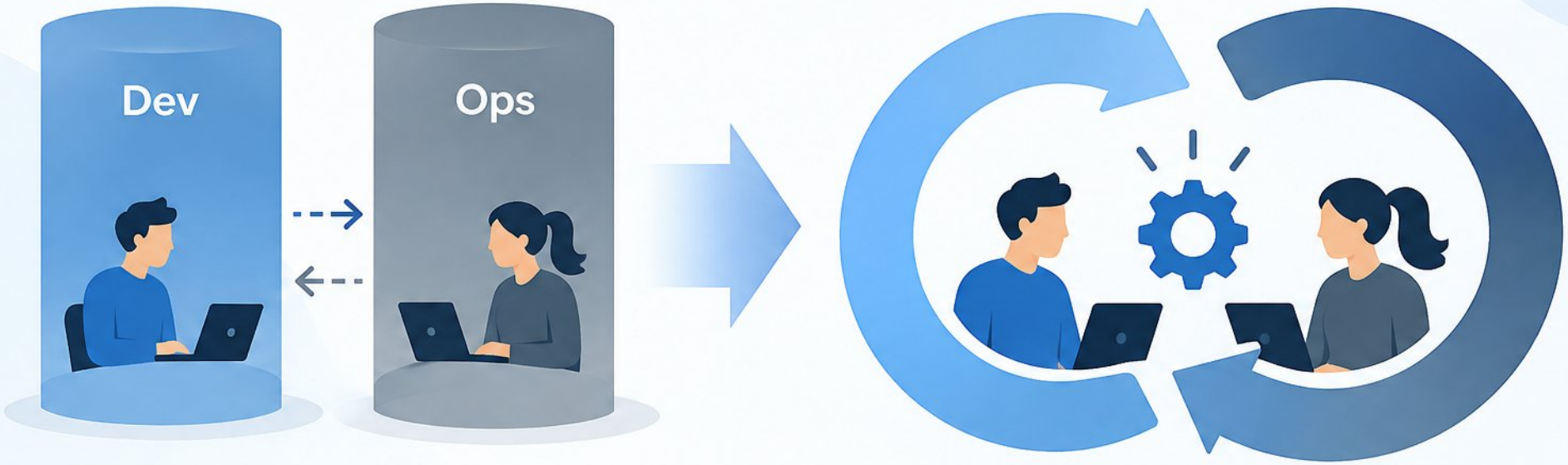
What Does a DevOps Engineer Do

- Builds and operates the delivery system: CI/CD pipelines, infrastructure as code, observability
- Ships software quickly and safely across many product teams
- Differs from SRE (reliability targets, incidents), platform (self-service platforms), cloud (architecture, cost)
- Roadmap: role, evolution, a real week, responsibilities, tools, ownership, skills, metrics, next steps
- Value: realistic entry picture for juniors, clearer handoffs for teams, real signals for managers



From Silos to Shared Ownership



- Dev and ops handoffs created release bottlenecks and slow incident recovery
- Core principles: collaboration, automation, continuous feedback, shared production responsibility
- DevOps is a culture and process shift first—the job title is one implementation
- Implemented as a dedicated role, embedded practice, or platform model
- Specialists own automation and production readiness across many product teams

The Daily Work: A Week in the Life

- Morning starts with dashboard and alert triage, then standup
- Work splits across reactive incidents, team enablement, and planned improvements
- Early-career engineers handle more toil and support requests at first
- Typical tasks: fix CI pipeline, review Terraform PR, provision database
- Sustainable on-call needs runbooks, escalation paths, and shift handoffs





