IaC Validation Tools and Workflows Overview

# Scripts

# validate\_IaC\_controls.py

This script conducts an in-depth compliance assessment of Infrastructure as Code (IaC) repositories, specifically focusing on alignment with defined INFRA controls.

## Core Capabilities

1. **Comprehensive Compliance Analysis:** Systematically evaluates Terraform configurations against established INFRA control policies.
2. **Automated Policy-to-Code Mapping:** Dynamically identifies which INFRA controls are being addressed within individual Terraform modules.
3. **Deviation Detection & Documentation:** Pinpoints non-compliant areas and produces detailed deviation logs for traceability.
4. **Structured Reporting:** Generates human-readable, markdown-formatted compliance reports suitable for audit or governance workflows.
5. **AI-Powered Insight**: Leverages Azure OpenAI to intelligently interpret code-to-policy relationships and detect nuanced compliance gaps.

# validate\_iac\_deployment\_architecture.py

Designed to ensure Terraform implementations are architecturally aligned, this script validates actual infrastructure code against architectural blueprints represented in PlantUML diagrams.

## Core Capabilities

1. **Architecture-Conformance Validation:** Verifies that deployed infrastructure faithfully implements the defined architecture.
2. **Design Compliance Reporting:** Produces detailed reports outlining alignment and discrepancies between design and implementation.
3. **Automated Remediation Guidance:** Offers precise, actionable remediation recommendations for non-compliant infrastructure modules.

# validate\_policy\_architecture\_compliance1.py

This script enforces architectural compliance by cross-referencing PlantUML-based designs with OPA (Open Policy Agent) policies.

## Core Capabilities

1. **Intelligent Policy Compliance Analysis:** Utilizes Azure OpenAI and structured GPT-4 prompts to validate design diagrams against policy definitions.
2. **Flexible Policy Metadata Handling**: Supports varied policy folder structures and extracts metadata from markdown indexes.
3. **Robust Compliance Reporting:** Produces structured markdown reports, highlighting compliance status, deviation rationale, and recommended remediations. Includes a dedicated error report for handling exceptions.
4. **CI/CD Integration Ready:** Designed for automation with clear exit codes, timestamped report generation, and efficient temp file cleanup.

# validate-architecture-policies.yaml

A GitHub Actions workflow that enforces architectural and policy compliance prior to pull request (PR) approval.

## Core Capabilities

1. Selective Validation Logic: Targets `.puml` (PlantUML) architecture file changes; Enforces ADR documentation; Validates control definitions before review.
2. Strict Branch Governance: Executes only on `develop` branch; Limited to draft PRs; Prevents execution on unapproved branches or PR states.
3. Insightful Output: Provides well-formatted CLI outputs; Clearly articulates validation failures; Optionally uploads compliance artifacts.
4. Secure Execution: Authenticates using a minimal-scope PAT (`PAT\_REPO\_WRITE`); Operates on GitHub-hosted runners.

# Workflows

# validate-IaC-deploymentarch-controls.yaml

This workflow is tailored for IaC compliance validation and intelligent pull request handling.

## Core Capabilities

1. **Efficient Triggering:** Executes only on infrastructure-related file changes.
2. **Compliance Artifacts:** Uploads validation reports for stakeholder review.
3. **Dynamic PR Management:** Automatically promotes compliant draft PRs to ready-for-review status.
4. **Security-Conscious Execution:** Authenticated via secure token.

