**Security Orchestration & Risk Platform  
High‑Level Design (HLD)**

Architecture • Threat Model • Data Model • Risk Scoring • Deployment • Roadmap

*August 22, 2025*

# 1. Executive Summary

This document defines the high‑level design for a modular security orchestration platform that discovers, scans, and prioritizes architectural risks across code, pipelines, containers, cloud/Kubernetes, and external exposure. The platform emphasizes Zero Trust, Policy‑as‑Code, evidence‑driven governance, and an asset/relationship graph to highlight blast radius and attack paths.

Key tenets:

* Zero Trust by default (OIDC/SAML SSO, RBAC/ABAC, least privilege, short‑lived tokens).
* Policy‑as‑Code (OPA/Rego) with exemptions requiring justification and expiry.
* Evidence & auditability (immutable artifacts, signed attestations, versioned policies).
* Data minimization and residency; encryption in transit/at rest; field‑level redaction.
* Noise control (exploitability, exposure, criticality, compensating controls).
* Extensible plug‑in SDK for scanners; event‑driven jobs; normalized findings model.

# 2. Scope & Goals

Goals: actionable, architecture‑aware risk reduction; fast MVP; OSS‑first; compliance‑ready.

Out of scope (initially): full SOAR playbooks, deep red‑team tooling, and full ASM breadth.

# 3. Architecture Overview

The platform uses a Spring Boot core with a React/Vue dashboard, Spring Cloud Gateway as an API gateway, and independent Spring Boot microservices for scanners (SAST, IaC, CI/CD, SBOM/Image, CSPM/KSPM). A queue (Kafka/RabbitMQ) schedules work; raw reports and SBOMs are stored in an object store; normalized findings and relationships are persisted in PostgreSQL, with a graph view for architectural analysis.

