CTO Factory Pipelines Overview

Version 1.0 – 31 oct. 2025

Standardized CI/CD & Security System — CTO Factory Automation Suite

Purpose

This document provides a full overview of the CTO Factory Pipelines, designed to offer a universal, zero-trust CI/CD and security framework that can be applied across any software project — backend, frontend, infrastructure, or documentation.

Each workflow enforces a reproducible, observable, and compliant delivery cycle with built-in DevSecOps best practices.

Architecture Summary

Layer	Workflow	Scope	Description
Build Pipeline	ci.yml	CI/CD	Validates, tests, builds, scans, deploys, and monitors projects automatically.
Security Pipeline	security.yml	DevSecOps	Performs deep code, dependency, and infrastructure security analysis.
Documentation	README.md	Knowledge	Explains how to integrate and reuse both pipelines across projects.

Workflow 1 — Bulletproof CI/CD

Overview

A multi-language CI/CD system with full automation and observability.

Main Phases

Phase	Purpose	Key Tools
1. Validation & Linting	Code structure, lint, secrets detection	super-linter, trivy, tflint
2. Testing	Unit, integration, E2E, coverage	go test, npm test, terraform plan
3. Build & Security Scan	Build docker image, SBOM generation, vulnerability scan	Docker, Trivy, Anchore
4. Deployment	Auto-deploy via SSH, Terraform, or MkDocs	appleboy/ssh-action, Terraform, MkDocs
5. Monitoring & Notification	Health checks, metrics, Slack alerts	bash, Slack, GitHub Summary

Features

- Auto-detects project type (Go, Node.js, Terraform)
- Multi-version matrix testing (Go 1.21/1.22, Node 18/20)
- Code coverage and SBOM artifacts automatically uploaded
- Zero-trust deployment via GitHub Secrets
- Slack + GitHub summary notifications on every deployment
- Concurrency control (no parallel builds on same ref)
- Docker cache reuse and cleanup for performance stability

Output Artifacts

File	Description
coverage.html	Visual coverage reports
bom.spdx.json	Software Bill of Materials
trivy-results.sarif	Vulnerability report (SARIF format)
metrics.md	Deployment performance metrics
$GITHUB_STEP_SUMMARY	Human-readable run summary

Workflow 2 — Security Suite

Overview

A standalone DevSecOps security framework to ensure code, container, and infrastructure integrity.

Main Phases

Phase	Purpose	Key Tools
1. Secrets & SAST	Secret leaks & static code analysis	Gitleaks, Super-Linter, govulncheck
2. Dependencies & Containers	Dependency and image scanning	Snyk, Trivy, Anchore
3. Infrastructure as Code (IaC)	Terraform compliance	Tfsec, TFLint
4. Reporting & Incident Automation	Summary, issue creation, Slack alert	GitHub Step Summary, peter-evans/create-issue, Slack

Features

- Full SAST + DAST + IaC scanning every week and on each PR
- SBOM generation for dependency traceability
- Auto-blocks builds with high/critical vulnerabilities
- Auto-creates GitHub Issue + Slack alert on failure
- Uploads audit artifacts (sbom, tfsec-results, trivy-results, security-summary)

Output Artifacts

File	Description
sbom.spdx.json	Inventory of all dependencies
trivy-results.sarif	Container vulnerability scan results
tfsec-results.sarif	Terraform IaC policy report
$GITHUB_STEP_SUMMARY	Human-readable audit summary
security-summary-*.zip	Archived report for compliance reviews

🧩 Integration Pattern

Each template repository (infra-template, backend-template, frontend-template, docs-template, monitoring-template) must include the following folder structure for GitHub workflows:.github

┣ 📂 workflows

┃ ┣ 📜 ci.yml

┃ ┗ 📜 security.yml

┗ 📜 README.md

The CI/CD pipelines automatically adapt to the repository type based on the presence of specific files:

- `go.mod` → Backend build & test
- `package.json` → Frontend or SDK build & test
- `terraform/main.tf` → Infrastructure validation & deployment
- `mkdocs.yml` → Documentation publishing

🔐 Secrets & Environment Variables

All secrets listed below are mandatory for full CI/CD functionality. They are stored securely in GitHub → Settings → Secrets and variables → Actions.

Secret Name	Purpose	Required In
CODECOV_TOKEN	Upload coverage to Codecov	CI/CD
OVH_TERRAFORM_BACKEND	Terraform remote backend config	CI/CD
OVH_PROJECT_ID	OVH Cloud project ID	CI/CD
SSH_HOST / SSH_USER / SSH_PRIVATE_KEY	Remote deployment	CI/CD
SLACK_WEBHOOK	Slack notifications	CI/CD
SNYK_TOKEN	Snyk vulnerability scanning	Security
SECURITY_SLACK_WEBHOOK	Security alerts	Security

🧾 Governance & Compliance Alignment

Our CI/CD automation is designed to ensure robust governance and compliance across various domains.

Domain	Coverage	Description
Security	✅	Automated SAST/DAST/IaC validation + SBOM
Quality	✅	Linting, testing, and coverage enforced
Observability	✅	Metrics, summaries, and Slack alerts
Performance	✅	Docker caching and concurrency control
Compliance	✅	SOC2, ISO 27001, and DevSecOps ready

🧭 Future Enhancements

We are continuously working to improve our CI/CD automation with the following planned features:

Version	Planned Feature
v1.1	Reusable workflow centralization (org/github-actions)
v1.2	Auto-tagging SBOM dependencies with CVE IDs
v1.3	Cost-aware CI/CD reporting (FinOps integration)

🪪 Maintainer Info

- **Maintained by:** CTO Factory Automation
- **Last Update:** 31 oct. 2025
- **Version:** v1.0
- **License:** MIT