Advanced Kubernetes Topologies

Multi-Tenancy • Federation • Multi-Cluster • Serverless • Databases • ML

Audience: Advanced / Intermediate (beginner-accessible)

Focus: Theory, architectures, decision frameworks, trade-offs

> Ziel: Architektur- und Entscheidungswissen für fortgeschrittene Kubernetes-Szenarien: Mandantenfähigkeit, Föderation, Multi-Cluster-Management, Serverless-Modelle, Datenbanken- und ML-Workloads. Keine Hands-on-Demos; Fokus auf Konzepte, Risiken und Entwurfsmuster.

Agenda

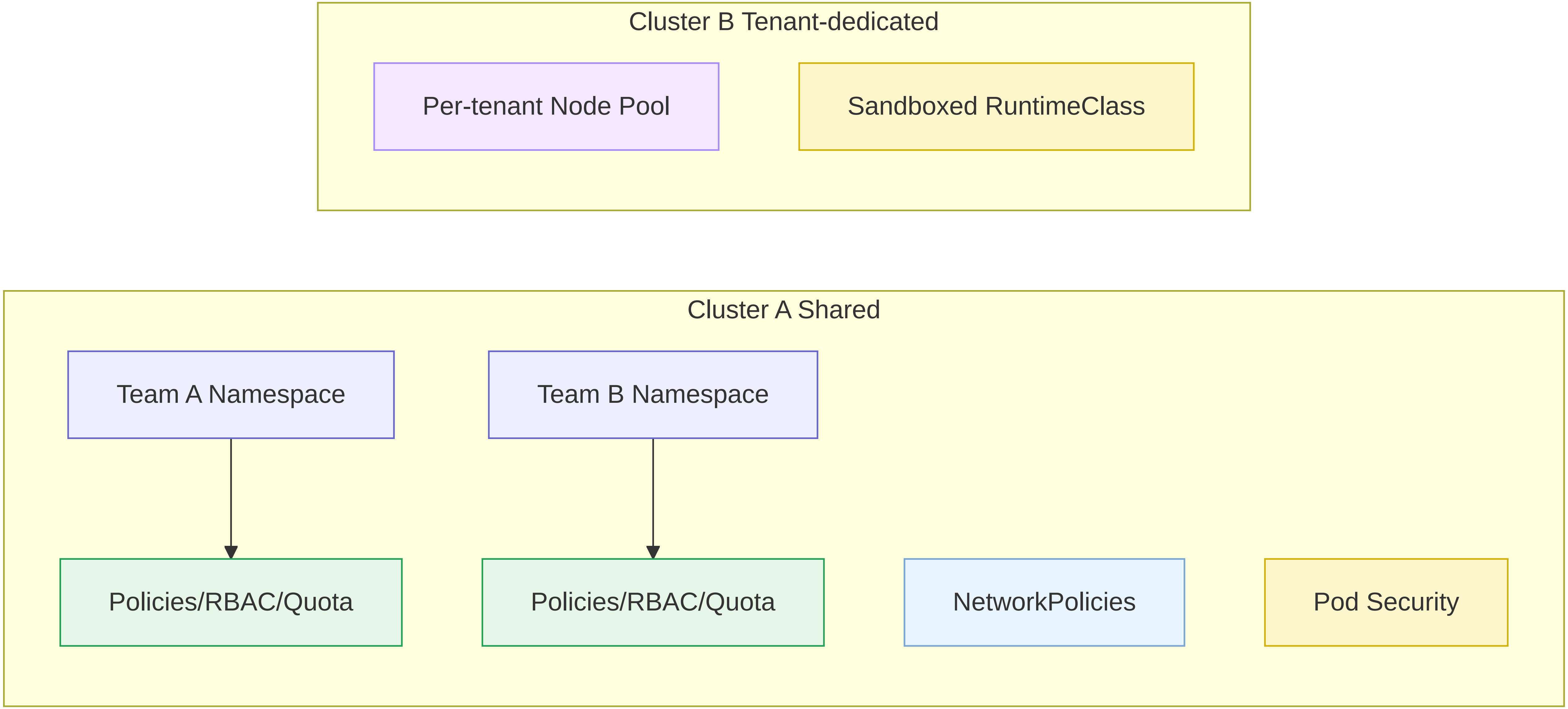
- Multi-Tenant Architectures
- Federation (KubeFed)
- Multi-Cluster Management Tools (Rancher, Anthos)
- Serverless on Kubernetes (Knative, OpenFaaS)
- Running Databases on Kubernetes (challenges, operators)
- Machine Learning on Kubernetes (Kubeflow, ML pipelines)

> Jede Sektion liefert ein konzeptionelles Diagramm und Kernleitfragen für Architekturentscheidungen.