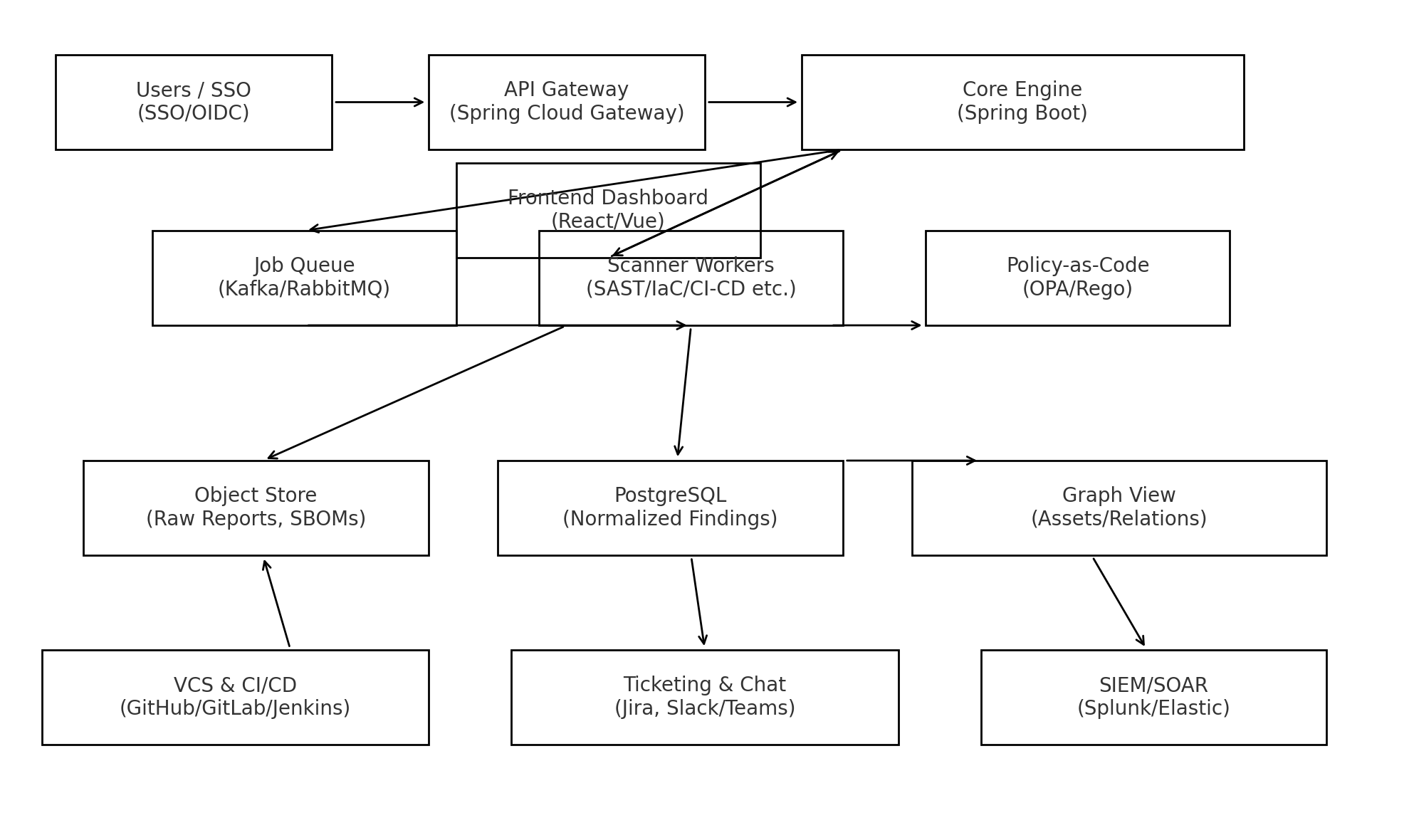


Figure 1: High‑level component interactions and integrations.

## 3.1 Components

1. **API Gateway:** Routing, auth offloading, rate limiting; service discovery to scanners.
2. **Core Engine:** Job orchestration, normalization, risk scoring, suppression/SLO governance, integrations.
3. **Scanner Workers:** SAST/IaC/CI‑CD/Container/CSPM/KSPM; pluggable via SDK; idempotent with content‑hashing.
4. **Queue:** Back‑pressure, retries, dead‑letter; scalable fan‑out for large repos/orgs.
5. **Storage:** Object store for raw artifacts; PostgreSQL for normalized data; optional graph store/view.
6. **Policy Engine:** OPA/Rego policies; compliance mappings; exemption workflow with expiry.
7. **Dashboard:** Executive and engineer views, triage workflows, reports/exports; SSO‑aware.

# 4. Threat Model

Assets: tenant data (findings, SBOMs), credentials/tokens, policies, audit logs, binaries, diagrams.

Trust boundaries: user browser ↔ gateway ↔ core; core ↔ workers; workers ↔ VCS/CI/cloud; multi‑tenant data.

Methodology: STRIDE with mitigations.

|  |  |  |
| --- | --- | --- |
| STRIDE | Risk Example | Mitigations |
| Spoofing | Token theft / SSO misuse | OIDC/OAuth2, short‑lived tokens, mTLS, client secrets in KMS |
| Tampering | Report/attestation alteration | Immutable object store, signatures (cosign/in‑toto), audit trails |
| Repudiation | Operator actions denied | Signed audit logs, time sync, non‑repudiation policies |
| Information Disclosure | PII leakage in logs | Data minimization, redaction, differential logging |
| Denial of Service | Scan storms / hot repos | Rate limits, quotas per tenant, autoscaling workers |
| Elevation of Privilege | Over‑broad roles/SCM tokens | RBAC/ABAC, scoped tokens, approval workflows |

# 5. Data Model

Normalized entities and key relations are depicted below; graph view supports attack‑path analytics.

