CodeKerdos

PROGRAM CURRICULUM

SRE + Platform Engineering Program

From production reliability to building and running an Internal Developer Platform

16	4	4	1
weekends	months	phases	capstone platform

PROGRAM STRUCTURE

Phase 1 · SRE Foundations	Weeks 1–7
Phase 2 · Advanced SRE	Weeks 8–10
Phase 3 · Platform Engineering	Weeks 11–13
Phase 4 · Engineering Depth & Career Readiness	Weeks 14–16
Track · System Design	Weeks 8–16, every weekend

Version 1.0 · October 2026

Hands-on labs every weekend · No cloud account required

PROGRAM OVERVIEW

Four months, one platform, built week by week

A 16-weekend, hands-on program that takes engineers from SRE fundamentals to designing, building and running an Internal Developer Platform. Every week builds on the same Kubernetes cluster, application and Git repository, so by the final weekend each participant has a working platform, a portfolio of code and design documents, and practice in every round of a senior SRE or platform engineering interview.

DURATION	16 weekends (4 months)
FORMAT	Live weekend sessions on Saturday and Sunday with guided labs, plus self-practice during the week
STRUCTURE	Phase 1 SRE Foundations (Weeks 1–7) · Phase 2 Advanced SRE (Weeks 8–10) · Phase 3 Platform Engineering (Weeks 11–13) · Phase 4 Engineering Depth & Career Readiness (Weeks 14–16)
SYSTEM DESIGN	A dedicated track from Week 8 to Week 16: a design topic and a practice problem every weekend, a full design week in Week 15 and a mock design round in Week 16
LABS	Hands-on every weekend, run locally on a Kubernetes cluster. No cloud account required.
CAPSTONE	Build and demo a mini Internal Developer Platform that survives a seeded production failure
ASSESSMENT	Weekly lab deliverables (60%) and capstone (40%)

Who this program is for

- Engineers in operations, DevOps, support or backend roles moving into SRE or platform engineering
- Working SRE and DevOps engineers who want depth in Kubernetes, observability, infrastructure as code and platform design
- Engineers preparing for senior SRE and platform engineering interviews

What you will be able to do

- Define SLIs, SLOs and error budgets, and turn them into alerts that page only when users are affected
- Debug production issues across Linux, networking, Kubernetes and application telemetry
- Ship changes safely with canary releases, GitOps and tested rollback
- Plan capacity, control cost, and design for disaster recovery with measured RTO and RPO
- Build an Internal Developer Platform with Terraform, Argo CD, Backstage and policy as code
- Write production automation in Python and Go, and design large-scale systems with real capacity estimates
- Perform with confidence across troubleshooting, system design, coding and behavioural interview rounds

HOW THE PROGRAM WORKS

Every weekend follows the same rhythm

Concepts on Saturday morning, hands-on work on Saturday afternoon, and Sunday to finish, review and practise. Each week ends with a deliverable committed to your own repository.

SATURDAY · SESSION 1	Concepts, architecture and real production examples
SATURDAY · SESSION 2	Guided hands-on lab on the shared lab environment
SUNDAY	Lab completion, deliverable review and, from Week 8, a 30-minute interview drill on the week's topic
DURING THE WEEK	Self-practice and the weekly deliverable (recommended 4–5 hours)

One environment, one repository

The program uses a single production-style payments application throughout. Phase 1 runs it with Prometheus, Alertmanager and Kubernetes; Phase 2 makes it observable, scalable and recoverable; Phase 3 wraps it in a self-service platform; Phase 4 adds automation and design depth. Nothing is thrown away between weeks, so the capstone is assembled from work already done rather than started from scratch.

Interview readiness, built in

Senior SRE and platform interviews typically include four kinds of rounds. Each one is practised repeatedly across the program, not only in the final week.