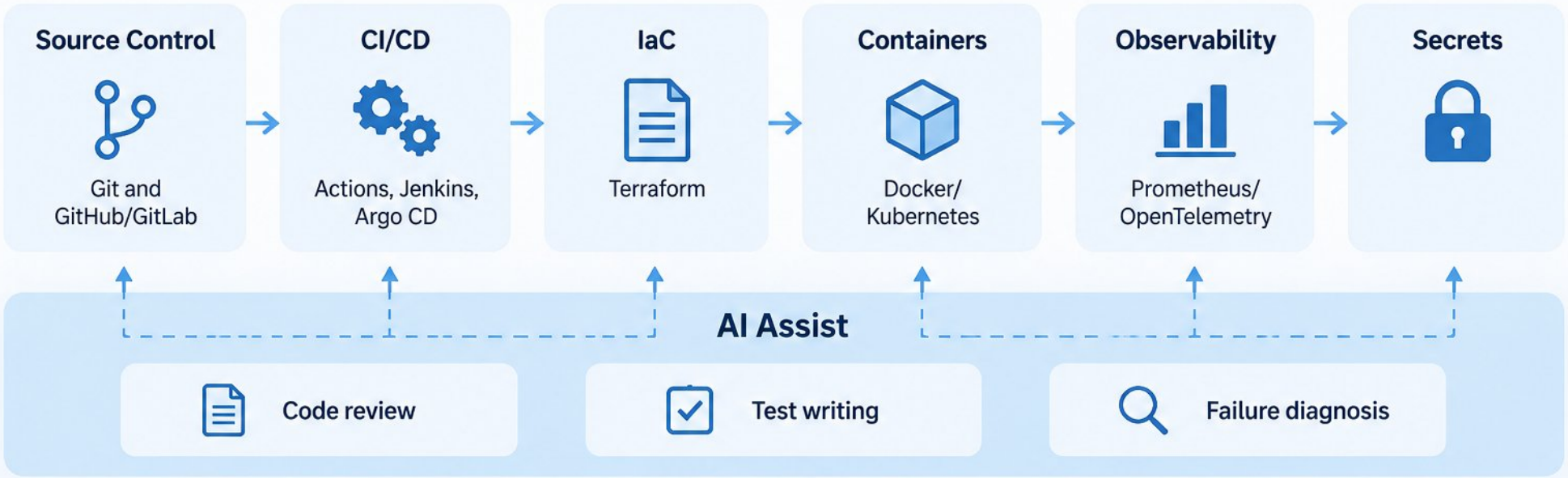
Core Responsibilities and Deliverables



- Own the CI/CD pipeline: builds, tests, artifacts, safe rollouts and rollback
- Manage infrastructure as code: reviewed plans, reproducible environments, drift detection
- Own observability: logs, metrics, traces, and alerts tied to SLOs
- Support security: automated scanning, secrets management, least privilege, signed artifacts and SBOMs
- Reduce toil: automate environment setup, secret rotation, database migrations, release notes

Toolchain Map and Where AI Fits

BUILD
LEARN
GROW



- Group tools by function: source control, CI/CD, IaC, containers, observability, secrets
- Git and GitHub/GitLab; Actions, Jenkins, Argo CD; Terraform; Docker/Kubernetes; Prometheus/OpenTelemetry
- Early-career Kubernetes: images, pods, deployments, services, config, safe rollouts
- AI in 2026: 48.8% use AI for a CI/CD task; only 8% run AI inside builds
- Top uses sit around the pipeline: code review, test writing, failure diagnosis
- AI raises change volume — review capacity (37%) and reliability judgment (36%) become bottlenecks
- Go deep on a few tools spanning source control, CI/CD, one cloud, and observability

A stronger toolchain builds a brighter you.



Collaboration Boundaries:
Who Owns What

CLEAR ROLES STRONGER OUTCOMES

- Developers own app code; DevOps owns pipelines and infrastructure automation
- SREs own reliability targets and incident leadership; security sets policy
- Avoid the gatekeeper trap: DevOps enables teams, it doesn't approve tickets
- Three models: centralized DevOps, embedded engineers, platform-as-product
- About 90% of organizations now run at least one internal platform
- Clear incident handoffs cut confusion and restore time



