

# Continuous Integration and Continuous Delivery Pipeline with 16 Gates

## A Comprehensive Implementation Guide

Jordan Suber

December 3, 2025

---

### Contents

<b>1</b>	<b>Executive Summary</b>	<b>4</b>
1.1	The 16 Gates Overview . . . . .	4
1.2	Key Benefits . . . . .	5
1.3	Target Audience . . . . .	6
<b>2</b>	<b>Source Code Management</b>	<b>7</b>
2.1	Gate 1: Source Code Version Control . . . . .	7
2.1.1	Overview . . . . .	7
2.1.2	Version Control System Selection . . . . .	7
2.1.3	Repository Configuration Requirements . . . . .	8
2.1.4	Git Hooks Implementation . . . . .	11
2.2	Gate 2: Optimum Branching Strategy . . . . .	13
2.2.1	Branching Strategy Comparison . . . . .	13
2.2.2	Trunk-Based Development (Recommended) . . . . .	13
2.2.3	GitFlow for Scheduled Releases . . . . .	15
2.3	Metrics and Compliance . . . . .	16
2.3.1	Key Metrics for SCM Gates . . . . .	16
<b>3</b>	<b>Build and Quality Gates</b>	<b>17</b>
3.1	Gate 3: Static Analysis . . . . .	17
3.1.1	Overview . . . . .	17
3.1.2	Static Analysis Tool Categories . . . . .	17
3.1.3	SonarQube Configuration . . . . .	18
3.1.4	Quality Gate Definition . . . . .	19
3.1.5	Multi-Language Linting Configuration . . . . .	21
3.2	Gate 4: 80% Code Coverage . . . . .	24
3.2.1	Overview . . . . .	24
3.2.2	Coverage Types . . . . .	24

3.2.3	Coverage Configuration Examples . . . . .	25
3.2.4	Coverage Enforcement in CI/CD . . . . .	29
<b>4</b>	<b>Security Scanning</b>	<b>31</b>
4.1	Gate 5: Vulnerability Scan . . . . .	31
4.1.1	Security Scanning Categories . . . . .	31
4.1.2	SAST Implementation with Semgrep . . . . .	32
4.1.3	Container Image Scanning with Trivy . . . . .	34
4.1.4	Infrastructure as Code Scanning . . . . .	35
4.2	Gate 6: Open Source Scan . . . . .	37
4.2.1	Software Composition Analysis . . . . .	37
4.2.2	OWASP Dependency-Check Configuration . . . . .	38
4.2.3	License Compliance Policy . . . . .	39
<b>5</b>	<b>Artifact Management</b>	<b>41</b>
5.1	Gate 7: Artifact Version Control . . . . .	41
5.1.1	Artifact Repository Architecture . . . . .	41
5.1.2	Semantic Versioning Strategy . . . . .	42
5.1.3	Container Image Tagging Strategy . . . . .	43
5.1.4	Artifact Integrity Verification . . . . .	45
<b>6</b>	<b>Infrastructure Provisioning</b>	<b>47</b>
6.1	Gate 8: Automatic Provision . . . . .	47
6.1.1	Infrastructure as Code with Terraform . . . . .	47
6.1.2	Terraform CI/CD Pipeline . . . . .	50
6.2	Gate 9: Immutable Servers . . . . .	53
6.2.1	Container Image Definition . . . . .	53
6.2.2	AMI Building with Packer . . . . .	55
<b>7</b>	<b>Integration and Performance Testing</b>	<b>58</b>
7.1	Gate 10: Integration Testing . . . . .	58
7.1.1	End-to-End Testing with Playwright . . . . .	58
7.1.2	API Integration Testing . . . . .	62
7.2	Gate 11: Performance Testing . . . . .	63
7.2.1	Load Testing with k6 . . . . .	63
7.2.2	Performance Gate Enforcement . . . . .	66
<b>8</b>	<b>Full Automation on Commit</b>	<b>69</b>
8.1	Gate 12: Full Automation on Commit . . . . .	69
8.1.1	Complete Pipeline Architecture . . . . .	69
8.1.2	Jenkins Declarative Pipeline . . . . .	71
<b>9</b>	<b>Production Release Strategy</b>	<b>77</b>
9.1	Gate 13: Zero Downtime Release . . . . .	77
9.1.1	Deployment Strategies Comparison . . . . .	77
9.1.2	Kubernetes Canary Deployment . . . . .	78

9.2	Gate 14: Feature Toggle . . . . .	81
9.2.1	Feature Flag Implementation . . . . .	81
<b>10</b>	<b>Change Management and Rollbacks</b>	<b>85</b>
10.1	Gate 15: Automated Change Order . . . . .	85
10.1.1	ServiceNow Integration . . . . .	85
10.2	Gate 16: Automated Rollback . . . . .	87
10.2.1	Rollback Automation with ArgoCD . . . . .	87
<b>11</b>	<b>Monitoring and Observability</b>	<b>92</b>
11.1	Pipeline Observability . . . . .	92
11.1.1	Pipeline Metrics Dashboard . . . . .	92
11.1.2	Prometheus Metrics Configuration . . . . .	93
11.2	Application Observability . . . . .	95
11.2.1	OpenTelemetry Integration . . . . .	95
<b>12</b>	<b>Compliance and Audit</b>	<b>97</b>
12.1	Audit Trail Requirements . . . . .	97
12.2	Compliance Framework Mapping . . . . .	97
<b>A</b>	<b>Tool Reference</b>	<b>98</b>
<b>B</b>	<b>Implementation Checklist</b>	<b>99</b>
B.1	Gate Implementation Checklist . . . . .	99
	<b>Conclusion</b>	<b>103</b>

## 1 Executive Summary

This document provides a comprehensive guide to implementing a robust Continuous Integration and Continuous Delivery (CI/CD) pipeline incorporating sixteen quality gates. These gates serve as automated checkpoints that ensure code quality, security, and operational readiness before software reaches production environments.

### 1.1 The 16 Gates Overview

The sixteen gates are organized into nine functional areas, each addressing critical aspects of the software delivery lifecycle:

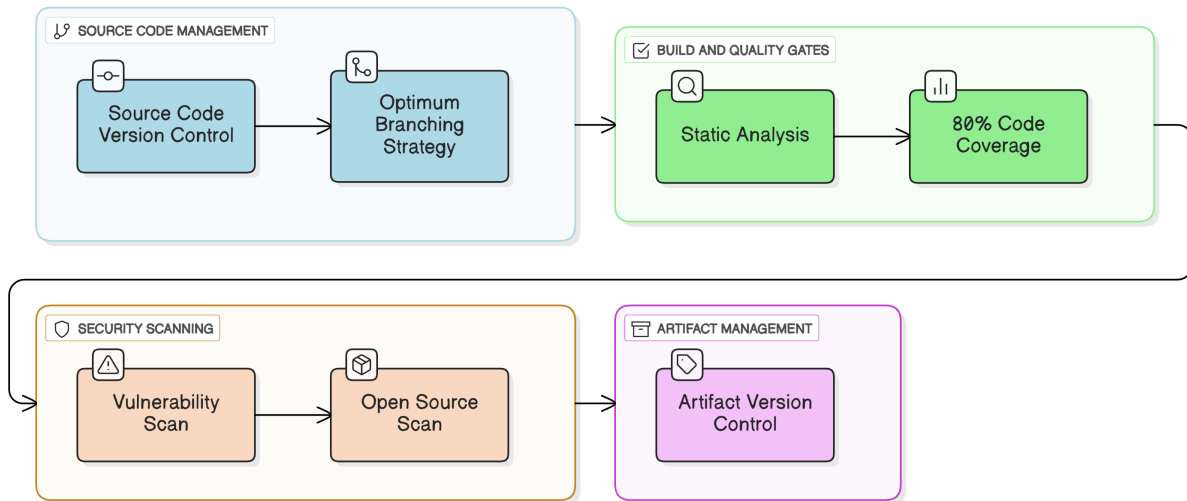


Figure 1: Software Development Pipeline Flow Chart showing the major stages of the software development lifecycle and their interactions.

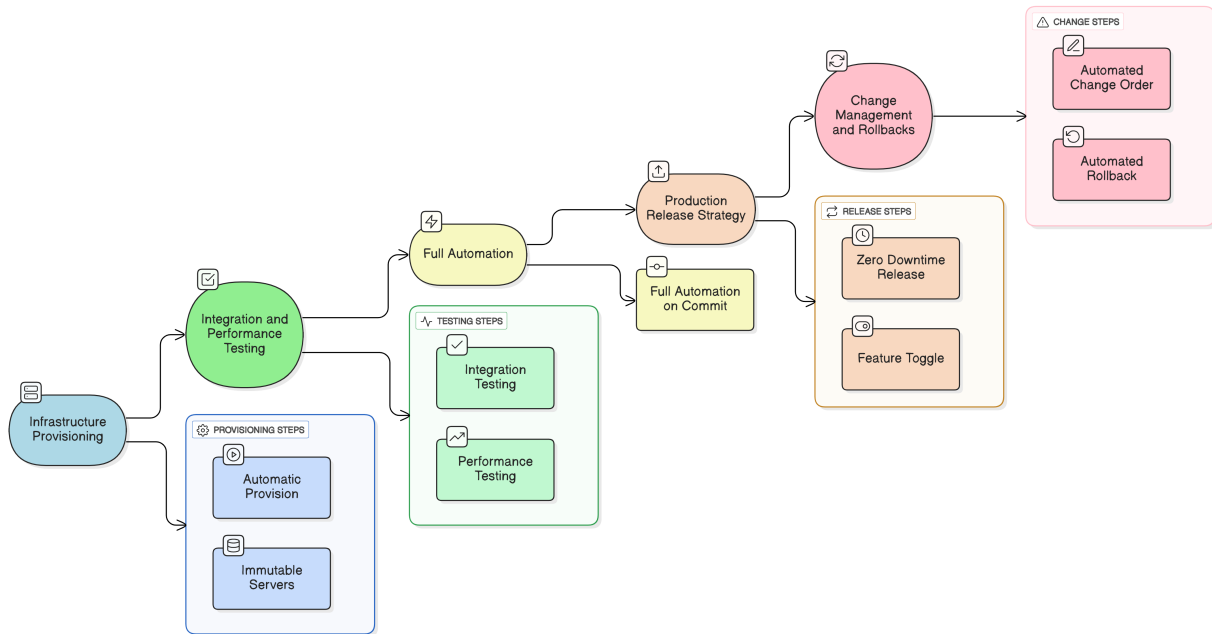


Figure 2: Infrastructure Provisioning and Release Flow highlighting automated infrastructure setup, testing, and release steps.

## 1.2 Key Benefits

Implementing these sixteen gates provides organizations with:

- **Reduced Risk:** Automated quality and security checks catch issues before they reach production
- **Faster Delivery:** Automation eliminates manual bottlenecks and reduces lead time
- **Improved Quality:** Consistent enforcement of coding standards and test coverage
- **Enhanced Security:** Continuous scanning for vulnerabilities and license compliance
- **Operational Resilience:** Zero-downtime deployments with automated rollback capabilities
- **Audit Compliance:** Automated change orders provide complete audit trails

### 1.3 Target Audience

This guide is intended for:

- DevOps Engineers designing and implementing CI/CD pipelines
- Software Architects establishing organizational delivery standards
- Engineering Managers overseeing software delivery practices
- Security Engineers integrating security into the development lifecycle
- Site Reliability Engineers ensuring production stability

## 2 Source Code Management

**Gates:**

Source Code Version Control

Optimum Branching Strategy

Source Code Management (SCM) forms the foundation of any CI/CD pipeline. These two gates ensure that all code changes are properly versioned, reviewed, and integrated using industry-standard practices.

### 2.1 Gate 1: Source Code Version Control

#### 2.1.1 Overview

Source Code Version Control establishes the fundamental infrastructure for tracking all changes to the codebase. This gate ensures that every modification is recorded, attributable, and reversible.

#### 2.1.2 Version Control System Selection

Table 1: Version Control Platform Comparison

Feature	GitHub	GitLab	Bitbucket
Hosting Options	Cloud, Enterprise	Cloud, Self-Hosted	Cloud, Data Center
CI/CD Integration	GitHub Actions	GitLab CI/CD	Bitbucket Pipelines
Code Review	Pull Requests	Merge Requests	Pull Requests
Security Scanning	Dependabot, CodeQL	SAST, DAST, SCA	Snyk Integration
Package Registry	GitHub Packages	GitLab Registry	Limited
Wiki/Docs	GitHub Wiki	GitLab Wiki	Confluence
Issue Tracking	GitHub Issues	GitLab Issues	Jira Integration
Pricing Model	Per-seat	Per-seat	Per-seat

### 2.1.3 Repository Configuration Requirements

Every repository must implement the following configurations:

**Branch Protection Rules** Branch protection prevents direct pushes to critical branches and enforces review requirements.

```
1 branches:
2   - name: main
3     protection:
4       required_pull_request_reviews:
5         required_approving_review_count: 2
6         dismiss_stale_reviews: true
7         require_code_owner_reviews: true
8         require_last_push_approval: true
9       required_status_checks:
10        strict: true
11        contexts:
12          - "ci/build"
13          - "ci/test"
14          - "ci/security-scan"
15          - "ci/lint"
16        enforce_admins: true
17        required_linear_history: true
18        allow_force_pushes: false
19        allow_deletions: false
20        required_conversation_resolution: true
21        required_signatures: true
```

Listing 1: GitHub Branch Protection Configuration (.github/settings.yml)

**CODEOWNERS Configuration** The CODEOWNERS file automatically assigns reviewers based on file paths.

```
1 # Default owners for everything in the repo
2 * @org/engineering-leads
3
4 # Frontend code
5 /src/frontend/ @org/frontend-team
6 /src/components/ @org/frontend-team
7
8 # Backend services
9 /src/api/ @org/backend-team
10 /src/services/ @org/backend-team
11
12 # Infrastructure and DevOps
13 /terraform/ @org/devops-team
14 /kubernetes/ @org/devops-team
15 /.github/ @org/devops-team
16 /Dockerfile* @org/devops-team
17
18 # Security-sensitive files
19 /src/auth/ @org/security-team
20 /src/crypto/ @org/security-team
21 *.pem @org/security-team
22 *.key @org/security-team
23
24 # Database migrations require DBA review
25 /migrations/ @org/dba-team @org/backend-team
26
27 # Documentation
28 /docs/ @org/tech-writers
29 *.md @org/tech-writers
```