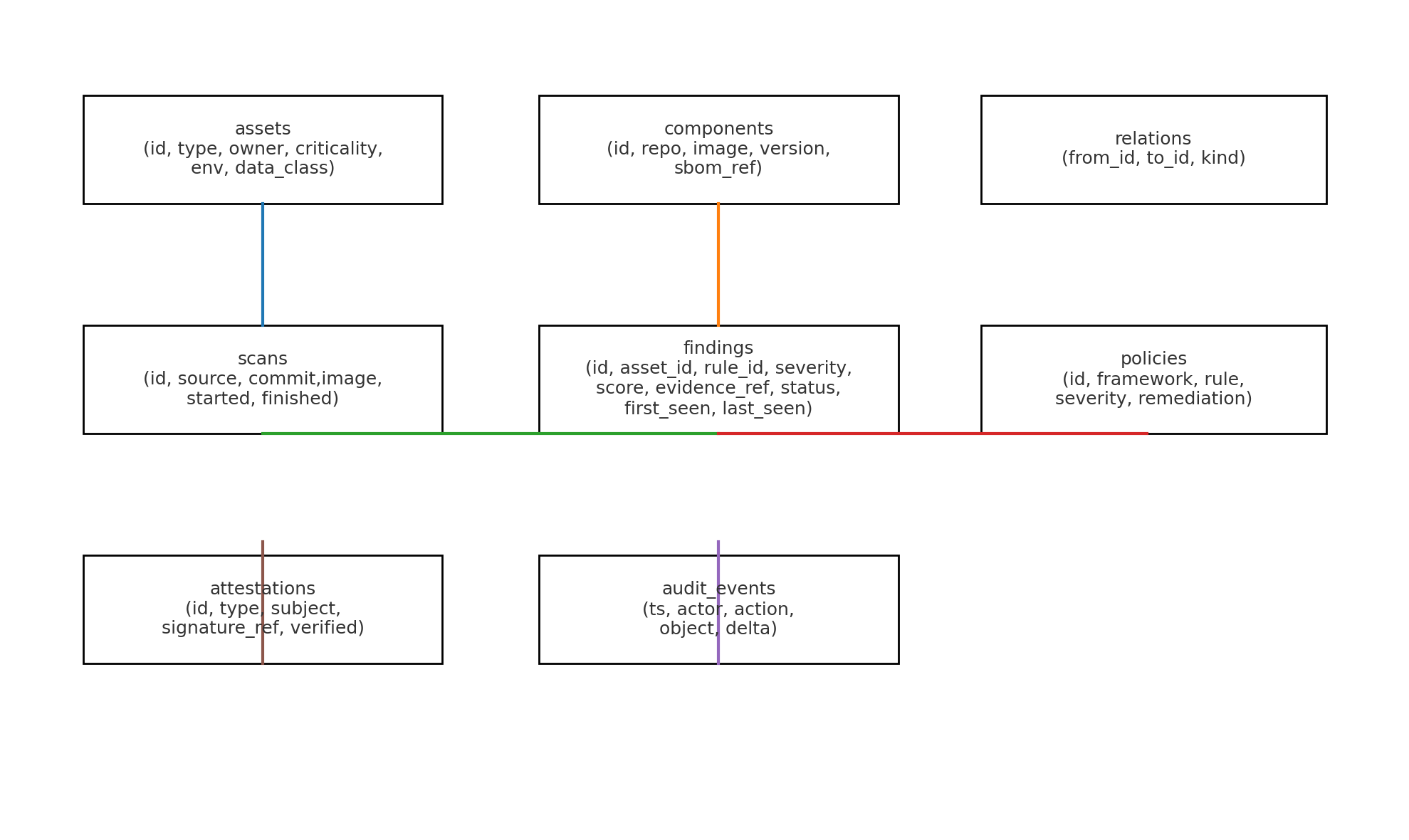


Figure 2: Core entities (assets, components, scans, findings, policies, attestations, audit).

# 6. Risk Scoring

Score = Severity × Exploitability × Exposure × Asset Criticality × Compensating Controls. Exploitability considers public exploits and KEV/EPSS‑like signals; Exposure captures internet‑facing reachability and lateral movement via the asset graph; Criticality is business/data tier; Controls reflect WAF, network policies, and runtime protections.

Prioritization: show 'Fix First' list with rationale and exact patch/config snippets.

# 7. Findings Lifecycle & Governance

**States:** New → Triaged → Assigned → Mitigated/Fixed → Verified → Closed; Won’t‑Fix with expiry.

SLOs: e.g., Critical on internet‑facing assets fixed ≤ 7 days; auto‑reopen on context change (new exploit, exposure change).