COLLABORATE • HAND OFF CLEARLY • ACHIEVE MORE TOGETHER

Skills, Certifications, and Career Path

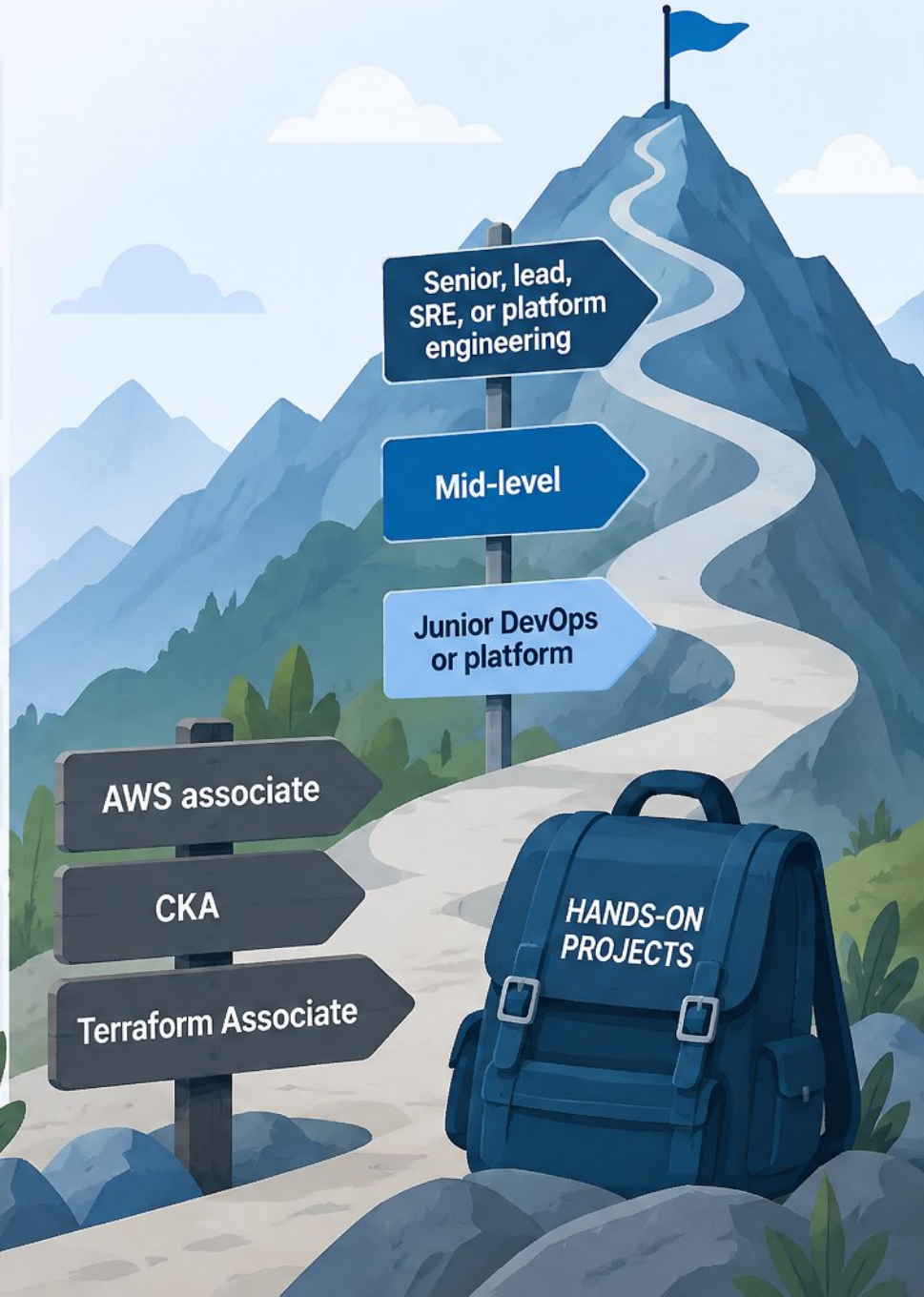
Foundations: Linux, networking, Git, one scripting language, cloud basics, containers, CI/CD

Daily soft skills: precise questions, documentation, incident communication, cross-team stakeholder management

Shipped projects beat certification count; AWS associate, CKA, and Terraform Associate are most recognized

Path: junior DevOps or platform → mid-level → senior, lead, SRE, or platform engineering

Spend about 20% of learning time on certs and 80% on two or three documented portfolio projects



Measuring Success and Avoiding Anti-Patterns

Measure. Learn. Improve. Keep Delivery Flowing.