Listing 2: CODEOWNERS File Example

**Commit Message Standards** Enforce conventional commit messages for automated changelog generation and semantic versioning.

```
1 module.exports = {
2   extends: ['@commitlint/config-conventional'],
3   rules: {
4     'type-enum': [
5       2,
6       'always',
7       [
8         'feat',      // New feature
9         'fix',       // Bug fix
10        'docs',       // Documentation changes
11        'style',      // Code style changes (formatting)
12        'refactor',   // Code refactoring
13        'perf',       // Performance improvements
14        'test',       // Adding or updating tests
15        'build',      // Build system changes
16        'ci',         // CI configuration changes
17        'chore',      // Maintenance tasks
18        'revert'      // Reverting previous commits
19      ]
20    ],
21    'scope-enum': [
22      2,
23      'always',
24      ['api', 'ui', 'db', 'auth', 'config', 'deps', 'infra']
25    ],
26    'subject-case': [2, 'always', 'lower-case'],
27    'subject-max-length': [2, 'always', 72],
28    'body-max-line-length': [2, 'always', 100],
29    'footer-max-line-length': [2, 'always', 100]
30  }
31 };
```

Listing 3: commitlint.config.js for Conventional Commits

### 2.1.4 Git Hooks Implementation

Pre-commit hooks enforce standards before code enters the repository.

```
1 repos:
2   - repo: https://github.com/pre-commit/pre-commit-hooks
3     rev: v4.5.0
4     hooks:
5       - id: trailing-whitespace
6       - id: end-of-file-fixer
7       - id: check-yaml
8       - id: check-json
9       - id: check-added-large-files
10        args: ['--maxkb=1000']
11       - id: check-merge-conflict
12       - id: detect-private-key
13       - id: no-commit-to-branch
14        args: ['--branch', 'main', '--branch', 'develop']
15
16   - repo: https://github.com/commitizen-tools/commitizen
17     rev: v3.13.0
18     hooks:
19       - id: commitizen
20        stages: [commit-msg]
21
22   - repo: https://github.com/gitleaks/gitleaks
23     rev: v8.18.1
24     hooks:
25       - id: gitleaks
26
27   - repo: local
28     hooks:
29       - id: lint
30        name: Run Linter
31        entry: npm run lint
32        language: system
33        types: [javascript, typescript]
34        pass_filenames: false
35
36       - id: unit-tests
37        name: Run Unit Tests
38        entry: npm run test:unit
39        language: system
40        pass_filenames: false
41        stages: [push]
```

Listing 4: .pre-commit-config.yaml

**Best Practice**

Always enable signed commits in your repository settings. This ensures that every commit can be cryptographically verified as coming from an authorized contributor. Use `git config --global commit.gpgsign true` to enable automatic signing.

## 2.2 Gate 2: Optimum Branching Strategy

### 2.2.1 Branching Strategy Comparison

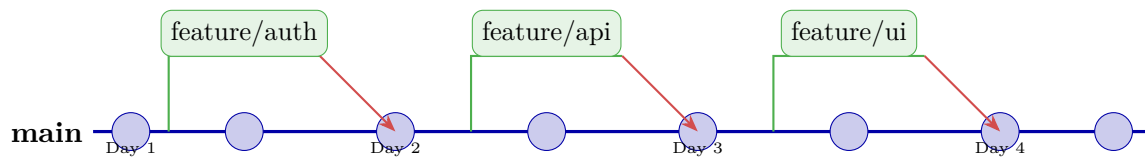
Organizations must select a branching strategy that aligns with their release cadence, team size, and deployment requirements.

Table 2: Branching Strategy Comparison

Strategy	Best For	Key Characteristics	Complexity
Trunk-Based	Continuous deployment, small teams	Short-lived branches, frequent integration	Low
GitFlow	Scheduled releases, large teams	Long-lived branches, explicit releases	High
GitHub Flow	Continuous deployment, web apps	Simple branching, PR-based workflow	Low
GitLab Flow	Environment-based deployment	Environment branches, upstream first	Medium
Release Flow	Large-scale projects, multiple releases	Topic branches, release branches	Medium

### 2.2.2 Trunk-Based Development (Recommended)

Trunk-Based Development is the recommended approach for teams practicing continuous deployment.



Feature branches live < 24 hours

## Key Principles

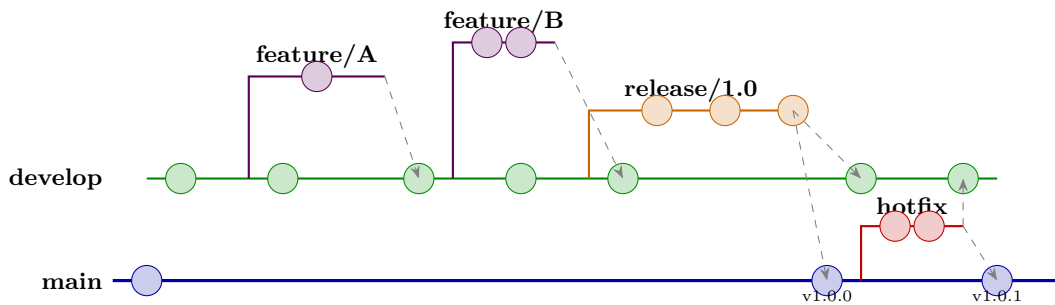
1. **Short-Lived Feature Branches:** Branches should be merged within 24 hours
2. **Feature Flags:** Use feature flags to hide incomplete features
3. **Continuous Integration:** Merge to trunk multiple times per day
4. **Release from Trunk:** Production deployments come directly from the main branch

```
1 name: Trunk-Based CI/CD
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10  validate:
11    runs-on: ubuntu-latest
12    steps:
13      - uses: actions/checkout@v4
14        with:
15          fetch-depth: 0
16
17      - name: Check Branch Age
18        if: github.event_name == 'pull_request'
19        run: |
20          BRANCH_AGE=$(git log -1 --format=%ct origin/main..HEAD)
21          CURRENT_TIME=$(date +%s)
22          AGE_HOURS=$(( (CURRENT_TIME - BRANCH_AGE) / 3600 ))
23          if [ $AGE_HOURS -gt 24 ]; then
24            echo "::warning::Branch is older than 24 hours. Consider
25              ↪ rebasing."
26          fi
27
28      - name: Verify Linear History
29        run: |
30          if ! git log --oneline --merges origin/main..HEAD | grep -q
31            ↪ .; then
32            echo "Linear history maintained"
33          else
34            echo "::error::Merge commits detected. Use rebase instead."
35            exit 1
36          fi
```

Listing 5: GitHub Actions Workflow for Trunk-Based Development

### 2.2.3 GitFlow for Scheduled Releases

GitFlow is appropriate for teams with scheduled release cycles and multiple supported versions.



```

1 # Branch naming patterns
2 branches:
3   main:
4     pattern: "^main$"
5     protected: true
6     deployment: production
7
8   develop:
9     pattern: "^develop$"
10    protected: true
11    deployment: staging
12
13   feature:
14     pattern: "^feature/[A-Z]+-[0-9]+-.*"
15     example: "feature/PROJ-123-add-user-authentication"
16     source: develop
17     target: develop
18     max_age_days: 14
19
20   release:
21     pattern: "^release/[0-9]+\.[0-9]+\.[0-9]+$"
22     example: "release/1.2.0"
23     source: develop
24     target: [main, develop]
25
26   hotfix:
27     pattern: "^hotfix/[A-Z]+-[0-9]+-.*"
28     example: "hotfix/PROJ-456-fix-critical-bug"
29     source: main
30     target: [main, develop]
31     priority: high

```

Listing 6: GitFlow Branch Naming Convention

**Warning**

GitFlow adds complexity and can slow down delivery if not managed properly. Reserve it for situations requiring multiple parallel release versions or strict release scheduling. For most web applications, simpler strategies like Trunk-Based Development are more appropriate.

## 2.3 Metrics and Compliance

### 2.3.1 Key Metrics for SCM Gates

Table 3: Source Code Management Metrics

Metric	Target	Warning	Measurement
Branch Age	< 24 hours	> 48 hours	Time since branch creation
PR Review Time	< 4 hours	> 24 hours	Time to first review
PR Merge Time	< 8 hours	> 48 hours	Time from open to merge
Review Approval Rate	> 95%	< 90%	PRs approved vs rejected
Commit Frequency	> 1/day/dev	< 1/week/dev	Commits per developer
Integration Frequency	> 1/day	< 1/week	Merges to main
Rollback Rate	< 5%	> 10%	Reverted deployments

## 3 Build and Quality Gates

**Gates:** Static Analysis 80% Code Coverage

Build and Quality Gates ensure that every code change meets established quality standards before it can progress through the pipeline. These gates enforce automated checks that would be impractical to perform manually.

### 3.1 Gate 3: Static Analysis

#### 3.1.1 Overview

Static analysis examines source code without executing it, identifying potential bugs, security vulnerabilities, code smells, and style violations. This gate prevents known categories of defects from entering the codebase.

#### 3.1.2 Static Analysis Tool Categories

Table 4: Static Analysis Tool Categories and Examples

Category	Tools	Purpose
Linters	ESLint, Pylint, RuboCop, golangci-lint	Style enforcement, basic error detection
Type Checkers	TypeScript, mypy, Flow	Type safety verification
Security Scanners	Semgrep, Bandit, Brakeman	Security vulnerability detection
Code Quality	SonarQube, CodeClimate	Comprehensive quality metrics
Complexity	Lizard, radon	Cyclomatic complexity analysis
Duplication	CPD, jscpd	Duplicate code detection

### 3.1.3 SonarQube Configuration

SonarQube provides comprehensive code quality analysis across multiple dimensions.

```
1 # Project identification
2 sonar.projectKey=mycompany_myproject
3 sonar.projectName=My Project
4 sonar.projectVersion=1.0
5
6 # Source configuration
7 sonar.sources=src
8 sonar.tests=tests
9 sonar.exclusions=**/node_modules/**,**/vendor/**,**/*.test.js
10
11 # Language-specific settings
12 sonar.javascript.lcov.reportPaths=coverage/lcov.info
13 sonar.python.coverage.reportPaths=coverage.xml
14 sonar.java.binaries=target/classes
15
16 # Quality gate settings
17 sonar.qualitygate.wait=true
18 sonar.qualitygate.timeout=300
19
20 # Issue severity thresholds
21 sonar.issue.ignore.multicriteria=e1,e2
22 sonar.issue.ignore.multicriteria.e1.ruleKey=javascript:S1135
23 sonar.issue.ignore.multicriteria.e1.resourceKey=**/*.spec.js
24 sonar.issue.ignore.multicriteria.e2.ruleKey=python:S1135
25 sonar.issue.ignore.multicriteria.e2.resourceKey=**/test_*.py
26
27 # Branch analysis
28 sonar.branch.name=${BRANCH_NAME}
29 sonar.branch.target=main
```

Listing 7: sonar-project.properties Configuration

### 3.1.4 Quality Gate Definition

```
1 quality_gate:
2   name: "Production Quality Gate"
3   conditions:
4     # Reliability
5     - metric: new_bugs
6       operator: GREATER_THAN
7       error: 0
8
9     - metric: new_reliability_rating
10      operator: GREATER_THAN
11      error: 1 # Must be A rating
12
13    # Security
14    - metric: new_vulnerabilities
15      operator: GREATER_THAN
16      error: 0
17
18    - metric: new_security_rating
19      operator: GREATER_THAN
20      error: 1 # Must be A rating
21
22    - metric: new_security_hotspots_reviewed
23      operator: LESS_THAN
24      error: 100 # All hotspots must be reviewed
25
26    # Maintainability
27    - metric: new_code_smells
28      operator: GREATER_THAN
29      error: 10 # Allow up to 10 new code smells
30
31    - metric: new_maintainability_rating
32      operator: GREATER_THAN
33      error: 1 # Must be A rating
34
35    - metric: new_technical_debt_ratio
36      operator: GREATER_THAN
37      error: 5 # Max 5% technical debt ratio
38
39    # Coverage
40    - metric: new_coverage
41      operator: LESS_THAN
42      error: 80 # Minimum 80% coverage on new code
43
44    - metric: new_line_coverage
45      operator: LESS_THAN
46      error: 80
47
48    # Duplication
```

```
49     - metric: new_duplicated_lines_density
50       operator: GREATER_THAN
51       error: 3  # Max 3% duplication
```

Listing 8: SonarQube Quality Gate Configuration

### 3.1.5 Multi-Language Linting Configuration

```
1 name: Static Analysis
2
3 on:
4   pull_request:
5     branches: [main, develop]
6
7 jobs:
8   lint-javascript:
9     runs-on: ubuntu-latest
10    steps:
11      - uses: actions/checkout@v4
12
13      - name: Setup Node.js
14        uses: actions/setup-node@v4
15        with:
16          node-version: '20'
17          cache: 'npm'
18
19      - run: npm ci
20
21      - name: Run ESLint
22        run: |
23          npx eslint . \
24            --format json \
25            --output-file eslint-report.json \
26            --max-warnings 0
27
28      - name: Upload ESLint Report
29        uses: actions/upload-artifact@v4
30        with:
31          name: eslint-report
32          path: eslint-report.json
33
34   lint-python:
35     runs-on: ubuntu-latest
36    steps:
37      - uses: actions/checkout@v4
38
39      - name: Setup Python
40        uses: actions/setup-python@v5
41        with:
42          python-version: '3.12'
43
44      - name: Install dependencies
45        run: |
46          pip install pylint mypy ruff bandit
47
48      - name: Run Ruff (fast linting)
```

```

49     run: ruff check . --output-format=json > ruff-report.json
50
51   - name: Run Pylint
52     run: |
53       pylint src/ \
54         --output-format=json \
55         --fail-under=8.0 \
56         > pylint-report.json || true
57
58   - name: Run mypy (type checking)
59     run: mypy src/ --strict --ignore-missing-imports
60
61   - name: Run Bandit (security)
62     run: bandit -r src/ -f json -o bandit-report.json
63
64 lint-go:
65   runs-on: ubuntu-latest
66   steps:
67     - uses: actions/checkout@v4
68
69     - name: Setup Go
70       uses: actions/setup-go@v5
71       with:
72         go-version: '1.22'
73
74     - name: Run golangci-lint
75       uses: golangci/golangci-lint-action@v4
76       with:
77         version: latest
78         args: --timeout=5m
79
80 sonarqube:
81   needs: [lint-javascript, lint-python, lint-go]
82   runs-on: ubuntu-latest
83   steps:
84     - uses: actions/checkout@v4
85       with:
86         fetch-depth: 0
87
88     - name: Download all reports
89       uses: actions/download-artifact@v4
90
91     - name: SonarQube Scan
92       uses: sonarsource/sonarqube-scan-action@master
93       env:
94         SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
95         SONAR_HOST_URL: ${ secrets.SONAR_HOST_URL }
96
97     - name: SonarQube Quality Gate
98       uses: sonarsource/sonarqube-quality-gate-action@master

