# CI/CD Migration Review: Jenkins to GitHub Actions

## 1. Migration Overview

**Key Steps Taken:** - Replaced Jenkins pipelines with GitHub Actions workflows. - Migrated credentials from Jenkins Credential Store to GitHub Secrets. - Switched metadata retrieval from Ant and sf\_extractor.jar to Salesforce CLI (sf). - Retained Fortify scanning (sourceanalyzer, BIRTReportGenerator, FPRUtility) but executed in GitHub-hosted runners. - Migrated Dependency Check pipeline from Ant-based tasks to GitHub Actions YAML.

**Security Changes:** - Authentication moved to **SFDX Auth URLs** managed in GitHub Secrets. - Jenkins proxy and token handling eliminated. - Branch protection and RBAC in GitHub replaced node-level access control.

**Jenkins Decommissioning:** - Jenkins agents and cloud-based executors removed. - Artifact management shifted from Jenkins archive to GitHub artifact store. - Ant utilities no longer required.

**Lingering Dependencies:** - No Jenkins dependencies remain; all pipelines converted.

**Typical Workflow (Fortify Integration):** - Checkout → Install Node + Salesforce CLI → Retrieve Metadata → Fortify translate + scan → Generate CSV/PDF/XLSX → Upload artifacts.

## 2. Access Control and Authentication

* **Permissions:** GitHub branch protection rules + workflow permissions (contents: read).
* **Workflow Access:** Restricted to collaborators with write/admin roles.
* **MFA:** Enforced at organization level.
* **Service Accounts:** Fortify and Salesforce access via GitHub Secrets; rotation recommended every 90 days.

## 3. Secrets Management

* **Storage:** GitHub Secrets for all sensitive values.
* **Prevention:** Secrets not echoed/logged; excluded from artifacts.
* **Scanning:** GitHub secret scanning enabled; alerts handled via immediate revocation and rotation.

## 4. Workflow and Pipeline Security

* **Runners:** GitHub-hosted (arc-small-container).
* **Hardening:** GitHub maintains OS patching; for future self-hosted, apply least privilege and segmentation.
* **Supply Chain Risks:** Major-version pinned actions (checkout@v4, setup-node@v4); recommendation to pin to SHA.
* **Triggers:** Manual workflow\_dispatch reduces attack surface.
* **Dependency Scanning:** Dependency Check pipeline retained; Dependabot for Node.js recommended.

## 5. Fortify Integration Specifics

* **Configuration:** Runs in container with Fortify client; OWASP Top 10 template reports generated.
* **Failure Handling:** Current setup doesn’t fail builds on findings; can be tightened with exit on critical/high.
* **Results Security:** Artifacts stored with 30-day retention, restricted access.
* **Testing:** Parity validated; same CLI commands ensure consistent detection.

## 6. Monitoring, Logging, and Incident Response

* **Logging:** GitHub run logs, audit logs; Fortify logs (translate.log, scan.log).
* **Monitoring:** 30-day artifact retention in GitHub Actions.
* **Alerts:** GitHub secret scanning + Dependabot; SIEM integration recommended.
* **Incident Response:** Secret revocation + rotation; reimage runners if compromised.
* **Integrations:** GitHub Security Dashboard, optional Splunk integration.

## 7. Best Practices and Risks

* **Implemented:** GitHub Secrets, least-privilege permissions, manual workflow dispatch.
* **Recommended:** OIDC for permissions, SHA pinning, workflow scanning (Checkov/KICS).
* **Risks:** Potential secret leakage if misused, artifact sensitivity, lack of enforced build failure on high issues.
* **Compliance:** GitHub Actions compliant with SOC 2, ISO 27001; internal audits recommended.
* **Training:** Updated runbooks for GitHub Actions; Salesforce CLI usage standardized.
* **Ongoing Reviews:** Quarterly reviews of pipeline security and Fortify parity.

## 8. Positives of Moving to GitHub Actions

* **Tighter Integration:** CI/CD directly tied to repository; eliminates external Jenkins maintenance.
* **Security Improvements:** Centralized GitHub RBAC, branch protection, MFA, secret scanning.
* **Reduced Infrastructure Overhead:** No separate Jenkins servers, agents, or Kubernetes management.
* **Developer Experience:** Simpler YAML-based workflows, faster onboarding, better artifact visibility.
* **Automation Ecosystem:** Access to GitHub Marketplace actions (Dependabot, CodeQL, etc.).

## 9. Missing Features in GitHub Actions

* **UI Richness:** Jenkins has richer plugin ecosystem (e.g., custom dashboards).
* **Job Chaining:** Jenkins pipeline syntax allows more complex orchestration.
* **Fine-grained Logs:** Jenkins console output can be more flexible for debugging.
* **Artifact Retention:** Limited to 90 days in GitHub (vs Jenkins unlimited with disk mgmt).
* **Runner Customization:** GitHub-hosted runners less customizable than Jenkins agents.

## Conclusion

The migration from Jenkins to GitHub Actions has **reduced infrastructure complexity**, **improved security posture**, and **streamlined DevOps workflows**. While some Jenkins-specific features (plugins, deep orchestration) are missing, the benefits of integration, governance, and reduced overhead outweigh the gaps. Ongoing improvements like **enforcing build failures on high vulnerabilities, OIDC adoption, and SHA pinning** will further mature the setup.