Week 8: Enterprise-Scale Virtualization and Management

NT524 — Cloud Architecture and Security

PhD. Nguyen Ngoc Tu

October 29, 2025

platform model — mô hình nền tảng; artifact management — quản trị hiện vật; orchestration — dàn dựng; capacity planning — hoạch định công suất

Learning Objectives

- Define the **platform model**—the orchestrated layer spanning the **control**, **compute**, **storage** planes, and the **network fabric**.
- Build secure artifact pipelines for **VM images** and **containers**, including signing, verification, and promotion.
- Provision infrastructure using **Terraform** or **Heat**, configure and deploy applications with **Ansible**, and manage lifecycle operations through **GitOps**.
- Evaluate and select **networking** and **storage** patterns with clear performance and reliability trade-offs.
- Apply **elastic scaling strategies** — horizontal, vertical, scheduled/predictive, and burst/over-provisioning models.

---

control plane — mặt phẳng điều khiển; compute plane — mặt phẳng tính toán; storage plane — mặt phẳng lưu trữ; network fabric — hạ tầng mạng

Module Map

1. Platform Model
2. Artifact Management
3. Orchestration
4. Network: Isolation, Scale, and Monitoring
5. Autoscaling: IaC, Resource Allocation & Management
6. Storage: Model Choice, Security, and Incident Response
7. Industrial Perspective

Platform Model — Orchestrated Concept (Overview)

Core idea: The platform *orchestrates* a closed control loop that continuously reconciles *desired state* with *real* capacity, policy, and risk.

- **Declare intent** → workloads, SLOs, budgets, and policies as code.
- **Map capabilities** → bind intent to topology/hardware (GPU/MIG, SR-IOV, storage tiers).
- **Place & connect** → schedule + program east–west/north–south paths.
- **Protect & govern** → identity, isolation, provenance, keys; quotas and guardrails.
- **Observe & adapt** → telemetry-driven scale, healing, and safe rollbacks across regions.

---

orchestrate — điều phối; closed control loop — vòng điều khiển khép kín; desired state — trạng thái mong muốn; east–west/north–south — luồng đông–tây/bắc–nam; guardrails — hàng rào bảo vệ; SLO — Service Level Objective

Example — Control Loop in Action

Scenario: A tenant requests a GPU workload with cost and policy constraints.

- **Intent (declaration):**
  - `spec: resources.gpu = 2`
  - `budget: $10/hr, SLO: latency < 50ms`
- **Controller decision:**
  - Match intent with available Multi-Instance GPU (MIG) profiles.
  - Check policy: tenant quota & isolation.
- **Actuators(actions) (actions):**
  - Scheduler places workload on node with free MIG slice.
  - Admission webhook (Gatekeeper) validates cost & isolation (comply with policy).
  - Runtime class sets sandboxed container or VM boundary.
- **Signals(feedbacks) (feedback):**
  - Telemetry shows GPU utilization & latency metrics.
  - Drift detected → controller rebalances or throttles workload.
- **Outcome:** System self-adjusts to meet SLO within policy and budget — achieving convergence to intent.

---

controller — bộ điều khiển; actuator — bộ thực thi; telemetry — số liệu giám sát; convergence — hội tụ trạng thái; Admission webhook — Kiểm soát tiếp nhận (cổng gác kiểm tra tuân thủ)

Control Plane — Reconciliation Contract

Objective: Continuously converge platform state to intent while enforcing policy and tenancy boundaries.

- **Inputs (intent)**: tenants/projects, quotas/budgets, catalogs, admission & policies-as-code.
- **Actuators(actions)**: API controllers, schedulers/placement, admission webhooks (OPA/Gatekeeper).
- **Signals(feedbacks)**: audit/drift events, saturation of shared services, policy violations.
- **Outcome**: safe, portable control across providers without leaking brand specifics.

---

reconciliation — đồng bộ hoá; admission control — kiểm soát tiếp nhận; tenancy — đơn vị thuê; drift — trôi cấu hình; OPA — Open Policy Agent

Control Plane — Cross-Platform

Function	OpenStack	AWS	Azure	GCP/VMware
Identity/RBAC	Keystone	IAM	Entra ID/RBAC	IAM / vCenter SSO
Image/Catalog	Glance	AMI	Managed Images	Compute Images / Content Library
Network Ctrl	Neutron	VPC	VNet	VPC / NSX-T
LB / API GW	Octavia	ELB/ALB/NLB	LB/AppGW	GCLB / NSX ALB
Metadata	Nova Metadata	IMDS	IMDS	IMDS / Guest Info

identity — danh tính; RBAC — phân quyền theo vai trò; metadata service — dịch vụ siêu dữ liệu; load balancer — cân bằng tải

Example — Compute Scheduling Loop in Action

Scenario: Multi GPU workloads compete for limited nodes under cost and isolation policies.

- **Inputs (requests):**
  - `teamA-train`: 2 GPUs, high priority, Service Level Objective (SLO): 50ms latency.
  - `teamB-infer`: 1 GPU, low priority, cost-sensitive.
  - Topology: 2 nodes with Non-Uniform Memory Access (NUMA) zones and MIG partitions.
- **Controller decision:**
  - Match GPU profiles (MIG/vGPU/passthrough) to workload specs.
  - Evaluate affinities & anti-affinities for locality and isolation.
  - Apply overcommit policy if total demand exceeds supply.
- **Actuators(actions) (actions):**
  - **VM/node placement:** assign `teamA` to node with direct PCIe GPU, `teamB` to MIG slice.
  - **Bin-packing (resource packing):** consolidate small jobs to improve utilization.
  - **Preemption(resource takeover):** evict(dismiss) low-priority pods when SLO violation detected.
  - **Runtime class:** enforce sandbox or VM for tenant isolation.
- **Signals(feedbacks) (feedback):**
  - GPU utilization 92%, queue wait increasing $\rightarrow$ scheduler throttles or rebalances.
  - Thermal sensor warning $\rightarrow$ migrate one job to cooler node.
- **Outcome:** Cluster achieves near-optimal GPU usage with fair scheduling and predictable SLOs under budget.

Compute Plane — Orchestrated Scheduling

Objective: Place workloads to meet SLO and cost while honoring isolation and topology.

- **Inputs**: requests/limits, affinities/anti-affinities, topology/NUMA, accelerator profiles (MIG/vGPU/passthrough).
- **Actuators(actions)**: VM/node placement, bin-packing/overcommit, preemption/queues, runtime class (ns/VM/sandbox).
- **Signals(feedbacks)**: utilization/queue time, thermal/device health, locality hits/misses.
- **Outcome**: predictable performance & fairness under policy and budget.

SLO — mục tiêu mức dịch vụ; affinity — ưa thích vị trí; overcommit — cấp phát vượt; preemption — truất quyền; NUMA — bộ nhớ không đồng nhất; Affinity/anti-affinities (Isolation-level): isolation at the placement level

Compute Plane — Hypervisors, Nodes, Accelerators

- **Hypervisors**: KVM (OpenStack/GCP), Nitro hypervisor (AWS), Hyper-V (Azure), ESXi (VMware).
- **VM profiles**: general, compute-optimized, memory-optimized, HPC, confidential VM.
- **Kubernetes**: managed (EKS/AKS/GKE) or self-managed on VMs/bare metal; cluster autoscaler.
- **Accelerators**: NVIDIA GPUs (A100/H100), vGPU/MIG, passthrough; FPGA options vary by provider.
- **Placement**: labels/taints (K8s), flavors/extra_specs (OpenStack), instance families/SKUs (CSPs).

