Kubernetes Scaling & Delivery

HPA • VPA • Cluster Autoscaler • Load Balancing • Multi-Zone/Cluster • Blue-Green • Canary • Progressive

Audience: Advanced / Intermediate (beginner-accessible)

Focus: Theory, control loops, patterns & trade-offs

Agenda

- Horizontal Pod Autoscaler (HPA)
- Vertical Pod Autoscaler (VPA)
- Cluster Autoscaler (CA)
- Load Balancing (internal, external)
- Multi-Zone & Multi-Cluster Deployments
- Blue-Green vs. Canary Deployments
- Advanced Deployment Strategies (progressive delivery)

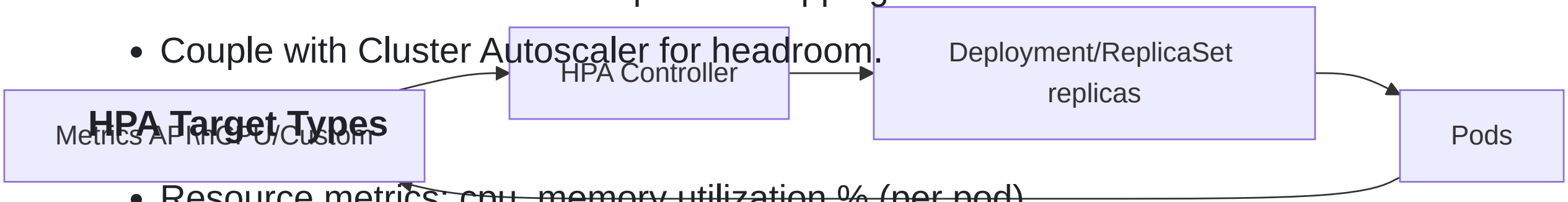
1) Horizontal Pod Autoscaler (HPA)

Concept & Signals

- **Goal:** Adjust replica count based on metrics.
- **Inputs:** metrics.k8s.io (CPU/Memory), custom/external metrics (Prometheus adapters).
- **Algorithm:** Current vs. target utilization → desired replicas (with stabilization windows).
- **Best for:** Stateless services with scale-out patterns.

Guidance:

- Use requests aligned with typical usage for CPU-% targets.
- Add cooldown/stabilization to prevent flapping.
- Couple with Cluster Autoscaler for headroom.



HPA Target Types

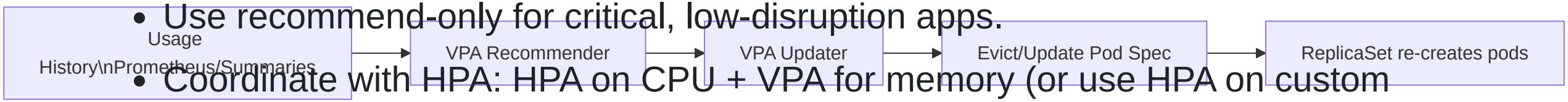
- Resource metrics: cpu, memory utilization % (per pod).
- Pods metrics: Rate per pod (e.g., RPS).
- Object metrics: Backend object’s metric (e.g., queue length of a Service).
- External metrics: Cloud/LB/queue systems (via adapter).

2) Vertical Pod Autoscaler (VPA)

Concept & Modes

- **Goal:** Recommend or set requests/limits per pod to right-size.
- **Modes:** Off (recommendations only), Auto (evict & restart to apply), Initial (apply at creation).
- **Components:** Recommender, Updater (evicts), Admission plugin (sets initial).

Guidance:

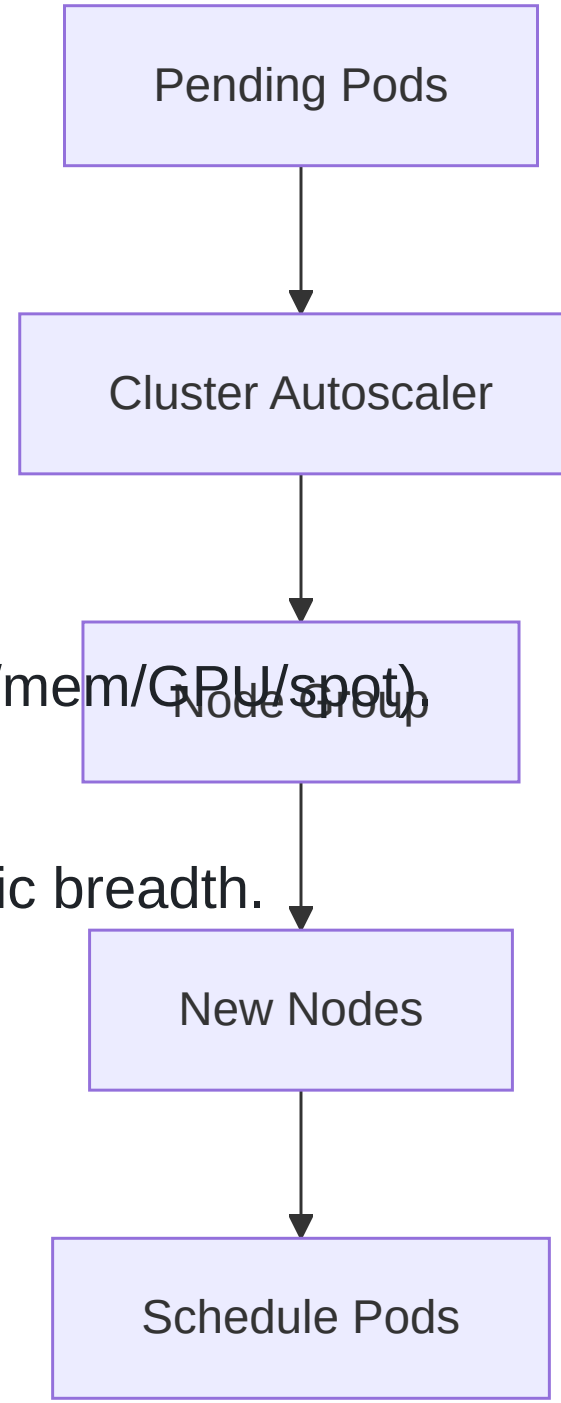


- Use recommend-only for critical, low-disruption apps.
- Coordinate with HPA: HPA on CPU + VPA for memory (or use HPA on custom metrics).
- Beware of frequent restarts; set min change thresholds.

3) Cluster Autoscaler (CA)

Concept & Triggers

- **Goal:** Adjust node count per node group to fit pending pods.
- **Scale-out:** Unschedulable pods → add nodes.
- **Scale-in:** Underutilized nodes → drain & remove (respect PDBs).
- **Scope:** Works with cloud provider autoscaling APIs or cluster API machine sets.



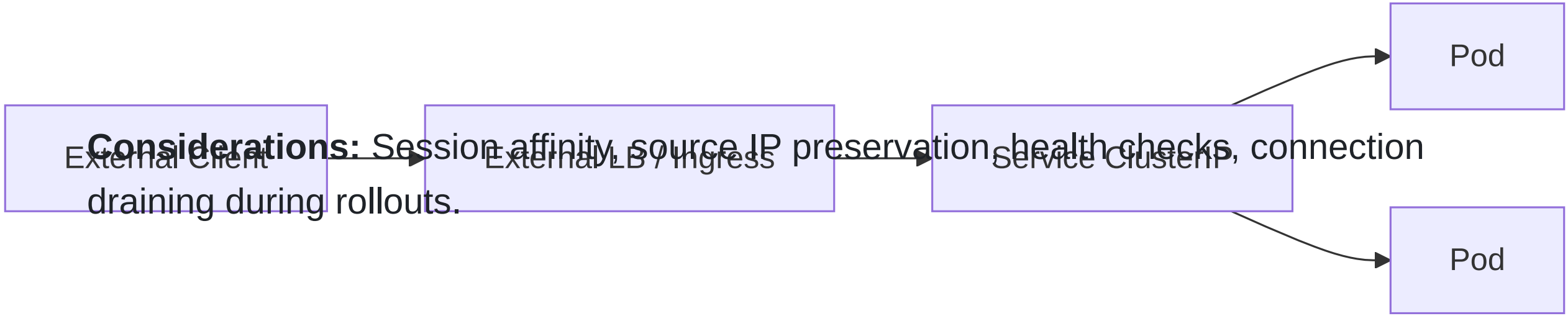
Best practices:

- Separate node pools (CPU/mem/GPU/spot).
- Ensure PDBs allow drain.
- Combine with HPA for elastic breadth.