1 Multi-Tenant Architectures

Tenancy Models & Isolation

- **Soft (namespaced) tenancy**: Teams/projects separated via namespaces, RBAC, quotas, NetworkPolicies, Pod Security, admission policies
- **Hard(er) tenancy**: Node pools per tenant, runtime sandboxing (gVisor/Kata), strict egress, per-tenant ingress & certs
- **Hardest (cluster-level) tenancy**: One cluster per tenant (strong blast-radius isolation), shared mgmt plane on top (cost ↑)


Cluster B Tenant-dedicated
Per-tenant Node Pool
Sandboxed RuntimeClass
Cluster A Shared
Team A Namespace
Team B Namespace
Policies/RBAC/Quota
Policies/RBAC/Quota
NetworkPolicies
Pod Security

Decision drivers: Risk profile, compliance, noisy neighbor tolerance, cost & ops maturity

> Modellwahl hängt von Risiko-/Compliance-Anforderungen und Teamreife ab. Soft Tenancy ist kosteneffizient, aber schwächer isoliert; Cluster-per-Tenant maximiert Isolation bei höheren Kosten.

Governance & Guardrails

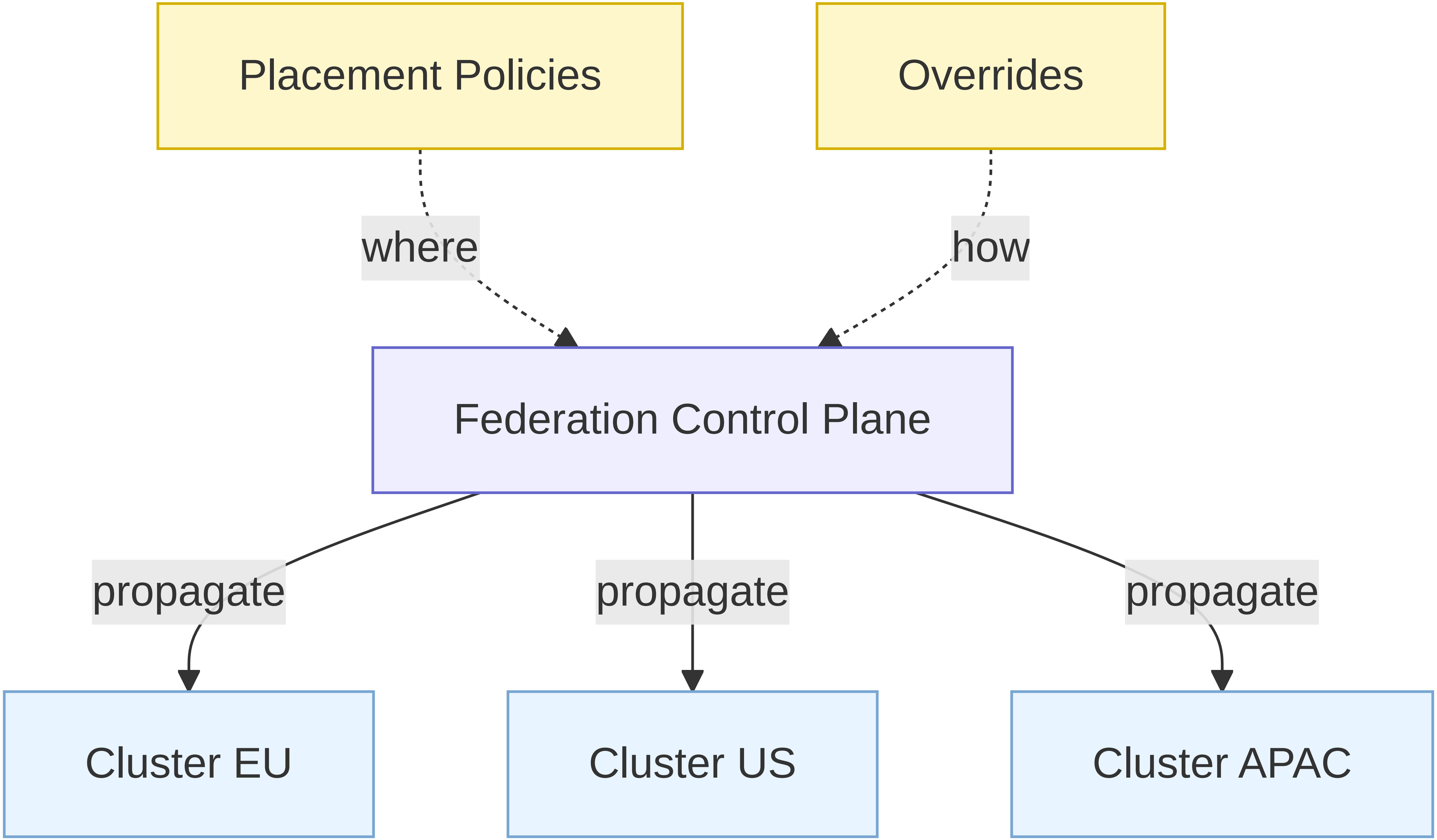
- **Identity & access:** SSO/OIDC → RBAC; least-privilege roles; namespace-bound contexts
- **Resources:** LimitRange & ResourceQuota per tenant; PriorityClasses; cost/usage metering
- **Policy:** OPA/Kyverno for image signing, namespaces labels, network defaults, pod security levels
- **Observability:** Per-tenant dashboards/alerts; log/metric multi-tenancy separation