ROUND	WHAT IT TESTS	WHERE IT IS PRACTISED
Troubleshooting	Structured debugging under pressure on live systems	Weeks 3, 5 and 7 drills; live troubleshooting round in Week 15
System design	Architecture, capacity estimates, failure modes and trade-offs	System design track: a problem every weekend from Week 8, design week in Week 15, mock round in Week 16
Coding for SREs	Practical scripting, automation and concurrency	Week 14 labs and timed practice; automation in the capstone
Behavioural and leadership	Ownership, incident leadership, influence and communication	Week 7 interview intensive; story bank and mock loop in Week 16

PROGRAM ROADMAP

16 weeks at a glance

WEEK	PHASE	THEME	HEADLINE LAB
1	Foundations	**Foundations: SLIs, SLOs, error budgets, toil**	SLO worksheet for a production-style service
2	Foundations	**Prometheus and observability**	PromQL and dashboards on the payments app
3	Foundations	**Linux, networking, logs and traces**	CPU, memory, disk and log incident debugging
4	Foundations	**Incidents and alerting**	Alertmanager routing, alert console, burn-rate alerts
5	Foundations	**Kubernetes for SREs**	Three-node kind cluster, 9 labs, blind incident drill
6	Foundations	**Shipping safely**	Argo Rollouts canary and blue/green with Prometheus analysis
7	Foundations	**Chaos and interviews**	Three-round game day; interview intensive
8	Advanced SRE	**Observability at scale with OpenTelemetry**	OTel pipeline to Prometheus, Loki and Tempo; SLOs as code
9	Advanced SRE	**Capacity, performance and cost**	k6 load test to saturation, KEDA, cost per service
10	Advanced SRE	**Resilience, disaster recovery and incident leadership**	Velero restore drill, Postgres failover, Chaos Mesh
11	Platform	**Terraform deep dive**	LocalStack infrastructure, drift, plan-on-PR pipeline
12	Platform	**Platform foundations, GitOps and platform APIs**	Cluster bootstrapped from Git; Crossplane claim
13	Platform	**Developer portal, golden paths and security as code**	One-click new service; signed images enforced
14	Career Ready	**Automation and coding for SREs**	Python/Go health checker; remediation controller
15	Career Ready	**System design deep dive**	Capacity estimates, design documents, practice rounds
16	Career Ready	**Capstone and interview readiness**	Capstone demo and full mock interview loop

Weeks 1–7 form a complete SRE foundation on their own. Weeks 8–16 build directly on the cluster, application and repository from those weeks, with the system design track running alongside every weekend.

SYSTEM DESIGN TRACK

System design, every weekend from Week 8

System design is taught as a continuous track, not a single session. Each weekend from Week 8 introduces one design topic tied to that week's technology and ends with a 30-minute practice problem. Week 15 brings it together in a full design week, and Week 16 closes with a timed mock design round.

A repeatable design framework

Every practice problem is answered with the same eight steps, so the approach becomes habit.

1	**Requirements and scale**	Clarify users, features, traffic and constraints before drawing anything
2	**SLOs first**	Agree availability, latency and durability targets; they drive every later choice
3	**Capacity estimates**	Requests per second, storage, bandwidth and machine count, with real numbers
4	**High-level architecture**	Components, data flow and APIs; the simplest design that meets the SLOs
5	**Data model and storage**	Access patterns, database choice, replication, partitioning and consistency
6	**Failure modes**	What breaks, the blast radius, and how the design degrades and recovers
7	**Observability**	The signals, SLO alerts and dashboards that prove the design works in production
8	**Rollout and cost**	Migration plan, safe release, and the cost of running it at scale

Building blocks covered

TRAFFIC	L4 and L7 load balancing, DNS, CDNs, API gateways, service discovery
DATA	SQL versus NoSQL, indexing, replication, sharding, hot keys, schema migrations
CONSISTENCY	Strong and eventual consistency, quorums, consensus and leader election (Raft)
PERFORMANCE	Caching strategies and invalidation, connection pooling, read and write paths
ASYNC PROCESSING	Queues and streams, backpressure, retries, idempotency, delivery guarantees
PROTECTION	Rate limiting, load shedding, timeouts, circuit breakers, graceful degradation
SCALE AND RESILIENCE	Autoscaling, multi-region active-active and active-passive, disaster recovery
PLATFORM	Multi-tenancy, control planes, CI/CD at scale, telemetry pipelines

SYSTEM DESIGN TRACK

Week-by-week design topics and problems

Each problem uses the system participants have just built that week, so the design discussion is grounded in something they have run, broken and fixed.

