DEVOPS ROADMAP

120 concepts to become a DevOps engineer

8 areas, 15 concepts each. Every concept explained in one line, the order to study each area, and two practice questions to test yourself.

How to use it: one area at a time, tick each concept only when you can explain it out loud in under a minute, then answer the two practice questions without notes.

@chalkesahil · chalkesahil.ai

AREA 01 / 08

Linux

Study order: Live in the terminal first; everything else runs on Linux.

01	**Filesystem layout**	/etc, /var, /home, /usr and what lives where.
02	**Permissions**	chmod, chown, users, groups and the rwx bits.
03	**Processes**	ps, top, kill and how signals stop a program.
04	**systemd**	Start, stop and enable services; read their status.
05	**Logs**	journalctl and /var/log to see what went wrong.
06	**Package managers**	apt and dnf to install and update software.
07	**Text tools**	grep, sed, awk, cut and sort on log files.
08	**Pipes and redirects**	Chain commands with | and send output to files.
09	**Environment variables**	PATH, export and where config comes from.
10	**SSH**	Keys, ssh-agent and logging in to servers safely.
11	**Disk and memory**	df, du, free and finding what fills a disk.
12	**cron**	Run a job on a schedule.
13	**Networking commands**	ip, ss, ping, curl and dig on a real box.
14	**File editing**	Enough vim or nano to fix a config on a server.
15	**sudo**	Run as root only when you must.

Practice

1. A service will not start. Which three commands do you run first?
2. Find the 5 biggest files on a full disk.

AREA 02 / 08

Networking

Study order: Learn how a request travels before you debug one.

01	**IP addresses**	IPv4, private ranges and what a subnet is.
02	**CIDR**	Read 10.0.0.0/16 and know how many hosts fit.
03	**Ports**	What 22, 80, 443 and 5432 usually are.
04	**TCP vs UDP**	Reliable connections versus fast datagrams.
05	**DNS**	How a name turns into an IP, and A, CNAME, TXT records.
06	**HTTP**	Methods, status codes and headers.
07	**HTTPS and TLS**	Certificates and why the padlock matters.
08	**Load balancers**	Spread traffic across many servers.
09	**Reverse proxy**	Nginx in front of your app.
10	**Firewalls**	Allow only the traffic you expect.
11	**NAT**	How private machines reach the internet.
12	**VPN**	Private access into a network.
13	**Latency**	Why distance and round trips slow requests.
14	**curl**	Test any endpoint from the terminal.
15	**Troubleshooting order**	DNS, then port, then TLS, then the app.

Practice

1. A site loads for you but not for a user. Walk through what you check.
2. Explain what happens when you type a URL and press Enter.

AREA 03 / 08

Git and scripting

Study order: Automate what you do twice; version everything.

01	**Commits**	Small, clear commits with good messages.
02	**Branches**	Work apart, then merge.
03	**Merge vs rebase**	Two ways to bring changes together.
04	**Pull requests**	Review before code reaches main.
05	**Resolving conflicts**	Read both sides and pick what is right.
06	**.gitignore**	Keep secrets and build files out of the repo.
07	**Tags and releases**	Mark the exact version that shipped.
08	**Bash scripts**	Variables, if, loops and exit codes.
09	**set -euo pipefail**	Make scripts stop on the first error.
10	**Python for automation**	Scripts that call APIs and parse JSON.
11	**YAML**	The format most DevOps tools are configured in.
12	**JSON and jq**	Read API output from the terminal.
13	**Makefiles**	One command to build, test and deploy.
14	**Secrets in code**	Never commit keys; use environment variables.
15	**Branching strategy**	Trunk-based versus long-lived branches.

Practice

1. Write a script that backs up a folder and deletes backups older than 7 days.
2. You committed a password. What do you do?

AREA 04 / 08

Docker

Study order: Package the app once so it runs the same everywhere.

01	**Images vs containers**	The template versus the running copy.
02	**Dockerfile**	FROM, RUN, COPY, CMD and what each does.
03	**Layers and caching**	Order instructions so builds stay fast.
04	**Multi-stage builds**	Build big, ship small.
05	**Ports and publishing**	Map a container port to the host.
06	**Volumes**	Keep data when the container is replaced.
07	**Environment config**	Pass settings in, do not bake them in.
08	**Docker networks**	Let containers talk by name.
09	**Docker Compose**	Run an app and its database together.
10	**Registries**	Push and pull images from Docker Hub or ECR.
11	**Image tags**	Never deploy latest; pin a version.
12	**Small base images**	Alpine or distroless for fewer bugs.
13	**Logs and exec**	docker logs and docker exec to debug.
14	**Health checks**	Tell Docker when the app is really ready.
15	**Non-root containers**	Run as a normal user for safety.

