Kubernetes Security

Authentication • Authorization (RBAC) • Network Policies • Security Contexts • Secrets • PSA vs. PSP • OPA/Kyverno

Audience: Advanced / Intermediate (beginner-accessible)

Focus: Theory-first, platform guardrails & policy models

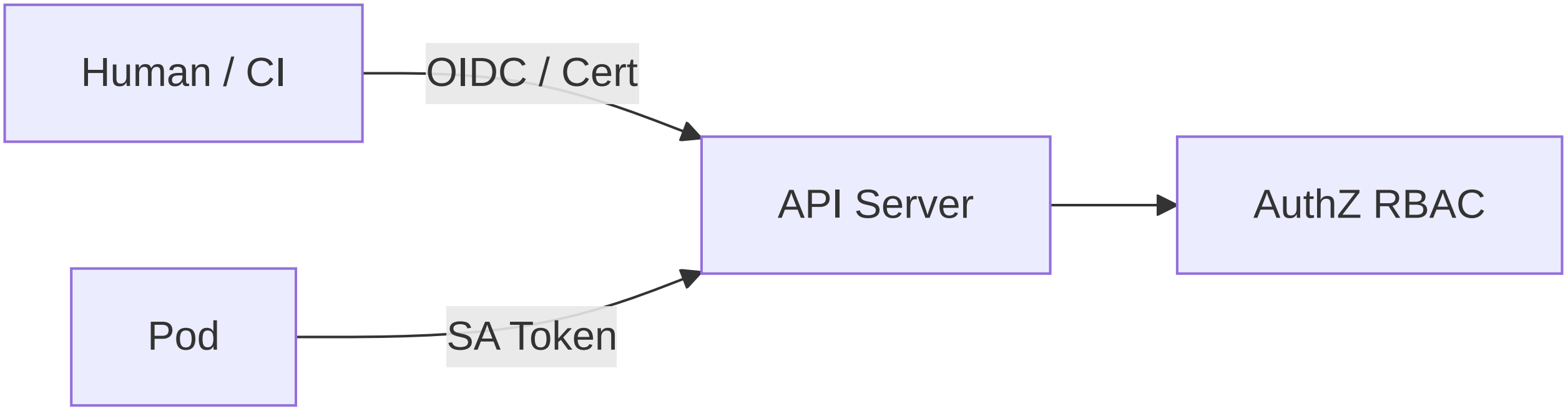
Agenda

- Authentication (Users, Service Accounts)
- Authorization (RBAC)
- Network Policies & Pod Security (overview)
- Security Contexts (runAsUser, capabilities)
- Secrets Management Best Practices
- Pod Security Admission (PSA) vs. deprecated PSP
- Kubernetes Security Tools (OPA/Gatekeeper, Kyverno)