---

hypervisor — trình siêu giám sát; passthrough — chuyển thẳng thiết bị; instance family — họ cấu hình; flavor — cấu hình ảo hoá

Example — Storage Plane in Action

Scenario: A multi-zone application requires encrypted, highly available storage.

- **Inputs:**
  - Persistent Volume Claim(PVC) requests a `gold` storage class with `RPO=0`, `RTO<15min`.
  - Encryption required with tenant-managed key (via KMS).
  - Locality hint: prefer same zone as compute pods.
- **Actuators(actions):**
  - CSI controller provisions SSD-backed volume from the `gold-tier`.
  - Automatic replication to a standby zone using Cinder or Ceph RBD mirroring.
  - Keys provisioned and rotated through Vault/KMS.
- **Signals(feedbacks):**
  - Monitor I/O latency and replication lag.
  - Detect quota nearing limit $\rightarrow$ trigger capacity alert.
- **Outcome:** Data stays encrypted, resilient, and policy-compliant even under failure or scale events.

CSI(Container Storage Interface) — giao diện lưu trữ containe; Ceph RBD — đĩa khối phân tán; mirroring — bản sao dữ liệu song song

Storage Plane — Data Orchestration

Objective: Bind claims to tiers with protection, encryption, and locality as part of the control loop.

- **Inputs**: Persistent Volume Claims(PVCs)/storage classes, Recovery Point Objective (RPO)/Recovery Time Objective(RTO), encryption and immutability requirements, locality hints.
- **Actuators(actions)**: Container Storage Interface(CSI)/Cinder dynamic provisioning, snapshots/clones, replication, KMS-backed keying.
- **Signals(feedbacks)**: latency/tail I/O, replication lag, durability SLO, quota usage.
- **Outcome**: recoverable, compliant data services with cost/tier awareness.

PVC — yêu cầu khối lưu trữ; storage class — hạng lưu trữ; RPO/RTO — mục tiêu mất dữ liệu/thời gian phục hồi; KMS — quản lý khoá

Storage Plane — Block, File, Object

Type	OpenStack/VMware	AWS/Azure	GCP
Block	Cinder / vSAN, VMFS	EBS / Managed Disks	Persistent Disk
File	Manila / NFS	EFS / Azure Files	Filestore
Object	Swift	S3 / Blob Storage	Cloud Storage
Snap/Backup	Cinder Snap/Backup	Snapshots/Backup Vault	Snapshots/Backup
Encryption	Barbican/KMS, vTPM	KMS/CMK, vTPM	KMS, CMEK

block storage — khối; file storage — tệp; object storage — đối tượng; snapshot — ảnh chụp; KMS — quản lý khoá

Example — Network Fabric in Action

Scenario: A multi-tenant AI platform exposes model APIs securely across regions.

- **Inputs:**
  - Service marked as `external` via Gateway API with `zero-trust` egress policy.
  - North–south traffic passes through L7 gateway (HTTPS, JWT validation).
  - East–west traffic restricted to project namespace using CNI policies.
- **Actuators(actions):**
  - CNI (OVN/eBPF) programs pod routes and applies NetworkPolicy for isolation.
  - Mesh layer (ambient mode) enforces mutual TLS (mTLS) and telemetry collection.
  - SR-IOV path enabled for GPU training jobs requiring low latency.
- **Signals(feedbacks):**
  - Observe end-to-end latency and packet loss via mesh metrics.
  - Detect policy hits/violations and egress bandwidth cost per tenant.
- **Outcome:** Secure, low-latency, cost-aware connectivity with consistent policies across regions.

CNI(Container Network Interface) — giao diện mạng container; OVN — Open Virtual Network; eBPF — mở rộng bộ lọc gói; SR-IOV — chia sẻ I/O ảo hoá; mTLS — xác thực hai chiều

Network Fabric — Intent-Driven Connectivity

Objective: Realize connectivity/security intent as programmable datapaths with SLO guarantees.

- **Inputs**: service exposure (internal/external), zero-trust policies, egress restrictions, multi-region topology.
- **Actuators(actions)**: CNI routes/policies (eBPF/OVN), Gateway API (L4/L7), mesh/ambient, SR-IOV/DPDK fast paths.
- **Signals(feedbacks)**: latency/loss, policy hits, congestion, egress cost.
- **Outcome**: predictable latency, least-privilege connectivity, cost-aware routing.

---

zero-trust — không tin cậy mặc định; egress — lưu lượng ra; service mesh/ambient — lưới dịch vụ/mesh không sidecar; Gateway API — chuẩn cổng dịch vụ

Example — Capability Mapping in Action

Scenario: A training job requires specific GPU profiles, fast network I/O, and compliant storage.

- **Inputs:**
  - Job requests `A100-40GB` GPUs with MIG partition `1g.5gb`.
  - Requires SR-IOV-enabled NIC for RDMA traffic and high-throughput data exchange.
  - Storage class must belong to the `compliant-gold` allowlist with QoS tier `premium`.
- **Actuators(actions):**
  - Scheduler filters nodes by labels/taints and GPU plugin reports (NVIDIA device plugin).
  - Admission webhook validates flavor `extra_specs` and compliance policy.
  - Device plugin binds MIG instance and configures SR-IOV VF for the pod/VM.
- **Signals(feedbacks):**
  - Allocation success/failure events streamed to telemetry service.
  - Monitor GPU temperature, NIC throughput, and storage IOPS against QoS target.
  - Automatic quarantine of unhealthy devices reported by node exporter.
- **Outcome:** Workload is scheduled only on nodes that exactly match declared capabilities — ensuring predictable performance and compliance.

MIG — chia tách GPU đa phiên bản; RDMA — truy cập bộ nhớ từ xa trực tiếp; VF — giao diện ảo SR-IOV; IOPS — số thao tác vào/ra mỗi giây

Capability Mapping — Feature Binding Loop

Objective: Advertise → schedule → bind → verify platform features as part of placement.

- **Inputs**: GPU type/MIG profile, NIC offloads (SR-IOV), storage QoS, compliance allowlists.
- **Actuators(actions)**: node labels/taints, flavor `extra_specs`, admission checks, device plugins.
- **Signals(feedbacks)**: allocation success/failure, QoS adherence, device health/telemetry.
- **Outcome**: correct-by-construction workload deployments.

---

capability mapping — ánh xạ năng lực hạ tầng; allowlist — danh sách cho phép; labels/taints — nhãn/ô nhiễm; QoS — chất lượng dịch vụ

Example — Policy-as-Code and Governance in Action

Scenario: A multi-team research platform enforces least-privilege access, budget limits, and compliance through automation.

