TECHWITHKOUSHIK

AI-FIRST CLOUD ENGINEERING

Technical Field Guide

A provider-agnostic roadmap for infrastructure, agents, FinOps, observability, security and autonomous operations

AI + IaC	MCP + Agents	Policy-as-Code
AI FinOps	OpenTelemetry	AIOps + SRE

WHO THIS IS FOR

Cloud engineers, DevOps engineers, SREs, platform engineers and architects who want to use AI without giving up engineering rigour, security or accountability.

WHAT YOU WILL BUILD

A safe AI-assisted delivery and operations model: intent -> generate -> test -> guard -> deploy -> observe -> triage -> remediate -> learn.

koushikmaji.com

20-page professional study guide • 2026 edition

01 / OPERATING MODEL

The AI-Augmented Cloud Engineer

Move from prompt experiments to governed engineering workflows.

The durable skill is not “prompting”. It is designing a controlled system in which AI can propose work, deterministic tools can verify it, and humans can approve risk. Use AI to increase throughput while preserving version control, testing, policy, audit and rollback.

The eight-stage control loop

Stage	AI contribution	Engineering control
1. Intent	Translate outcomes into a structured brief	SLO, security, data, budget and compliance requirements
2. Generate	Draft IaC, tests, docs and runbooks	Approved modules, provider constraints and coding standards
3. Validate	Explain plan output and likely risks	Syntax, unit, integration and contract tests
4. Guard	Summarise policy and cost findings	Policy-as-code, secret scanning, budget gates
5. Deploy	Prepare change plan and release notes	Human approval, signed artifacts, limited credentials
6. Observe	Summarise logs, metrics and traces	OpenTelemetry schema and data-quality controls
7. Triage	Build incident timeline and hypotheses	Evidence, confidence, blast radius and rollback
8. Learn	Draft post-incident actions	Owner, due date, measurable control improvement

OPERATING PRINCIPLE

Delegate repetitive, reversible and well-instrumented work first. Keep irreversible, ambiguous or high-impact changes under stronger human approval.

Career shift

- Syntax operator -> intent designer.
- Ticket resolver -> reliability systems engineer.
- Cloud bill reviewer -> unit-economics owner.
- Tool administrator -> AI control-plane architect.

02 / CONTEXT ENGINEERING

Write an Engineering Context Contract

An AI model cannot infer your platform standards safely. Supply them explicitly.

Before asking AI to generate infrastructure, create a machine-readable context pack. Treat this pack as version-controlled platform documentation. It should be reviewed like code because it directly shapes generated output.

Minimum context pack

Domain	Required context	Example evidence
Identity	Workload identity pattern, forbidden permissions, break-glass flow	Policy IDs, role boundaries, approval owners
Network	CIDR ownership, ingress/egress model, private endpoints, DNS	Network diagram, allowed routes, service catalogue
Data	Classification, region constraints, encryption, retention, backup	Data class, KMS policy, RPO/RTO
Reliability	SLO, scaling signal, dependency budget, failure modes	Availability target, latency SLI, chaos test
Cost	Budget, unit metric, expiry, commitment assumptions	Cost per request, monthly cap, owner tag
Delivery	Branch strategy, environments, test gates, deployment windows	Pipeline policy, required reviewers, rollback

Example: provider-agnostic intent contract

```
workload:
  name: checkout-api
  environment: production
  slo: {availability: 99.9%, p95_latency_ms: 300}
  data_class: confidential
  network: private-only
  identity: workload-identity; no-static-secrets
  budget: {monthly: 5000, unit: cost_per_1000_requests}
  change_policy: human-approval-required
```

PROMPT PATTERN

“Generate two architecture options from this contract. List assumptions, trust boundaries, failure modes, cost drivers and validation tests. Do not create resources that violate a stated constraint.”

03 / INFRASTRUCTURE AS CODE

AI-Assisted IaC Delivery Pipeline

AI produces a candidate change. The pipeline produces evidence.