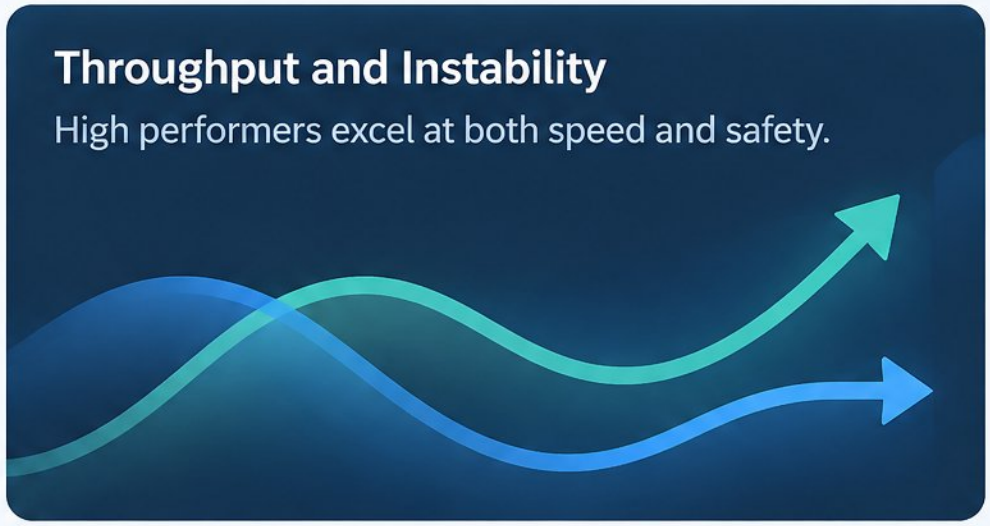
Five DORA metrics: deployment frequency, change lead time, failed deployment recovery time, change failure rate, deployment rework rate.

Throughput and instability: high performers excel at both speed and safety.

SLOs, error budgets, and blameless postmortems turn incidents into improvement.

Avoid anti-patterns: DevOps as ticket queue, unowned pipelines, alert fatigue, approval theater.

Safeguards: small reversible changes, automated rollbacks, low-noise alerts, current runbooks.



Common Anti-Patterns

DevOps as ticket queue

Unowned pipelines

Alert fatigue

Approval theater

Manager check: is DevOps speeding delivery or just adding approvals?

Key Takeaways and Next Steps

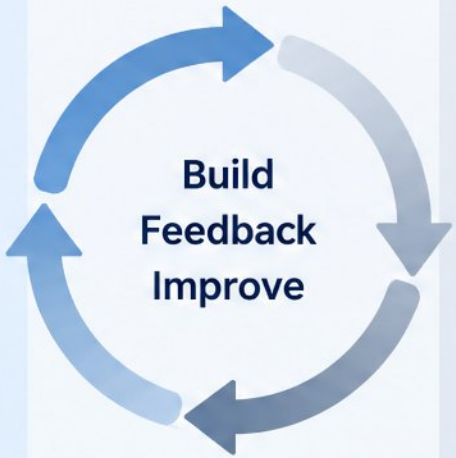
LEARN → APPLY → IMPROVE

Five takeaways

- delivery builder
- production-readiness owner
- developer enabler
- reliability guardian
- feedback champion

30-day starter

- Linux and Git fluency
- one CI/CD pipeline
- Terraform
- monitoring
- documentation



Manager checklist

- pipeline ownership
- sustainable on-call
- alert quality
- deployment speed
- recovery time

Follow-up topics

- Kubernetes deep dive
- platform engineering
- security automation

STRONGER SYSTEMS. HAPPIER TEAMS.

The 2026 Context:
What Changed and Why It Matters

- **AI adoption is near-universal:** 90% use AI at work, 80%+ report productivity gains
- **AI boosts delivery throughput** but still correlates with higher instability
- **DevOps maturity drives AI success;** mature practices unlock far greater returns
- **Platform engineering is now standard:** about 90% of organizations run at least one platform
- **The role shifts from operating tools** to designing, validating, and improving automated systems



Myths and Misconceptions to Correct Early

Clearer understanding leads to stronger teams and better outcomes.

Myth: DevOps is a tool or team name.	**Reality:** it is a culture and practice.
Myth: DevOps only does operations.	**Reality:** they write IaC, pipeline code, automation.
Myth: DevOps replaces SRE or platform engineering.	**Reality:** overlapping roles, different questions.
Myth: AI will remove the need for DevOps.	**Reality:** AI amplifies practice, raising review and safety needs.
Myth: You must master every tool.	**Reality:** depth in a few areas earns on-call trust.