4) Load Balancing

Internal & External Paths

- **In-cluster:**
  - Service (ClusterIP): kube-proxy (iptables/IPVS) or CNI dataplane (eBPF) balances to ready endpoints.
- **Node exposure:**
  - NodePort: Fixed port on every node.
  - LoadBalancer: Cloud/network LB → NodePort/ClusterIP.
  - Ingress/Gateway: L7 HTTP(S) routing at edge (TLS, host/path rules).



Considerations: Session affinity, source IP preservation, health checks, connection draining during rollouts.

5) Multi-Zone & Multi-Cluster Deployments

Resilience & Placement

Multi-Zone (single cluster):

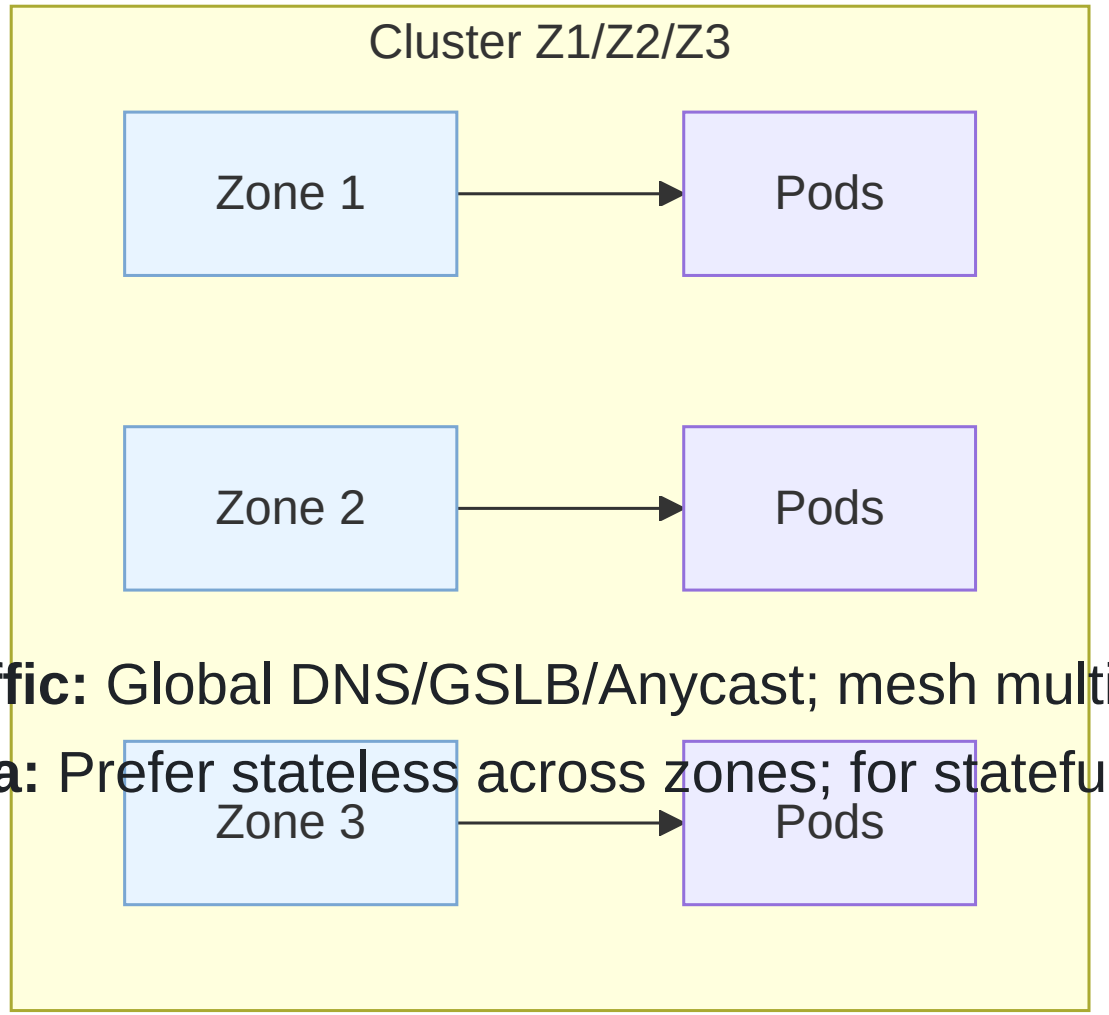
Topology Spread / Anti-Affinity across zones; zonal StorageClasses; zonal LBs.