WEEK	DESIGN TOPIC	PRACTICE PROBLEM
8	Telemetry pipelines: sampling, cardinality, storage tiers and retention	**Design observability for 200 microservices on a fixed budget**
9	Capacity estimation, autoscaling and load shedding	**Prepare a payments service for a 10x peak sale event in six weeks**
10	Replication, failover, multi-region design, RTO and RPO	**Your primary region is down: design the failover and walk through the first 30 minutes**
11	Infrastructure change safety: state, drift, blast radius and review	**Design infrastructure change management for 50 teams sharing one cloud estate**
12	Multi-tenancy, isolation and platform control planes	**Design a multi-tenant Kubernetes platform for 40 teams**
13	Software supply chain, identity, secrets and policy enforcement	**Design a secure delivery pipeline that produces audit evidence automatically**
14	Rate limiting, queues, retries and idempotency in code	**Implement a sliding-window rate limiter; design a distributed job scheduler**
15	Full framework, distributed systems fundamentals, non-abstract capacity design	**Global rate limiter, notification system, log ingestion at one million events per second, CI/CD platform for 500 engineers**
16	Timed mock round with written feedback	**Design an Internal Developer Platform**

How design answers are scored

Practice rounds and the Week 16 mock use the same rubric, so progress is visible from week to week.

DIMENSION	WHAT A STRONG ANSWER SHOWS
Requirements	Asks the right questions and states assumptions before designing
Estimation	Produces capacity numbers quickly and uses them to justify choices
Architecture	A clear, simple design that meets the SLOs, with sensible components and APIs
Reliability	Identifies failure modes, blast radius, degradation and recovery
Trade-offs	Compares alternatives and explains why one was chosen
Communication	Structured, calm and easy to follow; adapts when the interviewer pushes back

Track deliverables: a short design note each week from Week 8, two full design documents in Week 15, and a scored mock design round in Week 16.

PHASE 1 · WEEKS 1–7

SRE Foundations

The core of site reliability engineering: measure reliability, observe systems, debug Linux and networks, run incidents, operate Kubernetes and ship changes safely.

WEEK	THEME	WHAT YOU LEARN	HANDS-ON LAB
1	**Foundations**	SLIs, SLOs, error budgets and toil. How reliability targets are chosen and how they drive engineering decisions.	SLO worksheet: SLIs, targets and an error budget for a production-style service
2	**Prometheus and observability**	Metrics and PromQL, the golden signals, labels and cardinality, and dashboards that people actually use.	PromQL queries and dashboards on the payments application
3	**Linux, networking, logs and traces**	cgroups, processes and memory, packets and connections, log analysis with grep, awk and sed, and distributed traces.	Debug CPU, memory, disk and log-driven incidents on the payments application
4	**Incidents and alerting**	Alertmanager routing, inhibition and silences, SLO burn-rate alerting, page versus ticket, incident roles and blameless postmortems.	Alerting labs with an alert console and multi-window burn-rate alerts
5	**Kubernetes for SREs**	Pod lifecycle, probes, requests and limits, rollouts, PodDisruptionBudgets, HPA, and Service and DNS debugging.	Three-node kind cluster with Prometheus and Grafana, nine labs and a blind incident drill
6	**Shipping safely**	Deployment strategies, canary and blue/green releases, GitOps, progressive delivery, and rollback that works, including the database.	Argo Rollouts canary and blue/green gated by Prometheus analysis
7	**Chaos and interviews**	Chaos experiments with a hypothesis, game day design with incident roles, and interview preparation for SRE roles.	Three-round game day; interview intensive with live mock and resume clinic

PHASE 1 OUTCOME

You can define and defend SLOs, build alerting that pages on user impact, debug a Kubernetes service under pressure and release changes with a tested way back.

PHASE 2 · WEEKS 8–10

Advanced SRE

Take Phase 1 skills from one service to many: telemetry pipelines at scale, capacity and cost, and surviving data loss and regional failure.