- **Inputs:**
  - Organization defines tenants (projects) with separate budgets and IAM roles.
  - Compliance policies (SCP/Organization Policy) require encryption-at-rest and signed artifacts.
  - Guardrails specify who can deploy privileged workloads or access production namespaces.
- **Actuators(actions):**
  - OPA/Gatekeeper admission webhook validates deployment manifests against policy bundles.
  - IAM bindings grant least-privilege roles automatically based on team or namespace.
  - Secrets rotated through KMS and provenance verified before promotion to production.
- **Signals(feedbacks):**
  - Monitor access denials and configuration drift across tenants.
  - Track budget burn and quota usage via telemetry dashboards.
  - Audit logs feed into SIEM to detect policy violations or anomaly access.
- **Outcome:** Consistent governance, verifiable compliance, and secure operations — where safe defaults ("paved road") are enforced automatically, and exceptions are reviewed under control.

Tenancy, IAM, and Policy-as-Code — Guardrails

Objective: Enforce least-privilege and compliance continuously in the same loop as placement.

- **Inputs**: org structure, roles, budgets, compliance controls (SCP/Organization Policy).
- **Actuators(actions)**: IAM bindings, OPA/Gatekeeper admission, secret rotation with KMS, provenance/signing.
- **Signals(feedbacks)**: access denials, drift/vulns, budget burn, audit trails.
- **Outcome**: provable governance with paved-road defaults and controlled exceptions.

---

least privilege — đặc quyền tối thiểu; OPA — chính sách mở; provenance — nguồn gốc, xuất sứ; paved road — con đường mặc định an toàn

Example — Progressive Delivery and Runtime Control

Scenario: A production ML service runs across multiple zones with progressive rollout and self-healing.

- **Inputs:**
  - Error budget of `0.1%` downtime with daily SLO targets.
  - Scheduled upgrade window for model-serving deployment.
  - Game-day simulation prepared to test failover and backup restore paths.
- **Actuators(actions):**
  - Canary rollout deploys new model version to 5% of traffic before global release.
  - Blue-green environment ready for rollback if latency exceeds threshold.
  - Auto-remediation script restarts unhealthy pods and scales replicas within budget.
- **Signals(feedbacks):**
  - Monitor golden signals: latency, traffic, error rate, and saturation.
  - Observe failover metrics (RTO/RPO) and verify recovery time post-chaos test.
  - Alert if error budget burn rate exceeds 10% threshold.
- **Outcome:** Reliable, cost-aware operations that maintain SLO compliance even during failures — achieving faster recovery and safer releases.

---

progressive rollout — triển khai từng bước; rollback — quay lui phiên bản; burn rate — tốc độ tiêu hao ngân sách lỗi; failover — chuyển đổi dự phòng; chaos test — kiểm thử hỗn loạn

Failure Domains & Operations — Runtime Control

Objective: Sustain availability and delivery velocity via progressive change under error budgets.

- **Inputs**: SLO/error budgets, upgrade windows, chaos/game-day scenarios, backup policies.
- **Actuators(actions)**: rolling/blue-green, canary + surge capacity, auto-remediation/runbooks.
- **Signals(feedbacks)**: golden signals (latency, traffic, errors, saturation), failover health, recovery time.
- **Outcome**: resilient releases, rapid recovery, and cost-aware capacity planning.

---

error budget — ngưỡng lỗi cho phép; blue-green — triển khai song song hai stack (stack cũ và stack mới); canary — triển khai thử nghiệm một phần người dùng / nodes; auto-remediation — tự khắc phục; golden signals — tín hiệu vàng

Example — Unified Artifact Taxonomy and Lifecycle

Scenario: An enterprise platform team standardizes all build artifacts with a consistent taxonomy and signing pipeline.

- **Images:**
  - Golden VM templates built by Packer and stored in OpenStack Glance and AWS AMI.
  - Each image version tagged by Git commit and signed with KMS key before promotion.
- **Containers & Open Container Initiative (OCI) standards**
  - Docker/Helm/OPA bundles stored in Harbor (OCI registry) with provenance metadata.
  - WebAssembly filters and policy modules follow the same signing and SBOM verification policy.
- **System Packages:**
  - Internal APT/YUM repos mirror upstream sources with organization baselines.
  - Config bundles and OS hardening scripts managed as immutable packages.
- **Data & Schemas:**
  - Flyway pipelines control schema migrations; masked snapshots stored as versioned dumps.
  - Each dataset tagged with classification (public/internal/confidential) for policy enforcement.
- **Governance Artifacts:**
  - SBOMs (SPDX/CycloneDX) generated at build time and verified via in-toto attestations.
  - All signatures traceable to CI provenance for SLSA Level 3 compliance.
- **Outcome:** Unified artifact taxonomy ensures consistent promotion, verification, and compliance across multi-cloud environments.

Artifact Management — Scope & Taxonomy

Objective: Define the artifact surface area and taxonomy to standardize how we build, sign, store, and move assets across clouds.

- **Images**: VM images/templates (OpenStack Images/Glance, AWS AMI, Azure Compute Gallery (formerly SIG), GCP Images, VMware Templates/Content Library).
- **Containers & OCI**: images, Helm charts (OCI), OPA/Kyverno policy bundles, WASM modules.
- **Shared system files & packages**: private APT/YUM repos, config bundles, OS baselines.
- **Databases & data**: schema migrations (Flyway/Liquibase), seed data, masked snapshots, dumps.
- **Auxiliary artifacts**: SBOMs (SPDX/CycloneDX), attestations (in-toto/SLSA), signatures, policies.

---

artifact — hiện vật; template — khuôn mẫu; baseline — đường cơ sở; schema migration — thay đổi cấu trúc của cơ sở dữ liệu mà không làm mất dữ liệu hiện có; attestation — bản chứng thực (nguồn gốc / chuỗi cung ứng)

VM Images — Build, Catalog, Distribute (Cross Cloud)

Objective: Produce hardened, signed, versioned base images and distribute them consistently across accounts, regions, and platforms.

- **Build**: Packer/Image Builder hardened baselines (CIS), cloud-init/agents, compliance tests.
- **Catalog**: OpenStack Glance; AWS AMI & EC2 Image Builder; Azure Compute Gallery (formerly SIG); GCP Image Families; VMware Content Library.
- **Distribution**: share to accounts/subscriptions/projects; replicate across regions.
- **Trust**: image signing/verification (Glance signature props); Trusted/Confidential VM options per CSP.
- **Versioning**: semantic version + build metadata; immutable IDs; deprecate/EOL with grace.

---

image catalog — danh mục ảnh; replication — sao chép; deprecation — ngưng dùng; confidential VM — máy ảo bảo mật

Containers & OCI Artifacts — Build, Scan, Sign, Store

Objective: Ensure container supply-chain integrity via reproducible builds, vulnerability gating, signatures/attestations, and digest-based deploys.

- **Build**: BuildKit/Kaniko (daemonless), multi-arch, reproducible builds.
- **Scan**: Trivy/Grype/Snyk; block high severity; enforce base-image policies.
- **Sign & Attest**: Sigstore Cosign (keyless OIDC), in-toto attestations; Rekor transparency.
- **Registry**: Harbor, AWS ECR, Azure ACR, GCP Artifact Registry; immutable tags; org/geo replication.
- **Deploy by digest**: pin `sha256:...` for immutability; store SBOM as an OCI artifact.

