GKE Platform Architect Fieldbook

Simple Kubernetes explanations, production GKE patterns, and architecture practice

- GKE Platform Architect Fieldbook
- 00 · The GKE platform architect role
  - Explain it like I am five
  - The job loop
  - What strong evidence looks like
  - The key principle
- 01 · Kubernetes in simple terms
  - The small vocabulary
  - Requests and limits
  - Health
  - The declarative habit
- 02 · How GKE works
  - Responsibility boundary
  - Regional and zonal clusters
  - VPC-native networking
  - GKE and Google Cloud
  - Design question
- 03 · Autopilot or Standard
  - Autopilot
  - Standard
  - Decision table
  - Common error
- 04 · Cloud foundation and identity
  - Resource hierarchy
  - Two permission systems
  - Workload identity
  - Boundaries
  - Namespace warning
- 05 · GKE networking and isolation
  - Address plan
  - Private design
  - Traffic controls
  - Exposing services
- 06 · Designing Kubernetes workloads
  - Stateless service baseline
  - Graceful behavior
  - Configuration
  - Stateful work
  - Batch and events
- 07 · Scheduling, capacity, and autoscaling

    - Four scaling loops
    - Placement
    - Resources
    - Priority
    - Architect test
- 08 · Storage and data
    - Persistent storage
    - Database decision
    - Connection design
    - Backup
- 09 · GKE security architecture
    - Harden access
    - Harden Pods
    - Protect the supply chain
    - Network and data
    - Detect and respond
- 10 · Traffic, gateways, and service mesh
    - Traffic choices
    - Gateway API
    - Service mesh
    - Resilience
- 11 · Fleets and platform governance
    - Fleet capabilities
    - Platform team product
    - Config Sync and Policy Controller
    - Cluster count
- 12 · Delivery, GitOps, and the software supply chain
    - Golden path
    - Rollout strategy
    - Policy
    - Change safety
- 13 · Observability and SRE
    - Signals
    - SLO first
    - Kubernetes signals
    - Cardinality and cost
    - Runbooks
    - Learning loop
- 14 · Reliability, upgrades, and disaster recovery
    - Regional baseline
    - Upgrades
    - Failure design
    - Disaster recovery
- 15 · Cost, performance, and sustainability
    - Autopilot economics
    - Practical controls

    - Performance
    - Sustainability
    - Trade-off
- 16 · Real-life scenario: Global Service Platform
    - Situation
    - Architecture decision
    - Enterprise integration
    - Reliability
    - Operations and outcomes
    - Important trade-offs
- 17 · Interview, certification, and glossary
    - Professional Cloud Architect checkpoint
    - Compact glossary
    - Official references

GKE Platform Architect Fieldbook

> Independent learning material. Google Cloud and GKE are trademarks of Google LLC.

00 · The GKE platform architect role

Google Kubernetes Engine (GKE) is Google's managed Kubernetes service. A GKE architect decides how teams safely run containers at scale.

Explain it like I am five

Imagine an application is a toy made from many small boxes. A **container** is one box with everything a program needs. **Kubernetes** is the organizer that puts boxes on computers, replaces broken boxes, and makes more when people need them. **GKE** is Kubernetes where Google operates much of the organizer.

The architect decides which playground to use, who may enter, how boxes talk, what happens when a computer breaks, how updates stay safe, and how much it all costs.

The job loop

1. Discover outcomes, users, data, scale, recovery, security, and team skills.
2. Decide whether Kubernetes is actually needed.
3. Choose Autopilot or Standard and define cluster/project boundaries.
4. Design identity, network, workload, data, delivery, and operations.
5. Make trade-offs visible to developers, security, finance, and leadership.
6. Prove the design with tests, SLOs, game days, cost data, and recovery drills.

What strong evidence looks like

- A one-page platform and workload context diagram.
- A cluster-mode decision with rejected alternatives.
- Namespace, IAM, RBAC, and Workload Identity design.
- Network and traffic flow with default-deny policy.
- Resource requests, autoscaling, disruption, and topology policy.
- Supply-chain controls from source to signed image to running Pod.
- SLOs, dashboards, alerts, runbooks, and an upgrade plan.
- RTO/RPO, Backup for GKE restore test, and regional failure plan.
- Cost per request or tenant, not only a monthly invoice.