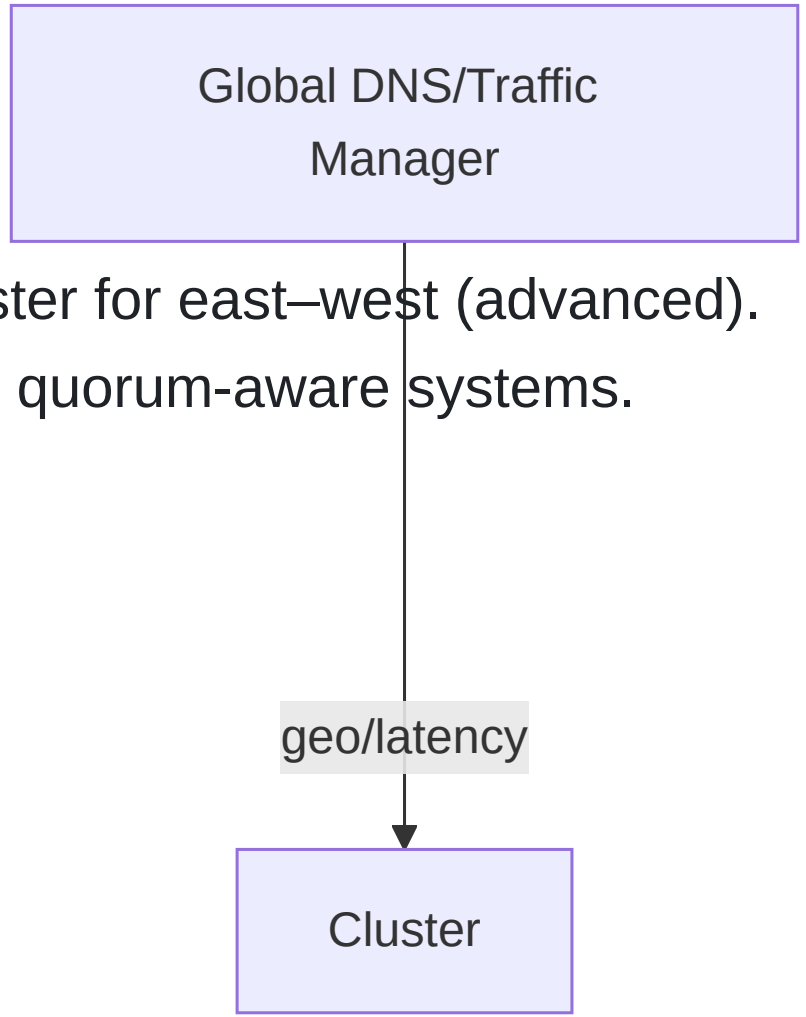
Failure domain: AZ-1 tolerance requires surplus capacity.

Multi-Cluster:

Reasons: Sovereignty, blast-radius, independent failure domains, differing configs.

Approaches: Per-cluster GitOps, KubeFed, or platform managers (Anthos/Rancher).