---

reproducible build — dựng tái lập; transparency log — sổ minh bạch; immutable tag — thẻ bất biến; digest pinning — cố định băm

Example — Container Artifact Lifecycle (Private + Public Cloud)

Scenario: Build, scan, sign, and deploy container images across hybrid environments with full supply-chain integrity.

- Developer triggers a CI pipeline (e.g., GitHub Actions or GitLab CI) using `Kaniko` for daemonless, reproducible multi-arch builds.
- The built image is scanned by `Trivy`; promotion is blocked if high-severity vulnerabilities are found.
- The resulting digest `sha256:...` is signed by `Cosign` (keyless OIDC) and attested via `in-toto` and `Rekor` transparency log.
- Signed images and SBOMs are pushed to `Harbor` as the private on-prem registry (integrated with OpenStack/Kolla or Kubernetes clusters).
- Harbor replicates trusted artifacts to public or regional registries (e.g., AWS ECR, Azure ACR, GCP Artifact Registry) for multi-cloud deployment.
- Kubernetes or OpenStack services deploy images by digest, ensuring immutability and traceable provenance even in air-gapped environments.

Harbor — registry nội bộ hỗ trợ scan, ký, và nhân bản đa vùng; air-gapped — môi trường cách ly mạng; Kaniko — công cụ build không cần daemon; Trivy — trình quét lỗ hổng container; Cosign — ký ảnh container không khoá; in-toto — chứng thực chuỗi cung ứng; Rekor — sổ minh bạch.

Shared System Files & Packages — Private Repos & Bundles

Objective: Control base OS state with signed private repos and versioned config bundles, ensuring integrity and safe rollout/rollback.

- **Private package repos**: APT/YUM with GNU Privacy Guard(GPG) signing; mirror approved upstreams; SBOM for base layer.
- **Config bundles**: versioned tar/zip of baselines (e.g., /etc hardening), distributed via object storage/CDN.
- **Runtime shares**: NFS/SMB services (EFS/Azure Files/Filestore/Manila) — treat mount content as versioned releases.
- **Integrity**: checksums (SHA-256), detached signatures; quarantine & staged rollout.
- **Retention**: object lock/immutability windows for critical baselines.

---

private repo — kho riêng; GPG signing — ký GPG; quarantine — vùng cách ly; object lock — khoá đối tượng

Example — Managing Private Repositories and Config Bundles

Scenario: Maintain signed OS packages and versioned configuration bundles for consistent and secure system baselines.

- A private `APT/YUM` repository mirrors approved upstream packages and re-signs them with an internal `GPG` key before publishing.
- Each base image build references this internal repo, embedding its `SBOM` for traceability and license compliance.
- Configuration bundles (e.g., hardened `/etc` templates, PAM settings, SSH policies) are packaged as versioned tarballs and stored in object storage with checksum metadata.
- Runtime services (e.g., `NFS/Manila` shares or `Azure Files`) deliver these bundles to **nodes as read-only mounts**, allowing controlled rollout and rollback by version.
- Integrity is enforced with detached signatures and SHA-256 verification before activation; failed verifications trigger quarantine staging.

APT/YUM — trình quản lý gói của Linux; GPG — công cụ ký số gói; SBOM — danh mục thành phần phần mềm; Manila — dịch vụ chia sẻ file của OpenStack; checksum — mã kiểm tra toàn vẹn; quarantine — vùng cách ly; rollback — quay lui phiên bản; NFS — Network File System; PAM — Pluggable Authentication Modules

Databases & Data — Migrations, Seed Data, Snapshots

Objective: Treat schema and data as first-class artifacts with checksummed migrations, protected seeds, and recoverable snapshots.

- **Migrations as artifacts**: Flyway/Liquibase scripts with checksums; semantic versioning; rollback plans.
- **Seed data**: masked/anonymized; stored as encrypted objects; checksum-verified before load.
- **Snapshots/dumps**: periodic logical/physical copies; encryption-at-rest with KMS/HSM; region replication.
- **Promotion gates**: schema diff checks in CI; policy to block destructive DDL outside maintenance windows.
- **Lineage**: track dataset/source commits; attach SBOM-like metadata (owner, source, PII class).

---

seed data — dữ liệu khởi tạo; checksum — mã kiểm; encryption-at-rest — mã hoá khi lưu; lineage — nguồn gốc dữ liệu

Signing, SBOM, Provenance & Verification

Objective: Produce machine-verifiable evidence (SBOMs, signatures, attestations) and enforce it at CI/admission with full auditability.

- **SBOM**: generate CycloneDX/SPDX for images/containers/packages; attach to registry/catalog.
- **Sign**: VM (Glance props); container (Cosign); packages (GPG); configs (detached sigs).
- **Provenance**: SLSA attestations; in-toto layout; build-system identity via OIDC.
- **Verify at admission**: K8s (OPA/Gatekeeper/Kyverno), OpenStack (policy on image props), CI gates.
- **Auditability**: Rekor/TLog references; immutable logs; reproducibility checks.

---

provenance — nguồn gốc, xuất xứ; admission — tiếp nhận; attest — chứng thực; auditability — khả năng kiểm toán

Promotion, Replication & Lifecycle Policies

Objective: Move artifacts safely across environments by digest with geo-replication, lifecycle controls, rollback plans, and compliance evidence.

- **Rings**: dev → staging → prod; promote by digest only; freeze mutable tags in prod.
- **Geo replication**: cross-region registry/bucket replication; warm standby for catalogs.
- **Lifecycle**: retention by class (gold/silver/bronze), auto-prune old minors, EOL notices.
- **Rollback**: N-2 image keep; blue/green or canary with fast revert triggers.
- **Compliance**: export manifests of deployed digests; periodic attestation re-validation.

---

retention — lưu giữ; rollback — quay lui; compliance — tuân thủ

Example — Artifact Promotion and Lifecycle Control

Scenario: A signed container image is promoted across dev–staging–prod rings with digest-based verification and controlled replication.

- The CI pipeline publishes a new image to the `dev` ring in Harbor with tag `:v1.4.0` and digest `sha256:....`
- After passing automated and security tests, the image is promoted to `staging` by its digest only — mutable tags remain frozen in production.
- `Harbor` automatically replicates the image and its SBOM across regions or isolated environments, maintaining a warm standby catalog for resilience.
- Lifecycle policies retain only the last two minor versions (N–2), while older digests are pruned or archived based on gold/silver/bronze classification.
- During production rollout, a `blue/green` deployment monitors error budgets; if thresholds are exceeded, rollback to the previous digest is triggered automatically.
- Compliance evidence (digest manifest, attestation, SBOM) is periodically exported for audit and re-validation.