> Governance ist zentral: konsistente Policies, Quoten, Identitäten und Observability-Prozesse pro Mandant.

2 Federation (KubeFed)

Purpose & Concepts

- **Goal:** Coordinate workloads/config across multiple clusters with declarative placement and propagation
- **Mechanics:** Federated types mirror native K8s kinds; a control plane propagates to member clusters based on policies
- **Use cases:** Multi-region HA, data sovereignty, blue/green by cluster, gradual regional rollouts


Placement Policies
Overrides
where
how
Federation Control Plane
propagate
propagate
propagate
Cluster EU
Cluster US
Cluster APAC

Key objects: FederatedDeployment/ConfigMap/Ingress, Placement (clusters), Overrides (per-cluster diffs)
Trade-offs: Added complexity, failure domains for the federator, lag & drift concerns

> Föderation abstrahiert Multi-Cluster als „eine Oberfläche“, erhöht aber Komplexität. Platzierungs- und Override-Policies sind die Kernelemente.

Federation vs. Alternatives

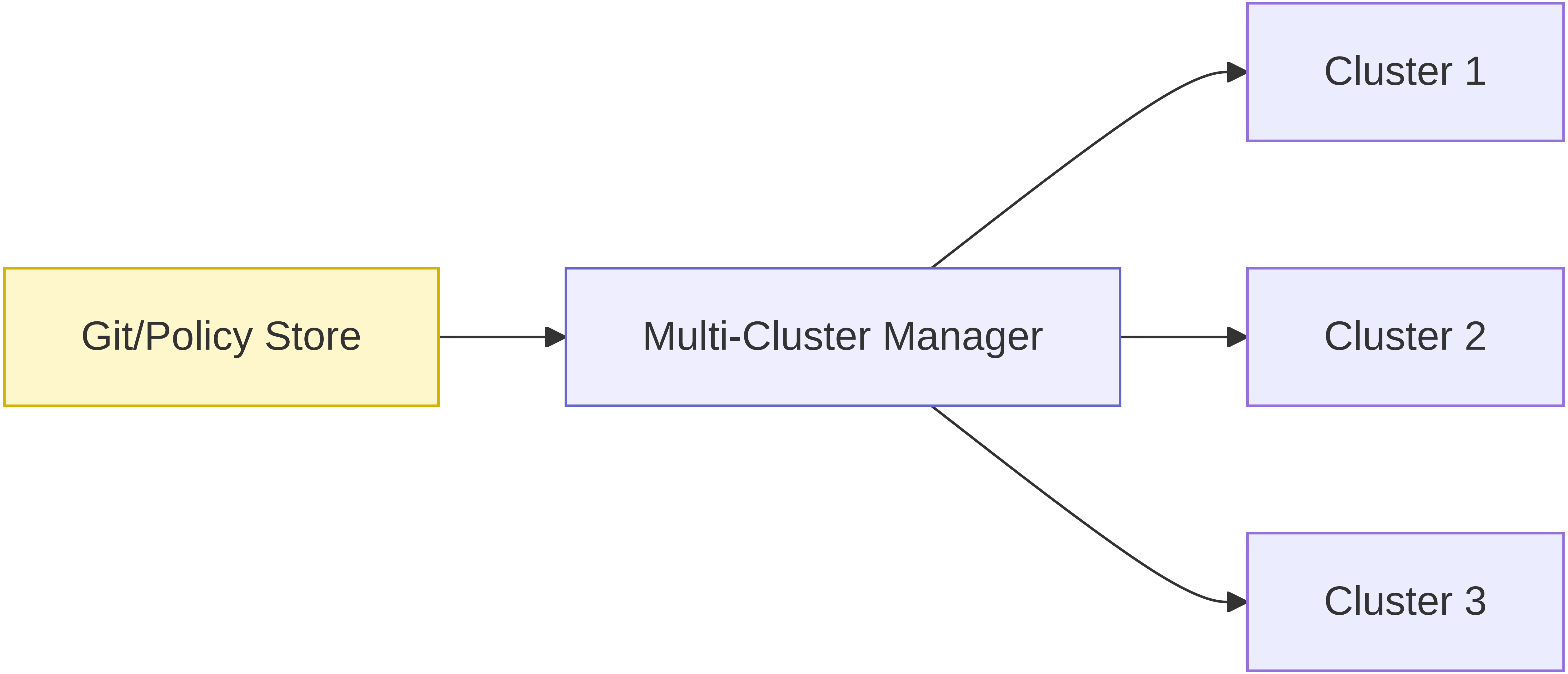
- **GitOps per cluster:** Independent controllers pulling env-specific repos (looser coupling)
- **Multi-cluster schedulers/meshes:** Traffic-level failover without object federation
- **Provider platforms (Anthos/AKS Arc):** Policy & config sync with vendor tooling

> Föderation ist nicht alternativlos: GitOps und Plattform-Tools können ähnliche Ziele mit anderer Kopplung erreichen.

3 Multi-Cluster Management Tools

Rancher, Anthos (and peers)

Capability	Rancher	Anthos (GKE + Config Mgmt)	Others (AKS Arc, EKS Blueprints)
Cluster lifecycle	Provision/import across clouds	Hybrid/multi-cloud attach	Varies by vendor
Policy/config sync	Fleet/GitOps	Config Sync, Policy Controller	Azure/GitOps/Policy, AWS add-ons
Access/IAM	Centralized auth, RBAC templates	Cloud IAM federation	Cloud-native
UI/ops	Rich multi-cluster UI	Deep GCP integration	Varies