The key principle

Kubernetes gives a common API for containers; it does not make an application secure, reliable, or inexpensive automatically. GKE removes some platform work, especially in Autopilot, but workload design remains your responsibility.

01 · Kubernetes in simple terms

Kubernetes continuously compares **desired state** with **actual state** and tries to close the gap. You declare what should exist; controllers reconcile it.

The small vocabulary

Object	Plain meaning
Pod	Smallest scheduled unit; one or more tightly coupled containers
Deployment	Keeps stateless Pods running and rolls out versions
StatefulSet	Stable identity and ordered behavior for stateful Pods
DaemonSet	Runs a Pod on selected nodes
Job / CronJob	Finite or scheduled work
Service	Stable network identity for a changing set of Pods
Gateway / Ingress	Routes external or internal application traffic
ConfigMap	Non-secret configuration
Secret	Kubernetes secret object; use careful encryption and access controls
Namespace	Logical scope for names, policy, quota, and team boundaries

Labels identify objects. Selectors find groups of objects. Controllers use selectors, so a wrong label can send traffic to the wrong Pods or prevent a Deployment from managing them.

Requests and limits

A CPU or memory **request** tells the scheduler what a Pod needs and influences capacity and cost. A **limit** is an upper bound. Missing or unrealistic values cause poor scheduling, evictions, throttling, or wasted nodes.

Health

- Startup probes protect slow-starting applications.
- Readiness probes decide whether a Pod receives traffic.
- Liveness probes decide whether a stuck container should restart.

A liveness probe must detect a condition a restart can fix. A readiness failure should remove traffic without creating a restart storm.

The declarative habit

Store reviewed manifests or templates in version control. Avoid making important production changes only with imperative commands. Git history should explain what changed, while audit logs explain who applied it.

02 · How GKE works

A GKE cluster has a Kubernetes **control plane** and **worker capacity**. Google manages the control plane. Your workloads run in Pods on managed compute.

Responsibility boundary

Google operates control-plane infrastructure and the managed GKE service. The customer still owns identities, IAM and RBAC, workload code, container images, Pod security, network policy, data protection, configuration, availability design, and incident response. In Standard, the customer also makes more node and node-pool decisions. In Autopilot, Google manages nodes, scaling, security settings, and many defaults.

Regional and zonal clusters

Autopilot clusters are regional. A regional Standard cluster replicates its control plane across zones and can spread nodes across zones. A zonal Standard cluster has a lower-availability control-plane design. Production systems should start from business availability requirements, not the cheapest cluster shape.

VPC-native networking

GKE is normally VPC-native: Pods and Services use alias IP ranges. Plan Pod, Service, node, and control-plane address space early. IP exhaustion is an architecture problem, not a late operations detail.

GKE and Google Cloud

GKE commonly integrates with Artifact Registry, Cloud Load Balancing, Cloud DNS, Cloud NAT, Cloud Armor, IAM, Secret Manager, Cloud KMS, Cloud Logging, Cloud Monitoring, Managed Service for Prometheus, Cloud SQL, Spanner, Pub/Sub, and Backup for GKE.

Design question

Do not ask only “How many clusters?” Ask which failures, trust boundaries, environments, teams, versions, quotas, and network domains must be isolated. Every extra cluster improves some isolation but adds fleet, delivery, upgrade, policy, and observability work.

03 · Autopilot or Standard

GKE offers two operating modes. The best default is the least operational control that still meets the requirement.

Autopilot

Google manages nodes, node provisioning, scaling, many security settings, and resource defaults. Workloads request CPU, memory, accelerators, and placement; GKE provisions capacity. Autopilot is optimized for most production workloads and is Google's recommended streamlined experience.

Choose it when the team wants Kubernetes APIs without operating node pools. Expect opinionated security constraints, automatic upgrades, and workload-based pricing. Verify compatibility for privileged agents, host access, specialized networking, or unsupported features.

Standard

You select node pools, machine types, operating systems, scaling, upgrades, taints, accelerators, and many low-level settings. Choose Standard when a proven requirement needs node-level control, a privileged DaemonSet, specialized hardware behavior, or a feature unavailable in Autopilot.