1) Authentication

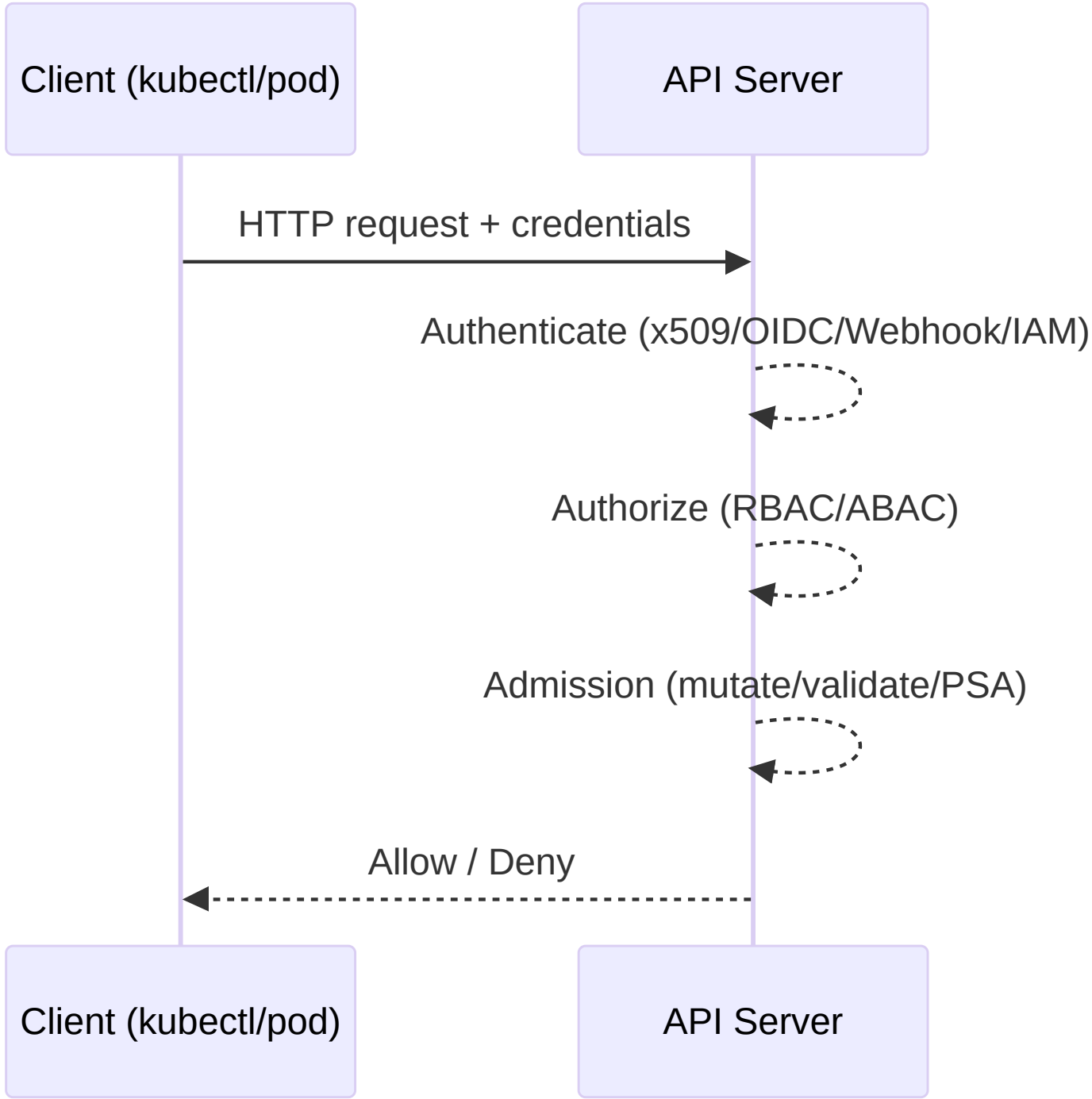
Users & Service Accounts (SAs)

- **Users (human/CI):** External to cluster; auth via client certs, OIDC, webhook token auth, cloud IAM plugins
- **ServiceAccounts (workload identity):** Namespaced identities for pods; tokens projected to pods (bound, short-lived recommended)
- **Best practice:** Centralize on OIDC for humans; use service account tokens (or external workload identity) for pods; rotate credentials


Human / CI
OIDC / Cert
API Server
AuthZ RBAC
Pod
SA Token

Auditing: Enable API audit logs; trace who did what, where, when

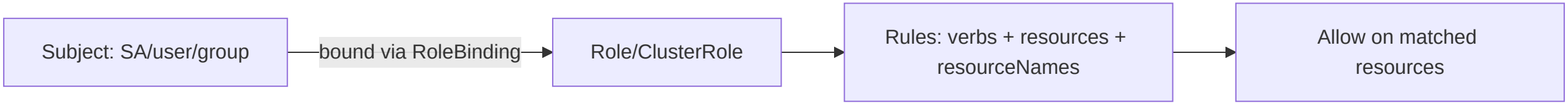
Auth Flow (Conceptual):


Client (kubectl/pod)
API Server
HTTP request + credentials
Authenticate (x509/OIDC/Webhook/IAM)
Authorize (RBAC/ABAC)
Admission (mutate/validate/PSA)
Allow / Deny
Client (kubectl/pod)
API Server