```

```
99      timeout-minutes: 5
100      env:
101        SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```

Listing 9: GitHub Actions Multi-Language Linting Workflow

## 3.2 Gate 4: 80% Code Coverage

### 3.2.1 Overview

Code coverage measures the percentage of code executed during automated testing. The 80% threshold represents an industry-standard balance between thoroughness and practicality.

### 3.2.2 Coverage Types

Table 5: Types of Code Coverage

Type	Description	Target
Line Coverage	Percentage of lines executed	Primary metric, $\geq 80\%$
Branch Coverage	Percentage of branches taken	Critical for conditionals, $\geq 75\%$
Function Coverage	Percentage of functions called	Should approach 100%
Statement Coverage	Percentage of statements run	Similar to line coverage
Condition Coverage	Each boolean evaluated both ways	Important for complex logic
MC/DC Coverage	Modified condition/decision	Required for safety-critical code

### 3.2.3 Coverage Configuration Examples

```
1 module.exports = {
2   collectCoverage: true,
3   collectCoverageFrom: [
4     'src/**/*.{js,jsx,ts,tsx}',
5     '!src/**/*.d.ts',
6     '!src/**/*.stories.{js,jsx,ts,tsx}',
7     '!src/**/*.test.{js,jsx,ts,tsx}',
8     '!src/**/*.index.{js,ts}',
9     '!src/mocks/**'
10  ],
11  coverageDirectory: 'coverage',
12  coverageReporters: ['text', 'lcov', 'html', 'cobertura'],
13  coverageThreshold: {
14    global: {
15      branches: 75,
16      functions: 80,
17      lines: 80,
18      statements: 80
19    },
20    // Stricter thresholds for critical paths
21    './src/auth/**/*.ts': {
22      branches: 90,
23      functions: 95,
24      lines: 90,
25      statements: 90
26    },
27    './src/payment/**/*.ts': {
28      branches: 95,
29      functions: 100,
30      lines: 95,
31      statements: 95
32    }
33  },
34  testMatch: [
35    '**/__tests__/**/*.[jt]s?(x)',
36    '**/?(*.)+(spec|test).[jt]s?(x)'
37  ],
38  testPathIgnorePatterns: [
39    '/node_modules/',
40    '/dist/'
41  ]
42 };
```

Listing 10: Jest Coverage Configuration (jest.config.js)

```
1 [tool.pytest.ini_options]
2 testpaths = ["tests"]
3 python_files = ["test_*.py", "*_test.py"]
4 python_functions = ["test_*"]
5 addopts = [
6     "-v",
7     "--tb=short",
8     "--strict-markers",
9     "--cov=src",
10    "--cov-report=term-missing",
11    "--cov-report=html:coverage_html",
12    "--cov-report=xml:coverage.xml",
13    "--cov-fail-under=80",
14    "--cov-branch"
15 ]
16
17 [tool.coverage.run]
18 branch = true
19 source = ["src"]
20 omit = [
21     "*/tests/*",
22     "*/__pycache__/*",
23     "*/migrations/*",
24     "*/.venv/*"
25 ]
26
27 [tool.coverage.report]
28 exclude_lines = [
29     "pragma: no cover",
30     "def __repr__",
31     "raise NotImplementedError",
32     "if __name__ == '__main__':",
33     "if TYPE_CHECKING:",
34     "@abstractmethod"
35 ]
36 fail_under = 80
37 show_missing = true
38 precision = 2
39
40 [tool.coverage.html]
41 directory = "coverage_html"
```

Listing 11: pytest Coverage Configuration (pyproject.toml)

```
1 <plugin>
2   <groupId>org.jacoco</groupId>
3   <artifactId>jacoco-maven-plugin</artifactId>
4   <version>0.8.11</version>
5   <executions>
6     <execution>
7       <id>prepare-agent</id>
8       <goals>
9         <goal>prepare-agent</goal>
10      </goals>
11    </execution>
12    <execution>
13      <id>report</id>
14      <phase>test</phase>
15      <goals>
16        <goal>report</goal>
17      </goals>
18    </execution>
19    <execution>
20      <id>check</id>
21      <goals>
22        <goal>check</goal>
23      </goals>
24      <configuration>
25        <rules>
26          <rule>
27            <element>BUNDLE</element>
28            <limits>
29              <limit>
30                <counter>LINE</counter>
31                <value>COVEREDRATIO</value>
32                <minimum>0.80</minimum>
33              </limit>
34              <limit>
35                <counter>BRANCH</counter>
36                <value>COVEREDRATIO</value>
37                <minimum>0.75</minimum>
38              </limit>
39            </limits>
40          </rule>
41          <rule>
42            <element>CLASS</element>
43            <limits>
44              <limit>
45                <counter>LINE</counter>
46                <value>COVEREDRATIO</value>
47                <minimum>0.70</minimum>
48              </limit>
49            </limits>
```

```
50         <excludes>
51             <exclude>*Exception</exclude>
52             <exclude>*Config</exclude>
53         </excludes>
54     </rule>
55 </rules>
56 </configuration>
57 </execution>
58 </executions>
59 </plugin>
```

Listing 12: JaCoCo Configuration for Java (pom.xml excerpt)

### 3.2.4 Coverage Enforcement in CI/CD

```
1 name: Test Coverage
2
3 on:
4   pull_request:
5     branches: [main]
6
7 jobs:
8   coverage:
9     runs-on: ubuntu-latest
10    steps:
11      - uses: actions/checkout@v4
12
13      - name: Setup Node.js
14        uses: actions/setup-node@v4
15        with:
16          node-version: '20'
17          cache: 'npm'
18
19      - run: npm ci
20
21      - name: Run Tests with Coverage
22        run: npm run test:coverage
23
24      - name: Check Coverage Thresholds
25        run: |
26          COVERAGE=$(cat coverage/coverage-summary.json | \
27            jq '.total.lines.pct')
28          echo "Line coverage: ${COVERAGE}%"
29
30          if (( $(echo "$COVERAGE < 80" | bc -l) )); then
31            echo "::error::Coverage ${COVERAGE}% is below 80% threshold"
32            exit 1
33          fi
34
35      - name: Upload Coverage to Codecov
36        uses: codecov/codecov-action@v4
37        with:
38          token: ${ secrets.CODECOV_TOKEN }
39          files: ./coverage/lcov.info
40          fail_ci_if_error: true
41          verbose: true
42
43      - name: Comment Coverage on PR
44        uses: actions/github-script@v7
45        with:
46          script: |
47            const fs = require('fs');
48            const summary = JSON.parse(
```

```
49         fs.readFileSync('coverage/coverage-summary.json')
50     );
51
52     const body = `## Coverage Report
53
54     | Metric | Coverage |
55     |-----|-----|
56     | Lines | ${summary.total.lines.pct}% |
57     | Branches | ${summary.total.branches.pct}% |
58     | Functions | ${summary.total.functions.pct}% |
59     | Statements | ${summary.total.statements.pct}% |`;
60
61     github.rest.issues.createComment({
62         issue_number: context.issue.number,
63         owner: context.repo.owner,
64         repo: context.repo.repo,
65         body: body
66     });
```

Listing 13: GitHub Actions Coverage Enforcement

**Best Practice**

Coverage metrics should be enforced differently for new code versus existing code. Require 80%+ coverage on new code while gradually improving legacy code coverage. Use SonarQube's "New Code" analysis to enforce stricter standards on changes.

**Warning**

High coverage does not guarantee high-quality tests. A codebase can achieve 100% coverage with tests that have no assertions. Always combine coverage metrics with mutation testing (using tools like Stryker or PIT) to verify test effectiveness.

## 4 Security Scanning

**Gates:** Vulnerability Scan Open Source Scan

Security scanning gates integrate security into the development lifecycle, implementing the “shift left” security philosophy. These gates automatically detect vulnerabilities before they reach production.

### 4.1 Gate 5: Vulnerability Scan

#### 4.1.1 Security Scanning Categories

Table 6: Security Scanning Types

Type	What It Scans	Tools	When to Run
SAST	Source code patterns	Semgrep, CodeQL, Checkmarx	Every commit
SCA	Dependencies	Snyk, Dependabot, OWASP DC	Every build
DAST	Running application	OWASP ZAP, Burp Suite	Post-deployment
Container	Container images	Trivy, Grype, Clair	Image build
IaC	Infrastructure code	Checkov, tfsec, KICS	Infrastructure changes
Secrets	Hardcoded secrets	Gitleaks, TruffleHog	Pre-commit, CI

### 4.1.2 SAST Implementation with Semgrep

```

1 rules:
2   # SQL Injection Detection
3   - id: sql-injection
4     patterns:
5     - pattern-either:
6       - pattern: |
7         $QUERY = "..." + $USER_INPUT + "..."
8         $DB.execute($QUERY)
9       - pattern: |
10        $DB.execute(f"...{$USER_INPUT}...")
11    message: "Potential SQL injection vulnerability"
12    severity: ERROR
13    languages: [python, javascript]
14    metadata:
15      cwe: "CWE-89"
16      owasp: "A03:2021"
17
18   # Hardcoded Secrets
19   - id: hardcoded-secret
20     pattern-regex: |
21       (?i)(password|secret|api_key|token)\s*=\s*['\"][^'\"]{8,}['\"]
22     message: "Potential hardcoded secret detected"
23     severity: ERROR
24     languages: [generic]
25     metadata:
26       cwe: "CWE-798"
27
28   # Insecure Deserialization
29   - id: insecure-pickle
30     patterns:
31     - pattern: pickle.loads($DATA)
32     - pattern-not-inside: |
33       if is_trusted($DATA):
34         ...
35     message: "Unsafe pickle deserialization"
36     severity: ERROR
37     languages: [python]
38     metadata:
39       cwe: "CWE-502"
40
41   # Path Traversal
42   - id: path-traversal
43     patterns:
44     - pattern: open($USER_INPUT)
45     - pattern-not: open(os.path.join($SAFE_DIR,
46       ↪ os.path.basename($USER_INPUT)))
47     message: "Potential path traversal vulnerability"
48     severity: WARNING

```

```
48   languages: [python]
49   metadata:
50     cwe: "CWE-22"
```

Listing 14: Semgrep Configuration (.semgrep.yml)

### 4.1.3 Container Image Scanning with Trivy

```
1 name: Container Security Scan
2
3 on:
4   push:
5     paths:
6       - 'Dockerfile*'
7       - '.dockerignore'
8       - 'docker-compose*.yaml'
9
10  jobs:
11    scan:
12      runs-on: ubuntu-latest
13      steps:
14        - uses: actions/checkout@v4
15
16        - name: Build Image
17          run: docker build -t myapp:${{ github.sha }} .
18
19        - name: Run Trivy Vulnerability Scanner
20          uses: aquasecurity/trivy-action@master
21          with:
22            image-ref: 'myapp:${{ github.sha }}'
23            format: 'sarif'
24            output: 'trivy-results.sarif'
25            severity: 'CRITICAL,HIGH'
26            vuln-type: 'os,library'
27            ignore-unfixed: true
28
29        - name: Upload Trivy Scan Results
30          uses: github/codeql-action/upload-sarif@v3
31          with:
32            sarif_file: 'trivy-results.sarif'
33
34        - name: Fail on Critical Vulnerabilities
35          uses: aquasecurity/trivy-action@master
36          with:
37            image-ref: 'myapp:${{ github.sha }}'
38            exit-code: '1'
39            severity: 'CRITICAL'
40            vuln-type: 'os,library'
```

Listing 15: Trivy Container Scanning Workflow

#### 4.1.4 Infrastructure as Code Scanning

```
1 name: IaC Security Scan
2
3 on:
4   pull_request:
5     paths:
6       - 'terraform/**'
7       - 'kubernetes/**'
8       - 'cloudformation/**'
9
10  jobs:
11    checkov:
12      runs-on: ubuntu-latest
13      steps:
14        - uses: actions/checkout@v4
15
16        - name: Run Checkov
17          uses: bridgecrewio/checkov-action@master
18          with:
19            directory: .
20            framework: terraform,kubernetes,cloudformation
21            output_format: sarif
22            output_file_path: checkov-results.sarif
23            soft_fail: false
24            skip_check: CKV_AWS_18,CKV_AWS_21 # Document exceptions
25            config_file: .checkov.yml
26
27        - name: Upload Checkov Results
28          uses: github/codeql-action/upload-sarif@v3
29          with:
30            sarif_file: checkov-results.sarif
```

Listing 16: Checkov IaC Security Scanning

```
1  # Checkov configuration
2  soft-fail: false
3  compact: true
4  framework:
5    - terraform
6    - kubernetes
7    - dockerfile
8
9  # Severity thresholds
10 hard-fail-on:
11   - CRITICAL
12   - HIGH
13
14 # Skip specific checks with justification
15 skip-check:
16   # Skip S3 versioning for ephemeral buckets
17   - CKV_AWS_21:
18     comment: "Ephemeral data bucket, versioning not required"
19     resources:
20       - aws_s3_bucket.temp_data
21
22 # Custom policies
23 external-checks-dir:
24   - ./security/custom-policies
25
26 # Output configuration
27 output:
28   - cli
29   - sarif
30   - json
```

Listing 17: Checkov Configuration (.checkov.yml)

## 4.2 Gate 6: Open Source Scan

### 4.2.1 Software Composition Analysis

Open Source Scan ensures that third-party dependencies do not introduce vulnerabilities or license compliance issues.