---

ring promotion — vòng triển khai; retention policy — chính sách lưu giữ; SBOM — danh mục thành phần phần mềm; attestation — chứng thực nguồn gốc; rollback — quay lui; compliance audit — kiểm toán tuân thủ

Reference Pipelines & Policy Gates (Cross Cloud)

Objective: Provide reference steps for multi-cloud artifact flows with enforceable gates and deploy-by-digest patterns.

```
# VM (AWS/Azure/GCP/VMware/OpenStack)
Build:   Packer/Image Builder  -> harden -> test
Sign:    Glance props (OS); AMI workflow (AWS);
         Azure Compute Gallery (formerly SIG); Images (GCP); OVF/CL (VMware)
Catalog: Publish to catalog (Glance/ACG/Images/CL) with version immutability
Promote: Share/replicate to prod accounts/subs/projects/regions
# Containers/OCI artifacts
Build:   BuildKit/Kaniko -> SBOM -> scan
Sign:    cosign sign --keyless IMAGE@sha256:...
Policy:  conftest/OPA, Kyverno verifyImages, gate on high CVE
Promote: registry->registry replication; deploy by digest
# DB/Data
Migrate: Flyway/Liquibase (checksum, idempotent)
Seed:    masked dumps (KMS encrypted), checksum verify
Snapshot: scheduled, geo-replicated, retention with object lock
```

Orchestration Layers — Aligned with Artifacts

Objective: Show how orchestration unifies **infra, artifacts, delivery, policy, and ops** into one policy-driven flow.

- **Infrastructure**: networks, subnets, routers, security groups, compute/storage pools (provisioned via IaC).
- **Artifact resolution**: select signed VM images (Glance/AMI/Azure Compute Gallery/GCP Images) and OCI digests.
- **Application delivery**: K8s (Helm/Kustomize via GitOps) or VM stacks (Heat/Terraform/CFN/Bicep).
- **Policy & admission**: enforce signed images, approved registries, allowed SKUs/flavors at deploy-time.
- **Operations**: progressive rollout, autoscaling, backups, DR runbooks, drift control.

---

orchestration — dàn dựng, điều phối; admission control — kiểm soát tiếp nhận; rollout — triển khai; drift — sai lệch cấu hình so vơi thông số thiết lập;

Cross-Cloud Orchestration Interfaces

Objective: Map neutral interfaces across clouds so pipelines stay portable and vendor-agnostic.

Interface	OpenStack	AWS	Azure	GCP/VMware
IaC engines	Heat/Terraform	CloudFormation/ Terraform	Bicep/ARM/ Terraform	Config Controller*/ Terraform/vRealize
Image catalogs	Glance	AMI/ EC2 Image Builder	Azure Compute Gallery	Images/Content Library
Secrets/KMS	Barbican/Vault	KMS/ Secrets Manager	Key Vault (Keys/Secrets)	Cloud KMS & Secret Manager/KMIP KMS
GitOps controllers	ArgoCD/Flux (runs on any Kubernetes)			
Admission and policy	K8s Gatekeeper/ Kyverno; Glance sig-enforce	SCP; OPA as add-on	Azure Policy; Gatekeeper	Organization Policy; Gatekeeper/NSX-T policies

interface — giao diện; catalog — danh mục; admission — tiếp nhận; SCP/Organization Policy — chính sách cấp tổ chức

Artifact-aware Provisioning & Policy Gates

Objective: Enforce artifact integrity at provisioning/admission via image/digest resolution and signed-only gates.

- **Resolve images**: look up by version → map to immutable IDs/digests per cloud/region.
- **Gate policies**: only signed VM images; only OCI digests from approved registries.
- **Admission**: K8s OPA/Gatekeeper/Kyverno; cloud org policies (SCP/Policy/Organization Policy).

Kyverno (verify image signatures, example):

```
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata: { name: verify-signed-images }
spec:
  rules:
  - name: verify-cosign
    match: { resources: { kinds: ["Pod"] } }
    verifyImages:
    - image: "registry.example.com/*"
      verifyDigest: true
      attestors:
      - entries:
```

GitOps Delivery by Digest (Concept)

Objective: Ensure the same, signed artifact is deployed consistently across all environments using Git as the single source of truth.

- **ArgoCD / Flux:** Continuously sync manifests from Git and deploy OCI images pinned by `sha256` digest.
- **App-of-Apps / ApplicationSet:** Parameterize environment differences (dev, staging, prod) while keeping version integrity.
- **VM stacks:** Reference VM image IDs (Glance, AMI, ACG, GCP) pinned to versioned digests or image hashes.

Goal: Fully declarative delivery pipeline — Git defines desired state, controllers ensure convergence to that state.

---

digest — băm nội dung; declarative — khai báo; source of truth — nguồn xác thực duy nhất

Progressive Delivery (K8s) & Blue/Green (VM)

Objective: Roll out changes safely with metric-gated canaries on K8s and blue/green for VM stacks.

- **K8s**: Argo Rollouts/canary; metric checks (Prometheus), automated pause/resume.
- **VM apps**: Heat/Terraform deploy a parallel stack for safe rollout; switch traffic to the new stack ("green"); retain the previous stack ("blue") for rollback.

Argo Rollouts — minimal canary:

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata: { name: web }
spec:
  replicas: 6
  strategy:
    canary:
      steps:
      - setWeight: 20
      - pause: { duration: 300 }
      - setWeight: 60
      - pause: {}
```

Autoscaling Orchestration — Pods, Nodes, VMs

Objective: Wire the control loops across layers and ensure pre-warmed artifacts minimize cold starts.

- **Pods**: HPA with resource/custom metrics; VPA for right-sizing.
- **Nodes**: Cluster Autoscaler (EKS/AKS/GKE/self-managed) with labels/taints for accelerators.
- **VM groups**: Senlin/ASG/VMSS/MIG; pre-warm pools using pinned artifacts.
- **Scheduled/predictive**: cron-based scale and ML-based forecasts; enforce quota and headroom.

---

right-sizing — điều chỉnh kích cỡ hợp lý; pre-warm — làm ấm trước; headroom — phần dự trữ

Data/State Orchestration — Migrations & Backups

Objective: Embed schema change and backup workflows into CI/CD so state evolves safely with code.

- **Migrations-as-code**: Flyway/Liquibase in CI/CD; checksums; maintenance windows.
- **Seed/fixtures**: masked dumps as encrypted objects; verify checksum before load.
- **Backups**: scheduled snapshots/backups; geo-replication; restore drills wired into pipelines.

CI step (Flyway) — example:

```
- name: Flyway migrate
  run: |
    export FLYWAY_URL=$DB_URL FLYWAY_USER=$DB_USER FLYWAY_PASSWORD=$DB_PASS
    flyway -locations=filesystem:db/migrations -baselineOnMigrate=true migrate
```

---

fixture — dữ liệu mẫu; restore drill — diễn tập khôi phục; maintenance window — khung bảo trì

Compliance, Drift Control & DR Orchestration

Objective: Treat compliance, drift checks, and DR drills as first-class pipeline stages, not afterthoughts.

- **Drift**: terraform plan + driftctl; Azure Policy/AWS Config/GCP Config Validator.
- **Compliance gates**: conftest/OPA on IaC; image signature enforcement; SBOM presence.
- **DR runbooks**: infra re-provision + data restore; traffic failover; time-boxed RTO checks.

Pipeline gates (pseudo):