2) Authorization (RBAC)

Role-Based Access Control

- **Subjects:** Users, groups, service accounts
- **Roles:** Sets of permissions (verbs on resources) scoped to namespace
- **ClusterRoles:** Cluster-scoped or namespace-applied reusable roles
- **Bindings:** (Cluster)RoleBinding = subject ↔️ role association


Subject: SA/user/group
bound via RoleBinding
Role/ClusterRole
Rules: verbs + resources + resourceNames
Allow on matched resources

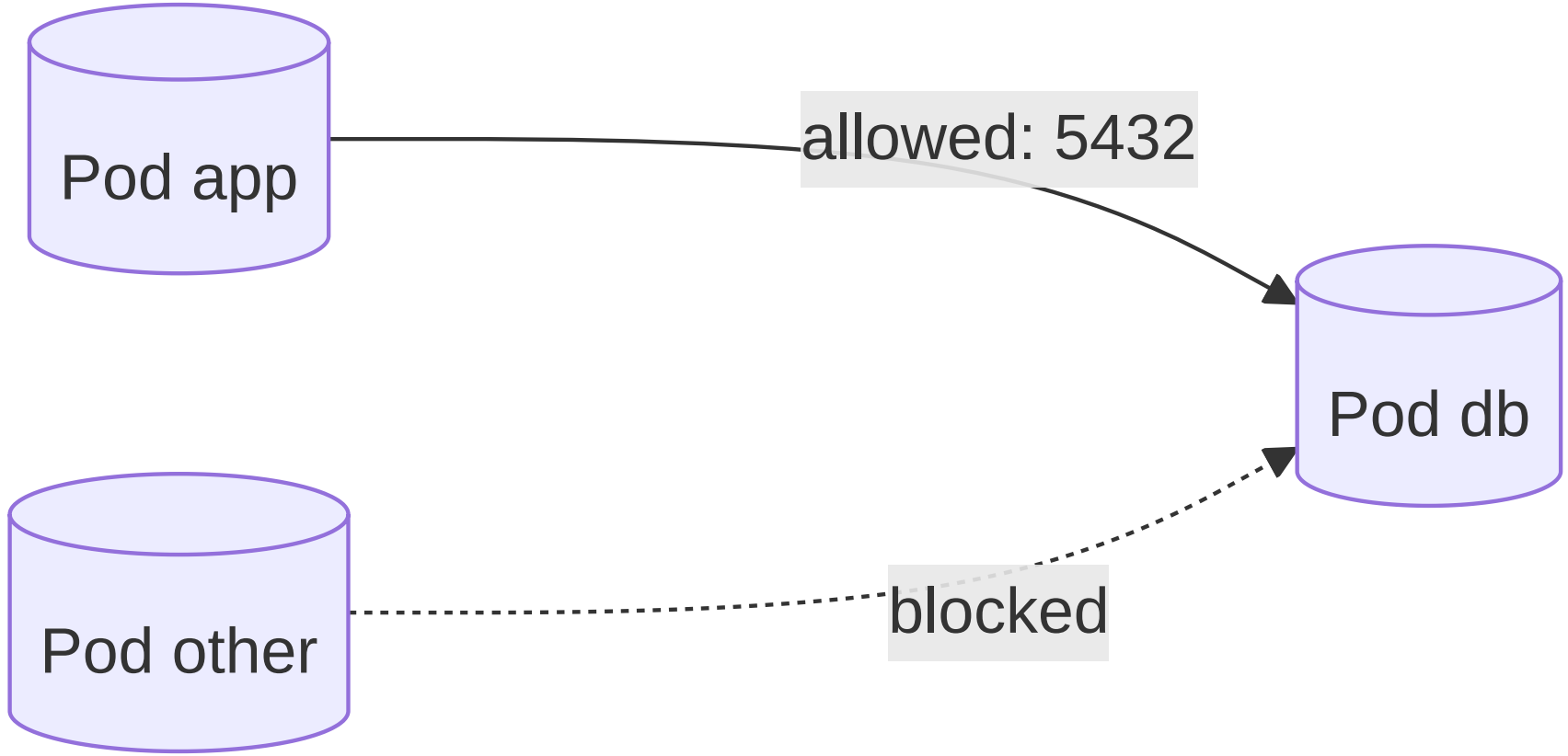
Principles: Least privilege; separate read vs write; break-glass roles; use label selectors only where supported via admission