```
1 # Snyc policy file
2 version: v1.25.0
3
4 # Ignore specific vulnerabilities with justification
5 ignore:
6   SNYK-JS-LODASH-567746:
7     - '*':
8       reason: 'Not exploitable in our usage context'
9       expires: '2025-01-01T00:00:00.000Z'
10      created: '2024-01-01T00:00:00.000Z'
11
12 # Patch vulnerabilities when fixes aren't available
13 patch:
14   SNYK-JS-HOEK-12345:
15     - 'request > hawk > hoek':
16       patched: '2024-06-01T00:00:00.000Z'
17
18 # Language-specific settings
19 language-settings:
20   python:
21     python-version: '3.12'
22
23 # License policies
24 license-policies:
25   severity:
26     - GPL-3.0: high
27     - AGPL-3.0: critical
28     - unlicensed: critical
29   allow:
30     - MIT
31     - Apache-2.0
32     - BSD-2-Clause
33     - BSD-3-Clause
34     - ISC
```

Listing 18: Snyc Configuration (.snyc)

### 4.2.2 OWASP Dependency-Check Configuration

```

1 name: Dependency Security Scan
2
3 on:
4   schedule:
5     - cron: '0 6 * * *' # Daily at 6 AM
6   push:
7     paths:
8       - '**/package*.json'
9       - '**/requirements*.txt'
10      - '**/Pipfile*'
11      - '**/pom.xml'
12      - '**/build.gradle*'
13
14 jobs:
15   dependency-check:
16     runs-on: ubuntu-latest
17     steps:
18       - uses: actions/checkout@v4
19
20       - name: Run OWASP Dependency-Check
21         uses: dependency-check/Dependency-Check_Action@main
22         with:
23           project: 'MyProject'
24           path: '.'
25           format: 'ALL'
26           args: >
27             --failOnCVSS 7
28             --enableRetired
29             --enableExperimental
30             --suppression ./dependency-suppression.xml
31
32       - name: Upload Dependency-Check Report
33         uses: actions/upload-artifact@v4
34         with:
35           name: dependency-check-report
36           path: reports/
37
38       - name: Publish to DefectDojo
39         run: |
40           curl -X POST "${{ secrets.DEFECTDOJO_URL }}
41             ↪ }}/api/v2/import-scan/" \
42             -H "Authorization: Token ${{ secrets.DEFECTDOJO_TOKEN }}" \
43             -F "scan_type=Dependency Check Scan" \
44             -F "file=@reports/dependency-check-report.xml" \
45             -F "engagement=${{ secrets.DEFECTDOJO_ENGAGEMENT }}" \
46             -F "verified=false" \
47             -F "active=true"

```

Listing 19: OWASP Dependency-Check GitHub Action

### 4.2.3 License Compliance Policy

Table 7: License Compliance Categories

Category	Licenses	Policy
Permissive	MIT, Apache 2.0, BSD	<b>Approved</b> for all use
Weak Copyleft	LGPL, MPL	<b>Approved</b> with attribution
Strong Copyleft	GPL, AGPL	<b>Restricted</b> - requires review
Proprietary	Commercial licenses	<b>Restricted</b> - requires procurement
Unknown	No license specified	<b>Blocked</b> - cannot use

```

1 name: License Compliance
2
3 on:
4   pull_request:
5     paths:
6       - '**/package*.json'
7       - '**/requirements*.txt'
8       - '**/go.mod'
9
10  jobs:
11    license-check:
12      runs-on: ubuntu-latest
13      steps:
14        - uses: actions/checkout@v4
15
16        # Node.js license check
17        - name: Check npm licenses
18          run: |
19            npx license-checker \
20              --production \
21              --json \
22              --out licenses.json \
23              --failOn "GPL-3.0;AGPL-3.0;UNLICENSED"
24
25        # Python license check
26        - name: Check pip licenses
27          run: |
28            pip install pip-licenses
29            pip-licenses \
30              --format=json \
31              --output-file=python-licenses.json \
32              --fail-on="GPL-3.0;AGPL-3.0"
33
34        # Go license check
35        - name: Check Go licenses
36          run: |
37            go install github.com/google/go-licenses@latest

```

```
38     go-licenses check ./... \  
39         --disallowed_types=restricted,forbidden  
40  
41     - name: Generate SBOM  
42       run: |  
43         npx @cyclonedx/cyclonedx-npm \  
44           --output-file sbom.json \  
45           --output-format json  
46  
47     - name: Upload SBOM  
48       uses: actions/upload-artifact@v4  
49       with:  
50         name: sbom  
51         path: sbom.json
```

Listing 20: License Compliance Workflow

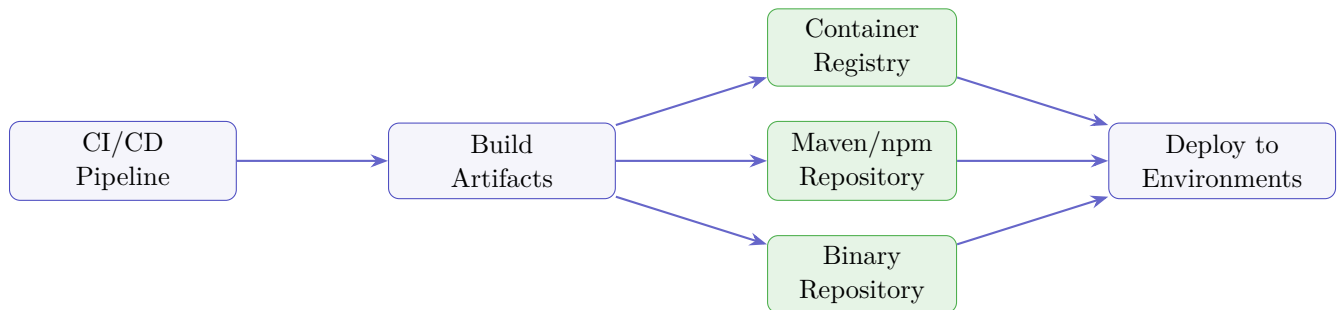
## 5 Artifact Management

**Gates:** Artifact Version Control

Artifact Management ensures that build outputs are properly versioned, stored, and retrievable for deployment to any environment.

### 5.1 Gate 7: Artifact Version Control

#### 5.1.1 Artifact Repository Architecture



### 5.1.2 Semantic Versioning Strategy

```
1 // .releaserc.json
2 {
3   "branches": ["main", {"name": "beta", "prerelease": true}],
4   "plugins": [
5     ["@semantic-release/commit-analyzer", {
6       "preset": "conventionalcommits",
7       "releaseRules": [
8         {"type": "feat", "release": "minor"},
9         {"type": "fix", "release": "patch"},
10        {"type": "perf", "release": "patch"},
11        {"breaking": true, "release": "major"},
12        {"type": "docs", "release": false},
13        {"type": "style", "release": false},
14        {"type": "refactor", "release": "patch"},
15        {"type": "test", "release": false},
16        {"type": "ci", "release": false}
17      ]
18    }],
19    "@semantic-release/release-notes-generator",
20    "@semantic-release/changelog",
21    ["@semantic-release/npm", {
22      "npmPublish": true
23    }],
24    ["@semantic-release/git", {
25      "assets": ["package.json", "CHANGELOG.md"],
26      "message": "chore(release): ${nextRelease.version}"
27    }],
28    "@semantic-release/github"
29  ]
30 }
```

Listing 21: Semantic Versioning with Semantic Release

### 5.1.3 Container Image Tagging Strategy

```

1 name: Build and Push Container
2
3 on:
4   push:
5     branches: [main]
6     tags: ['v*']
7
8 env:
9   REGISTRY: ghcr.io
10  IMAGE_NAME: ${ github.repository }
11
12 jobs:
13   build:
14     runs-on: ubuntu-latest
15     permissions:
16       contents: read
17       packages: write
18
19     steps:
20       - uses: actions/checkout@v4
21
22       - name: Set up Docker Buildx
23         uses: docker/setup-buildx-action@v3
24
25       - name: Log in to Container Registry
26         uses: docker/login-action@v3
27         with:
28           registry: ${ env.REGISTRY }
29           username: ${ github.actor }
30           password: ${ secrets.GITHUB_TOKEN }
31
32       - name: Extract metadata
33         id: meta
34         uses: docker/metadata-action@v5
35         with:
36           images: ${ env.REGISTRY }/${ env.IMAGE_NAME }
37           tags: |
38             # Branch name
39             type=ref,event=branch
40             # Git short SHA
41             type=sha,prefix=sha-
42             # Semantic version from tag
43             type=semver,pattern={{version}}
44             type=semver,pattern={{major}}.{{minor}}
45             type=semver,pattern={{major}}
46             # Latest tag for main branch
47             type=raw,value=latest,enable=${ github.ref ==
               ↪ 'refs/heads/main' }}

```

```
48         # Build timestamp
49         type=raw,value={{date 'YYYYMMDD-HH:mm:ss'}}
50
51     - name: Build and push
52       uses: docker/build-push-action@v5
53       with:
54         context: .
55         push: true
56         tags: ${{ steps.meta.outputs.tags }}
57         labels: ${{ steps.meta.outputs.labels }}
58         cache-from: type=gha
59         cache-to: type=gha,mode=max
60         build-args: |
61             VERSION=${{ github.ref_name }}
62             COMMIT=${{ github.sha }}
63             BUILD_DATE=${{ github.event.head_commit.timestamp }}
64
65     - name: Sign the container image
66       run: |
67         cosign sign --yes \
68           ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}@${{
69             ↪ steps.build.outputs.digest }}
70
71     env:
72       COSIGN_EXPERIMENTAL: 1
```

Listing 22: Docker Image Build and Push Workflow

### 5.1.4 Artifact Integrity Verification

```

1  name: Artifact Integrity
2
3  jobs:
4    sign:
5      runs-on: ubuntu-latest
6      steps:
7        - name: Generate SBOM
8          uses: anchore/sbom-action@v0
9          with:
10             path: .
11             format: cyclonedx-json
12             output-file: sbom.json
13
14        - name: Sign SBOM with Sigstore
15          run: |
16            cosign sign-blob \
17              --yes \
18              --bundle sbom.bundle \
19              sbom.json
20
21        - name: Attest SBOM to image
22          run: |
23            cosign attest \
24              --yes \
25              --predicate sbom.json \
26              --type cyclonedx \
27              ${ env.REGISTRY }}/${ env.IMAGE_NAME }}@${{
28                ↪ needs.build.outputs.digest }}
29
30        - name: Generate SLSA Provenance
31          uses:
32            ↪ slsa-framework/slsa-github-generator/.github/workflows/generator_conta
33          with:
34            image: ${ env.REGISTRY }}/${ env.IMAGE_NAME }}
35            digest: ${ needs.build.outputs.digest }}
36
37    verify:
38      needs: sign
39      runs-on: ubuntu-latest
40      steps:
41        - name: Verify image signature
42          run: |
43            cosign verify \
44              --certificate-identity-regexp ".*@mycompany.com" \
45              --certificate-oidc-issuer
46            ↪ https://token.actions.githubusercontent.com \
47            ${ env.REGISTRY }}/${ env.IMAGE_NAME }}:latest

```

```
46     - name: Verify SBOM attestation
47       run: |
48         cosign verify-attestation \
49           --type cyclonedx \
50           --certificate-identity-regexp ".*@mycompany.com" \
51           --certificate-oidc-issuer
52             ↪ https://token.actions.githubusercontent.com \
               ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}@${{
               ↪ needs.build.outputs.digest }}
```

Listing 23: Artifact Signing and Verification

## 6 Infrastructure Provisioning

**Gates:** Automatic Provision    Immutable Servers

Infrastructure Provisioning gates ensure that deployment environments are created automatically, consistently, and without manual intervention.

### 6.1 Gate 8: Automatic Provision

#### 6.1.1 Infrastructure as Code with Terraform

```

1 terraform {
2   required_version = ">= 1.6.0"
3
4   required_providers {
5     aws = {
6       source = "hashicorp/aws"
7       version = "~> 5.0"
8     }
9     kubernetes = {
10      source = "hashicorp/kubernetes"
11      version = "~> 2.24"
12    }
13  }
14
15  backend "s3" {
16    bucket      = "mycompany-terraform-state"
17    key          = "environments/production/terraform.tfstate"
18    region      = "us-east-1"
19    encrypt      = true
20    dynamodb_table = "terraform-state-lock"
21  }
22 }
23
24 # EKS Cluster
25 module "eks" {
26   source = "terraform-aws-modules/eks/aws"
27   version = "~> 19.0"
28
29   cluster_name      = var.cluster_name
30   cluster_version   = "1.28"
31
32   vpc_id            = module.vpc.vpc_id
33   subnet_ids        = module.vpc.private_subnets
34
35   cluster_endpoint_public_access = true
36
37   eks_managed_node_groups = {
38     general = {

```

```
39     name           = "general-workloads"
40     instance_types = ["m6i.large"]
41     min_size        = 2
42     max_size        = 10
43     desired_size    = 3
44
45     labels = {
46         workload-type = "general"
47     }
48
49     tags = {
50         Environment = var.environment
51         Terraform   = "true"
52     }
53 }
54 }
55
56 # Enable IRSA
57 enable_irsa = true
58
59 tags = local.common_tags
60 }
61
62 # Application Load Balancer
63 module "alb" {
64     source = "terraform-aws-modules/alb/aws"
65     version = "~> 9.0"
66
67     name           = "${var.project_name}-alb"
68     load_balancer_type = "application"
69     vpc_id         = module.vpc.vpc_id
70     subnets       = module.vpc.public_subnets
71     security_groups = [module.alb_sg.security_group_id]
72
73     listeners = {
74         https = {
75             port           = 443
76             protocol       = "HTTPS"
77             certificate_arn = var.acm_certificate_arn
78
79             forward = {
80                 target_group_key = "app"
81             }
82         }
83     }
84
85     target_groups = {
86         app = {
87             name_prefix = "app-"
88             protocol    = "HTTP"
```

```
89     port                = 8080
90     target_type         = "ip"
91
92     health_check = {
93         enabled          = true
94         healthy_threshold = 2
95         unhealthy_threshold = 3
96         interval         = 30
97         path              = "/health"
98         port              = "traffic-port"
99     }
100 }
101 }
102
103 tags = local.common_tags
104 }
```

Listing 24: Terraform Main Configuration (main.tf)

### 6.1.2 Terraform CI/CD Pipeline

```
1 name: Terraform Infrastructure
2
3 on:
4   push:
5     branches: [main]
6     paths:
7       - 'terraform/**'
8   pull_request:
9     branches: [main]
10    paths:
11      - 'terraform/**'
12
13 env:
14   TF_VERSION: '1.6.0'
15   TF_WORKING_DIR: 'terraform/environments/production'
16
17 jobs:
18   validate:
19     runs-on: ubuntu-latest
20     steps:
21       - uses: actions/checkout@v4
22
23       - name: Setup Terraform
24         uses: hashicorp/setup-terraform@v3
25         with:
26           terraform_version: ${ env.TF_VERSION }
27
28       - name: Terraform Format Check
29         run: terraform fmt -check -recursive
30         working-directory: terraform
31
32       - name: Terraform Init
33         run: terraform init -backend=false
34         working-directory: ${ env.TF_WORKING_DIR }
35
36       - name: Terraform Validate
37         run: terraform validate
38         working-directory: ${ env.TF_WORKING_DIR }
39
40       - name: Run tfsec
41         uses: aquasecurity/tfsec-action@v1.0.0
42         with:
43           working_directory: terraform
44           soft_fail: false
45
46       - name: Run Checkov
47         uses: bridgecrewio/checkov-action@master
48         with:
```