Recommended repository structure

```
infra/
  modules/          # versioned, reusable components
  stacks/           # environment compositions
  policy/           # OPA/Conftest rules
  tests/            # native IaC tests + integration tests
  context/          # architecture and security contracts
  runbooks/         # deploy, rollback, recovery
  .ci/              # plan, scan, cost, sign, approve
```

PR gate sequence

Gate	Typical check	Failure behaviour
Format + validate	Syntax, provider lock, module inputs	Fail immediately
Static analysis	Deprecated fields, risky patterns, naming	Fail or annotate by severity
Plan review	Resource adds/changes/destroys, replacements	Require explicit destructive-change approval
Policy	Identity, public access, encryption, regions, tags	Block non-compliant plan
Cost	Delta, budget threshold, commitment impact	Block or route to FinOps owner
Tests	Plan assertions, integration and post-deploy smoke	Fail and preserve logs
Supply chain	Dependency scan, provenance, artifact signature	Reject untrusted artifact

AI review prompt

```
Review this plan as a senior cloud engineer.
Return: destructive changes; public exposure; privilege expansion;
data movement; state risk; expected cost delta; rollback steps;
and the exact evidence used for each finding.
```

DO NOT

Let AI execute an unreviewed apply, modify state directly, or use a production-wide credential. Separate plan permissions from apply permissions.

04 / TESTING

Build an IaC Testing Pyramid

Generated infrastructure needs software-engineering discipline.

Use multiple test layers because each catches a different class of failure. OpenTofu supports a test command with assertions against plans or state; integration tests may create real resources, so cleanup and cost controls are essential. [S10]

Layer	Purpose	Examples
Static	Fast feedback without creating resources	fmt, validate, lint, schema checks, secret scan
Unit / plan	Assert planned properties and module behaviour	Encryption enabled, public access false, tags present
Policy	Evaluate organisation invariants on plan JSON	Allowed regions, instance families, IAM restrictions
Integration	Create isolated resources and test real behaviour	Private endpoint reachable only from approved network
Contract	Verify compatibility with platform interfaces	Outputs, DNS, identity claims, telemetry labels
Post-deploy	Prove runtime health and rollback	Synthetic request, SLO check, drift scan

Illustrative plan assertion (adapt to your IaC tool)

```
run "private_storage" {
  command = plan
  assert {
    condition     = output.public_access_enabled == false
    error_message = "Storage must not be public"
  }
}
```

AI USE CASE

Ask AI to derive test cases from the intent contract, then require a human to map every critical requirement to at least one deterministic test.

Test data rules

- Use sandbox accounts/projects/subscriptions and explicit TTL tags.
- Generate unique names; never reuse production identifiers.
- Budget integration-test environments and verify teardown.
- Store test evidence with the pull request.

05 / POLICY-AS-CODE

Turn Guardrails into Executable Decisions

Use policy engines to decide; use AI to explain the decision.

Open Policy Agent is a general-purpose policy engine that can evaluate structured input and return structured decisions across CI/CD, Kubernetes, microservices and APIs. [S5]

Illustrative Rego-style policy

```
package cloud.guardrails

deny[msg] {
  r := input.resource_changes[_]
  r.type == "object_storage"
  r.change.after.public_access == true
  msg := sprintf("%s enables public access", [r.address])
}

deny[msg] {
  some i
  input.resource_changes[i].change.after.tags.Owner == ""
  msg := "Owner tag is required"
}
```

Design policies at three levels

Level	Examples	Response
Mandatory	No public confidential data; no wildcard admin; encryption required	Block
Risk-based	Large instance, cross-region data, privileged container	Require owner approval
Advisory	Missing optimisation, weak naming, low utilisation forecast	Annotate and recommend

SEPARATION OF DUTIES

AI may translate a policy failure into plain language and propose a fix. AI should not silently waive the policy that rejected its own output.

06 / SECURITY

Threat-Model the AI Cloud Control Plane

Cloud access turns an LLM mistake into an infrastructure incident.

OWASP's 2025 LLM risk catalogue highlights classes including prompt injection, sensitive information disclosure, supply-chain risk, improper output handling, excessive agency and unbounded consumption. [S11] NIST AI RMF provides a Govern, Map, Measure and Manage structure for AI risk. [S7]

Threat	Cloud example	Control
Prompt injection	A ticket or README instructs the agent to run an unsafe tool	Treat retrieved content as untrusted; isolate instructions from data
Sensitive disclosure	Prompt includes secrets, customer data or plan output	Redact, classify, minimise context; disable content capture by default
Improper output handling	Model output is passed directly to shell or API	Schema validate; allow-list commands and arguments
Excessive agency	Agent receives broad production admin access	Least privilege, per-tool scopes, approval and TTL credentials
Supply chain	Generated code imports an untrusted module/image	Pin, scan, sign and verify provenance
Unbounded consumption	Retry loop creates cost or resource explosion	Token, tool-call, time and budget quotas

Security acceptance criteria

- Every tool call is attributable to user, agent, environment and change request.
- Production write operations require explicit policy decision and approval record.
- Sensitive prompt/response capture is off unless a documented use case enables it.
- The system can revoke the agent credential and stop execution immediately.