```
Stage: Validate -> OPA/conftest on TF/Helm
Stage: Security -> verify signatures/SBOM
Stage: Plan     -> terraform plan (no-drift) / heat stack-validate
Stage: Deploy   -> progressive rollout or blue/green
Stage: DR Test  -> restore snapshot, run probes, publish report
```

---

gate — cổng kiểm soát; failover — chuyển mạch dự phòng; RTO — mục tiêu thời gian khôi phục

Isolation Tiers & Tenancy

Objective: Establish tenant segmentation and a default-deny posture across east–west and north–south paths, uniformly for VMs, containers, and bare metal.
Scope: runtime network design & monitoring; orchestration/policy gates are covered elsewhere.

- **Tenant-level segmentation**: VPC/VNet/Project networks; VRF-style routing domains per tenant.
- **East–west isolation**: overlay segments (Geneve/VXLAN), Security Groups/ACLs, K8s NetworkPolicy.
- **North–south controls**: ingress/egress gateways, NAT/SNAT, WAF, DDoS edge.
- **Default-deny posture**: least-privilege rules; micro-segmentation for app tiers.
- **Hybrid isolation**: consistent policies across VMs, containers, and bare metal.

---

isolation — cô lập; tenancy — người thuê; east–west traffic — lưu lượng ngang; north–south traffic — lưu lượng vào/ra; default-deny — mặc định từ chối; micro-segmentation — vi phân đoạn

Multi-Region Topology & Interconnect

Objective: Design regional topologies and interconnect/route policies that isolate failures and meet latency SLOs with deterministic failover.

- **Regions & AZs**: fault domains with independent control/data planes; avoid shared fate.
- **Hub–spoke vs mesh**: centralize shared services (hub) or scoped meshes for latency SLOs.
- **WAN options**: private interconnects, cloud WANs, or IPSec/GRE/WireGuard overlays.
- **Routing**: BGP/EVPN for scale; regional summarization; communities/NO_EXPORT to contain routes.
- **Failover**: GSLB/anycast DNS, health-based traffic steering; account for data gravity.

---

multi-region — đa vùng; availability zone — vùng sẵn sàng; hub–spoke — trục–nan hoa; interconnect — liên kết; anycast — phát đa điểm đồng đích

Scaling Patterns & Performance

Objective: Select and tune datapaths, load-balancing patterns, and egress controls to sustain throughput targets and p95/p99 latency at scale.

- **Overlays at scale**: Geneve/VXLAN with NIC offload; MTU headroom for encap; ECMP on inner 5-tuple.
- **High-throughput paths**: SR-IOV/DPDK, XDP/eBPF fast paths; NUMA alignment for NFV dataplanes.
- **Load balancing**: L4 NLB vs L7 ALB/API GW; internal vs external; consistent hashing & zonal shards.
- **Egress control**: per-tenant NAT pools, egress gateways, DNS egress policies, QoS/rate limits.
- **Capacity signals**: p95/p99 latency, drops/retransmits, connection churn, SYN backlog.

MTU — đơn vị truyền tối đa; SR-IOV — ảo hoá chia sẻ theo hàng; DPDK — bộ công cụ gói dữ liệu; QoS — chất lượng dịch vụ; backlog — hàng chờ

Service Exposure, Policy, and Private Access

Objective: Expose services securely with zero-trust edges, least-privilege egress, private endpoints, and identity-bound mesh policies (runtime scope; admission lives elsewhere).

- **Ingress**: API gateways/WAF, L7 routing, canary headers; mutual TLS for zero-trust edges.
- **Egress**: dedicated egress subnets, DNS egress policies, approved destinations only.
- **Private endpoints**: provider backbone–native service links (no public internet paths).
- **Mesh policies**: mTLS, authz, rate limiting, and traffic shaping for east–west flows.
- **Identity alignment**: SPIFFE/SVID or mTLS SANs to bind service identity to network policy.

---

private endpoint — điểm cuối riêng tư; zero-trust — không tin cậy mặc định; mTLS — TLS hai chiều; rate limiting — giới hạn tốc độ; identity — danh tính

TAP, SPAN/ERSPAN & Packet Mirroring

Objective: Mirror traffic responsibly for security/forensics using the right primitive (TAP/SPAN/ERSPAN) and capacity safeguards.

- **TAP (physical)**: hardware copy for out-of-band sensors; ideal for lossless capture.
- **SPAN/ERSPAN**: switch/software mirroring; ERSPAN encapsulates for remote analyzers.
- **Cloud equivalents**: VPC Traffic Mirroring (AWS), vTAP/Port Mirroring (Azure), Packet Mirroring (GCP), NSX/OVN mirrors.
- **Targets**: IDS/IPS (Zeek/Suricata), PCAP recorders, DLP engines; throttle to avoid overload.

OVS mirror (example):

```
# Mirror a tenant port to an analyzer on br-int
ovs-vsctl \
  -- --id=@s get Port qvo-tenant123 \
  -- --id=@a get Port span-analyzer0 \
  -- --id=@m create Mirror name=m0 select-src-port=@s output-port=@a \
  -- set Bridge br-int mirrors=@m
```

TAP — cổng sao chép phần cứng; SPAN — sao chép cổng chuyển mạch; ERSPAN — SPAN từ xa có bao gói;

Network Observability & SLOs

Objective: Operate a telemetry → pipeline → SLO loop that detects regressions early and drives actionable runbooks.

- **Telemetry**: flow logs (VPC/VNet/GCP), IPFIX/sFlow/NetFlow; eBPF for K8s (Hubble).
- **Signals(feedbacks)**: latency (p95/p99), error rate, saturation, tail drops, retransmits.
- **Pipelines**: span/mirror → collectors (Kafka/OTel) → processors (Zeek/Suricata) → SIEM.
- **SLOs & alerts**: burn-rate policies; black-box probes per region; synthetic path tests.
- **Runbooks**: packet loss triage, PMTU checks, asymmetric routing, noisy-neighbor isolation.

---

observability — quan sát hệ thống; IPFIX — xuất lưu lượng IP; tail drop — rơi gói đuôi; PMTU — MTU đường dẫn; runbook — sổ tay vận hành

Autoscaling Principles & Control Loops

Objective: Design SLO-driven control loops (reactive/scheduled/predictive) that scale across pods → nodes → VMs → data, with stability heuristics and artifact alignment.

- **Scopes**: pods → nodes → VMs → data services (DB throughput, storage IOPS).
- **Control loops**: reactive (metrics-triggered), scheduled (calendar peaks), predictive (forecast-based).
- **SLO-driven**: act on p95/p99 latency, error rate, saturation; scale only when SLO is at risk.
- **Artifact alignment**: use *pre-warmed* VM images/OCI digests (immutable, signed) to cut cold-starts.
- **Stability**: cooldown, hysteresis, min-ready seconds; damped thresholds to prevent flapping.
- **IaC/GitOps**: autoscaling configs are versioned; promote per environment (dev → prod).

---

autoscaling — tự co giãn; reactive — phản ứng; predictive — dự báo; cooldown — thời gian hạ nhiệt; hysteresis — trễ đàn hồi

Cross-Cloud Autoscaling Primitives

Objective: Identify equivalent scaling building blocks across platforms so behaviors remain portable and enforceable by policy.

- **Kubernetes**: HPA/VPA; Cluster Autoscaler (CA); KEDA for event/queue-based scaling.
- **VM groups**: OpenStack Senlin; AWS Auto Scaling Groups (ASG + Warm Pools); Azure VMSS; GCP MIG.
- **Data services**: read replicas, shard/partition units; storage throughput/IOPS tiers (EBS/PD/Managed Disks).
- **Accelerators**: GPU-aware scheduling (labels/taints), NVIDIA device plugin, MIG, time-slicing/MPS.
- **Quotas/limits**: tenant CPU/RAM/GPU/IOPS ceilings; budgets & max floors/ceilings for scale.

---

warm pool — nhóm ấm; device plugin — phần bổ trợ thiết bị; shard — mảnh dữ liệu; quota — hạn ngạch

Kubernetes Autoscaling — HPA/VPA/CA/KEDA (Concepts)

Objective: Apply minimal, safe patterns to combine HPA/VPA/CA/KEDA without instability or budget drift.

- **HPA**: resource/custom metrics; target utilization or RPS; respect PodDisruptionBudget.
- **VPA**: *Off* / *Auto* / *Initial*; combine with HPA via min/max caps.
- **Cluster Autoscaler**: adds/removes nodes from unschedulable pods; label/taint pools for GPUs/SR-IOV.
- **KEDA**: scales on external events (queue length, lag, HTTP_RPS).

---

unschedulable — không thể lên lịch; taint — vết đánh dấu; PDB — ngân sách gián đoạn; lag — độ trễ hàng đợi

Kubernetes Autoscaling — HPA + VPA (Minimal Example)

HPA + VPA (minimal):

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata: { name: web }
spec:
  scaleTargetRef: { apiVersion: apps/v1, kind: Deployment, name: web }
  minReplicas: 3
  maxReplicas: 30
  metrics:
  - type: Resource
    resource: { name: cpu, target: { type: Utilization, averageUtilization: 70 } }
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata: { name: web }
spec:
  targetRef: { apiVersion: apps/v1, kind: Deployment, name: web }
  resourcePolicy:
    containerPolicies:
    - containerName: "*"
      minAllowed: { cpu: "250m", memory: "256Mi" }
      maxAllowed: { cpu: "4", memory: "8Gi" }
```