```
49     directory: terraform
50     framework: terraform
51
52   plan:
53     needs: validate
54     runs-on: ubuntu-latest
55     steps:
56       - uses: actions/checkout@v4
57
58       - name: Setup Terraform
59         uses: hashicorp/setup-terraform@v3
60         with:
61           terraform_version: ${ env.TF_VERSION }
62
63       - name: Configure AWS Credentials
64         uses: aws-actions/configure-aws-credentials@v4
65         with:
66           role-to-assume: ${ secrets.AWS_ROLE_ARN }
67           aws-region: us-east-1
68
69       - name: Terraform Init
70         run: terraform init
71         working-directory: ${ env.TF_WORKING_DIR }
72
73       - name: Terraform Plan
74         id: plan
75         run: |
76           terraform plan \
77             -input=false \
78             -out=tfplan \
79             -no-color
80         working-directory: ${ env.TF_WORKING_DIR }
81
82       - name: Upload Plan
83         uses: actions/upload-artifact@v4
84         with:
85           name: terraform-plan
86           path: ${ env.TF_WORKING_DIR }/tfplan
87
88       - name: Post Plan to PR
89         if: github.event_name == 'pull_request'
90         uses: actions/github-script@v7
91         with:
92           script: |
93             const output = '#### Terraform Plan
94             \\\'\'\'
95             ${ steps.plan.outputs.stdout }
96             \\\'\'\'\'
97             github.rest.issues.createComment({
98               issue_number: context.issue.number,
```

```
99         owner: context.repo.owner,
100         repo: context.repo.repo,
101         body: output
102     });
103
104     apply:
105       needs: plan
106       if: github.ref == 'refs/heads/main' && github.event_name == 'push'
107       runs-on: ubuntu-latest
108       environment: production
109       steps:
110         - uses: actions/checkout@v4
111
112         - name: Setup Terraform
113           uses: hashicorp/setup-terraform@v3
114           with:
115             terraform_version: ${ env.TF_VERSION }
116
117         - name: Configure AWS Credentials
118           uses: aws-actions/configure-aws-credentials@v4
119           with:
120             role-to-assume: ${ secrets.AWS_ROLE_ARN }
121             aws-region: us-east-1
122
123         - name: Download Plan
124           uses: actions/download-artifact@v4
125           with:
126             name: terraform-plan
127             path: ${ env.TF_WORKING_DIR }
128
129         - name: Terraform Init
130           run: terraform init
131           working-directory: ${ env.TF_WORKING_DIR }
132
133         - name: Terraform Apply
134           run: terraform apply -auto-approve tfplan
135           working-directory: ${ env.TF_WORKING_DIR }
```

Listing 25: Terraform GitHub Actions Workflow

## 6.2 Gate 9: Immutable Servers

### 6.2.1 Container Image Definition

```

1  # syntax=docker/dockerfile:1.6
2
3  # =====
4  # Stage 1: Dependencies
5  # =====
6  FROM node:20-alpine AS deps
7  WORKDIR /app
8
9  # Install dependencies only when needed
10 COPY package.json package-lock.json ./
11 RUN --mount=type=cache,target=/root/.npm \
12     npm ci --only=production
13
14 # =====
15 # Stage 2: Builder
16 # =====
17 FROM node:20-alpine AS builder
18 WORKDIR /app
19
20 COPY --from=deps /app/node_modules ./node_modules
21 COPY . .
22
23 # Build application
24 RUN npm run build
25
26 # =====
27 # Stage 3: Production
28 # =====
29 FROM gcr.io/distroless/nodejs20-debian12 AS production
30
31 # Labels for container metadata
32 LABEL org.opencontainers.image.title="My Application" \
33       org.opencontainers.image.description="Production application" \
34       org.opencontainers.image.vendor="MyCompany" \
35       org.opencontainers.image.source="https://github.com/mycompany/myapp"
36
37 WORKDIR /app
38
39 # Copy only production artifacts
40 COPY --from=builder /app/dist ./dist
41 COPY --from=builder /app/node_modules ./node_modules
42 COPY --from=builder /app/package.json ./
43
44 # Set environment
45 ENV NODE_ENV=production \
46     PORT=8080
47

```

```
48 # Non-root user (distroless uses nonroot by default)
49 USER nonroot
50
51 # Health check
52 HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
53     CMD ["/nodejs/bin/node", "-e",
54         ↪ "require('http').get('http://localhost:8080/health')"]
55 EXPOSE 8080
56
57 CMD ["dist/server.js"]
```

Listing 26: Production Dockerfile with Multi-Stage Build

### 6.2.2 AMI Building with Packer

```

1  packer {
2    required_plugins {
3      amazon = {
4        version = ">= 1.2.0"
5        source  = "github.com/hashicorp/amazon"
6      }
7      ansible = {
8        version = ">= 1.1.0"
9        source  = "github.com/hashicorp/ansible"
10     }
11   }
12 }
13
14 source "amazon-ebs" "app" {
15   ami_name      = "myapp-${var.version}-${timestamp}"
16   instance_type = "t3.medium"
17   region        = var.aws_region
18
19   source_ami_filter {
20     filters = {
21       name =
22         ↪ "ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"
23       root-device-type = "ebs"
24       virtualization-type = "hvm"
25     }
26     owners      = ["099720109477"] # Canonical
27     most_recent = true
28   }
29   ssh_username = "ubuntu"
30
31   launch_block_device_mappings {
32     device_name = "/dev/sda1"
33     volume_size = 30
34     volume_type = "gp3"
35     encrypted   = true
36     delete_on_termination = true
37   }
38
39   tags = {
40     Name       = "myapp-${var.version}"
41     Version    = var.version
42     BuildDate  = timestamp()
43     GitCommit  = var.git_commit
44     Environment = var.environment
45   }
46 }
47

```

```
48 build {
49     sources = ["source.amazon-ebs.app"]
50
51     provisioner "shell" {
52         inline = [
53             "sudo apt-get update",
54             "sudo apt-get install -y python3 python3-pip"
55         ]
56     }
57
58     provisioner "ansible" {
59         playbook_file = "ansible/playbook.yml"
60         extra_arguments = [
61             "--extra-vars", "app_version=${var.version}"
62         ]
63     }
64
65     provisioner "shell" {
66         inline = [
67             "sudo apt-get clean",
68             "sudo rm -rf /var/lib/apt/lists/*",
69             "sudo rm -rf /tmp/*"
70         ]
71     }
72
73     post-processor "manifest" {
74         output      = "manifest.json"
75         strip_path  = true
76     }
77 }
78
79 variable "version" {
80     type      = string
81     description = "Application version"
82 }
83
84 variable "git_commit" {
85     type      = string
86     description = "Git commit SHA"
87 }
88
89 variable "aws_region" {
90     type      = string
91     default   = "us-east-1"
92 }
93
94 variable "environment" {
95     type      = string
96     default   = "production"
97 }
```

---

Listing 27: Packer Configuration for Immutable AMI

## 7 Integration and Performance Testing

**Gates:** Integration Testing Performance Testing

Integration and Performance Testing gates verify that the application functions correctly in a production-like environment and meets performance requirements.

### 7.1 Gate 10: Integration Testing

#### 7.1.1 End-to-End Testing with Playwright

```
1 import { defineConfig, devices } from '@playwright/test';
2
3 export default defineConfig({
4   testDir: './e2e',
5   fullyParallel: true,
6   forbidOnly: !!process.env.CI,
7   retries: process.env.CI ? 2 : 0,
8   workers: process.env.CI ? 4 : undefined,
9   reporter: [
10    ['html', { outputFolder: 'playwright-report' }],
11    ['junit', { outputFile: 'test-results/junit.xml' }],
12    ['json', { outputFile: 'test-results/results.json' }]
13  ],
14
15  use: {
16    baseURL: process.env.BASE_URL || 'http://localhost:3000',
17    trace: 'on-first-retry',
18    screenshot: 'only-on-failure',
19    video: 'retain-on-failure',
20  },
21
22  projects: [
23    {
24      name: 'chromium',
25      use: { ...devices['Desktop Chrome'] },
26    },
27    {
28      name: 'firefox',
29      use: { ...devices['Desktop Firefox'] },
30    },
31    {
32      name: 'webkit',
33      use: { ...devices['Desktop Safari'] },
34    },
35    {
36      name: 'mobile-chrome',
37      use: { ...devices['Pixel 5'] },
```

```
38     },
39     {
40       name: 'mobile-safari',
41       use: { ...devices['iPhone 12'] },
42     },
43   ],
44
45   webServer: process.env.CI ? undefined : {
46     command: 'npm run start',
47     url: 'http://localhost:3000',
48     reuseExistingServer: !process.env.CI,
49   },
50 });
```

Listing 28: Playwright Configuration (playwright.config.ts)

```
1 import { test, expect } from '@playwright/test';
2
3 test.describe('User Authentication Flow', () => {
4   test.beforeEach(async ({ page }) => {
5     await page.goto('/');
6   });
7
8   test('should allow user to login', async ({ page }) => {
9     // Navigate to login
10    await page.click('[data-testid="login-button"]');
11
12    // Fill credentials
13    await page.fill('[data-testid="email-input"]', 'test@example.com');
14    await page.fill('[data-testid="password-input"]', 'password123');
15
16    // Submit form
17    await page.click('[data-testid="submit-button"]');
18
19    // Verify successful login
20    await
21      ↪ expect(page.locator('[data-testid="user-menu"]')).toBeVisible();
22    await expect(page).toHaveURL(/.*dashboard/);
23  });
24
25  test('should show error for invalid credentials', async ({ page }) =>
26    ↪ {
27    await page.click('[data-testid="login-button"]');
28    await page.fill('[data-testid="email-input"]', 'wrong@example.com');
29    await page.fill('[data-testid="password-input"]', 'wrongpassword');
30    await page.click('[data-testid="submit-button"]');
31
32    await expect(page.locator('[data-testid="error-message"]'))
33      .toContainText('Invalid credentials');
34  });
35
36  test('should complete checkout flow', async ({ page }) => {
37    // Login first
38    await page.goto('/login');
39    await page.fill('[data-testid="email-input"]', 'test@example.com');
40    await page.fill('[data-testid="password-input"]', 'password123');
41    await page.click('[data-testid="submit-button"]');
42
43    // Add item to cart
44    await page.goto('/products');
45    await page.click('[data-testid="product-card"]:first-child');
46    await page.click('[data-testid="add-to-cart"]');
47
48    // Proceed to checkout
49    await page.click('[data-testid="cart-icon"]');
```

```
48     await page.click('[data-testid="checkout-button"]');
49
50     // Complete payment
51     await page.fill('[data-testid="card-number"]', '4242424242424242');
52     await page.fill('[data-testid="card-expiry"]', '12/25');
53     await page.fill('[data-testid="card-cvc"]', '123');
54     await page.click('[data-testid="pay-button"]');
55
56     // Verify order confirmation
57     await expect(page.locator('[data-testid="order-confirmation"]'))
58       .toBeVisible({ timeout: 10000 });
59   });
60 }
```

Listing 29: Playwright E2E Test Example

### 7.1.2 API Integration Testing

```
1 name: API Integration Tests
2
3 on:
4   deployment_status:
5
6 jobs:
7   api-tests:
8     if: github.event.deployment_status.state == 'success'
9     runs-on: ubuntu-latest
10    steps:
11      - uses: actions/checkout@v4
12
13      - name: Run Newman Tests
14        run: |
15          npx newman run \
16            tests/api/collection.json \
17            --environment tests/api/environments/${{
18              ↪ github.event.deployment.environment }}.json \
19            --reporters cli,junit,htmlextra \
20            --reporter-junit-export results/junit.xml \
21            --reporter-htmlextra-export results/report.html \
22            --iteration-count 3 \
23            --delay-request 100
24
25      - name: Upload Test Results
26        uses: actions/upload-artifact@v4
27        if: always()
28        with:
29          name: api-test-results
30          path: results/
31
32      - name: Contract Testing with Pact
33        run: |
34          npx pact-broker can-i-deploy \
35            --pacticipant MyService \
36            --version ${github.sha} \
37            --to-environment ${github.event.deployment.environment }
```

Listing 30: API Integration Tests with Newman/Postman

## 7.2 Gate 11: Performance Testing

### 7.2.1 Load Testing with k6

```
1 import http from 'k6/http';
2 import { check, sleep, group } from 'k6';
3 import { Rate, Trend } from 'k6/metrics';
4
5 // Custom metrics
6 const errorRate = new Rate('errors');
7 const apiLatency = new Trend('api_latency');
8
9 // Test configuration
10 export const options = {
11   scenarios: {
12     // Smoke test
13     smoke: {
14       executor: 'constant-vus',
15       vus: 1,
16       duration: '1m',
17       startTime: '0s',
18     },
19     // Load test
20     load: {
21       executor: 'ramping-vus',
22       startVUs: 0,
23       stages: [
24         { duration: '2m', target: 100 }, // Ramp up
25         { duration: '5m', target: 100 }, // Steady state
26         { duration: '2m', target: 200 }, // Peak load
27         { duration: '5m', target: 200 }, // Sustained peak
28         { duration: '2m', target: 0 },   // Ramp down
29       ],
30       startTime: '1m',
31     },
32     // Stress test
33     stress: {
34       executor: 'ramping-vus',
35       startVUs: 0,
36       stages: [
37         { duration: '2m', target: 500 },
38         { duration: '5m', target: 500 },
39         { duration: '2m', target: 0 },
40       ],
41       startTime: '20m',
42     },
43   ],
44
45   thresholds: {
46     // Response time thresholds
47     'http_req_duration': ['p(95)<500', 'p(99)<1000'],
```

```

48   'http_req_duration{endpoint:api}': ['p(95)<200'],
49   'http_req_duration{endpoint:static}': ['p(95)<100'],
50
51   // Error rate thresholds
52   'errors': ['rate<0.01'],           // <1% error rate
53   'http_req_failed': ['rate<0.01'],
54
55   // Throughput
56   'http_reqs': ['rate>100'],         // >100 RPS
57 },
58 };
59
60 const BASE_URL = __ENV.BASE_URL || 'https://api.example.com';
61
62 export default function () {
63   group('API Endpoints', function () {
64     // Health check
65     const healthRes = http.get(`${BASE_URL}/health`, {
66       tags: { endpoint: 'health' },
67     });
68     check(healthRes, {
69       'health check status is 200': (r) => r.status === 200,
70     });
71
72     // User listing
73     const usersRes = http.get(`${BASE_URL}/api/v1/users`, {
74       tags: { endpoint: 'api' },
75       headers: { 'Authorization': 'Bearer ${__ENV.API_TOKEN}' },
76     });
77
78     const usersOk = check(usersRes, {
79       'users status is 200': (r) => r.status === 200,
80       'users response has data': (r) => r.json('data') !== undefined,
81       'users latency < 500ms': (r) => r.timings.duration < 500,
82     });
83
84     errorRate.add(!usersOk);
85     apiLatency.add(usersRes.timings.duration);
86
87     // Create resource
88     const createRes = http.post(
89       `${BASE_URL}/api/v1/items`,
90       JSON.stringify({
91         name: `test-item-${Date.now()}`,
92         value: Math.random() * 100,
93       }),
94       {
95         headers: {
96           'Content-Type': 'application/json',
97           'Authorization': 'Bearer ${__ENV.API_TOKEN}',

```