RBAC Patterns

- Team roles: view, edit, admin per namespace; avoid cluster-admin for routine
- CI/CD: Narrowed ClusterRoles for deployments only (no secrets/privileged ops)
- Impersonation for audits: --as, --as-group to test policies

3) Network Policies & Pod Security (Overview)

- **NetworkPolicy:** Namespaced L3/L4 allow-lists (Ingress/Egress) by pod/namespace selectors and IPBlocks
- **Effect:** Once any policy selects a pod, deny-by-default for unspecified directions/ports
- **Pod Security (runtime posture):** Linux user/caps/seccomp/AppArmor/SELinux set per Pod/Container (via securityContext)



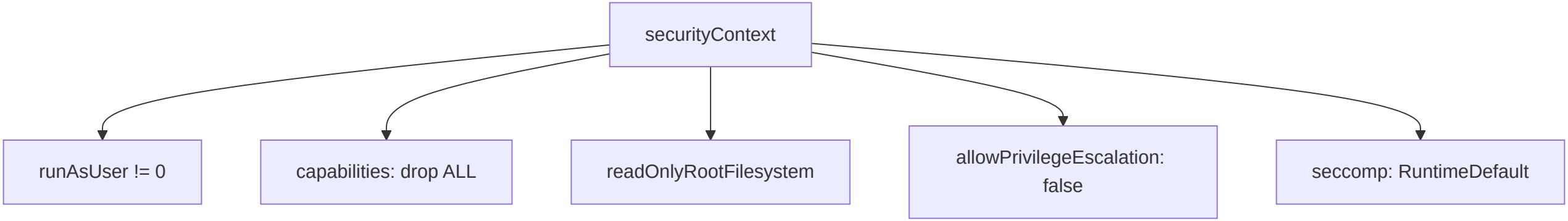
NetworkPolicy: allow
app → db:5432

Combine: NP (network plane) + Pod Security (process/container plane)

4) Security Contexts

Process & Container Hardening

- **Pod/Container securityContext:**
  - User/Group: runAsUser, runAsGroup, fsGroup
  - Capabilities: Drop all; add minimal (NET_BIND_SERVICE etc.)
  - Privilege: allowPrivilegeEscalation: false, privileged: false
  - Filesystem: readOnlyRootFilesystem: true
  - Seccomp: seccompProfile: RuntimeDefault (or custom)
  - SELinux/AppArmor: Confinement profiles where available


securityContext
runAsUser != 0
capabilities: drop ALL
readOnlyRootFilesystem
allowPrivilegeEscalation: false
seccomp: RuntimeDefault

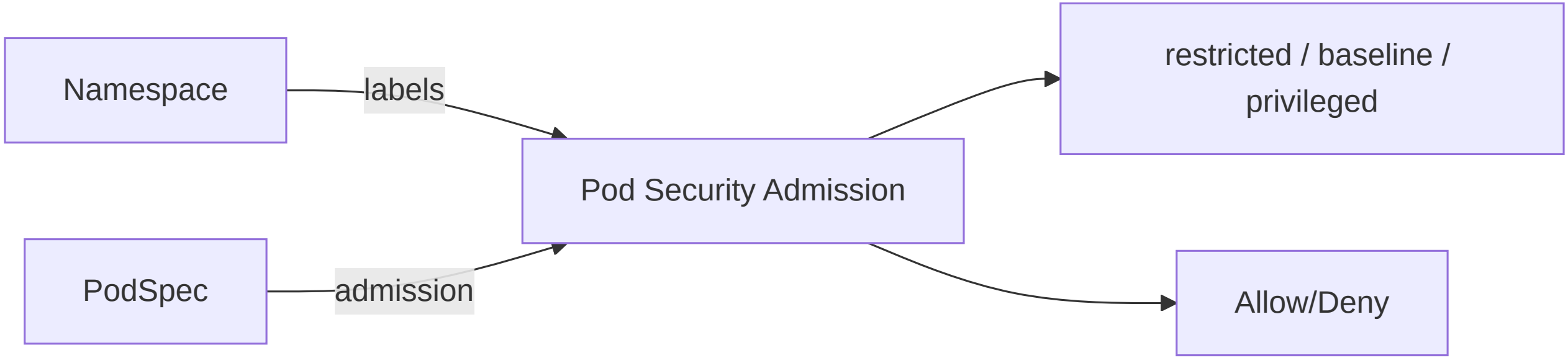
Guidance: Enforce via PSA and validating policies (OPA/Kyverno)