Decision table

Requirement	Start with
Most web, API, worker, and batch containers	Autopilot
Minimal platform operations	Autopilot
Direct node configuration or privileged host agent	Standard
Exact node purchase/capacity strategy	Standard
Kubernetes learning lab needing node access	Standard or local Kubernetes

Common error

“Standard gives more control” is not enough. Control has a carrying cost: capacity planning, upgrades, node security, repair, bin packing, and on-call knowledge. Record the exact capability that pays for that cost.

Official source: GKE Autopilot overview.

04 · Cloud foundation and identity

The safest cluster is still unsafe if the organization and identity foundation is weak.

Resource hierarchy

Google Cloud resources sit under organization, folders, and projects. Use folders for policy and ownership. Use projects as strong boundaries for IAM, billing, quotas, APIs, and failure. Separate production from non-production and consider dedicated networking, security, logging, and platform projects.

Two permission systems

Google Cloud IAM controls access to projects, GKE APIs, and Google Cloud resources. Kubernetes RBAC controls actions inside a cluster. A user may have permission to discover a cluster but not edit workloads—or the reverse through poor configuration. Design both paths together.

Workload identity

Use Workload Identity Federation for GKE so Pods obtain short-lived identities for Google Cloud APIs. Do not distribute service-account key files inside images or Kubernetes Secrets.

Boundaries

- Give people group-based access, not individual permanent grants.
- Use least-privilege IAM and namespace-scoped RBAC.
- Separate platform administrators from application deployers.
- Protect powerful Kubernetes groups such as `system:masters`.
- Apply Organization Policy constraints and audit changes.
- Centralize Cloud Audit Logs in a protected logging project.

Namespace warning

A namespace is useful for names, quota, RBAC, and policy, but it is not always a hard hostile-tenant boundary. Strongly untrusted tenants may require separate clusters or projects, depending on the threat model.

05 · GKE networking and isolation

GKE networking connects Pods, Services, Google Cloud load balancers, VPCs, the internet, and hybrid systems.

Address plan

Plan non-overlapping primary node ranges and secondary Pod/Service ranges. Estimate maximum Pods per node, cluster growth, peering, Shared VPC, hybrid networks, and future clusters. Use flexible Pod CIDR features where appropriate, but do not depend on late rescue from a poor IP plan.

Private design

Private nodes have no external IP addresses. Use Cloud NAT for controlled outbound internet access and Private Google Access for Google APIs where appropriate. Restrict control-plane access with DNS-based endpoints, authorized networks, or private access according to the current GKE model.

Traffic controls

VPC firewall rules control virtual-machine and network-level flows. Kubernetes NetworkPolicy controls Pod-level communication when enforcement is enabled. GKE Dataplane V2, based on eBPF, is the recommended dataplane and is the Autopilot default.

Start with default deny, then allow required DNS, identity metadata, ingress, monitoring, and service-to-service paths. Test policies before enforcement.

Exposing services

Use Gateway API or Ingress to provision application load balancing. Use `Service` type `LoadBalancer` for L4 traffic. Put Cloud Armor in front of public HTTP services. Use Cloud DNS for names and Certificate Manager or managed certificates for TLS.

Official source: GKE networking best practices.

06 · Designing Kubernetes workloads

A production workload is more than a Deployment and a Service.

Stateless service baseline

Use a Deployment with multiple replicas, rolling-update strategy, readiness, startup and liveness probes, resource requests, sensible limits, topology spread, Pod disruption budget, security context, dedicated service account, and controlled configuration.

Graceful behavior

When Kubernetes terminates a Pod it sends a signal and waits for a grace period. The application should stop accepting new work, finish or hand off safe work, close connections, and exit. Align load-balancer draining, readiness, signal handling, and termination grace.

Configuration

Keep environment-specific configuration outside images. ConfigMaps are not secret stores. For sensitive values, prefer Secret Manager integration or carefully governed Kubernetes Secrets with encryption, access, and rotation.

Stateful work

Kubernetes can run stateful software, but a managed database often reduces risk. If state must run in GKE, define stable identity, storage class, replication, anti-affinity, quorum, backup, upgrades, disruption, and recovery. A StatefulSet does not create database replication for you.