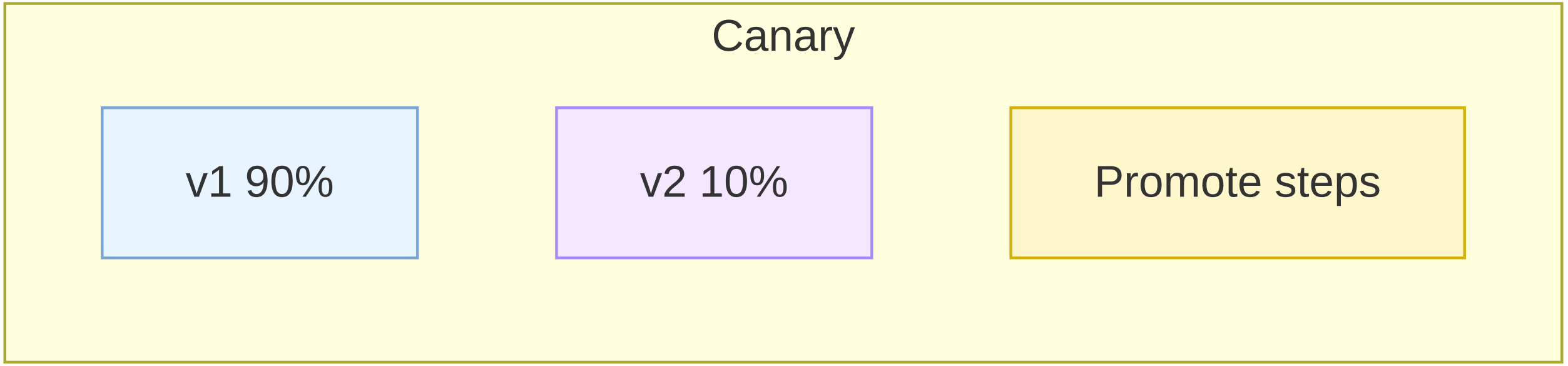
Traffic: Global DNS/GSLB/Anycast; mesh multi-cluster for east–west (advanced).

Data: Prefer stateless across zones; for stateful use quorum-aware systems.

6) Blue-Green vs. Canary Deployments

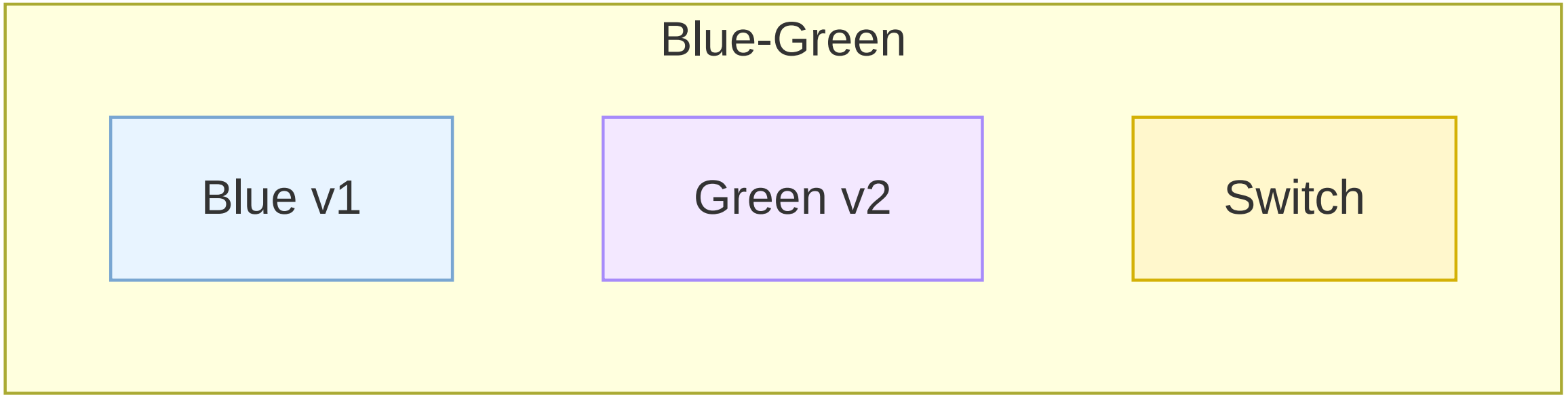
Comparison

Strategy	How it works	Pros	Cons	When to use
Blue-Green	Run two envs (blue=old, green=new); switch traffic atomically	Fast rollback, simple mental model	Double capacity; state sync	Big releases, infra changes
Canary	Gradually shift % of traffic to new version	Low risk, observe metrics	More moving parts; needs SLOs	Frequent small releases



Controls: Ingress weights, service mesh routes, or LB target weighting.