WEEK 8 — Observability at scale with OpenTelemetry

Builds on Weeks 2–4

SATURDAY · SESSION 1	The OpenTelemetry model: traces, metrics, logs and context propagation. Collector pipelines, head versus tail sampling, Grafana Alloy, Loki and Tempo. Cardinality budgets and the cost of telemetry.
SATURDAY · SESSION 2	**Lab:** instrument the payments application with the OpenTelemetry SDK, route telemetry through a Collector to Prometheus, Loki and Tempo, and move from a slow trace straight to its logs.
SUNDAY	SLOs as code with Sloth and OpenSLO: generate burn-rate alerts from YAML and compare them with the hand-written Week 4 rules. Overview of eBPF-based observability.
DELIVERABLE	**An slo/ folder in your repository that generates alerts for two services**
INTERVIEW DRILL	“Design observability for 200 microservices on a fixed budget.”

WEEK 9 — Capacity, performance and cost

Builds on Weeks 5 and 8

SATURDAY · SESSION 1	Capacity planning: headroom, saturation and Little’s Law. Load test types: smoke, load, stress and soak. HPA versus VPA versus KEDA. Node autoscaling concepts with Cluster Autoscaler and Karpenter.
SATURDAY · SESSION 2	**Lab:** load-test the payments application to saturation with k6, find the bottleneck from your Week 8 dashboards, and tune requests, limits and HPA.
SUNDAY	Event-driven scaling with KEDA. FinOps fundamentals with OpenCost: cost per namespace and per request, and rightsizing.
DELIVERABLE	**A one-page capacity plan and a cost-per-service report**
INTERVIEW DRILL	“A peak sale event is six weeks away. How do you prepare?”

WEEK 10	**Resilience, disaster recovery and incident leadership** Builds on Weeks 4, 6 and 7
SATURDAY · SESSION 1	RTO and RPO design, backup versus replication, and regional failover patterns. Stateful reliability: replication lag, failover and safe schema migrations.
SATURDAY · SESSION 2	**Lab:** back up to in-cluster MinIO with Velero, delete a namespace, restore it and measure recovery time. Fail over a CloudNativePG primary under load.
SUNDAY	Chaos engineering as a practice: Chaos Mesh experiments with steady-state hypotheses. Incident leadership at scale: commander handoffs, stakeholder communication, status pages and on-call health. Production Readiness Reviews.
DELIVERABLE	**A disaster recovery runbook with measured RTO and RPO, and a Production Readiness Review for the payments service**
INTERVIEW DRILL	“Your primary region is down. Walk through the first 30 minutes.”

PHASE 2 OUTCOME

You can run telemetry pipelines across many services, plan capacity and cost with evidence, and recover from data loss and failover with measured RTO and RPO.

PHASE 3 · WEEKS 11–13

Platform Engineering

Turn everything run by hand into a self-service platform: infrastructure as code, GitOps, a developer portal and security built into the golden path.

WEEK 11 — Terraform deep dive

Builds on Week 5

SATURDAY · SESSION 1	Terraform for SREs: HCL, providers, resources and data sources, variables, outputs and locals, count versus for_each, and lifecycle rules. State management: remote backends, locking, import, and moved and removed blocks. Terraform and OpenTofu.
SATURDAY · SESSION 2	**Lab:** build AWS-style infrastructure on LocalStack (network, storage, IAM and a lock table) with remote state. Make a manual change, detect the drift and reconcile it.
SUNDAY	Module design, versioning and composition. Testing with terraform test and Terratest. Policy checks with Checkov. Plan on pull request, apply on merge. Rebuild the Week 5 cluster and monitoring stack with the kind and Helm providers.
DELIVERABLE	**A versioned module repository and a CI pipeline that posts the plan to the pull request and blocks on policy failures**
INTERVIEW DRILL	"Design infrastructure change management for 50 teams sharing one cloud estate."