VM Autoscaling & Warm Pools (IaC Examples)

Objective: Reduce cold-starts and variance by pinning immutable images and maintaining warm capacity aligned to demand windows.

- **Immutable boot**: pin AMI/ACG/Image IDs; user-data pulls signed configs.
- **Warm pools**: keep *N* pre-initialized VMs to reduce cold-start latency.
- **Scheduled actions**: scale windows aligned to business calendars.

Terraform — AWS ASG (excerpt):

```
resource "aws_autoscaling_group" "web" {
  desired_capacity = 6
  min_size = 3
  max_size = 30
  launch_template { id = aws_launch_template.web.id, version = "$Latest" }
  warm_pool { min_size = 3 }
  target_group_arns = [aws_lb_target_group.web.arn]}
resource "aws_autoscaling_policy" "cpu70" {
  autoscaling_group_name = aws_autoscaling_group.web.name
  policy_type = "TargetTrackingScaling"
  target_tracking_configuration {
    predefined_metric_specification { predefined_metric_type = "ASGAverageCPUUtilization" }
    target_value = 70
```

Resource Allocation, Quotas & Runtime Guards

Objective: Encode classes, quotas, and runtime guards so scaling remains fair, isolated, and budgeted under contention.

- **Classes & flavors**: standard/perf/gpu/io; CPU policy (dedicated/shared), hugepages, SR-IOV exposure.
- **Bin-packing vs spread**: utilization vs fault-domain resilience; anti-affinity for replicas.
- **Quotas & budgets**: per-tenant CPU/RAM/GPU/IOPS; egress/cost budgets to cap scale.
- **GPU specifics**: MIG profiles, time-slicing/MPS, NUMA alignment; preemption policies for queues.
- **Storage**: dynamic IOPS/throughput classes; burst credits; throttle guards for noisy neighbors.

---

bin-packing — đóng gói tối ưu; anti-affinity — chống đồng vị trí; preemption — truất quyền; noisy neighbor — hàng xóm ồn

Predictive & Cost-Aware Scaling — Signals & Policies

Objective: Forecast safely, cap spend, and maintain stability with post-scale verification and headroom policies.

- **Signals**: request rate, concurrency, queue lag, CPU run-queue, GC time, cache miss.
- **Forecasting**: seasonality (daily/weekly), EWMA/ARIMA; ML models with min/max guardrails.
- **Policies**: scheduled overrides; cost caps; SLO burn-rate triggers; surge buffers for step loads.
- **Stability**: cooldown, step functions, partial rollouts; verify post-scale with synthetic probes.
- **What-if**: capacity modeling (headroom $\geq$ 20–30%); pre-warm pools in peak seasons.

EWMA — trung bình mũ có trọng số; burn-rate — tốc độ tiêu hao SLO; surge — dồn tải; headroom — phần dự trữ

Predictive & Cost-Aware Scaling — KEDA Example

Objective: Implement event-driven scaling with guardrails using KEDA (queue length trigger) and conservative cooldown. **KEDA ScaledObject — queue length (excerpt):**

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata: { name: worker-queue }
spec:
  scaleTargetRef: { name: worker }
  pollingInterval: 5
  cooldownPeriod: 60
  minReplicaCount: 2
  maxReplicaCount: 50
  triggers:
  - type: aws-sqs-queue
    metadata:
      queueURL: "https://sqs.amazonaws.com/ACCT/queue"
      queueLength: "100" # target per-replica messages
```

KEDA — tự co giãn theo sự kiện; cooldown — thời gian hạ nhiệt; per-replica — mỗi bản sao

Storage — Classes, Designs, Recovery

Objective: Choose the right storage class and recovery strategy for each workload tier, scoped strictly to the data layer.
Scope: data layer only (no orchestration/network policy here).

- **Ephemeral vs Persistent**: root ephemeral for stateless; Cinder/Manila, Filestore, EFS/Azure Files for stateful shares.
- **Instance-backed vs Volume-backed**: prefer boot-from-volume for snapshot/clone efficiency across clouds.
- **Snapshots & Backups**: scheduled snapshots; off-cluster/off-account backups; verify restores.
- **RPO/RTO trade-offs**: align to business tiers; document runbooks and DR tests per region.

---

ephemeral — tạm thời; persistent — bền vững; snapshot — ảnh chụp; backup — sao lưu; runbook — sổ tay vận hành

Model Choice & Workload Mapping

Objective: Map workload I/O patterns to the correct block/file/object services and media (local NVMe vs NVMe-oF) with clear tiering rules.

- **Block**: low-latency I/O (DB/VM disks); classes: gp3/io2 (AWS), Premium SSD v2 (Azure), PD Balanced/Extreme (GCP), Cinder (OpenStack), vSAN/VMFS (VMware).
- **File**: shared POSIX/SMB (ML training sets, home dirs); EFS/Azure Files/Filestore/Manila/CephFS.
- **Object**: lakes/archives/artifacts; S3/Blob/Cloud Storage/Swift with lifecycle policies.
- **Local NVMe vs NVMe-oF**: local for ultra-low latency; NVMe-oF for pooled performance and elasticity.
- **Patterns**: put write-ahead logs on fast block; offload cold data to object; cache layers for read-heavy workloads.

---

block storage — lưu trữ khối; file storage — lưu trữ tệp; object storage — lưu trữ đối tượng; NVMe-oF — NVMe qua hạ tầng mạng

Security — Encryption, Keys, Immutability, Access

Objective: Enforce end-to-end data protection (encryption, keying, immutability, access) with auditable controls at the storage boundary.

- **Encryption at rest & in transit**: LUKS/dm-crypt, TLS in-flight; CMEK/CMK via KMS/HSM; vTPM for VM disks.
- **Key management**: rotate, split duties, envelope encryption for app secrets; short-lived creds.
- **Immutability**: S3/Object Lock, Blob immutability, GCS retention locks; WORM buckets for backups.
- **Access control**: IAM/RBAC least-privilege; signed URLs and bucket policies; per-tenant encryption contexts.
- **Audit & detection**: storage access logs, anomaly alerts (spikes in deletes/reads), quarantine pipelines.

---

encryption at rest — mã hoá khi lưu; envelope encryption — mã hoá bao; immutability — bất biến; least privilege — đặc quyền tối thiểu

Data Protection — Snapshots, Backups, Replication

Objective: Achieve application-consistent protection with verifiable restores and right-sized replication/lifecycle policies.

- **Application-consistent**: quiesce DB (fsfreeze, VSS, pre/post hooks) before snapshot; PITR via WAL/binlog.
- **Backups**: encrypted cross-region/cross-account copies; periodic restore drills with checksum validation.
- **Replication**: async vs sync (latency/cost trade-offs); object CRR/replication rules; Ceph tiers/CRUSH rulesets.
- **Lifecycle & retention**: 3–2–1–1–0 rule; tiering (hot/warm/cold); legal hold for investigations.
- **Catalog**: inventory volumes/buckets with version, owner, sensitivity, and recovery class.

PITR — khôi phục theo thời điểm; replication — sao chép; lifecycle — vòng đời; retention — lưu giữ

Incident Response — Ransomware & Data-Loss Playbook

Objective: Contain destructive events quickly and restore data integrity from immutable backups with a practiced playbook.

- **Immediate**: freeze destructive IAM changes; block keys; isolate impacted subnets/projects.
- **Containment**: enforce object immutability; snapshot-now critical volumes; stop automated deletions.
- **Triage & scope**: diff access logs vs baseline; identify compromised principals and blast radius.
- **Restore**: recover from immutable backups into clean accounts/tenants; re-key; verify with checksums.
- **After-action**: rotate credentials, patch findings, update runbooks, schedule game-day re-tests.

Runbook skeleton (pseudo):