Gates: Error rates, latency SLOs, synthetic checks.



7) Advanced Deployment Strategies

Progressive Delivery (Theory)

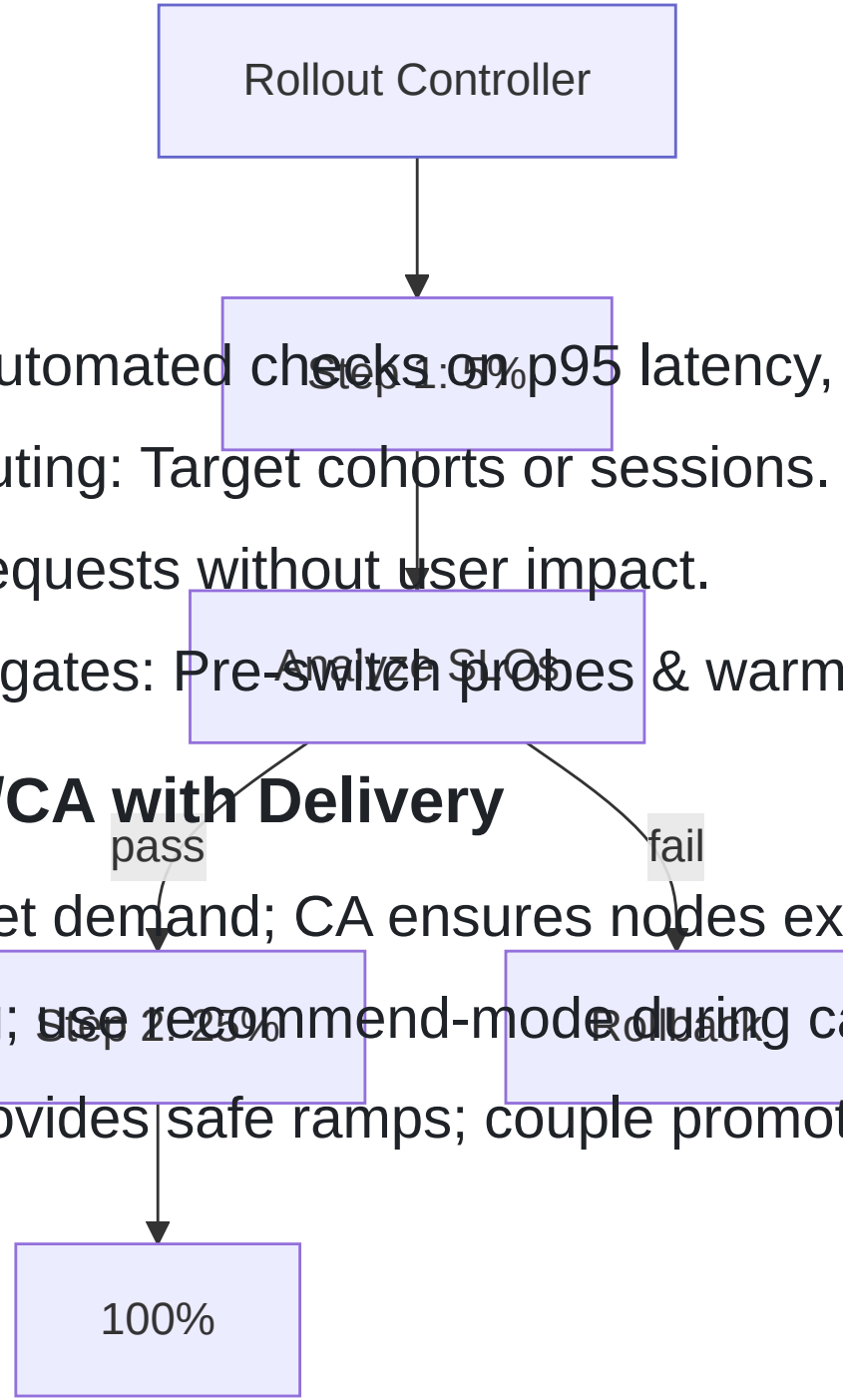
- **Idea:** Automate promotion/rollback based on metrics and pre/post checks.
- **Controllers:** Argo Rollouts, Flagger (with Istio/Linkerd/NGINX/Contour).
- **Policies:** Step weights, max surge/unavailable, analysis templates, abort thresholds.