Batch and events

Use Jobs and CronJobs for finite work, with concurrency and history limits. For event-driven scale-to-zero use cases, evaluate Knative-based platforms or KEDA where supported and governed. Never let an uncontrolled retry create an infinite expensive workload.

07 · Scheduling, capacity, and autoscaling

Scheduling places Pods on capacity that satisfies resources and placement rules.

Four scaling loops

- Horizontal Pod Autoscaler (HPA) changes Pod replicas from metrics.
- Vertical Pod Autoscaler (VPA) recommends or changes Pod requests.
- Cluster autoscaler changes Standard node-pool size.
- Node auto-provisioning or Autopilot creates suitable capacity.

These loops interact. HPA cannot help if Pods are unschedulable and no capacity can arrive. Scaling from zero may add startup delay. Downstream systems may fail before the cluster reaches its maximum.

Placement

Use node selectors or affinity for required hardware. Use taints and tolerations to reserve capacity. Use topology spread and Pod anti-affinity for fault distribution. Avoid rules so strict that Pods cannot schedule during failure.

Resources

Set requests from measured usage plus headroom. A too-low CPU request harms scheduling and HPA behavior; a too-high request wastes capacity. Memory limits can cause OOM kills. CPU limits can throttle. Treat defaults as a starting point.

Priority

PriorityClass helps important Pods schedule under pressure, but preemption can remove lower-priority work. Reserve it for explicit business importance and test overload behavior.

Architect test

Load-test the entire scaling path: metrics collection, HPA reaction, node provisioning, image pulling, startup, readiness, downstream capacity, and scale-down. A YAML value is not proof of elasticity.

08 · Storage and data

Containers are disposable. Business data is not.

Persistent storage

PersistentVolumeClaims request storage through a StorageClass. GKE commonly uses Persistent Disk or Hyperdisk CSI drivers. Zonal volumes live in one zone; regional persistent disks synchronously replicate across zones where supported. Filestore provides managed NFS. Cloud Storage FUSE exposes object data through a file interface for suitable workloads, but object semantics are not a normal POSIX disk.

Database decision

Prefer Cloud SQL, AlloyDB, Spanner, Bigtable, Memorystore, Firestore, or another managed data service when it meets requirements. Running a database in GKE is reasonable only when Kubernetes portability or special software needs justify operating replication, patching, backup, failover, and capacity.

Connection design

Use private connectivity where appropriate. Prefer Workload Identity over key files. Use connection pooling and bounded concurrency. A thousand new Pods can become a database outage if every Pod opens many connections.

Backup

Backup for GKE captures selected Kubernetes resources and persistent volume data. It does not capture cluster configuration, node pools, or container images. Preserve infrastructure-as-code and images separately. Restore into a prepared target cluster and test the full application.

Official source: Backup for GKE.

09 · GKE security architecture

Use layers: identity, admission, workload, network, data, supply chain, detection, and response.

Harden access

Use groups, IAM, Kubernetes RBAC, and least privilege. Use Workload Identity Federation for Pods. Restrict control-plane and node exposure. Avoid broad `cluster-admin`, service-account keys, default service accounts, and anonymous access.

Harden Pods

Run as non-root, drop Linux capabilities, prevent privilege escalation, use a read-only root filesystem where possible, select seccomp, avoid host namespaces and hostPath, and enforce Pod Security Standards or

Policy Controller rules. Autopilot enforces many constraints by default.

Protect the supply chain

Build in controlled pipelines, scan source and images, store images in Artifact Registry, generate provenance/SBOMs, sign approved artifacts, and use Binary Authorization to restrict deployment. Pin images by digest for high-assurance releases.

Network and data

Use private nodes, Dataplane V2, default-deny NetworkPolicy, Cloud Armor, encryption, Secret Manager, Cloud KMS, and minimal egress. Remember that network policy needs an enforcement dataplane.

Detect and respond

Use Cloud Audit Logs, GKE security posture, Security Command Center, Cloud Logging, Cloud Monitoring, and policy/audit findings. Define containment: disable identity, quarantine namespace/workload, block image, preserve evidence, rotate secrets, redeploy trusted artifacts, and review blast radius.

