AiOpsOne

The DevSecOps Pipeline Cheatsheet

Every gate, the tool that owns it, and what should **fail the build** versus what should merely **report** — distilled from a real six-gate pipeline run against a deliberately vulnerable app. Every number verified against a real run.

THE SIX GATES

Gate	Tool that owns it	What it catches	Fail vs report	Trigger
Secrets	gitleaks (+ GitGuardian/ggshield)	Committed credentials, high-entropy strings	BLOCK on new	PR (diff) + **scheduled full-history**
SAST	Semgrep + **your own sink rules** (CodeQL / Snyk Code)	Injection, XSS, hardcoded secrets in *your* code	BLOCK on ERROR	Every PR
SCA	Trivy `fs` (+ Dependabot to remediate)	Vulnerable dependencies	BLOCK HIGH/CRIT, `--ignore-unfixed`	PR + Dependabot PRs
IaC	Checkov (gate) + `trivy config`	Public buckets, wildcard IAM, open security groups	BLOCK	On `**.tf` change
Container	hadolint + Dockle + Trivy image	Dockerfile smells, CIS, image CVEs, secrets in a layer	BLOCK (Dockle fatal, Trivy HIGH/CRIT)	On `docker/**` change
DAST	ZAP baseline (PR); ZAP full + Nuclei (scheduled)	Vulnerabilities in the running app	baseline REPORTS; active on schedule	PR (baseline) + scheduled (active)

THE ONE RULE FOR WHAT BLOCKS

> **Block on findings that are specific, actionable, and low-false-positive. Report everything else.** Gate on fixable findings only (`--ignore-unfixed` / `--only-fixed`) — a gate that fires on things people cannot fix gets switched off, and **a bypassed gate is worth less than no gate** because it looks like coverage.

FAST — EVERY PULL REQUEST

- gitleaks (scans the **diff**)
- Semgrep (auto rules + your sink rules)
- Trivy `fs` — dependencies
- Checkov — Terraform
- hadolint + Dockle + Trivy image
- ZAP **baseline** (passive, `-I`)

SLOW — ON A SCHEDULE

- **Full-history** secrets scan (the diff gate misses history)
- ZAP **full/active** scan (it is literally an attack)
- Nuclei with **your own** templates
- Image re-scan — new CVEs land against old builds

ACTIVE SCANS NEVER GATE A PR SOMEONE IS WAITING ON.

GREEN IS NOT SAFE — READ WHAT A GATE ACTUALLY SCANNED

Green check	Why it is green — and why that is not "secure"
ZAP baseline — 0 failures	Passive: it never sends an attack. Zero failures on an app with a working SQL injection. A smoke alarm that did not go off, on a fire it cannot see.

Green check	Why it is green — and why that is not "secure"
terraform validate — Success	Syntax only. Passes on a public S3 bucket, an `Action:"*"` IAM policy, and a security group open to the world. Valid is not secure.
gitleaks on push — no leaks	Scans the push **diff** (often 1 commit), not history. A secret already in `main` is out of scope — caught only by the scheduled full-history scan.
SBOM job — success	It *produces* the bill of materials; it does not gate. Green means "the record exists", not "the image is safe".

TOOL STATUS — 2026 (RE-CHECK QUARTERLY)

Alive & maintained: Trivy, Checkov, Grype, Syft, Semgrep, gitleaks, hadolint, Dockle, ZAP, Nuclei, CodeQL, Dependabot.

Frozen — do not rely on: **tfsec** (folded into Trivy; last release March 2026). Migrate to Checkov — Trivy's config scanner has **no IAM-wildcard rule**, so following the official tfsec→Trivy path silently drops a full-admin finding.

Archived — remove: **Terrascan** (Nov 2025).

> **Check it yourself, don't trust a banner:**
> `curl -s https://api.github.com/repos/<org>/<repo>`
> Read `archived` and `pushed_at`. A frozen project is more dangerous than an archived one — it still looks alive.

NEVER PUBLISH A RAW SCAN COUNT

> On the **identical image**, same afternoon: **Trivy 7,696** · **Grype 3,995** · **Docker Scout 836**. A CVE count is a fact about your *scanner and the date*, not your image. Gate on one tool, audit with another, and never put a raw number in an SLA or a board slide without naming the tool and the date beside it.

COPY-PASTE — THE WORKFLOWS ARE PUBLIC

Every workflow behind this cheatsheet runs for real on the intentionally-vulnerable lab. Clone it, read the YAML, reuse it:

- `github.com/jaybilgaye/aiopsone-range` → `.github/workflows/`
- `devsecops-pipeline.yml` — the monolith (one app, one team)
- `iac-security.yml` and `container-security.yml` — focused & reusable via `workflow_call` (many repos, many owners)

ONE REPO, ONE TEAM → MONOLITH. MANY REPOS OR OWNERS → REUSABLE WORKFLOWS. MOST ORGS RUN BOTH.