```
98     },
99     tags: { endpoint: 'api' },
100   }
101 );
102
103   check(createRes, {
104     'create status is 201': (r) => r.status === 201,
105   });
106 });
107
108   sleep(1);
109 }
110
111 export function handleSummary(data) {
112   return {
113     'summary.json': JSON.stringify(data, null, 2),
114     'summary.html': htmlReport(data),
115   };
116 }
```

Listing 31: k6 Load Test Script (load-test.js)

### 7.2.2 Performance Gate Enforcement

```

1 name: Performance Testing Gate
2
3 on:
4   workflow_call:
5     inputs:
6       environment:
7         required: true
8         type: string
9       base_url:
10        required: true
11        type: string
12
13 jobs:
14   performance-test:
15     runs-on: ubuntu-latest
16     steps:
17       - uses: actions/checkout@v4
18
19       - name: Setup k6
20         run: |
21           curl -L
22             ↪ https://github.com/grafana/k6/releases/download/v0.48.0/k6-v0.48.0-1
23             ↪ | tar xz
24           sudo mv k6-*/k6 /usr/local/bin/
25
26       - name: Run Load Tests
27         run: |
28           k6 run \
29             --out json=results.json \
30             --out csv=results.csv \
31             -e BASE_URL=${{ inputs.base_url }} \
32             -e API_TOKEN=${{ secrets.API_TOKEN }} \
33             tests/performance/load-test.js
34
35       - name: Analyze Results
36         id: analyze
37         run: |
38           # Extract key metrics
39           P95=$(jq '.metrics.http_req_duration.values["p(95)"]'
40             ↪ results.json)
41           P99=$(jq '.metrics.http_req_duration.values["p(99)"]'
42             ↪ results.json)
43           ERROR_RATE=$(jq '.metrics.http_req_failed.values.rate'
44             ↪ results.json)
45           RPS=$(jq '.metrics.http_reqs.values.rate' results.json)
46
47           echo "p95_latency=$P95" >> $GITHUB_OUTPUT
48           echo "p99_latency=$P99" >> $GITHUB_OUTPUT

```

```

44     echo "error_rate=$ERROR_RATE" >> $GITHUB_OUTPUT
45     echo "requests_per_second=$RPS" >> $GITHUB_OUTPUT
46
47     # Check thresholds
48     if (( $(echo "$P95 > 500" | bc -l) )); then
49         echo "::error::P95 latency ($P95 ms) exceeds 500ms
50             ↳ threshold"
51         echo "passed=false" >> $GITHUB_OUTPUT
52         exit 1
53     fi
54
55     if (( $(echo "$ERROR_RATE > 0.01" | bc -l) )); then
56         echo "::error::Error rate ($ERROR_RATE) exceeds 1%
57             ↳ threshold"
58         echo "passed=false" >> $GITHUB_OUTPUT
59         exit 1
60     fi
61
62     echo "passed=true" >> $GITHUB_OUTPUT
63
64 - name: Upload Results to Grafana Cloud
65   run: |
66     curl -X POST \
67       -H "Authorization: Bearer ${ secrets.GRAFANA_TOKEN }" \
68       -H "Content-Type: application/json" \
69       -d @results.json \
70       ${ secrets.GRAFANA_K6_CLOUD_URL }
71
72 - name: Post Results to PR
73   if: github.event_name == 'pull_request'
74   uses: actions/github-script@v7
75   with:
76     script: |
77       const body = `## Performance Test Results
78
79       | Metric | Value | Threshold | Status |
80       |-----|-----|-----|-----|
81       | P95 Latency | ${ steps.analyze.outputs.p95_latency }ms
82       ↳ | <500ms | ${ steps.analyze.outputs.passed == 'true'
83       ↳ && 'Pass' || 'Fail' } |
84       | P99 Latency | ${ steps.analyze.outputs.p99_latency }ms
85       ↳ | <1000ms | ${ steps.analyze.outputs.passed ==
86       ↳ 'true' && 'Pass' || 'Fail' } |
87       | Error Rate | ${ steps.analyze.outputs.error_rate }% |
88       ↳ | <1% | ${ steps.analyze.outputs.passed == 'true' &&
89       ↳ 'Pass' || 'Fail' } |
90       | Throughput | ${
91       ↳ steps.analyze.outputs.requests_per_second } RPS |
92       ↳ | >100 | ${ steps.analyze.outputs.passed == 'true' &&
93       ↳ 'Pass' || 'Fail' } |`;

```

```
83
84     github.rest.issues.createComment({
85         issue_number: context.issue.number,
86         owner: context.repo.owner,
87         repo: context.repo.repo,
88         body: body
89     });
```

Listing 32: Performance Testing Gate Workflow

## 8 Full Automation on Commit

**Gates:** Full Automation on Commit

This gate ensures that the entire pipeline executes automatically without manual intervention from commit to deployment-ready artifact.

### 8.1 Gate 12: Full Automation on Commit

#### 8.1.1 Complete Pipeline Architecture

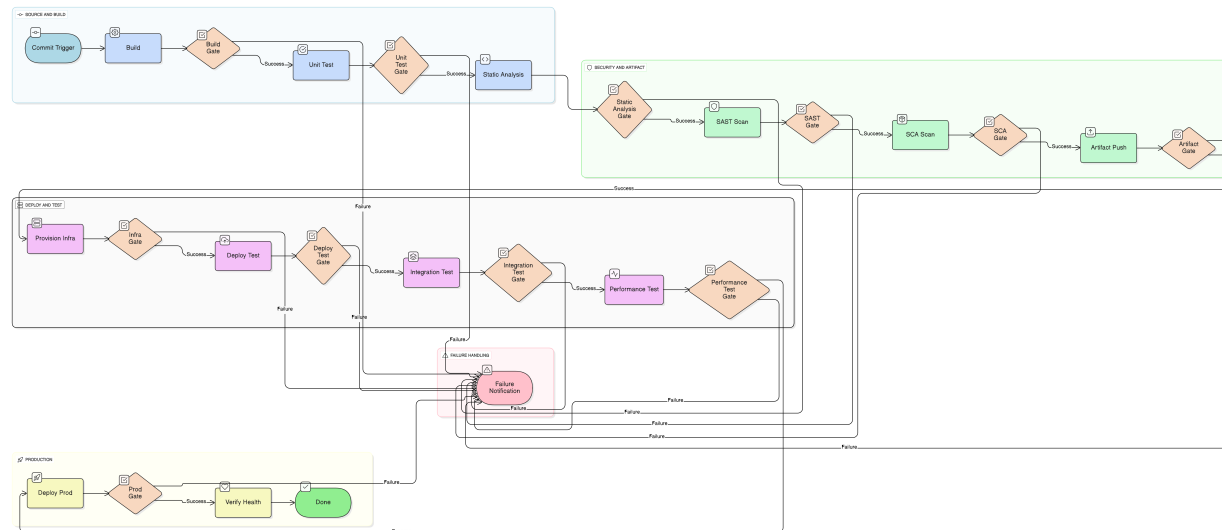


Figure 3: End-to-End CI/CD Pipeline Flow illustrating the complete path from code commit through deployment and verification.

### 8.1.2 Jenkins Declarative Pipeline

```
1 pipeline {
2     agent {
3         kubernetes {
4             yaml '''
5                 apiVersion: v1
6                 kind: Pod
7                 spec:
8                     containers:
9                         - name: node
10                          image: node:20-alpine
11                          command: [cat]
12                          tty: true
13                         - name: docker
14                          image: docker:24-dind
15                          securityContext:
16                              privileged: true
17                         - name: terraform
18                          image: hashicorp/terraform:1.6
19                          command: [cat]
20                          tty: true
21             '''
22         }
23     }
24
25     environment {
26         DOCKER_REGISTRY = 'ghcr.io/mycompany'
27         IMAGE_NAME = 'myapp'
28         SONAR_HOST = credentials('sonar-host-url')
29         SONAR_TOKEN = credentials('sonar-token')
30     }
31
32     options {
33         timeout(time: 60, unit: 'MINUTES')
34         disableConcurrentBuilds()
35         buildDiscarder(logRotator(numToKeepStr: '10'))
36         timestamps()
37     }
38
39     stages {
40         stage('Checkout') {
41             steps {
42                 checkout scm
43                 script {
44                     env.GIT_COMMIT_SHORT = sh(
45                         script: 'git rev-parse --short HEAD',
46                         returnStdout: true
47                     ).trim()
48                 }
49             }
50         }
51     }
52 }
```

```
48         env.VERSION =  
49             ↪ "${env.BUILD_NUMBER}-${env.GIT_COMMIT_SHORT}"  
50     }  
51 }  
52  
53 stage('Build') {  
54     steps {  
55         container('node') {  
56             sh 'npm ci'  
57             sh 'npm run build'  
58         }  
59     }  
60 }  
61  
62 stage('Quality Gates') {  
63     parallel {  
64         stage('Unit Tests') {  
65             steps {  
66                 container('node') {  
67                     sh 'npm run test:coverage'  
68                 }  
69             }  
70             post {  
71                 always {  
72                     junit 'coverage/junit.xml'  
73                     publishHTML([  
74                         reportDir: 'coverage/lcov-report',  
75                         reportFiles: 'index.html',  
76                         reportName: 'Coverage Report'  
77                     ])  
78                 }  
79             }  
80         }  
81  
82         stage('Static Analysis') {  
83             steps {  
84                 container('node') {  
85                     sh 'npm run lint -- --format json -o  
86                         ↪ lint-results.json'  
87                     withSonarQubeEnv('SonarQube') {  
88                         sh '''  
89                             npm run sonar -- \  
90                             -Dsonar.projectVersion=${VERSION}  
91                         '''  
92                     }  
93                 }  
94             }  
95         }  
96     }  
97 }
```

```

96         stage('Security Scan') {
97             steps {
98                 container('node') {
99                     sh 'npm audit --json > npm-audit.json ||
100                        ↪ true'
101                     sh 'npx snyk test --json >
102                        ↪ snyk-results.json || true'
103                 }
104             }
105         }
106
107         stage('Quality Gate Check') {
108             steps {
109                 timeout(time: 5, unit: 'MINUTES') {
110                     waitForQualityGate abortPipeline: true
111                 }
112             }
113         }
114
115         stage('Build Container') {
116             steps {
117                 container('docker') {
118                     sh """
119                         docker build \
120                         --build-arg VERSION=${VERSION} \
121                         --build-arg COMMIT=${GIT_COMMIT} \
122                         -t
123                         ↪ ${DOCKER_REGISTRY}/${IMAGE_NAME}:${VERSION}
124                         ↪ \
125                         -t ${DOCKER_REGISTRY}/${IMAGE_NAME}:latest \
126                         .
127                     """
128                 }
129             }
130
131             stage('Container Security Scan') {
132                 steps {
133                     container('docker') {
134                         sh """
135                             trivy image \
136                             --exit-code 1 \
137                             --severity CRITICAL,HIGH \
138                             --ignore-unfixed \
139                             ${DOCKER_REGISTRY}/${IMAGE_NAME}:${VERSION}
140                         """
141                     }
142                 }
143             }
144         }
145     }
146 }

```

```

142     }
143
144     stage('Push Artifact') {
145         when {
146             branch 'main'
147         }
148         steps {
149             container('docker') {
150                 withCredentials([usernamePassword(
151                     credentialsId: 'docker-registry',
152                     usernameVariable: 'DOCKER_USER',
153                     passwordVariable: 'DOCKER_PASS'
154                 )]) {
155                     sh """
156                         echo \${DOCKER_PASS} | docker login
157                             ↪ \${DOCKER_REGISTRY} -u \${DOCKER_USER}
158                             ↪ --password-stdin
159                         docker push
160                             ↪ \${DOCKER_REGISTRY}/${IMAGE_NAME}:${VERSION}
161                         docker push
162                             ↪ \${DOCKER_REGISTRY}/${IMAGE_NAME}:latest
163                     """
164                 }
165             }
166         }
167     }
168
169     stage('Deploy to Test') {
170         when {
171             branch 'main'
172         }
173         steps {
174             container('terraform') {
175                 dir('terraform/environments/test') {
176                     sh 'terraform init'
177                     sh "terraform apply -auto-approve
178                         ↪ -var='image_tag=${VERSION}'"
179                 }
180             }
181         }
182     }
183
184     stage('Integration Tests') {
185         when {
186             branch 'main'
187         }
188         steps {
189             container('node') {
190                 sh 'npm run test:e2e'
191             }
192         }
193     }

```

```

187     }
188 }
189
190 stage('Performance Tests') {
191     when {
192         branch 'main'
193     }
194     steps {
195         sh """
196             k6 run \
197                 -e BASE_URL=${TEST_URL} \
198                 tests/performance/load-test.js
199         """
200     }
201 }
202
203 stage('Deploy to Production') {
204     when {
205         branch 'main'
206     }
207     input {
208         message "Deploy to production?"
209         ok "Deploy"
210         parameters {
211             choice(
212                 name: 'DEPLOYMENT_STRATEGY',
213                 choices: ['canary', 'blue-green', 'rolling'],
214                 description: 'Select deployment strategy'
215             )
216         }
217     }
218     steps {
219         container('terraform') {
220             dir('terraform/environments/production') {
221                 sh 'terraform init'
222                 sh """
223                     terraform apply -auto-approve \
224                         -var='image_tag=${VERSION}' \
225                         -var='deployment_strategy=${DEPLOYMENT_STRATEGY}'
226                 """
227             }
228         }
229     }
230 }
231 }
232
233 post {
234     always {
235         cleanWs()
236     }

```