07 / MCP AND AGENTS

Design Narrow MCP Tool Contracts

Standardised connectivity is valuable only when the exposed capabilities are safe.

MCP connects AI hosts, clients and servers and supports resources, prompts and tools. The 2026 roadmap emphasises production concerns such as transport scalability and agent communication. [S2][S3]

Tool contract anatomy

```
tool: propose_iac_change
mode: read-write-to-branch-only
input: {repo, branch, intent_contract, module_allowlist}
output: {commit, plan_summary, policy_results, cost_delta}
permissions: [read_repo, write_feature_branch, run_plan]
forbidden: [merge, apply, state_write, secret_read]
approval: required_before_merge
audit: full
```

Tool class	Example	Default posture
Read-only	Get plan, query inventory, retrieve telemetry	Allow with scoped identity and audit
Drafting	Create branch, propose patch, create ticket	Allow in non-production workspace
Change	Apply approved plan, scale service, rotate secret	Policy + human approval + rollback
Destructive	Delete state/resource, alter routing, revoke identity	Deny by default; break-glass workflow

MCP SAFETY RULE

Do not expose a generic shell or unrestricted cloud API as the first tool. Expose task-specific tools with typed input, explicit environment, bounded output and policy checks.

08 / IDENTITY

Identity and Approval Architecture for AI Agents

The agent must be easier to stop and audit than a human administrator.

Recommended identity chain

Step	Control	Evidence
User -> AI host	User authentication, tenant and role context	User id, session id, request purpose
Host -> agent	Short-lived agent session and task scope	Agent id, model/version, task id
Agent -> tool	Per-tool identity or delegated token	Tool name, arguments hash, policy decision
Tool -> cloud API	Workload identity, environment boundary, least privilege	Cloud audit event, resource id, change result
Approval	Independent approver for high-risk change	Approver, timestamp, diff and expiry

Permission design

- Separate read, plan, propose, approve and apply roles.
- Use short-lived credentials; avoid embedding provider keys in prompts or agent memory.
- Bind permissions to environment and resource scope, not only API verb.
- Require a fresh approval if the plan changes after review.
- Add a kill switch that revokes active sessions and blocks new tool calls.

AUDIT EVENT

Record: actor, agent, model, prompt/template version, tool, input hash, resource, policy decision, approval, result, latency, cost and rollback status.

09 / FINOPS

AI FinOps and FOCUS-Based Cost Data

Optimise for business value, not only lower bills.

FOCUS is an open specification that normalises billing datasets across AI, cloud, SaaS and data-centre vendors. FOCUS 1.4 was ratified on 4 June 2026 and expands provider-agnostic billing use cases. [S12] The FinOps Foundation's 2026 report says AI cost management is the number-one skillset teams need to develop. [S1]

Cost model for AI-enabled cloud workflows

```
cost_per_successful_task =
  model_inference
+ retrieval_and_vector_search
+ agent_tool_calls
+ compute_and_accelerators
+ data_transfer
+ observability_and_evaluation
+ failed_attempts_and_retries
```

Metric	Engineering question	Action
Cost / successful task	Are retries and tool loops inflating spend?	Cap steps; improve tool selection and validation
Input / output tokens	Is context or response larger than needed?	Trim context; summarise; cache
GPU utilisation	Is expensive capacity doing useful work?	Batch, queue, schedule or right-size
Cost / environment	Are dev and test resources idle?	TTL, schedules and ownership
Effective cost	Are commitments and discounts allocated correctly?	Use normalised cost data and unit allocation

SHIFT LEFT

Attach estimated cost delta and unit-cost impact to the pull request. Budget exceptions need an owner, business reason and expiry.