# 8. Integrations

SCM/CI: GitHub/GitLab/Jenkins/Azure DevOps; PR annotations, required checks.

Ticketing/Chat: Jira/Azure Boards; Slack/Teams deep links.

SIEM/SOAR: Splunk/Elastic/Chronicle exports and webhooks for playbooks.

# 9. Deployment Topologies

SaaS multi‑tenant: per‑tenant schema or RLS; region‑pinned storage (e.g., ap‑southeast‑2 for AU).

Private/Air‑gapped: offline scanner bundles; mirrored registries; optional agent; license server; strict egress controls.

Backups & DR: defined RPO/RTO; cross‑region encrypted backups; restore drills.

# 10. Roadmap

MVP (6–8 weeks):

* Engine + Dashboard + PostgreSQL + Liquibase
* SCA/SAST: Dependency‑Check + Semgrep (repo URL → results)
* IaC: Checkov + Trivy Config
* CI/CD: TruffleHog + Semgrep rules for pipelines
* Integrations: GitHub/GitLab + Jira, basic Slack
* Risk scoring v1 + triage workflow; CSV/PDF exports

Phase 2:

* SBOM (Syft) + Image scanning (Trivy/Grype), registry hooks
* Policy‑as‑Code packs + exemption workflow with expiry
* Asset graph + basic attack‑path views
* VEX/KEV exploitability flags

Phase 3:

* CSPM/KSPM (cloud API pulls, kube‑bench, Polaris) + drift detection
* Admission control (Gatekeeper/Kyverno) + PR autofix for IaC
* Provenance/signing (in‑toto, cosign), SLSA‑aligned builds

Phase 4 (optional/advanced):

* DAST (ZAP) and API security linting
* Falco runtime detections; SOAR webhooks
* ML‑assisted dedup/triage & fix ranking

# 11. Acceptance Criteria (Samples)

|  |  |  |
| --- | --- | --- |
| Feature | Acceptance Criteria | Evidence |
| Repo Scan (SAST/SCA) | Given a public repo URL, when a scan is triggered, then results appear under the project with severity, file/line, and remediation; duplicate findings are deduped across commits. | UI screenshot; normalized records; raw report in object store |
| IaC Scan | Given a Terraform/K8s directory, policy violations are listed with rule IDs and suggested fixes; a PR autofix can be generated for simple misconfigurations. | PR with patch; policy evaluation logs |
| Risk Scoring | Critical items reflect exploitability and exposure (internet‑facing) and sort to the top; changing asset exposure updates priority. | Before/after score snapshot; graph path showing exposure |
| SSO & RBAC | SSO login works; users see only their tenant/projects; roles enforce least privilege. | SSO test log; role matrix; access tests |

# 12. Sequence Flows

12.1 Repo Scan Flow

1. User submits repo URL and selects scanners.
2. Engine creates a job with content hash; enqueues to Queue.
3. Worker checks cache; clones repo if needed; runs Semgrep/Dependency‑Check.
4. Raw artifacts → Object store; parsed findings → PostgreSQL; relations updated.
5. Risk score computed; tickets/notifications emitted; dashboard updated.

12.2 Cloud/K8s Posture Flow

1. Connector assumes read‑only role; pulls cloud/K8s configs.
2. Policies evaluated (OPA/Rego, Checkov, kube‑bench/Polaris).
3. Findings normalized and linked to assets; drift vs. IaC computed.
4. Optional PRs generated for IaC fixes; posture score updated.
5. Evidence exported for compliance where applicable.