```
237     success {
238         slackSend(
239             channel: '#deployments',
240             color: 'good',
241             message: "SUCCESS: ${env.JOB_NAME} ${env.BUILD_NUMBER}"
242         )
243     }
244     failure {
245         slackSend(
246             channel: '#deployments',
247             color: 'danger',
248             message: "FAILED: ${env.JOB_NAME} ${env.BUILD_NUMBER}"
249         )
250     }
251 }
252 }
```

Listing 33: Complete Jenkins Pipeline (Jenkinsfile)

## 9 Production Release Strategy

**Gates:**

Zero Downtime Release

Feature Toggle

Production Release Strategy gates ensure that deployments to production occur safely without service interruption and with the ability to control feature exposure.

### 9.1 Gate 13: Zero Downtime Release

#### 9.1.1 Deployment Strategies Comparison

Table 8: Deployment Strategy Comparison

Strategy	Pros	Cons	Best For
Rolling	Simple, resource efficient	Slow rollback, version mixing	Stateless services
Blue/Green	Instant rollback, no mixing	2x resources required	Critical services
Canary	Gradual risk exposure	Complex traffic management	High-traffic systems
A/B Testing	User-specific testing	Requires feature flags	User experience changes
Shadow	No user impact	Complex data handling	Major refactors

### 9.1.2 Kubernetes Canary Deployment

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Rollout
3  metadata:
4    name: myapp
5  spec:
6    replicas: 10
7    revisionHistoryLimit: 3
8    selector:
9      matchLabels:
10       app: myapp
11    template:
12      metadata:
13        labels:
14          app: myapp
15      spec:
16        containers:
17          - name: myapp
18            image: ghcr.io/mycompany/myapp:latest
19            ports:
20              - containerPort: 8080
21            resources:
22              requests:
23                memory: "256Mi"
24                cpu: "250m"
25              limits:
26                memory: "512Mi"
27                cpu: "500m"
28            readinessProbe:
29              httpGet:
30                path: /health/ready
31                port: 8080
32                initialDelaySeconds: 5
33                periodSeconds: 5
34            livenessProbe:
35              httpGet:
36                path: /health/live
37                port: 8080
38                initialDelaySeconds: 15
39                periodSeconds: 10
40
41    strategy:
42      canary:
43        maxSurge: "25%"
44        maxUnavailable: 0
45
46        # Canary steps
47        steps:
48          - setWeight: 5
```

```

49     - pause: { duration: 2m }
50
51     - setWeight: 10
52     - pause: { duration: 5m }
53
54     - setWeight: 25
55     - pause: { duration: 10m }
56
57     - setWeight: 50
58     - pause: { duration: 15m }
59
60     - setWeight: 75
61     - pause: { duration: 10m }
62
63     - setWeight: 100
64
65     # Traffic routing
66     trafficRouting:
67       istio:
68         virtualService:
69           name: myapp-vsvc
70           routes:
71             - primary
72
73     # Analysis during rollout
74     analysis:
75       templates:
76         - templateName: success-rate
77         - templateName: latency
78
79       startingStep: 1 # Start analysis at step 1
80
81       args:
82         - name: service-name
83           value: myapp
84
85     ---
86     apiVersion: argoproj.io/v1alpha1
87     kind: AnalysisTemplate
88     metadata:
89       name: success-rate
90     spec:
91       args:
92         - name: service-name
93       metrics:
94         - name: success-rate
95           interval: 1m
96           count: 5
97           successCondition: result[0] >= 0.99
98           failureLimit: 3

```

```

99     provider:
100       prometheus:
101         address: http://prometheus:9090
102         query: |
103           sum(rate(
104             http_requests_total{
105               service="{{args.service-name}}",
106               status=~"2.."
107             }[5m]
108           )) /
109           sum(rate(
110             http_requests_total{
111               service="{{args.service-name}}"
112             }[5m]
113           ))
114
115 ---
116 apiVersion: argoproj.io/v1alpha1
117 kind: AnalysisTemplate
118 metadata:
119   name: latency
120 spec:
121   args:
122     - name: service-name
123   metrics:
124     - name: latency-p99
125       interval: 1m
126       count: 5
127       successCondition: result[0] < 500
128       failureLimit: 3
129       provider:
130         prometheus:
131           address: http://prometheus:9090
132           query: |
133             histogram_quantile(0.99,
134               sum(rate(
135                 http_request_duration_seconds_bucket{
136                   service="{{args.service-name}}"
137                 }[5m]
138               )) by (le)
139             ) * 1000

```

Listing 34: Argo Rollouts Canary Configuration

## 9.2 Gate 14: Feature Toggle

### 9.2.1 Feature Flag Implementation

```
1 // feature-flags.ts
2 import * as LaunchDarkly from 'launchdarkly-node-server-sdk';
3
4 const client = LaunchDarkly.init(process.env.LAUNCHDARKLY_SDK_KEY!);
5
6 export interface FeatureFlags {
7   newCheckoutFlow: boolean;
8   enhancedSearch: boolean;
9   betaFeatures: boolean;
10  maintenanceMode: boolean;
11 }
12
13 export async function getFeatureFlags(
14   userId: string,
15   userAttributes: Record<string, any>
16 ): Promise<FeatureFlags> {
17   await client.waitForInitialization();
18
19   const user: LaunchDarkly.LDUser = {
20     key: userId,
21     custom: {
22       ...userAttributes,
23       tier: userAttributes.subscriptionTier,
24       country: userAttributes.country,
25     },
26   };
27
28   return {
29     newCheckoutFlow: await client.variation(
30       'new-checkout-flow',
31       user,
32       false
33     ),
34     enhancedSearch: await client.variation(
35       'enhanced-search',
36       user,
37       false
38     ),
39     betaFeatures: await client.variation(
40       'beta-features',
41       user,
42       false
43     ),
44     maintenanceMode: await client.variation(
45       'maintenance-mode',
46       user,
47       false
```

```
48     ),
49   };
50 }
51
52 // Usage in API route
53 export async function handleCheckout(req: Request, res: Response) {
54   const flags = await getFeatureFlags(req.user.id, {
55     subscriptionTier: req.user.tier,
56     country: req.user.country,
57     signupDate: req.user.createdAt,
58   });
59
60   if (flags.maintenanceMode) {
61     return res.status(503).json({
62       error: 'Service temporarily unavailable',
63     });
64   }
65
66   if (flags.newCheckoutFlow) {
67     return newCheckoutHandler(req, res);
68   }
69
70   return legacyCheckoutHandler(req, res);
71 }
```

Listing 35: LaunchDarkly Feature Flag Integration

```
1 # feature-flags.yaml
2 feature-flags:
3   new-checkout-flow:
4     description: "New checkout experience with improved UX"
5     type: boolean
6     default: false
7     targeting:
8       rules:
9         - name: "Beta Users"
10           conditions:
11             - attribute: tier
12               operator: equals
13               value: "beta"
14             variation: true
15
16         - name: "Internal Users"
17           conditions:
18             - attribute: email
19               operator: endsWith
20               value: "@mycompany.com"
21             variation: true
22
23         - name: "Gradual Rollout"
24           conditions:
25             - attribute: key
26               operator: segmentMatch
27               value: "gradual-rollout-10-percent"
28             variation: true
29
30     fallthrough:
31       variation: false
32
33   metrics:
34     - key: checkout-conversion
35       type: conversion
36     - key: checkout-revenue
37       type: numeric
38
39   enhanced-search:
40     description: "AI-powered search with semantic understanding"
41     type: boolean
42     default: false
43     prerequisites:
44       - key: beta-features
45         variation: true
46     targeting:
47       rules:
48         - name: "Premium Users"
49           conditions:
```

```
50         - attribute: tier
51           operator: in
52           values: ["premium", "enterprise"]
53         variation: true
54     fallthrough:
55         percentage_rollout:
56             - variation: true
57               weight: 20000 # 20%
58             - variation: false
59               weight: 80000 # 80%
```

Listing 36: Feature Flag Configuration Schema

## 10 Change Management and Rollbacks

**Gates:** Automated Change Order    Automated Rollback

Change Management gates ensure that all production changes are properly documented and that the system can automatically recover from failed deployments.

### 10.1 Gate 15: Automated Change Order

#### 10.1.1 ServiceNow Integration

```

1 name: Create Change Order
2
3 on:
4   workflow_call:
5     inputs:
6       environment:
7         required: true
8         type: string
9       version:
10        required: true
11        type: string
12     outputs:
13       change_number:
14         description: "ServiceNow change number"
15         value: ${ jobs.create-change.outputs.change_number }
16
17 jobs:
18   create-change:
19     runs-on: ubuntu-latest
20     outputs:
21       change_number: ${ steps.create.outputs.change_number }
22
23   steps:
24     - name: Generate Change Details
25       id: details
26       run: |
27         # Gather deployment information
28         echo "summary=Deploy ${ inputs.version } to ${ inputs.environment }" >> $GITHUB_OUTPUT
29         echo "start_time=$(date -u +"%Y-%m-%dT%H:%M:%SZ")" >> $GITHUB_OUTPUT
30         echo "end_time=$(date -u -d "+1 hour" +"%Y-%m-%dT%H:%M:%SZ")" >> $GITHUB_OUTPUT
31
32     - name: Create ServiceNow Change
33       id: create
34       run: |

```

```

35     RESPONSE=$(curl -s -X POST \
36         "${{ secrets.SERVICENOW_INSTANCE
           ↪ }}/api/sn_chg_rest/change/standard" \
37     -H "Authorization: Bearer ${{ secrets.SERVICENOW_TOKEN }}" \
38     -H "Content-Type: application/json" \
39     -d '{
40         "short_description": "${{ steps.details.outputs.summary
           ↪ }}",
41         "description": "Automated deployment from CI/CD
           ↪ pipeline\n\nVersion: ${{ inputs.version
           ↪ }}\nEnvironment: ${{ inputs.environment }}\nCommit:
           ↪ ${{ github.sha }}\nPipeline: ${{ github.server_url
           ↪ }}/${{ github.repository }}/actions/runs/${{
           ↪ github.run_id }}",
42         "category": "Software",
43         "service_offering": "Application Services",
44         "assignment_group": "DevOps Team",
45         "requested_by": "${{ github.actor }}",
46         "start_date": "${{ steps.details.outputs.start_time }}",
47         "end_date": "${{ steps.details.outputs.end_time }}",
48         "risk": "low",
49         "impact": "3",
50         "u_change_type": "standard",
51         "u_rollback_plan": "Automated rollback via ArgoCD",
52         "cmdb_ci": "myapp-${{ inputs.environment }}"
53     }')
54
55     CHANGE_NUMBER=$(echo $RESPONSE | jq -r '.result.number')
56     echo "change_number=$CHANGE_NUMBER" >> $GITHUB_OUTPUT
57     echo "Change order created: $CHANGE_NUMBER"
58
59 - name: Attach Evidence
60   run: |
61       # Attach test results
62       curl -X POST \
63         "${{ secrets.SERVICENOW_INSTANCE
           ↪ }}/api/now/attachment/file" \
64     -H "Authorization: Bearer ${{ secrets.SERVICENOW_TOKEN }}" \
65     -H "Content-Type: multipart/form-data" \
66     -F "table_name=change_request" \
67     -F "table_sys_id=${{ steps.create.outputs.change_sys_id }}"
           ↪ \
68     -F "file=@test-results/summary.pdf"

```

Listing 37: Automated Change Order Creation

## 10.2 Gate 16: Automated Rollback

### 10.2.1 Rollback Automation with ArgoCD

```

1  apiVersion: argoproj.io/v1alpha1
2  kind: Application
3  metadata:
4    name: myapp-production
5    namespace: argocd
6    annotations:
7      notifications.argoproj.io/subscribe.on-health-degraded.slack:
          ↪ deployments
8      notifications.argoproj.io/subscribe.on-sync-failed.slack:
          ↪ deployments
9  spec:
10   project: production
11   source:
12     repoURL: https://github.com/mycompany/myapp-manifests
13     targetRevision: HEAD
14     path: environments/production
15
16   destination:
17     server: https://kubernetes.default.svc
18     namespace: production
19
20   syncPolicy:
21     automated:
22       prune: true
23       selfHeal: true
24       allowEmpty: false
25
26   syncOptions:
27     - CreateNamespace=true
28     - PrunePropagationPolicy=foreground
29     - PruneLast=true
30
31   retry:
32     limit: 3
33     backoff:
34       duration: 5s
35       factor: 2
36       maxDuration: 3m
37
38   # Rollback configuration
39   revisionHistoryLimit: 10
40
41  ---
42  apiVersion: argoproj.io/v1alpha1
43  kind: ApplicationSet
44  metadata:
45    name: myapp-rollback-monitor

```

```
46 spec:
47   generators:
48     - list:
49       elements:
50         - app: myapp-production
51
52   template:
53     metadata:
54       name: '{{app}}-rollback-monitor'
55     spec:
56       project: production
57       source:
58         repoURL: https://github.com/mycompany/rollback-automation
59         targetRevision: HEAD
60         path: monitors
61       helm:
62         parameters:
63           - name: targetApp
64             value: '{{app}}'
65           - name: healthCheckInterval
66             value: "30s"
67           - name: rollbackThreshold
68             value: "3" # Failed health checks before rollback
```

Listing 38: ArgoCD Application with Auto-Rollback

```

1 name: Automated Rollback
2
3 on:
4   workflow_dispatch:
5     inputs:
6       reason:
7         description: 'Reason for rollback'
8         required: true
9       target_version:
10        description: 'Version to rollback to (leave empty for previous)'
11        required: false
12
13    # Triggered by monitoring alerts
14    repository_dispatch:
15      types: [rollback-triggered]
16
17 jobs:
18   rollback:
19     runs-on: ubuntu-latest
20     environment: production
21
22     steps:
23       - name: Determine Rollback Target
24         id: target
25         run: |
26           if [ -n "${{ inputs.target_version }}" ]; then
27             echo "version=${{ inputs.target_version }}" >>
28               ↪ $GITHUB_OUTPUT
29           else
30             # Get previous successful deployment
31             VERSION=$(curl -s \
32               "${{ secrets.ARGOCD_SERVER
33               ↪ }}/api/v1/applications/myapp-production/revisions" \
34               -H "Authorization: Bearer ${{ secrets.ARGOCD_TOKEN }}" \
35               | jq -r '.revisions[1].revision')
36             echo "version=$VERSION" >> $GITHUB_OUTPUT
37           fi
38
39       - name: Create Rollback Change Order
40         id: change
41         run: |
42           RESPONSE=$(curl -s -X POST \
43             "${{ secrets.SERVICENOW_INSTANCE
44             ↪ }}/api/sn_chg_rest/change/emergency" \
45             -H "Authorization: Bearer ${{ secrets.SERVICENOW_TOKEN }}" \
46             -H "Content-Type: application/json" \
47             -d '{
48               "short_description": "Emergency rollback to ${{
49               ↪ steps.target.outputs.version }}"',