Official source: GKE security best practices.

10 · Traffic, gateways, and service mesh

Kubernetes Services create stable discovery for changing Pods.

Traffic choices

- ClusterIP: internal virtual service.
- Headless Service: direct endpoint discovery.
- LoadBalancer Service: Layer 4 load balancing.
- Gateway API / Ingress: Layer 7 routing and Google Cloud load balancers.
- Multi Cluster Services/Ingress: selected fleet-wide discovery and traffic.

Use readiness gates and health checks so traffic reaches healthy Pods. Preserve client IP only when the requirement justifies configuration and trade-offs.

Gateway API

Gateway API separates infrastructure ownership from application routes. A platform team can own Gateway classes and shared gateways while application teams own permitted HTTPRoutes. This is usually clearer than letting every namespace create unrelated load balancers.

Service mesh

A mesh can add mutual TLS, traffic policy, telemetry, and service identity. Cloud Service Mesh can manage parts of this. A mesh also adds proxies or ambient components, policy, upgrades, latency, resource use, and debugging complexity. Adopt it for explicit cross-service requirements, not because microservices exist.

Resilience

Use timeouts, bounded retries, circuit breaking, load shedding, graceful degradation, and idempotency in the application and traffic layer. Do not allow multiple retry layers to multiply requests.

11 · Fleets and platform governance

A fleet logically groups clusters so teams can apply multi-cluster capabilities and consistent governance.

Fleet capabilities

Fleets support Config Sync, Policy Controller, team management, multi-cluster traffic, fleet Workload Identity, Cloud Service Mesh, and centralized views. Some capabilities assume namespace, service, or identity sameness across clusters. Treat those assumptions as architecture decisions.

Platform team product

The platform should offer paved roads:

- approved cluster/project blueprints;
- namespace and identity onboarding;
- secure workload templates;
- shared Gateway, DNS, certificate, and observability patterns;
- CI/CD and Artifact Registry;
- policy with clear errors and exceptions;
- cost allocation and quotas;
- runbooks, SLOs, support, and version lifecycle.

Config Sync and Policy Controller

Config Sync reconciles declarative configuration from Git, OCI, or Helm sources. Policy Controller evaluates and enforces constraints. Roll policy out in audit or dry-run mode, then canary clusters, then wider enforcement. One wrong global policy can stop every deployment.

Cluster count

Use one cluster where teams can safely share and isolation is adequate. Add clusters for environment, regulatory, region, version, failure, or strong tenant boundaries. Do not create one cluster per team without measuring the operational and cost impact.

Official source: GKE fleets.

12 · Delivery, GitOps, and the software supply chain

Production delivery should turn reviewed source into a traceable running image.

Golden path

1. Developer changes source and tests.
2. CI builds once, scans, creates provenance and an SBOM.
3. Artifact Registry stores the immutable image.
4. Policy checks manifests and image trust.
5. A delivery controller deploys the same digest through environments.
6. Readiness and SLO signals decide promotion.
7. Rollback or roll-forward uses a known good artifact.

Cloud Build or another approved CI system can build. Cloud Deploy, Config Sync, Argo CD, or another governed controller can deliver. Choose one clear owner for desired state so two reconcilers do not fight.

Rollout strategy

Rolling updates are the baseline. Use canary or blue/green when risk and traffic control justify them. Define `maxSurge`, `maxUnavailable`, readiness, draining, and rollback behavior from capacity and availability needs.

Policy

Validate schemas, required labels, security contexts, resources, allowed registries, signed images, and prohibited host access before production. Make policy fast, explainable, versioned, and testable.

Change safety

Separate application rollout from irreversible data migration. Use backward- compatible schemas and expand/migrate/contract steps. A Kubernetes rollback cannot undo corrupted data.

13 · Observability and SRE

Observability answers what users experience, why it happens, and what to do.

Signals

Collect metrics, logs, traces, events, audit logs, profiles, and business outcomes. GKE integrates with Cloud Logging and Cloud Monitoring. Managed Service for Prometheus supports Prometheus metrics and queries. OpenTelemetry provides portable instrumentation.

SLO first

Define service-level indicators such as successful request ratio and latency. Set an SLO and error budget. Alert on fast or sustained error-budget burn rather than every transient metric. Infrastructure alarms should be actionable causes.