10 / GPU ENGINEERING

Schedule Accelerators by Requirement, Not Habit

GPU economics is a platform-engineering problem.

Kubernetes 1.34 made the core Dynamic Resource Allocation APIs generally available. DRA allows workloads to request device properties while the scheduler allocates actual devices, supporting specialised resources such as GPUs and FPGAs. [S13]

Accelerator request contract

Field	Example	Why
Workload type	training / batch inference / online inference	Determines interruption and latency tolerance
Device properties	memory, topology, accelerator class	Avoids allocating an unsuitable device
Priority + deadline	P1 online; batch by 06:00 UTC	Supports queue and pre-emption policy
Checkpoint path	object://checkpoints/job-123	Enables recovery and spot/pre-emptible capacity
Utilisation target	GPU > 65%, memory > 70%	Creates optimisation signal
Budget + owner	£500 run cap; team-platform	Stops unowned consumption

AI-assisted optimisation

- Forecast queue demand from historical workloads.
- Detect low utilisation caused by input pipeline, batch size or memory bottlenecks.
- Recommend workload placement, batching, autoscaling or smaller accelerators.
- Never terminate training solely from one metric; verify checkpoint and workload owner policy.

CLOUD-AGNOSTIC VIEW

Even outside Kubernetes, model accelerator capacity as a schedulable pool with typed capabilities, quotas, queueing, ownership, telemetry and cost.

11 / MODEL SERVING

Engineer Reliable Inference Platforms

The model is one component inside a production service.

Serving control plane

Concern	Telemetry	Design response
Latency	Time to first token, p50/p95/p99, queue time	Autoscale on queue + concurrency; route by latency SLO
Quality	Task success, evaluation score, fallback rate	Route complex tasks to stronger model; evaluate changes
Cost	Tokens, cache hit, tool calls, cost per task	Cache, batch, shorten context, use smaller model
Capacity	Concurrency, accelerator memory, saturation	Admission control, quotas, warm pools
Reliability	Timeout, retry, circuit breaker, provider error	Fallback model/provider, idempotent tool actions
Safety	Blocked output, policy denial, sensitive-data event	Input/output validation and human escalation

Routing pseudocode

```
if task.risk == "high": require_human_review()
elif task.context_tokens > model.context_limit: summarise_or_retrieve()
elif latency_slo_tight and small_model_quality_ok: route(small_model)
else: route(standard_model)
apply_budget_cap(task); record_trace(task)
```

DESIGN FOR FAILURE

Model providers, vector stores and tools will fail independently. Use timeouts, bounded retries, idempotency keys, circuit breakers and a degraded mode.

12 / OBSERVABILITY

OpenTelemetry for AI and Agent Workflows

Instrument the model call, the tool call and the business outcome.

OpenTelemetry provides vendor-neutral metrics, logs and traces. Its GenAI semantic conventions cover spans, metrics and events for GenAI clients and MCP-related operations. The OpenTelemetry project notes that prompt and response content capture is opt-in because it may contain sensitive data. [S4][S14][S15]

Recommended span hierarchy

```
request / user-task
  agent.invoke
    retrieval.query
    gen_ai.chat
    execute_tool: plan_infrastructure
      cloud_api.read
    execute_tool: create_pull_request
  evaluation.result
```

Signal	Attributes to standardise
Model call	operation, requested/response model, input/output tokens, latency, error
Agent	agent id, task id, step count, tool-call count, outcome
Tool	tool name, environment, argument hash, result status, approval id
Retrieval	index, query, document ids, relevance score, tenant
Cost	estimated cost, currency, allocation key, successful-task id
Safety	policy result, sensitive-content flag, human escalation

PRIVACY DEFAULT

Capture metadata first. Enable full prompts, completions or tool arguments only after data classification, access control, retention and redaction are defined.

13 / AIOPS

Evidence-Driven Incident Correlation

A useful AI incident narrative must be explainable and falsifiable.