```

```

46         "description": "Automated rollback triggered\n\nReason:
           ↳ ${ inputs.reason ||
           ↳ github.event.client_payload.reason }}\nTarget: ${
           ↳ steps.target.outputs.version }}",
47         "justification": "${ inputs.reason ||
           ↳ github.event.client_payload.reason }}",
48         "risk": "high",
49         "impact": "2"
50     })
51     echo "change_number=$(echo $RESPONSE | jq -r
           ↳ '.result.number')" >> $GITHUB_OUTPUT
52
53 - name: Execute Rollback
54   run: |
55     # Rollback using ArgoCD
56     argocd app rollback myapp-production ${
           ↳ steps.target.outputs.version }} \
57     --server ${ secrets.ARGOC_D_SERVER }} \
58     --auth-token ${ secrets.ARGOC_D_TOKEN }}
59
60 - name: Wait for Rollback
61   run: |
62     argocd app wait myapp-production \
63     --health \
64     --timeout 600 \
65     --server ${ secrets.ARGOC_D_SERVER }} \
66     --auth-token ${ secrets.ARGOC_D_TOKEN }}
67
68 - name: Verify Rollback
69   run: |
70     # Run smoke tests
71     npm run test:smoke -- --url ${ secrets.PRODUCTION_URL }}
72
73 - name: Update Change Order
74   run: |
75     curl -X PATCH \
76     "${ secrets.SERVICENOW_INSTANCE
           ↳ }}/api/now/table/change_request/${
           ↳ steps.change.outputs.change_sys_id }}" \
77     -H "Authorization: Bearer ${ secrets.SERVICENOW_TOKEN }}" \
78     -H "Content-Type: application/json" \
79     -d '{
80       "state": "closed",
81       "close_code": "successful",
82       "close_notes": "Rollback completed successfully.
           ↳ Application restored to version ${
           ↳ steps.target.outputs.version }}"
83     },'
84
85 - name: Notify Team

```

```
86     uses: slackapi/slack-github-action@v1
87     with:
88       channel-id: 'deployments'
89       payload: |
90         {
91           "blocks": [
92             {
93               "type": "header",
94               "text": {
95                 "type": "plain_text",
96                 "text": "Rollback Completed"
97             }
98           ],
99           {
100             "type": "section",
101             "fields": [
102               {"type": "mrkdwn", "text":
103                 ↪ "*Environment:*\\nProduction"},
104               {"type": "mrkdwn", "text": "*Version:*\\n${{
105                 ↪ steps.target.outputs.version }}"},
106               {"type": "mrkdwn", "text": "*Change:*\\n${{
107                 ↪ steps.change.outputs.change_number }}"},
108               {"type": "mrkdwn", "text": "*Reason:*\\n${{
109                 ↪ inputs.reason ||
110                 ↪ github.event.client_payload.reason }}"}
```

Listing 39: Automated Rollback Workflow

## 11 Monitoring and Observability

While not explicitly one of the sixteen gates, comprehensive monitoring and observability are essential for the effective operation of the CI/CD pipeline and production systems.

### 11.1 Pipeline Observability

#### 11.1.1 Pipeline Metrics Dashboard

Table 9: Key Pipeline Metrics

Metric	Formula	Target	DORA Category
Deployment Frequency	Deploys / Time	Daily+	Throughput
Lead Time for Changes	Commit to Deploy	< 1 day	Throughput
Change Failure Rate	Failed / Total	< 15%	Stability
Mean Time to Recovery	Incident to Recovery	< 1 hour	Stability
Build Success Rate	Passed / Total	> 95%	Quality
Test Pass Rate	Passed / Total	> 99%	Quality
Security Issue Density	Issues / KLOC	< 0.1	Security

### 11.1.2 Prometheus Metrics Configuration

```
1 # prometheus-rules.yaml
2 groups:
3   - name: cicd-pipeline
4     interval: 30s
5     rules:
6       # Deployment frequency
7       - record: cicd:deployments:rate_1d
8         expr: |
9           sum(increase(
10             deployment_total{environment="production"}[1d]
11           ))
12
13       # Lead time for changes (in hours)
14       - record: cicd:lead_time:avg_1d
15         expr: |
16           avg(
17             deployment_lead_time_seconds{environment="production"}
18           ) / 3600
19
20       # Change failure rate
21       - record: cicd:change_failure_rate:ratio_7d
22         expr: |
23           sum(increase(
24             deployment_total{status="failed",
25               ↪ environment="production"}[7d]
26           )) /
27           sum(increase(
28             deployment_total{environment="production"}[7d]
29           ))
30
31       # Build duration
32       - record: cicd:build_duration:p95_1h
33         expr: |
34           histogram_quantile(0.95,
35             sum(rate(
36               build_duration_seconds_bucket[1h]
37             )) by (le, pipeline)
38           )
39
40       # Alerts
41       - alert: HighChangeFailureRate
42         expr: cicd:change_failure_rate:ratio_7d > 0.15
43         for: 1h
44         labels:
45           severity: warning
46         annotations:
47           summary: "High change failure rate detected"
```

```
47         description: "Change failure rate is {{ $value |  
           ↪ humanizePercentage }}"  
48  
49     - alert: SlowLeadTime  
50       expr: cird:lead_time:avg_1d > 24  
51       for: 2h  
52       labels:  
53         severity: warning  
54       annotations:  
55         summary: "Lead time exceeds 24 hours"  
56         description: "Average lead time is {{ $value |  
           ↪ humanizeDuration }}"
```

Listing 40: Pipeline Metrics Exporter Configuration

## 11.2 Application Observability

### 11.2.1 OpenTelemetry Integration

```

1 import { NodeSDK } from '@opentelemetry/sdk-node';
2 import { getNodeAutoInstrumentations } from
  ↳ '@opentelemetry/auto-instrumentations-node';
3 import { OTLPTraceExporter } from
  ↳ '@opentelemetry/exporter-trace-otlp-grpc';
4 import { OTLPMetricExporter } from
  ↳ '@opentelemetry/exporter-metrics-otlp-grpc';
5 import { PeriodicExportingMetricReader } from
  ↳ '@opentelemetry/sdk-metrics';
6 import { Resource } from '@opentelemetry/resources';
7 import { SemanticResourceAttributes } from
  ↳ '@opentelemetry/semantic-conventions';
8
9 const resource = new Resource({
10   [SemanticResourceAttributes.SERVICE_NAME]: process.env.SERVICE_NAME
11     ↳ || 'myapp',
12   [SemanticResourceAttributes.SERVICE_VERSION]: process.env.VERSION ||
13     ↳ 'unknown',
14   [SemanticResourceAttributes.DEPLOYMENT_ENVIRONMENT]:
15     ↳ process.env.ENVIRONMENT || 'development',
16 });
17
18 const sdk = new NodeSDK({
19   resource,
20
21   traceExporter: new OTLPTraceExporter({
22     url: process.env.OTEL_EXPORTER_OTLP_ENDPOINT ||
23       ↳ 'http://otel-collector:4317',
24   }),
25
26   metricReader: new PeriodicExportingMetricReader({
27     exporter: new OTLPMetricExporter({
28       url: process.env.OTEL_EXPORTER_OTLP_ENDPOINT ||
29         ↳ 'http://otel-collector:4317',
30     }),
31     exportIntervalMillis: 60000,
32   }),
33
34   instrumentations: [
35     getNodeAutoInstrumentations({
36       '@opentelemetry/instrumentation-fs': { enabled: false },
37       '@opentelemetry/instrumentation-http': {
38         requestHook: (span, request) => {
39           span.setAttribute('http.request.id',
40             ↳ request.headers['x-request-id']);
41         },
42       },
43     }),
44   ],
45 });

```

```
37     }),
38   ],
39 });
40
41 sdk.start();
42
43 process.on('SIGTERM', () => {
44   sdk.shutdown()
45     .then(() => console.log('Tracing terminated'))
46     .catch((error) => console.error('Error terminating tracing', error))
47     .finally(() => process.exit(0));
48 });
```

Listing 41: OpenTelemetry Configuration (tracing.ts)

## 12 Compliance and Audit

Maintaining compliance requires comprehensive audit trails and evidence collection throughout the CI/CD pipeline.

### 12.1 Audit Trail Requirements

Table 10: Audit Trail Data Points

Event Type	Required Data	Retention
Code Changes	Author, reviewer, timestamp, diff	7 years
Build Execution	Inputs, outputs, logs, duration	3 years
Test Results	Pass/fail, coverage, duration	3 years
Security Scans	Findings, severity, remediation	7 years
Deployments	Version, environment, approver	7 years
Access Changes	User, permissions, timestamp	7 years
Incidents	Timeline, impact, resolution	7 years

### 12.2 Compliance Framework Mapping

Table 11: Gate Mapping to Compliance Frameworks

Gate	SOC 2	ISO 27001	PCI DSS
Version Control	CC6.1, CC8.1	A.12.1.2	6.4.3, 6.4.5
Static Analysis	CC7.1	A.14.2.1	6.3.2
Code Coverage	CC7.1	A.14.2.8	6.3.2
Vulnerability Scan	CC6.1, CC7.1	A.12.6.1	6.5, 11.2
Open Source Scan	CC6.6	A.14.2.5	6.3.2
Change Order	CC6.1, CC8.1	A.12.1.2	6.4.5
Rollback	CC7.4	A.12.3.1	6.4.5.4

## A Tool Reference

Table 12: CI/CD Tool Reference

Category	Tool	Purpose	License
Version Control	Git	Distributed VCS	GPL-2.0
Version Control	GitHub	Git hosting, CI/CD	Commercial
Version Control	GitLab	Git hosting, CI/CD	MIT/Commercial
CI/CD	Jenkins	Automation server	MIT
CI/CD	GitHub Actions	CI/CD workflows	Commercial
CI/CD	GitLab CI	CI/CD pipelines	MIT/Commercial
CI/CD	ArgoCD	GitOps deployment	Apache-2.0
Static Analysis	SonarQube	Code quality	LGPL-3.0
Static Analysis	ESLint	JavaScript linting	MIT
Static Analysis	Semgrep	SAST scanning	LGPL-2.1
Testing	Jest	JavaScript testing	MIT
Testing	Playwright	E2E testing	Apache-2.0
Testing	k6	Load testing	AGPL-3.0
Security	Trivy	Container scanning	Apache-2.0
Security	Snyk	SCA/SAST	Commercial
Security	OWASP DC	Dependency check	Apache-2.0
Infrastructure	Terraform	IaC provisioning	MPL-2.0
Infrastructure	Packer	Image building	MPL-2.0
Containers	Docker	Containerization	Apache-2.0
Containers	Kubernetes	Orchestration	Apache-2.0
Monitoring	Prometheus	Metrics collection	Apache-2.0
Monitoring	Grafana	Visualization	AGPL-3.0
Feature Flags	LaunchDarkly	Feature management	Commercial
Feature Flags	Unleash	Feature toggles	Apache-2.0

## B Implementation Checklist

### B.1 Gate Implementation Checklist

#### Gate 1: Source Code Version Control

- ☐ Git repository configured
- ☐ Branch protection enabled
- ☐ CODEOWNERS file created
- ☐ Signed commits enforced
- ☐ Pre-commit hooks installed

#### Gate 2: Optimum Branching Strategy

- ☐ Branching strategy documented
- ☐ Branch naming conventions enforced
- ☐ PR templates created
- ☐ Review requirements configured

#### Gate 3: Static Analysis

- ☐ Linters configured for all languages
- ☐ SonarQube project created
- ☐ Quality profiles defined
- ☐ CI integration completed

#### Gate 4: 80% Code Coverage

- ☐ Test framework configured
- ☐ Coverage tool integrated
- ☐ Thresholds enforced in CI
- ☐ Coverage reporting enabled

**Gate 5: Vulnerability Scan**

- ☐ SAST tool configured
- ☐ Container scanning enabled
- ☐ IaC scanning configured
- ☐ Secret scanning enabled

**Gate 6: Open Source Scan**

- ☐ SCA tool configured
- ☐ License policy defined
- ☐ SBOM generation enabled
- ☐ Vulnerability database updated

**Gate 7: Artifact Version Control**

- ☐ Artifact repository configured
- ☐ Semantic versioning implemented
- ☐ Image signing enabled
- ☐ Retention policies defined

**Gate 8: Automatic Provision**

- ☐ IaC templates created
- ☐ State management configured
- ☐ CI/CD integration completed
- ☐ Security scanning enabled

**Gate 9: Immutable Servers**

- ☐ Container images optimized
- ☐ AMI/image building automated
- ☐ No SSH access in production
- ☐ Configuration via environment

**Gate 10: Integration Testing**

- ☐ E2E test suite created
- ☐ Test environment provisioned
- ☐ Contract testing implemented
- ☐ Test data management defined

**Gate 11: Performance Testing**

- ☐ Load test scripts created
- ☐ Performance thresholds defined
- ☐ Baseline metrics established
- ☐ Results tracking configured

**Gate 12: Full Automation on Commit**

- ☐ Pipeline fully automated
- ☐ No manual gates before staging
- ☐ Failure notifications configured
- ☐ Pipeline metrics tracked

**Gate 13: Zero Downtime Release**

- ☐ Deployment strategy selected
- ☐ Health checks configured
- ☐ Traffic shifting automated
- ☐ Canary analysis enabled

**Gate 14: Feature Toggle**

- ☐ Feature flag service configured
- ☐ SDK integrated in application
- ☐ Targeting rules defined
- ☐ Kill switches implemented

**Gate 15: Automated Change Order**

- ☐ ITSM integration configured
- ☐ Change templates created
- ☐ Evidence collection automated
- ☐ Approval workflows defined

**Gate 16: Automated Rollback**

- ☐ Rollback triggers defined
- ☐ Health check thresholds set
- ☐ Rollback automation tested
- ☐ Communication automated

## Conclusion

Implementing a CI/CD pipeline with sixteen gates requires significant investment in tooling, automation, and organizational change. However, the benefits in terms of software quality, delivery speed, security posture, and operational reliability make this investment worthwhile for organizations committed to software excellence.

## Key Success Factors

1. **Executive Sponsorship:** Leadership must champion the initiative and allocate resources
2. **Incremental Adoption:** Implement gates progressively rather than all at once
3. **Automation First:** Eliminate manual steps wherever technically feasible
4. **Metrics-Driven:** Track DORA metrics to measure improvement
5. **Continuous Improvement:** Regularly review and refine gate criteria
6. **Culture Change:** Foster a culture of shared responsibility for quality

## Next Steps

Organizations beginning this journey should:

1. Assess current CI/CD maturity against this framework
2. Identify gaps and prioritize gates based on risk and value
3. Create a phased implementation roadmap
4. Establish baseline metrics for comparison
5. Begin with foundational gates (SCM, Build, Test)
6. Progressively add security, infrastructure, and release gates
7. Continuously measure and improve