LESS MYTHS. STRONGER PRACTICES.

A Realistic First 90 Days in the Role



First 30 days

First 30 days: learn the environment, shadow on-call, read runbooks

Days 31–60

Days 31–60: own a small pipeline area, fix recurring toil

Days 61–90

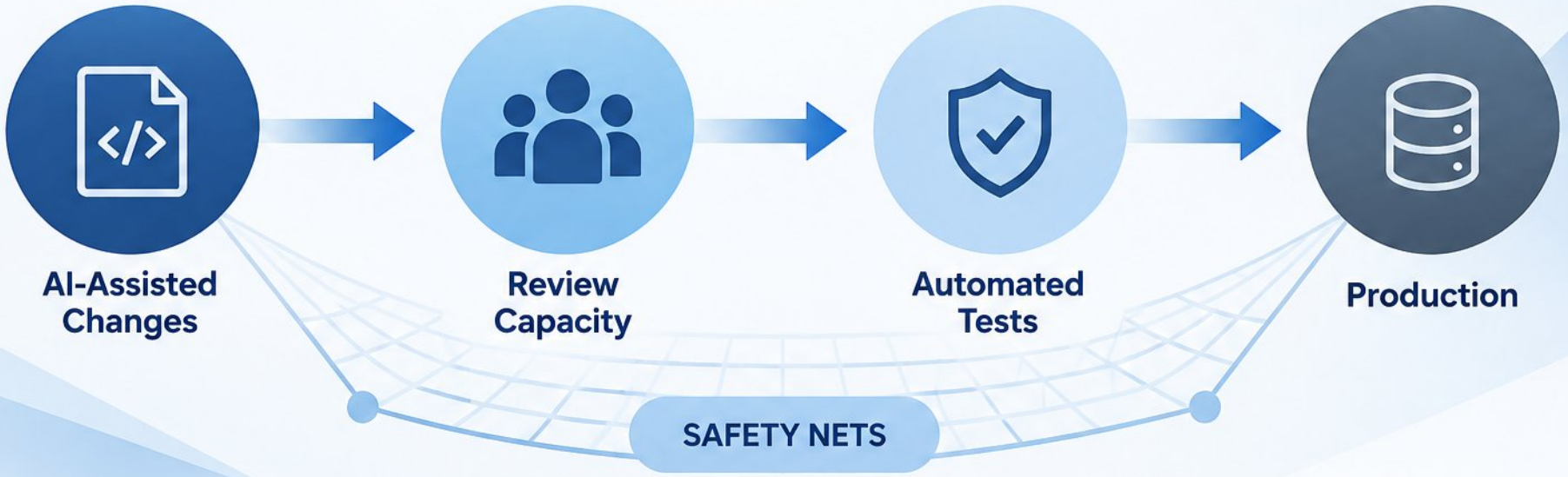
Days 61–90: ship an improvement like a reusable pipeline template

Strong juniors ask specific questions, document everything, verify AI changes

Managers: set clear on-call load and protected project time

Working With AI-Assisted Delivery Safely

- Most 2026 AI use sits around the pipeline: code review, test writing, failure diagnosis
- Only 8% run AI steps inside builds; ~49% use AI for at least one CI/CD task
- Bottleneck moved downstream: 37% cite review capacity, 36% reliability judgment
- Trust but verify: about 30% of practitioners report little or no trust in AI code
- Safety nets: strong tests, mature version control, fast feedback, clear AI policy
- Executives and practitioners disagree on AI in incident management: 74% vs 39%



Wrap-Up: Applying This to Your Team

Early career

- Early career: one cloud, one CI/CD tool, one IaC tool, small documented project

Teams

- Teams: agree explicitly on pipeline, infrastructure code, observability, incident ownership

Managers

- Managers: judge by throughput, stability, on-call sustainability, developer experience—not tickets

Next steps

- Next steps: Kubernetes deep dive, platform engineering, security automation

Great DevOps engineers make others more effective and leave systems safer

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/0ef8f603/public