Practice

1. Containerise a small web app with its database using Compose.
2. Your image is 1.2 GB. How do you shrink it?

AREA 05 / 08

CI/CD

Study order: Every push should build, test and be ready to ship.

01	**Continuous integration**	Build and test on every push.
02	**Continuous delivery**	Every passing build can go to production.
03	**Pipelines**	Stages that run in order.
04	**GitHub Actions**	Workflows, jobs, steps and runners.
05	**Triggers**	Run on push, pull request or a schedule.
06	**Build artifacts**	Save what a stage produced for the next one.
07	**Caching dependencies**	Make pipelines faster.
08	**Secrets in CI**	Store keys in the CI's secret store.
09	**Automated tests**	Unit tests first, then integration tests.
10	**Linting**	Catch mistakes before review.
11	**Image builds in CI**	Build and push a Docker image per commit.
12	**Environments**	Dev, staging and production with approvals.
13	**Blue-green deploys**	Switch traffic to a new copy at once.
14	**Canary releases**	Send a little traffic to the new version first.
15	**Rollbacks**	Go back fast when a release breaks.

Practice

1. Build a pipeline that tests, builds an image and deploys to staging.
2. A deploy broke production. How should rollback work?

AREA 06 / 08

Kubernetes

Study order: Learn the core objects before Helm and operators.

01	**Pods**	The smallest thing Kubernetes runs.
02	**Deployments**	Keep N copies of a pod running.
03	**ReplicaSets**	What a Deployment uses under the hood.
04	**Services**	A stable address for a set of pods.
05	**Ingress**	Route outside traffic to services.
06	**ConfigMaps**	Config kept outside the image.
07	**Secrets**	Sensitive config, handled with care.
08	**Namespaces**	Split a cluster between teams or apps.
09	**kubectl**	get, describe, logs, apply and rollout.
10	**Probes**	Liveness and readiness checks.
11	**Resource requests and limits**	CPU and memory per container.
12	**Rolling updates**	Replace pods a few at a time.
13	**Persistent volumes**	Storage that outlives a pod.
14	**Helm**	Package and template Kubernetes apps.
15	**Autoscaling**	Add pods when load goes up.

Practice

1. A pod is in CrashLoopBackOff. Walk through how you debug it.
2. Deploy an app with 3 replicas behind a Service and an Ingress.

AREA 07 / 08

Cloud and Terraform

Study order: Pick one cloud, learn it well, then write it as code.

01	**Regions and zones**	Where your servers physically run.
02	**IAM**	Users, roles and least privilege.
03	**Compute**	Virtual machines like EC2.
04	**Object storage**	S3 for files and backups.
05	**VPC**	Your private network in the cloud.
06	**Security groups**	Firewall rules for cloud servers.
07	**Managed databases**	RDS instead of running your own.
08	**Managed Kubernetes**	EKS, GKE or AKS.
09	**Cost basics**	What costs money and how to spot waste.
10	**Infrastructure as code**	Create cloud resources from files, not clicks.
11	**Terraform providers**	Plugins that talk to each cloud.
12	**plan and apply**	See the change, then make it.
13	**Terraform state**	How Terraform remembers what it built.
14	**Modules**	Reuse the same setup across projects.
15	**Remote state and locking**	Let a team share state safely.

Practice

1. Write Terraform for a VPC, a server and a security group.
2. Your cloud bill doubled this month. How do you find out why?

AREA 08 / 08

Monitoring and SRE

Study order: If you cannot see it, you cannot fix it.

01	**Metrics**	Numbers over time: CPU, requests, errors.
02	**Logs**	Events with detail, searched when things break.
03	**Traces**	Follow one request across services.
04	**Prometheus**	Collect and query metrics.
05	**Grafana**	Dashboards for those metrics.
06	**Alerts**	Page a human only for real problems.
07	**The four golden signals**	Latency, traffic, errors and saturation.
08	**SLIs**	What you measure about user experience.
09	**SLOs**	The target for that measure.
10	**Error budgets**	How much failure you can afford.
11	**Incident response**	Who does what when production breaks.
12	**On-call**	Rotations, runbooks and handovers.
13	**Postmortems**	Learn from an outage without blame.
14	**Backups and restores**	A backup counts only if a restore works.
15	**Capacity planning**	Grow before you run out.

Practice

1. Define SLIs and an SLO for a login API.
2. Your alert fires at 3 AM for a full disk. What do you do now and tomorrow?