Kubernetes signals

Watch unschedulable Pods, restart/OOM rate, readiness, CPU throttling, memory, node pressure, HPA limits, control-plane/API errors, workload latency, queue age, and persistent-volume behavior.

Cardinality and cost

Unbounded labels such as user ID create expensive, unusable metrics. Structure logs, set retention, sample traces deliberately, and keep labels bounded.

Runbooks

Every urgent alert needs an owner, impact statement, dashboards, safe diagnosis, mitigation, escalation, and verification. Automate repeatable safe actions, but keep audit and rollback.

Learning loop

Blameless postmortems identify contributing conditions and concrete follow-up. Track action ownership. Reliability improves when incident learning changes code, policy, tests, and platform defaults.

14 · Reliability, upgrades, and disaster recovery

GKE control-plane availability is only one part of application reliability.

Regional baseline

Use regional clusters for important production workloads, spread Pods and capacity across zones, keep multiple replicas, define disruption budgets, and remove single-zone storage or downstream dependencies when the requirement demands it.

Upgrades

Use release channels, maintenance windows, and exclusions. GKE upgrades all clusters over time; you can control timing but cannot stay unsupported forever. Test new versions and APIs in lower environments or canary clusters. Monitor deprecated APIs and ensure Pod disruption budgets do not make maintenance impossible.

Failure design

Test Pod deletion, node drain, zone capacity loss, dependency timeouts, DNS failure, quota exhaustion, image-pull failure, and bad configuration. Use topology spread, graceful termination, safe retries, and capacity headroom.

Disaster recovery

Start from RTO and RPO. Preserve:

- clusters and cloud infrastructure as code;
- manifests and policy in version control;
- images and provenance in durable registries;
- workload resources and volumes with Backup for GKE where appropriate;
- external database backups and replication;
- DNS, secrets, certificates, and recovery permissions.

Restore into a predesigned target, validate data and traffic, and practice. A backup plan that has never restored is a hope, not evidence.

15 · Cost, performance, and sustainability

Cost is a workload property created by requests, replicas, nodes, storage, traffic, observability, and management choices.

Autopilot economics

Autopilot usually charges from requested workload resources and manages bin packing. Accurate requests are still essential. Standard charges for node capacity, so utilization, machine type, node pools, autoscaling,

reservations, committed use, and Spot VMs matter.

Practical controls

- Label namespaces and workloads for cost allocation.
- Set ResourceQuota and LimitRange.
- Right-size requests from measurements and VPA recommendations.
- Use HPA and node scaling with tested boundaries.
- Schedule non-production down where possible.
- Use Spot Pods/VMs only for interruption-tolerant work.
- Control logging volume and retention.
- Review cross-zone, internet, and load-balancer data transfer.
- Track cost per request, customer, job, or training run.

Performance

Measure p50/p95/p99 latency, throughput, saturation, startup, scaling, image pull, DNS, service mesh overhead, storage, and downstream limits. Tune the application before only buying more CPU.

Sustainability

Efficient requests, autoscaling, managed services, modern machine families, fewer idle clusters, shorter jobs, sensible data retention, and efficient code reduce both spend and resource use.

Trade-off

The cheapest single-zone cluster may violate availability. Maximum redundancy may waste budget. State the business outcome purchased by each reliability cost.

16 · Real-life scenario: Global Service Platform

This fictional scenario shows how a GKE architect supports a global enterprise.

Situation

ToolWorks sells equipment in 55 countries. Customer web/mobile channels use APIs running on GKE. SAP owns orders and product prices, Salesforce owns sales activity, and Workday owns employee/territory data. Campaign traffic grows ten times. Teams currently deploy manually and one bad cluster role can affect all applications.

Goals: faster delivery, 99.95% checkout availability, controlled personal data, no duplicate orders, recovery within one hour with no more than five minutes of data loss, and clear platform cost.

Architecture decision

Use separate production and non-production projects in a governed folder, Shared VPC, central logging/security projects, and regional Autopilot clusters. Autopilot reduces node operations; no workload has a proven node-level need. Fleets group clusters, while Config Sync and Policy Controller roll out platform configuration through audit, canary, and enforcement stages.