Correlation pipeline

Phase	Method	Output
Normalise	Common service, environment, deployment and trace identifiers	Comparable signals
Reduce noise	Deduplicate, suppress known maintenance, group symptoms	Event clusters
Add topology	Service graph, dependency, ownership and change data	Blast-radius context
Generate hypotheses	LLM summarisation + statistical/graph correlation	Ranked likely causes
Test hypothesis	Query telemetry, compare baseline, inspect recent change	Evidence for/against
Recommend action	Runbook, risk, approval, rollback, validation	Actionable incident ticket

Structured incident output

```
incident = {
  "impact": "checkout p95 > 1.2s",
  "likely_root_cause": "DB connection pool exhaustion",
  "confidence": 0.82,
  "evidence": ["trace span wait", "pool metric 100%", "deploy +8m"],
  "safe_action": "rollback deployment",
  "validation": "p95 < 300ms for 10m"
}
```

HUMAN CHECK

Correlation is not causation. Show competing hypotheses and evidence gaps. A confident narrative without traceable evidence is not an incident diagnosis.

14 / REMEDIATION

Design a Safe Autonomous Remediation Loop

Autonomy should earn trust one bounded action at a time.

State machine

```
DETECT -> CORRELATE -> PROPOSE -> POLICY_CHECK -> APPROVE?
  -> EXECUTE -> VERIFY -> {RESOLVED | ROLLBACK | ESCALATE}

Every transition emits an audit event.
Timeout or uncertainty moves to ESCALATE.
```

Automation level	Example	Required control
Recommend	Draft rollback command and validation query	Evidence and risk score
Approve then run	Restart stateless workload; scale replicas	Human approval, idempotency, verification
Auto-run low risk	Clear known safe cache; restart failed sidecar	Rate limit, blast-radius cap, rollback
Never autonomous by default	Delete data, alter routing, rotate root identity	Break-glass process and independent approval

Verification contract

- Success metric and observation window are defined before execution.
- Action has an idempotency key and maximum retry count.
- Rollback is tested and uses a separate control path.
- The remediation stops if customer impact worsens.
- A post-action summary links evidence, approval, command and result.

15 / RAG FOR OPERATIONS

Ground AI in Approved Engineering Knowledge

RAG can improve relevance, but retrieval quality and document trust must be engineered.

Knowledge sources

Source	Index metadata	Risk
Runbooks	service, owner, version, tested date, environment	Stale or unsafe instructions
Architecture decisions	system, decision, status, superseded-by	Old decision returned as current
Post-incident reviews	service, cause, action, severity, date	Sensitive customer or personnel data
IaC modules	module, version, provider, security tier	Unapproved or vulnerable module
Policies	policy id, version, severity, exception owner	Mismatch between text and executable policy

Retrieval pipeline

```
query -> classify task and tenant -> apply access filter
-> hybrid retrieve -> rerank -> freshness filter
-> cite document + version -> generate answer
-> validate against executable policy/tool result
```

DEFENCE AGAINST INDIRECT INJECTION

Treat retrieved documents as untrusted data. Do not allow a runbook, web page or ticket to redefine system instructions or authorise a tool call.

Quality metrics

- Retrieval recall for known incident questions.
- Citation correctness and document freshness.
- Access-control leakage test.
- Answer usefulness and action safety.

16 / EVALUATION

Evaluate AI Like a Production Dependency

A demo that works once is not an engineering control.

Evaluation matrix

Dimension	Offline test	Online signal
Correctness	Golden IaC review findings; known incident root causes	Accepted recommendation rate; escaped defects
Safety	Prompt injection, unsafe tool request, secret leakage tests	Policy denials, human overrides, rollback rate
Reliability	Timeout, malformed output, tool failure, provider outage	Success rate, retry count, degraded-mode usage
Cost	Token and tool budget per scenario	Cost per successful task and per team
Latency	P95 task completion under synthetic load	P95/P99 by workflow and model
Explainability	Evidence and citation completeness	Uncited recommendation rate

Release gate

```
Release only if:
- policy bypass rate == 0 in security suite
- critical finding recall >= agreed threshold
- cost per successful task <= budget
- rollback and kill switch pass
- human reviewers approve evidence quality
```

MODEL CHANGE = SOFTWARE CHANGE

Pin model and prompt/template versions where possible. Re-run evaluation when the model, prompt, tool schema, retrieval index or policy changes.

17 / PLATFORM ENGINEERING

Create Golden Paths for AI-Enabled Cloud Work

The platform should make the safe path the easiest path.