Git/Policy Store
Multi-Cluster Manager
Cluster 1
Cluster 2
Cluster 3

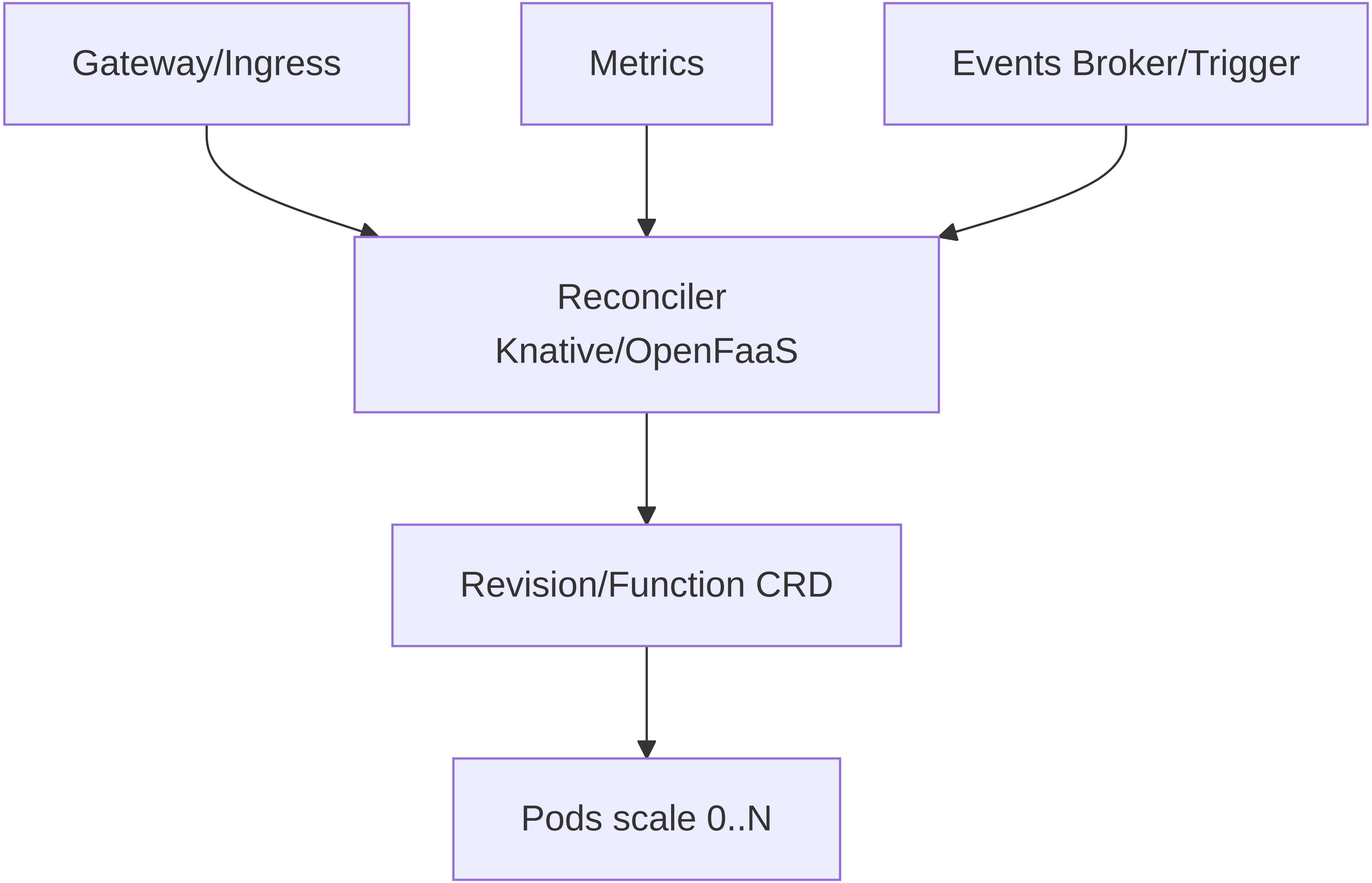
Decision drivers: Heterogeneity of environments, need for centralized policy, vendor lock-in tolerance

> Werkzeugwahl abhängig von Heterogenität und Governance-Anforderungen. Rancher ist herstellerneutral; Anthos integriert tief in GCP.

4 Serverless on Kubernetes

Concepts & Components (Knative, OpenFaaS)

- **Abstraction:** Developer focuses on functions/services; platform handles scaling, routing, and revisions
- **Knative Serving:** Revisions, Routes, Configurations; scale-to-zero, autoscale on requests (KPA/HPA)
- **Eventing (Knative):** Brokers/Triggers for event-driven apps; CloudEvents
- **OpenFaaS:** Functions as CRDs with templates, autoscaling, gateway & Prometheus integration


Gateway/Ingress
Metrics
Events Broker/Trigger
Reconciler
Knative/OpenFaaS
Revision/Function CRD
Pods scale 0..N

Use cases: Bursty traffic, APIs with idle periods, event-driven pipelines

Trade-offs: Cold starts, limited long-running connections, platform complexity

> Serverless auf K8s skaliert bedarfsorientiert und entkoppelt die App vom Infrastrukturbetrieb. Wichtig: Cold-Start, State-Management, Observability.

