Below is a **ready-to-use, production-grade set of Jira / Confluence templates** plus **OPA (Open Policy Agent) / policy-as-code examples** aligned with your **healthcare, multi-product, HIPAA, Zero Trust architecture**.

This is how mature orgs **scale architecture governance without meetings**.

📘 Confluence Templates

1️⃣ Architecture Decision Record (ADR) Template

Page Title: ADR-XXX –
Space: Architecture / Decisions

# ADR-XXX – <Decision Title>

## Status

Proposed | Approved | Superseded | Deprecated

## Context

What problem are we solving? What constraints exist (clinical, regulatory, technical)?

## Decision

What is the decision? Be explicit.

## Options Considered

1. Option A – Pros / Cons

2. Option B – Pros / Cons

3. Option C – Pros / Cons

## Consequences

### Positive

-

### Negative

-

## Guardrails

- Required patterns

- Prohibited patterns

- Follow-up actions

## Owners

Architect:

Approvers:

## Date

YYYY-MM-DD

2️⃣ Solution Architecture Review Template

# Solution Architecture Review

## Solution Overview

Product / Program:

Business outcome:

Scope:

## Architecture Diagram

(C4 L2/L3 diagram link)

## Key Decisions (ADR links)

- ADR-001
- ADR-002

## Integration Patterns

- API / Event / Batch / Façade
- Platform services used

## Security & Compliance

- PHI flows documented (Y/N)
- Threat model completed (link)
- IAM integration (Y/N)

## NFRs

Availability:

Latency:

RTO/RPO:

## Risks & Mitigations

- Risk:
- Mitigation:

## Review Outcome

Approved | Approved with conditions | Rejected

3️⃣ Data Product Definition Template

# Data Product: <Name>

## Domain

Care / Claims / Member / Provider / Ops

## Owner

Product owner:

Tech owner:

## Input Sources

- Event streams

- APIs

- CDC

- Batch

## Output Interfaces

- Tables

- APIs

- Events

- Features

## Quality SLAs

Freshness:

Completeness:

Accuracy:

## Security

PHI: Yes/No

Masking: Yes/No

Access policy:

## Consumers

- Analytics
- ML
- GenAI
- External

4️⃣ **Security Architecture Review Template**

# Security Architecture Review

## Scope

Application / Data / AI / Integration

## Data Classification

Public / Internal / PII / PHI

## Controls

- IAM (OIDC, MFA)
- Encryption
- Audit logging

- DLP

- Network segmentation

## Threat Model

STRIDE / ATT&CK (link)

## Findings

- Finding:

- Severity:

- Remediation:

## Approval

Security Lead:

Date:

5️⃣ Runbook Template (Ops + Audit)

# Runbook – <Service Name>

## Purpose

What this service does

## Dependencies

- Platform services

- External systems

## Startup / Shutdown

Steps...

## Incident Response

- Alert types

- Escalation path

- Manual fallback

## DR Procedure

RTO / RPO

Restore steps

## Audit Evidence

Log location:

Retention:

🎫 **Jira Templates**

1️⃣ **Architecture Review Ticket (ARCH-Review)**

Issue Type: Architecture Review
Workflow: Draft → Review → Approved → Closed

Fields

- Solution name
- Diagram link
- ADR links
- PHI (Y/N)
- Platform services used

- Target release
- Risk level

2️⃣ Architecture Exception Request (ARCH-Exception)

Title: Architecture Exception – <Reason>

Description:

What standard is violated?

Why is exception needed?

Expiry date:

Mitigation plan:

Approval:

EA / Security / Platform

Auto-close date:

3️⃣ Security Finding Ticket

Issue Type: Security Finding
Severity: Critical / High / Medium / Low
SLA: Auto-calculated

4️⃣ Data Quality Issue

Issue Type: Data Quality
Fields

- Data product
- Rule violated

- Impacted consumers
- SLA breach (Y/N)

🔐 OPA / Policy-as-Code Examples

(Enforced in CI/CD, Terraform, Kubernetes, API gateway)

1️⃣ Block direct EHR access

```
package architecture.rules


deny[msg] {
 input.resource.type == "service"
 input.resource.endpoint == "ehr"
 msg := "Direct EHR access is prohibited. Use Integration Facade."
}
```

2️⃣ Enforce encryption for PHI

```
package security.phi


deny[msg] {
 input.data.classification == "PHI"
 not input.encryption.at_rest
 msg := "PHI data must be encrypted at rest"
}
```

3️⃣ Require IAM integration

```
package security.iam
```

```
deny[msg] {
  input.service.exposed == true
  not input.auth.oidc
  msg := "All exposed services must use OIDC authentication"
}
```

4️⃣ Block shared databases

```
package architecture.db

deny[msg] {
  count(input.databases.shared) > 0
  msg := "Shared databases are not allowed"
}
```

5️⃣ Require ADR for high-risk changes

```
package governance.adr

deny[msg] {
  input.change.risk == "high"
  not input.adr.exists
  msg := "High-risk changes require an approved ADR"
}
```

6️⃣ Prevent ungoverned GenAI usage

```
package ai.governance
```

```
deny[msg] {
  input.ai.enabled == true
  not input.ai.rag_enabled
  msg := "GenAI must use approved RAG sources"
}
```

7️⃣ Require audit logging for PHI

```
package security.audit


deny[msg] {
  input.data.classification == "PHI"
  not input.logging.audit_enabled
  msg := "Audit logging is mandatory for PHI access"
}
```

⚙️ Where These Policies Run

- Terraform (OPA / Sentinel)
- Kubernetes admission controller
- CI/CD pipelines
- API gateway
- Data pipelines
- ML model registration
- GenAI gateways

🧠 How This Scales Governance

- No meetings for basic compliance
- Architects define rules once
- Pipelines enforce forever
- Teams move faster
- Audits become trivial

🎯 **Interview-ready one-liner**

“I scale architecture by converting standards into Jira workflows, Confluence templates, and OPA policies, so governance happens automatically and teams move fast without breaking compliance.”