WEEK 12 — Platform foundations, GitOps and platform APIs

Builds on Weeks 6 and 11

SATURDAY · SESSION 1	What platform engineering is: platform as a product, Team Topologies, Internal Developer Platform reference architecture, golden paths versus guardrails, and DORA and adoption metrics. GitOps at scale: Argo CD app-of-apps, ApplicationSets, Helm versus Kustomize, and environment promotion.
SATURDAY · SESSION 2	**Lab:** Terraform creates the cluster and installs Argo CD; Argo CD manages everything else from Git. Promote changes from dev to stage to prod by pull request, with no manual kubectl.
SUNDAY	Multi-tenancy with namespaces, RBAC, quotas and network policies. CRDs, controllers and operators. Crossplane compositions and claims, and when to choose them over Terraform modules. Kubernetes upgrade strategy.
DELIVERABLE	**A platform charter, a GitOps-managed cluster and a Crossplane claim that developers can request**
INTERVIEW DRILL	"Design a multi-tenant Kubernetes platform for 40 teams."

WEEK 13	**Developer portal, golden paths and security as code** Builds on Weeks 8 and 12
SATURDAY · SESSION 1	Backstage: software catalog, scaffolder templates, TechDocs and plugins. Measuring developer experience.
SATURDAY · SESSION 2	**Lab:** build a golden-path template. One click creates the repository, CI pipeline, Argo CD application, SLOs and dashboards for a new service.
SUNDAY	Software supply chain: SBOMs with Syft, image signing with cosign, and SLSA levels. Kyverno admission policies. Secrets with Vault and External Secrets. Certificate automation with cert-manager. Mapping controls to frameworks such as HIPAA and PCI DSS.
DELIVERABLE	**A working "create a service" flow, with unsigned images blocked at admission**
INTERVIEW DRILL	"Design a secure delivery pipeline that produces audit evidence automatically."

PHASE 3 OUTCOME

You can provision infrastructure through reviewed code, run a multi-tenant GitOps platform, and give developers a secure golden path from template to production.

PHASE 4 · WEEKS 14–16

Engineering Depth & Career Readiness

Add the engineering depth senior roles expect: production automation in Python and Go, large-scale system design, and a capstone presented like a real design review.

WEEK 14 — Automation and coding for SREs

Builds on Weeks 3 and 12

SATURDAY · SESSION 1	Python for SRE tooling: parsing logs and metrics, API clients with timeouts, retries and backoff, concurrency, command-line tools and tests. Go fundamentals for infrastructure tools: goroutines, channels and context. Common SRE coding patterns.
SATURDAY · SESSION 2	**Lab:** build a concurrent health-checker CLI that polls endpoints with timeouts and retries, exposes Prometheus metrics and ships with tests.
SUNDAY	Kubernetes automation: client libraries and a small controller that remediates a known failure pattern, with safety guardrails. Runbook automation and AI-assisted operations: where they help and where they do not. Timed practice: log parsing, sliding-window rate limiter, LRU cache and maintenance-window scheduling.
DELIVERABLE	**A health-checker CLI and a remediation controller, both with tests, in your repository**
INTERVIEW DRILL	45-minute practical coding round

WEEK 15 — System design deep dive

Builds on all previous weeks

SATURDAY · SESSION 1	A repeatable design framework: requirements, SLOs, capacity estimates, architecture, data, failure modes, observability, rollout and cost. Distributed systems fundamentals: replication, partitioning, consistency models, consensus, caching, queues and backpressure, idempotency, rate limiting, and active-active versus active-passive multi-region.
SATURDAY · SESSION 2	Non-abstract design: capacity estimates with real numbers for requests per second, storage, bandwidth and machine count. **Worked example:** a log ingestion pipeline at one million events per second.
SUNDAY	Paired practice rounds: global rate limiter, notification system with SLOs, metrics platform, and a CI/CD platform for 500 engineers. Live troubleshooting round on the lab cluster. Peer review against a scoring rubric.
DELIVERABLE	**Two written design documents with capacity estimates**
INTERVIEW DRILL	Full 60-minute system design round