Serverless Best-Practice (Theory)

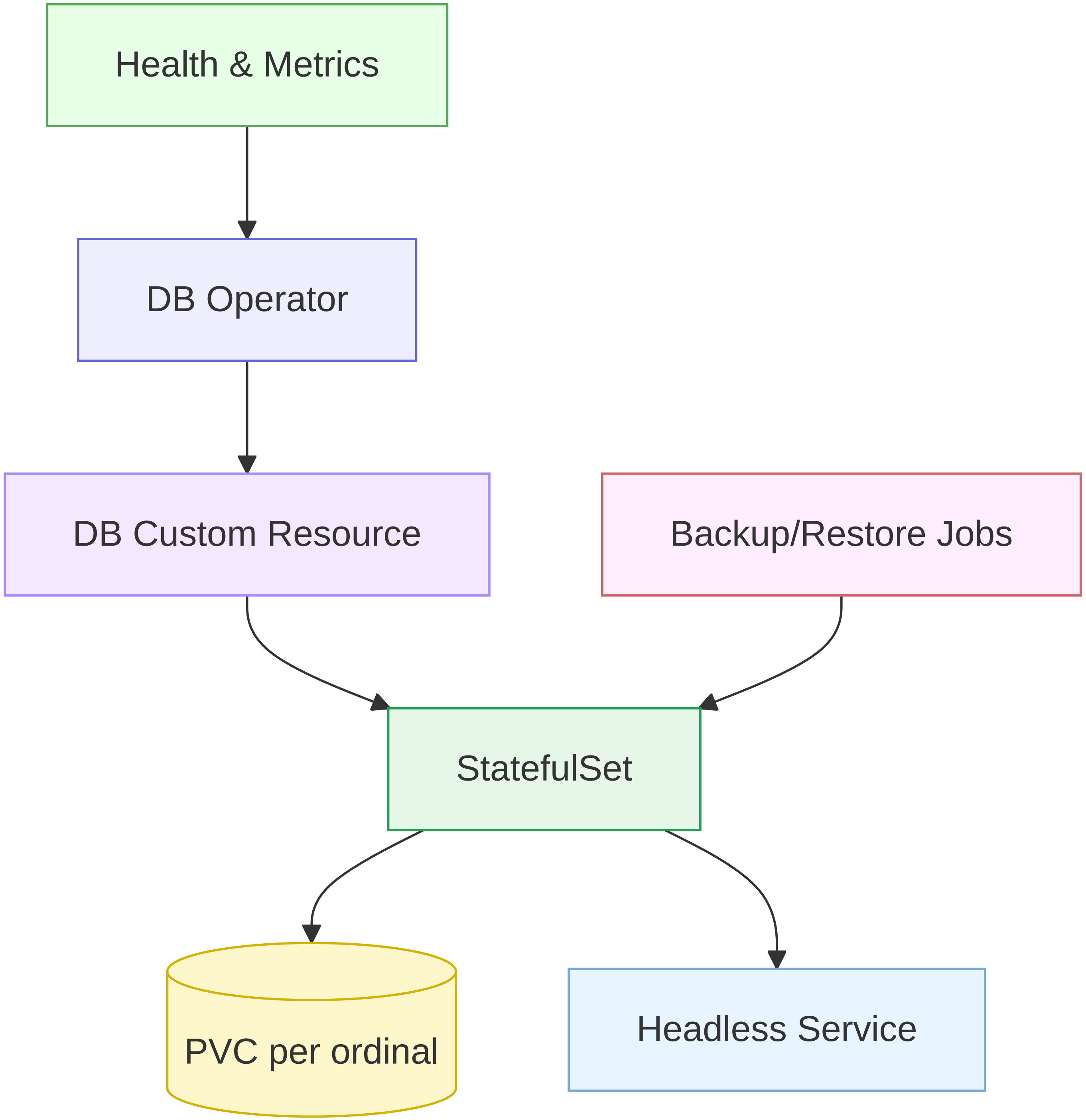
- Small, stateless handlers; idempotent; externalize state
- Pre-warm or min-scale for latency-sensitive paths
- Use mesh/ingress for canary & traffic shaping; enforce resource limits

> Funktionen klein und zustandslos halten, Latenz durch Min-Scale steuern, Traffic über Mesh/Ingress kontrollieren.