Platform products to expose

Product	Self-service interface	Built-in controls
AI-ready service	Template or portal request	Identity, network, telemetry, budget, deployment pipeline
Inference endpoint	Model class + SLO + quota request	Routing, autoscaling, content policy, cost allocation
RAG index	Data source + classification + owner	Access filters, retention, freshness and evaluation
Agent tool	Typed tool contract + environment	Least privilege, audit, approval and rate limits
GPU job	Workload, device needs, deadline, budget	Queue, DRA/capacity policy, checkpoint and TTL
Incident assistant	Service + time window + impact	Read-only telemetry, citations, confidence and escalation

CNCF guidance on production AI engineering highlights reliable serving, GPU scheduling, observability, model versioning and governance as infrastructure concerns. [S9]

PLATFORM KPI

Measure time-to-first-safe-deployment, policy failure rate, unit cost, SLO attainment, developer adoption and operational toil - not only number of templates.

18 / ROADMAP

12-Week Cloud Engineer Journey

Build one integrated capstone instead of collecting disconnected tutorials.

Weeks	Build	Technical evidence
1-2	Intent contract + provider-agnostic architecture	SLO, threat model, cost unit, failure modes and ADR
3-4	AI-assisted IaC module	Plan, tests, module versioning and destructive-change review
5-6	Policy and supply-chain gates	OPA rules, secret scan, provenance and approval path
7-8	FinOps + accelerator controls	FOCUS-aligned cost data, PR delta, GPU/job budget and TTL
9-10	OpenTelemetry + AIOps triage	Trace schema, incident timeline, evidence-backed hypothesis
11	MCP read-only agent	Typed tools, per-tool identity, audit events and evaluation
12	Safe remediation demo	Approve-execute-verify-rollback state machine

Capstone acceptance checklist

- No production-wide credential exists.
- Every generated change is version-controlled and tested.
- Policy and cost decisions are attached to the pull request.
- Telemetry links request -> agent -> tool -> cloud API -> outcome.
- High-risk changes require independent approval.
- Rollback and kill switch have been demonstrated.
- A dashboard reports quality, reliability, safety, cost and latency.

PORTFOLIO OUTPUT

Publish the architecture, control model, evaluation design and a sanitised demo. Recruiters and technical leaders value evidence of judgement more than a list of AI tools.

19 / REFERENCE

Sources, Standards and Further Study

Primary references used for the technical claims in this guide.

Reference	Link
[S1] FinOps Foundation - State of FinOps 2026	https://data.finops.org/
[S2] Model Context Protocol - Specification	https://modelcontextprotocol.io/specification/draft
[S3] Model Context Protocol - 2026 Roadmap	https://modelcontextprotocol.io/development/roadmap
[S4] CNCF - OpenTelemetry graduation announcement	https://www.cncf.io/announcements/2026/05/21/cloud-native-computing-foundation-announces-opentelemetrys-graduation-solidifying-status-as-the-de-facto-observability-standard/
[S5] Open Policy Agent documentation	https://www.openpolicyagent.org/docs
[S6] SLSA supply-chain security framework	https://slsa.dev/
[S7] NIST AI Risk Management Framework	https://www.nist.gov/itl/ai-risk-management-framework
[S8] FinOps Foundation - FinOps for AI Overview	https://www.finops.org/wg/finops-for-ai-overview/
[S9] CNCF - Cloud-native platform under production AI	https://www.cncf.io/blog/2026/03/26/the-platform-under-the-model-how-cloud-native-powers-ai-engineering-in-production/
[S10] OpenTofu testing workshop reference	https://www.massdriver.cloud/blogs/opentofu-foundations-testing-and-validation
[S11] OWASP Top 10 for LLM Applications 2025	https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/
[S12] FOCUS 1.4 specification page	https://focus.finops.org/focus-specification/
[S13] Kubernetes 1.34 Dynamic Resource Allocation GA	https://kubernetes.io/blog/2025/09/01/kubernetes-v1-34-dra-updates/
[S14] OpenTelemetry GenAI semantic conventions repository	https://github.com/open-telemetry/semantic-conventions-genai
[S15] OpenTelemetry - Inside the LLM Call	https://opentelemetry.io/blog/2026/genai-observability/

FINAL MESSAGE

AI is not a substitute for cloud engineering fundamentals. AI increases the value of architecture, identity, policy, testing, observability, cost management and operational judgement.

TechWithKoushik | koushikmaji.com