WEEK 16	**Capstone and interview readiness** Brings the full program together
SATURDAY · SESSION 1	Capstone build window with mentor reviews. Running the platform as a product: adoption metrics, platform SLOs, and presenting impact to leadership.
SATURDAY · SESSION 2	Behavioural and leadership rounds: a story bank drawn from incidents and the capstone, covering ownership, influence and conflict. Resume, GitHub portfolio and design documents as evidence of impact.
SUNDAY	Capstone demos (20 minutes plus 10 minutes of questions). Full mock interview loop: troubleshooting, system design, coding and behavioural. Graduation.
DELIVERABLE	**Capstone demo and mock interview scorecard**
INTERVIEW DRILL	Complete mock loop with written feedback

PHASE 4 OUTCOME

You can write and test production automation, design large-scale systems with real numbers, and present your platform and your experience across a full interview loop.

CAPSTONE PROJECT

Build a platform, then break it on purpose

Each participant, or pair, builds and demos a mini Internal Developer Platform for a fictional healthcare or payments company. A new service goes from a portal click to production, then survives a seeded failure.

Minimum scope

1	Infrastructure provisioned from Terraform modules through a plan-on-PR pipeline	Week 11
2	New service created from the Backstage golden-path template	Week 13
3	Deployed by Argo CD with a signed image and admission policies enforced	Weeks 12–13
4	OpenTelemetry telemetry, SLOs and burn-rate alerts generated automatically	Week 8
5	Load-tested, with a short capacity and cost note	Week 9
6	Seeded failure handled through alert, rollback or restore, with a blameless postmortem	Weeks 6, 10
7	One automated remediation and a written design document	Weeks 14–15

Evaluation rubric

CRITERION	WEIGHT	WHAT EXCELLENT LOOKS LIKE
Golden path works end to end	20%	A new service is live in under 15 minutes with no manual steps
Reliability design	20%	SLOs, alerts and error-budget policy are coherent and justified
Security and compliance	15%	Supply chain, secrets and policies are mapped to a named standard
Operability under failure	20%	The failure is handled calmly and the postmortem has real action items
Automation	10%	Remediation is safe, tested and protected by guardrails
Communication	15%	Architecture, trade-offs and impact are explained clearly to technical and non-technical audiences

Program assessment combines the weekly lab deliverables (60%) with the capstone (40%).

TOOLS AND ENVIRONMENT

What you will work with

OBSERVABILITY	Prometheus, Grafana, Alertmanager, Loki, Tempo, OpenTelemetry, Grafana Alloy, Sloth
KUBERNETES AND DELIVERY	Kubernetes (kind), Helm, Kustomize, Argo CD, Argo Rollouts, KEDA
RELIABILITY AND RESILIENCE	k6, Chaos Mesh, Velero, MinIO, CloudNativePG, OpenCost
INFRASTRUCTURE AS CODE	Terraform, OpenTofu, LocalStack, Terratest, Checkov, Crossplane
PLATFORM AND SECURITY	Backstage, Kyverno, cosign, Syft, HashiCorp Vault, External Secrets, cert-manager
LANGUAGES AND TOOLING	Linux, Bash, Python, Go, Git, GitHub Actions, Docker

Prerequisites

- Comfortable with the Linux command line and basic networking concepts
- Working knowledge of Git and Docker
- Some scripting experience in Bash or Python. Go is taught from the basics in Week 14.

Lab environment

LAPTOP	8 GB RAM minimum (16 GB recommended), 4 or more CPU cores, 50 GB free disk space (SSD recommended)
8 GB SETUP	Every lab is designed to run on an 8 GB laptop: a single-node cluster, one tool stack at a time, and pre-built images for heavier tools. A tuned memory configuration is provided in the setup guide. The full capstone can optionally run in a cloud environment such as GitHub Codespaces.
OPERATING SYSTEM	Windows with WSL2 and Docker Desktop, macOS or Linux
LOCAL TOOLS	Docker, kind, kubectl, Helm, Terraform and k6. A setup guide is shared before Week 1, with pre-flight checks before Weeks 5 and 8.
CLOUD ACCOUNT	Not required. All labs run locally; AWS-style labs use LocalStack.
REPOSITORY	Each participant maintains one Git repository that grows week by week into the capstone