Cloud Load Balancing and Gateway API expose services; Cloud Armor protects the public edge. Private nodes, Cloud NAT, Private Google Access, Dataplane V2, and default-deny NetworkPolicy control paths. Each workload has a dedicated Kubernetes service account mapped through Workload Identity Federation.

Artifact Registry holds scanned immutable images. CI builds once and produces provenance. Binary Authorization permits approved images. Cloud Deploy canaries releases and stops promotion when SLO burn is high.

Enterprise integration

Customer APIs commit channel transactions to a managed database and publish orders to Pub/Sub. A GKE worker consumes idempotently and calls the governed SAP integration API. Pub/Sub buffers campaign spikes and failed messages enter a dead-letter path with reconciliation. Salesforce and Workday changes arrive through approved APIs/files into normalized events. No service reads another system's database directly.

Reliability

Deployments use at least three replicas, topology spread, readiness/startup probes, graceful termination, and disruption budgets. HPA scales from request and queue signals. Autopilot provides capacity; load tests include cold scale-up. Cloud SQL/Spanner design follows data requirements and the agreed RPO.

Backup for GKE protects Kubernetes resources and eligible volumes; IaC rebuilds clusters and cloud services; registries preserve images. A prepared secondary region receives restored configuration/data during scheduled game days. DNS and traffic failover are tested, not assumed.

Operations and outcomes

Cloud Monitoring dashboards show checkout SLO, latency, Pub/Sub backlog, SAP errors, Pod restarts, unschedulable Pods, rollout status, security findings, and cost per order. Measure duplicate orders, recovery time, deployment frequency, change failure rate, policy violations, and idle requested CPU.

Important trade-offs

- Autopilot removes node management but limits privileged/node-level patterns.
- Async SAP integration protects checkout but introduces a pending order state.
- A fleet creates consistency but a bad global policy creates wide blast radius.
- Multi-region recovery meets the business target but requires rehearsed data, identity, secrets, DNS, and operations—not only a second cluster.

17 · Interview, certification, and glossary

Use **CLUSTER** to answer design questions:

- **C**ontext: outcome, users, data, scale, existing systems.
- **L**evels of reliability: SLO, RTO, RPO, failure scope.
- **U**sers and identities: IAM, RBAC, workload identity, tenants.
- **S**ervices and state: workloads, traffic, storage, dependencies.
- **T**rust and threats: supply chain, network, policy, secrets.
- **E**lasticity and economics: requests, scaling, quota, unit cost.
- **R**ollout and recovery: delivery, observability, upgrades, DR.

Professional Cloud Architect checkpoint

Google's standard Professional Cloud Architect exam is two hours with 50–60 multiple-choice and multiple-select questions. It includes two case studies; case-study questions make up 20–30%. This site uses one 60-question original GKE-heavy mock and two fictional case studies. It is practice, not copied exam content.

Official source: Professional Cloud Architect certification.

Compact glossary

Term	Plain meaning
Autopilot	GKE mode where Google manages node infrastructure and many defaults
Standard	GKE mode with customer-managed node pools and more control
Cluster	Kubernetes control plane plus registered compute
Node	Worker machine that runs Pods
Pod	Smallest Kubernetes scheduled unit
Deployment	Reconciles and rolls out stateless Pods
StatefulSet	Workload with stable identity and ordered behavior

Term	Plain meaning
Service	Stable discovery and virtual endpoint for Pods
Gateway API	Kubernetes traffic-routing API with role separation
HPA	Changes Pod replica count from metrics
VPA	Recommends or changes Pod resource requests
PDB	Limits voluntary concurrent Pod disruption
RBAC	Kubernetes role-based authorization
IAM	Google Cloud authorization
WIF	Workload Identity Federation; keyless workload access
CSI	Container Storage Interface
CNI	Container Network Interface
SLO	Measurable reliability target
RTO	Maximum acceptable recovery time
RPO	Maximum acceptable data loss measured in time

Official references

- GKE documentation
- GKE best practices
- Google Cloud Well-Architected Framework
- GKE security best practices
- Autopilot and Standard comparison
- GKE fleets

Features, versions, limits, and certification formats change. Verify production details in current official documentation.