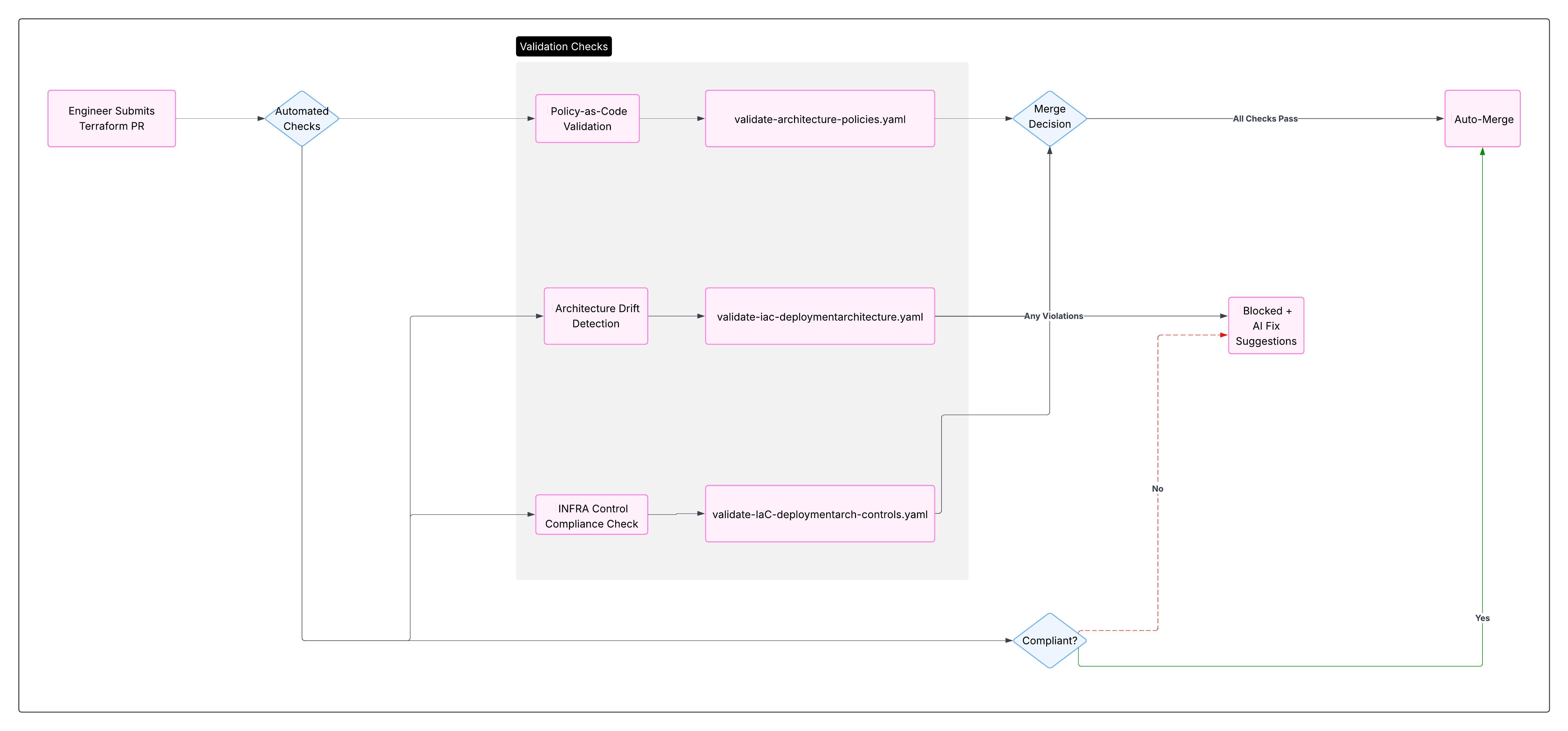
# validate-iac-deploymentarchitecture.yml

This workflow validates Terraform configurations against deployment architecture standards to enforce shift-left security—capturing architectural misalignments early in the development lifecycle.

## Core Capabilities

1. **Proactive Compliance Triggers:** Executes only on relevant infrastructure changes; Ensures strict adherence to deployment architecture principles.
2. **Python-Based Validation Engine:** Runs `validate\_iac\_deployment\_architecture.py`; Requires Python 3.10 and specific dependencies (`requirements.txt`).
3. **Artifact Retention for Audit:** Always stores compliance reports—even on validation failure.
4. **On-Demand Execution Support:** Supports manual trigger via `workflow\_dispatch` for ad-hoc compliance checks.

**Sample diagram representing the workflow:**

****