# 13. Platform Security

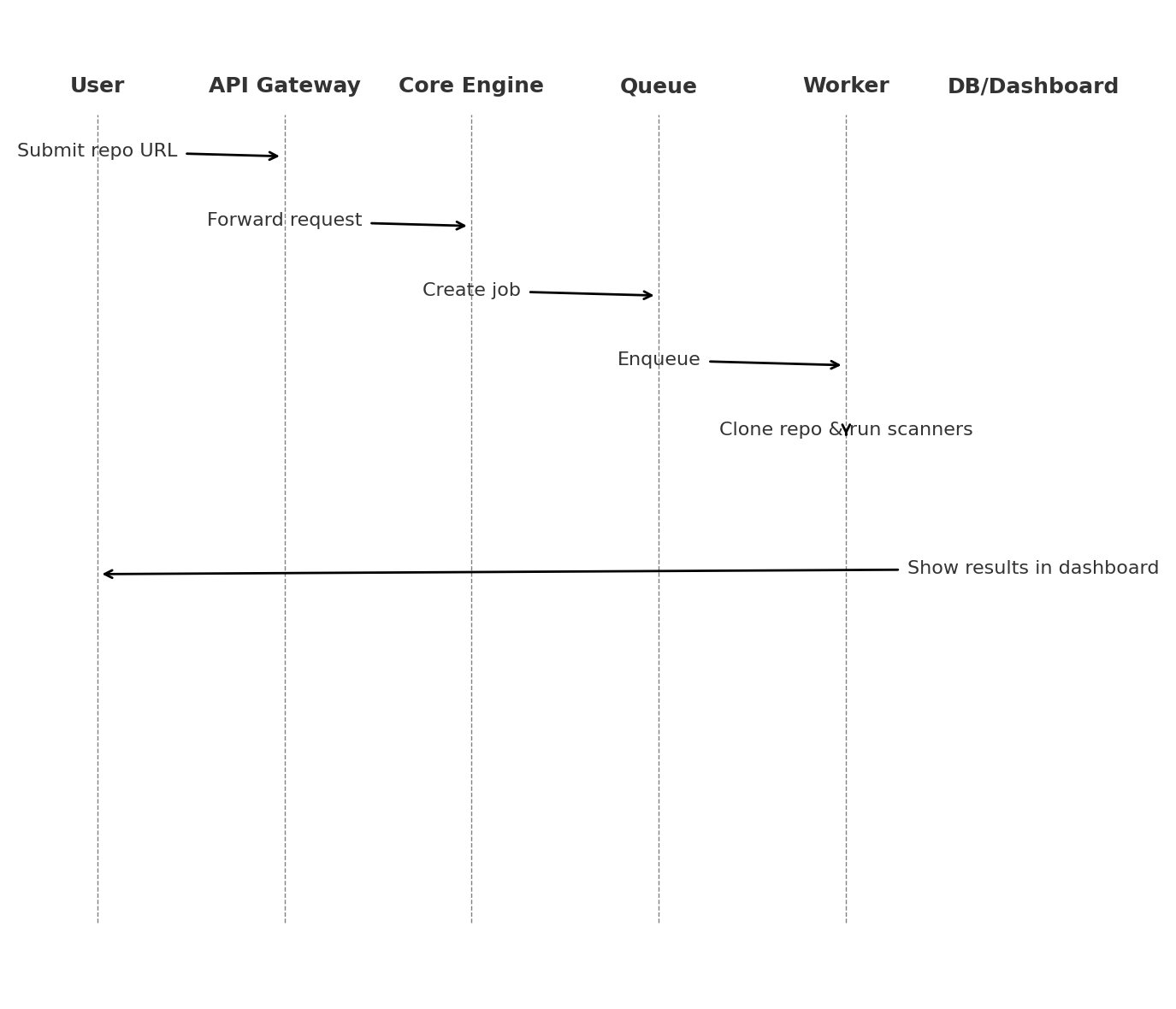
Tenant isolation (schema/RLS), KMS‑backed encryption, key rotation; hardened images; signed releases; backups/DR; rate limiting and anomaly detection; privacy controls and data retention.