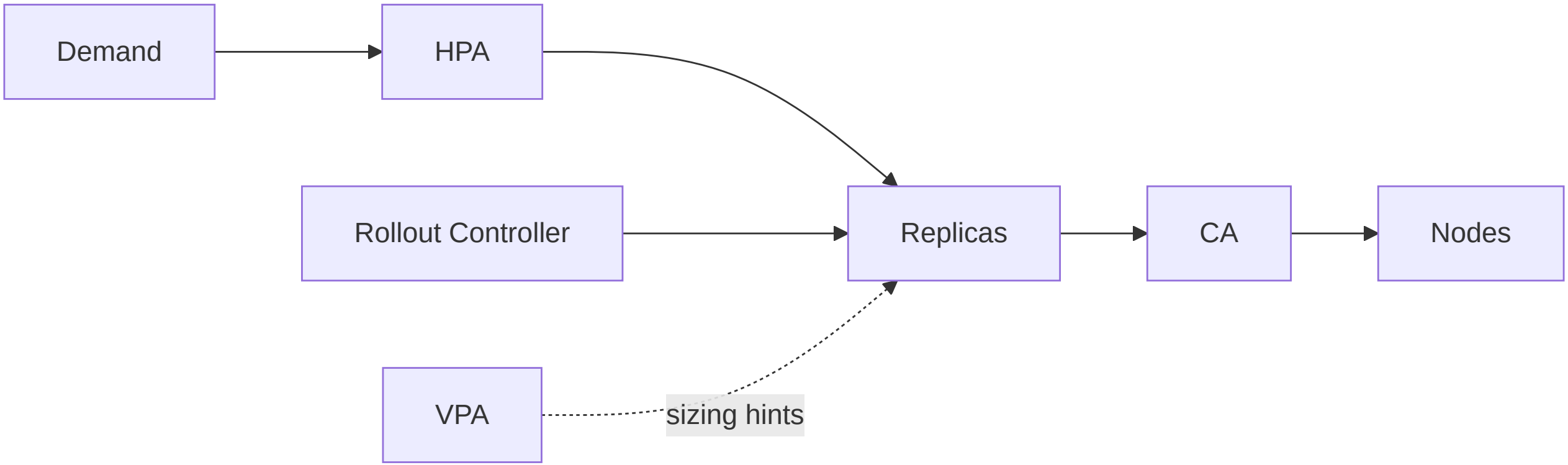
Techniques:

- Canary with analysis: Automated checks on p95 latency, error rates, saturation.
- A/B & header-based routing: Target cohorts or sessions.
- Shadow traffic: Mirror requests without user impact.
- Blue-Green with health gates: Pre-switch probes & warm-up.

Orchestrating HPA/VPA/CA with Delivery

- HPA scales pods to meet demand; CA ensures nodes exist to host them.
- VPA informs right-sizing; use recommend-mode during canary to avoid churn.
- Progressive delivery provides safe ramps; couple promotion with alert silences and SLO burn-rate checks.




Demand
HPA
Rollout Controller
Replicas
CA
Nodes
VPA
sizing hints

Patterns & Anti-Patterns (Summary)

Do:

- Use HPA on stable metrics; add stabilization.
- Keep VPA recommend-only for critical paths; apply during maintenance.
- Separate node pools; enable CA with safe PDBs.
- Choose blue-green for big-bang changes; canary/progressive for frequent releases.
- Automate promotions/rollbacks via SLO-aware analysis.

Avoid:

- HPA targets on noisy or high-latency metrics.
- VPA + HPA both controlling CPU aggressively (feedback loops).
- Single-zone deployments for user-critical services.

References & Further Reading

- **Kubernetes:** HPA/VPA/CA concepts; Topology & Spread Constraints
- **Progressive Delivery:** Argo Rollouts, Flagger; Service Mesh routing
- **Ingress/Gateway API weighting techniques**
- **Autoscaling:** metrics adapters & best practices