5) Secrets Management Best Practices

- **Storage realities:** Kubernetes Secrets are base64-encoded in etcd; enable encryption at rest (EncryptionConfiguration)
- **Access control:** RBAC least-privilege; avoid broad get/list on secrets
- **Mount vs. env:** Prefer file mounts for large/rotating secrets; avoid long-lived env vars
- **External managers:** Integrate cloud KMS / secret managers or CSI Secrets Store driver
- **Rotation:** Short-lived tokens/certs; automate rotation and revoke on compromise
- **Backups:** Treat etcd/secret backups as sensitive; encrypt and restrict access

6) Pod Security Admission (PSA) vs. PodSecurityPolicy (PSP)

- **PSP (deprecated):** Complex cluster-wide objects validating pod specs; removed in recent releases
- **PSA:** Namespace-level labels enforce Baseline/Restricted (or Privileged) policy sets at admission

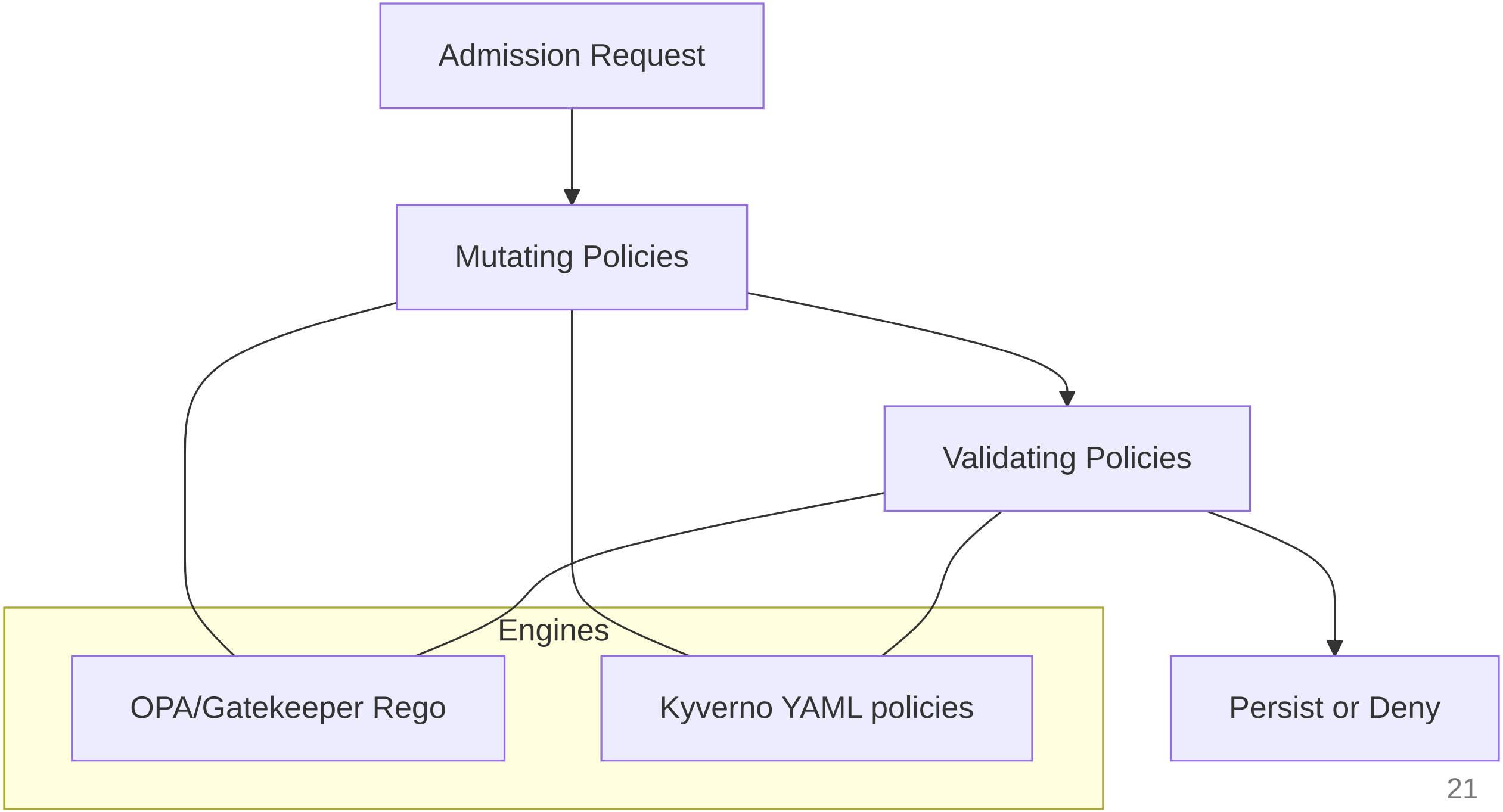

Namespace
labels
PodSpec
admission
Pod Security Admission
restricted / baseline / privileged
Allow/Deny

- **Modes:** enforce, audit, warn (can combine)
- **Recommended:** Default to restricted for most namespaces; escalate via exception process
- **Gaps:** PSA is policy bundles; use OPA/Kyverno for custom constraints

7) Kubernetes Security Tools

OPA/Gatekeeper & Kyverno (Policy-as-Code)

- **OPA Gatekeeper:** Rego-based constraints/templates; validating/mutating (v3 supports mutations); strong for cross-resource logic
- **Kyverno:** Kubernetes-native policy language (YAML), easier for cluster admins; validate/mutate/generate/sync


Admission Request
Mutating Policies
Validating Policies
Engines
OPA/Gatekeeper Rego
Kyverno YAML policies
Persist or Deny