# 14. Open‑Source Tooling Map

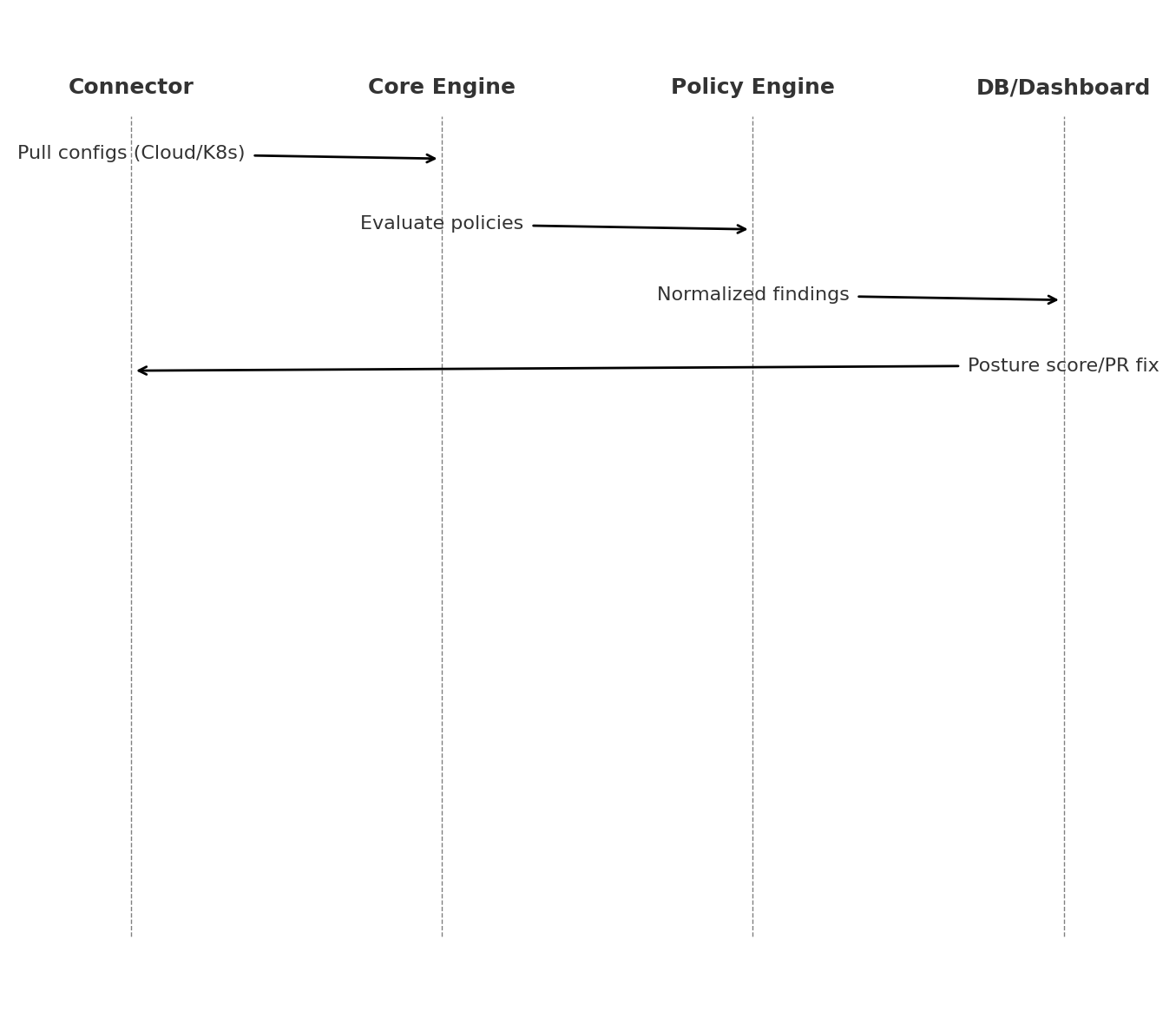
* SAST/SCA: Semgrep, SpotBugs, OWASP Dependency‑Check
* IaC: Checkov, Trivy (config)
* Containers: Syft/Grype, Trivy (images)
* K8s: kube‑bench, Polaris/kube‑score, Gatekeeper/Kyverno
* Runtime: Falco
* Supply chain: in‑toto, cosign, SBOM via Syft
* Secrets: TruffleHog, gitleaks
* Policy: OPA/Rego

# Appendix: Visual Diagrams

## A1. Repo Scan Sequence Diagram



## A2. Cloud/K8s Scan Sequence Diagram



## A3. Example Attack Path Graph

