles, layers and registries
- Config, networks and volumes, then Compose for multi-container apps

MILESTONE

Your app builds into an image, runs as a container, and keeps its data after the container dies.

QAIS'S NOTE

"The container started" doesn't mean "the app works". Check it properly.

FREE START

docs.docker.com, Get Started

06 · WEEKS 6-7

AWS, and your first cloud

One cloud, properly. You can pick up Azure or GCP later, because the ideas transfer.

- IAM first: users, roles, policies, least privilege
- EC2, S3 and VPC, then where databases and load balancers fit
- Logs, metrics and cost. Yes, cost. Set a billing alert on day one.

MILESTONE

You can draw your app on AWS and explain its access, network path, storage and cost.

QAIS'S NOTE

Don't memorise 200 service names. Know why each resource in your app exists.

FREE START

AWS Skill Builder (free tier courses)

07 · WEEK 8

Terraform, and infrastructure as code

Stop clicking around the console. Describe your infrastructure in files you can review and rebuild.

- → Providers, resources, variables, outputs, modules
- → Read the plan before you apply, especially deletions
- → State: what it is, why it's sensitive, how teams share it

MILESTONE

You can read a plan, spot what's being replaced, and explain why state needs protecting.

QAIS'S NOTE

Never apply a plan you haven't read. That's how prod gets deleted.

FREE START

developer.hashicorp.com/terraform/tutorials

08 · WEEK 9

CI/CD with GitHub Actions

A push to Git triggers checks, a build and a deploy, automatically and visibly.

- → Triggers, jobs and steps, in a sensible order
- → Continuous integration: fast feedback on every change
- → Secrets, permissions, approvals, and how to roll back

MILESTONE

You can walk through your pipeline from PR to deploy, including what happens when a stage fails.

QAIS'S NOTE

A green pipeline only proves the checks you wrote. Make the checks mean something.

FREE START

docs.github.com/actions

09 · WEEKS 10-11

Kubernetes, and running containers

You describe the state you want, and Kubernetes works to keep it there. It's the tool everyone wants on their CV and the one that exposes weak foundations fastest.

- → Pods, Deployments, nodes, replicas and rolling updates
- → Services, ConfigMaps, Secrets, storage
- → Status, events and logs: is it the app, scheduling, config or network?

MILESTONE

You can explain how traffic reaches your app in the cluster and where you'd look when a pod is unhealthy.

QAIS'S NOTE

I hold four Kubernetes certs. It made me feel stupid every day at first. That's normal. Push through.

FREE START

kubernetes.io/docs/tutorials

HOW IT ALL CONNECTS

Git	A change is recorded and reviewed
GitHub Actions	Checks run, delivery starts
Docker	The app is packaged as an image
AWS + Kubernetes	Infra hosts it, the cluster runs it
Terraform	Defines the infrastructure underneath it all

Stuck already? Book a free 1:1 call with the team Book →

PART 3 · PROOF IT WORKS

One project. Done properly.

One small app you deliver end to end beats ten tutorial clones. Make it pass these four tests:

01 REPEATABLE

Can you rebuild it?

Code and infrastructure in version control. Someone else could set it up.

02 EXPLAINABLE

Can you defend a choice?

Why this tool, and what trade-off did you make?

03 OBSERVABLE

Can you see a problem?

You know what healthy looks like and where to look when it isn't.

04 REVIEWABLE

Can someone follow it?

A README that explains the purpose, the architecture and the outcome.

THE 5 MISTAKES I SEE EVERY WEEK

1	**Tutorial hell.** 40 hours watched, zero things built. Close the video and rebuild it from memory.
2	**Learning 5 tools at once.** AWS, Python, K8s and Terraform in one week means none of them stick.
3	**Skipping the foundation.** Kubernetes before Linux and networking is pain.
4	**"Familiar with" on your CV.** Say what you ran, what broke and what you did.
5	**Doing it alone.** Nobody gets consistent alone. Get feedback while the work is still small.

Stuck already? Book a free 1:1 call with the team Book →

PROOF IT WORKS

Different starts.
Real offers.

200+ members hired, as reported by CoderCo on its programme research page.

"I was stuck in an IT Service Desk role for two years and felt lost. Since joining in May, I've secured two offers - Infrastructure and Systems Engineer at AWS. The AWS interview actually felt easier than the prep sessions."

EID ABOKOR · SYSTEMS ENGINEER AT AWS

"I'm happy to announce I received and accepted a job offer as a cloud support engineer after a long five-stage interview."

SARMAD ALI · CLOUD SUPPORT ENGINEER

AND ME

Failed A-levels. Dropped out of uni in my first term. Now 22, leading a DevOps team in Dubai. Nobody asked about my degree. They asked what I'd built.

Student accounts. Individual results vary.

YOUR NEXT STEP

You've got the route. Don't walk it alone.

This roadmap gives you the order. What it can't give you is someone checking your work, telling you what's missing, and getting you ready for the interview.

That's what CoderCo is: live sessions, project reviews, interview prep, and a community of people on exactly the same path. It's the room that got me consistent.

01
Where are you?
New to tech, already in IT, or stuck connecting the tools?

02
What's the gap?
Understanding, projects, feedback, or interview confidence?

03
What would help?
Structure, people, or someone to hold you to it?

Book your free 1:1 call

Tell us where you're starting, and we'll tell you exactly what your next step is.

Book my free call →

Qais