5 Running Databases on Kubernetes

Challenges & Patterns

- **Challenges:** Persistent performance (IOPS/latency), node/topology constraints, failover correctness, backup/restore, split-brain risk
- **Patterns:** StatefulSets with volumeClaimTemplates, topology-aware storage classes, PDBs & anti-affinity, operator-managed lifecycle


Health & Metrics
DB Operator
DB Custom Resource
Backup/Restore Jobs
StatefulSet
PVC per ordinal
Headless Service

Operator responsibilities: Bootstrapping, upgrades, leader elections, backups, PITR, topology & failover orchestration
When to prefer external DBaaS: Strict SLOs, heavy IO, compliance controls, ops capacity limits

> DBs im Cluster sind möglich, aber anspruchsvoll. Operatoren kapseln Betriebslogik. Bei hohen SLOs oder knappen Ops-Ressourcen ist DBaaS oft sinnvoller.

Storage & Topology Considerations

- **StorageClasses:** WaitForFirstConsumer, zonal replication, RWX vs. RWO semantics
- **Performance:** Pin pods via node/zone affinity; ensure sufficient IOPS; monitor fs/cache hit rates
- **Safety:** Quorum-aware rollouts, fencing, strict PDBs, backups verified regularly

> Topologie und Performance sind kritisch: korrekte Platzierung, IOPS-Planung und Rollout-Sicherheit (Quorum/Fencing).

6 Machine Learning on Kubernetes

Kubeflow & ML Pipelines

- **Why K8s for ML:** Reproducible environments, scalable training/inference, GPU scheduling, portable pipelines
- **Kubeflow:** KFServing/KServe for inference, Pipelines for DAG workflows, Notebooks, training operators (TFJob, PyTorchJob)

Pipeline concepts: DAGs, artifacts/metadata, caching of steps, experiment tracking
Serving: Canary models, A/B routes, autoscaling; GPUs/TPUs for training/inference

> Kubeflow strukturiert den ML-Lebenszyklus: Datenvorbereitung, Training, Registry, Serving und Rückkopplung über Metriken/Drift.


Data Source
Preprocess Step
Metrics/Drift
Training Job
TFJob/PyTorchJob
Model Registry
KServe Inference Service

ML Ops Considerations (Theory)

- **Data governance:** Lineage, versioning, access control, PII handling
- **Resource mgmt:** GPU node pools, quota, priority for training vs. prod inference
- **Observability:** Model metrics (latency, accuracy), drift detection, rollback to previous model

> ML erfordert zusätzliche Governance (Daten/Modelle) und Ressourcensteuerung (GPU-Pools). Observability muss Modellqualität abbilden (Drift/Accuracy).

Decision Frameworks (Summary)

- **Multi-tenancy:** Start namespaced; escalate to dedicated node pools or clusters as risk/compliance grows
- **Federation vs. GitOps:** Use KubeFed for object-level propagation; otherwise prefer per-cluster GitOps + traffic failover
- **Multi-cluster mgmt:** Choose by heterogeneity & governance needs; weigh lock-in vs. neutrality
- **Serverless:** Ideal for bursty, stateless, event-driven tasks; manage cold starts and state externally
- **Databases:** Operators enable in-cluster DBs, but DBaaS often wins for high SLOs
- **ML on K8s:** Kubeflow/KServe standardize pipelines & serving; plan GPU pools and model governance

> Zusammenfassende Leitlinien für Architekturentscheidungen über die sechs

References & Further Reading

- **Multi-Tenancy:** CNCF WG Multi-Tenancy, OPA/Kyverno best practices
- **Federation:** KubeFed documentation & design docs
- **Multi-Cluster:** Rancher, Anthos/Config Sync, AKS Arc, EKS Blueprints
- **Serverless:** Knative Serving/Eventing, OpenFaaS
- **Databases:** Operators (e.g., Zalando Postgres, Percona, Crunchy), StatefulSets
- **ML:** Kubeflow, KServe, Argo Workflows

> Primärquellen für vertiefte Planung. Immer aktuelle API-Versionen, Treiberfähigkeiten und Projektreife beachten.