```
Step 1: Freeze -> IAM change freeze, block KMS key usage
Step 2: Snapshot -> app-consistent snapshots of affected volumes
Step 3: Contain -> bucket/object lock, quarantine prefixes
Step 4: Restore -> clean tenant, verify checksums, cutover
```

Industry Patterns & Elastic Parameters

- **Workload classes**: interactive APIs, streaming, batch/ML training, NFV/data plane.
- **Elastic parameters (core)**: min/max replicas; target utilization; cooldown; hysteresis; evaluation interval; lookback window.
- **Capacity guards**: headroom (%); surge buffer; warm pool size; pre-warmed image/digest IDs.
- **Resource knobs**: CPU policy (shared/dedicated), GPU profiles (MIG/time-slicing), IOPS/throughput tiers.
- **SLO alignment**: p95/p99 latency targets; error-budget burn-rate thresholds (fast/slow).
- **Cost controls**: per-env budget caps; spot/preemptible share; egress ceilings; storage tiering rules.

---

elasticity — độ đàn hồi; headroom — phần dự trữ; hysteresis — trễ đàn hồi; warm pool — nhóm ấm; burn rate — tốc độ tiêu hao ngân sách lỗi

Cross-Cloud Mapping of Elastic Knobs

Layer/Knob	AWS	Azure	GCP	OpenStack/VM
VM groups (scale)	ASG Target-Tracking, Warm Pools	VMSS autoscale	MIG autoscaler	Senlin/VM DRS K8s CA
Pods (scale)	EKS HPA/VPA	AKS HPA/VPA	GKE HPA/VPA	K8s HPA/VPA
Target utilization	CPU/ALB RequestCount	CPU/Requests	CPU/Requests	HPA metrics/cu tom
Cooldown/hysteresis	ASG cooldown	VMSS cooldown	MIG cooldown	HPA stabilizatio Window
Accelerators	p*-gpu + MPS	NC/ND SKUs	A2/H100 + MPS	flavors+extra_s / device plugin
IOPS/throughput	EBS gp3/io2	Premium SSD v2	PD Balanced/Extreme	Cinder QoS vSAN policy
Warm capacity	Warm Pools	Pre-provisioned VMSS	Instance templates + min-Ready	Pre-warmed in ages; PDBs

Elastic Sizing Playbook & Example

Step 1 — Baseline capacity per unit. Measure per-instance sustainable throughput C (req/s) at latency $\leq$ p95 target.

Step 2 — Choose utilization target. $u_t \in [0.5, 0.75]$ for headroom; pick safety factor $s \in [1.2, 1.4]$.

Step 3 — Compute minima. With peak rate λ_{peak}:

$$N_{\min} = \left\lceil \frac{\lambda_{\text{peak}}}{C \cdot u_t \cdot s} \right\rceil$$

Step 4 — Buffers. Warm pool $W = \max(1, \lfloor 0.15 \cdot N_{\min} \rfloor)$; surge buffer $B = \lceil 0.2 \cdot N_{\min} \rceil$.

Step 5 — Control loop. Evaluation interval $T_e = 15{-}30$s; lookback $L = 3{-}5 \times T_e$; cooldown $T_c = 2{-}5$ min; hysteresis $\Delta = 10{-}20\%$.

Worked example. λ_{peak}=9,000 req/s, C=500 req/s, u_t=0.7, s=1.3
$\Rightarrow N_{\min} = \left\lceil \frac{9000}{500 \cdot 0.7 \cdot 1.3} \right\rceil = \lceil 19.8 \rceil = 20.$
Warm pool W=3, surge buffer B=4. Pin images/digests for pre-warm; set HPA target=70%, stabilization window=60s, CA enabled.

Week 8 Wrap-Up — Enterprise-Scale Virtualization and Management

Theme: Platform engineering integrates infrastructure, artifacts, orchestration, and elasticity into one policy-driven control loop.

- **Platform Model — Orchestrated Foundation:** Control, compute, storage planes, and network fabric operate under a continuous reconciliation loop aligning desired state, capacity, and policy.
- **Artifact Management — Trust and Reproducibility:** Build, sign, verify, and promote VM images / OCI containers by digest; enforce provenance and immutability across environments.
- **Orchestration — Declarative Delivery and Policy Control:** Provision via Terraform / Heat, configure with Ansible, deploy through GitOps (ArgoCD / Flux); enforce signed-only admission and drift control.
- **Network Storage — Resilient Runtime Planes:** Intent-driven connectivity, least-privilege segmentation, encrypted and immutable data protection with auditable recovery paths.
- **Elastic Scaling — SLO-Driven Control Loops:** Reactive, scheduled, and predictive autoscaling from pods $\rightarrow$ nodes $\rightarrow$ VMs $\rightarrow$ data; pre-warmed artifacts and cost-aware capacity policies.
- **Enterprise Goal:** Achieve secure, compliant, and reproducible operations through unified orchestration, policy-as-code, and continuous verification.