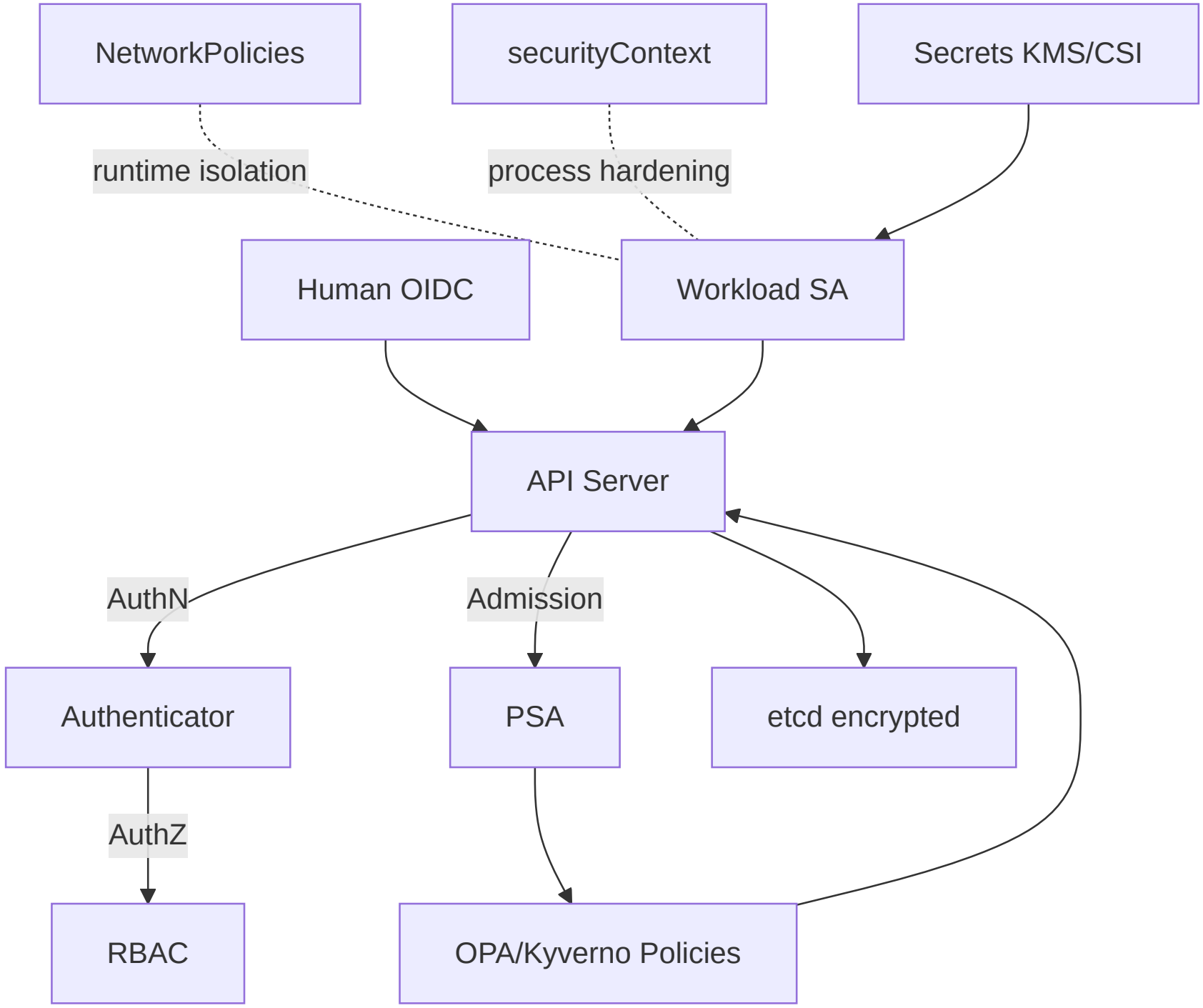
Use cases: Enforce signed images, forbid :latest, require labels/annotations, ensure runAsNonRoot, default seccomp, restrict host networking, quota guardrails

Policy Strategy (Guidance):

- Shift-left: Validate with policy in CI (conftest/kyverno CLI)
- Admission gates: Fail-closed on high-severity misconfigs; allow audit/warn for migration
- Exceptions: Controlled via labels/PolicyExceptions; time-bound with approvals
- Observability: Export policy violations; monitor deny rates and latency

Putting It Together

Reference Security Architecture (Theory)


NetworkPolicies
securityContext
Secrets KMS/CSI
runtime isolation
process hardening
Human OIDC
Workload SA
API Server
AuthN
Admission
Authenticator
PSA
etcd encrypted
AuthZ
RBAC
OPA/Kyverno Policies

Layers: AuthN/Z → Admission (PSA + policies) → Runtime hardening (securityContext) → Network isolation (NetPol) → Secret hygiene

Design Takeaways

- Centralize AuthN (OIDC) and separate human vs. workload identities
- Enforce least privilege RBAC; test with impersonation
- Apply PSA restricted by default; extend with OPA/Kyverno for custom rules
- Harden pods with securityContext; deny-by-default networking with explicit allows
- Treat secrets as high-risk data; encrypt, minimize, rotate, and prefer external KMS/CSI

References & Further Reading

- **Kubernetes Docs:** AuthN/AuthZ, RBAC, Admission, Pod Security Admission
- **OPA/Gatekeeper & Rego language; Kyverno policies**
- **NetworkPolicies & CNI capabilities**
- **Secrets:** EncryptionConfiguration, Secrets Store CSI driver
- **Benchmark:** CIS Kubernetes Benchmark (as a checklist)