

Continuous Integration and Continuous Delivery Pipeline with 16 Gates

A Comprehensive Implementation Guide

Jordan Suber

December 3, 2025

Contents

1 Executive Summary	4
1.1 The 16 Gates Overview	4
1.2 Key Benefits	5
1.3 Target Audience	6
2 Source Code Management	7
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
3 Build and Quality Gates	17
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
4	Security Scanning	31
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
5	Artifact Management	41
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
6	Infrastructure Provisioning	47
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
7	Integration and Performance Testing	58
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
8	Full Automation on Commit	69
8.1	Gate 12: Full Automation on Commit	69
8.1.1	Complete Pipeline Architecture	69
8.1.2	Jenkins Declarative Pipeline	71
9	Production Release Strategy	77
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
10	Change Management and Rollbacks	85
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
11	Monitoring and Observability	92
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
12	Compliance and Audit	97
12.1	Audit Trail Requirements	97
12.2	Compliance Framework Mapping	97
A	Tool Reference	98
B	Implementation Checklist	99
B.1	Gate Implementation Checklist	99
Conclusion		103

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:

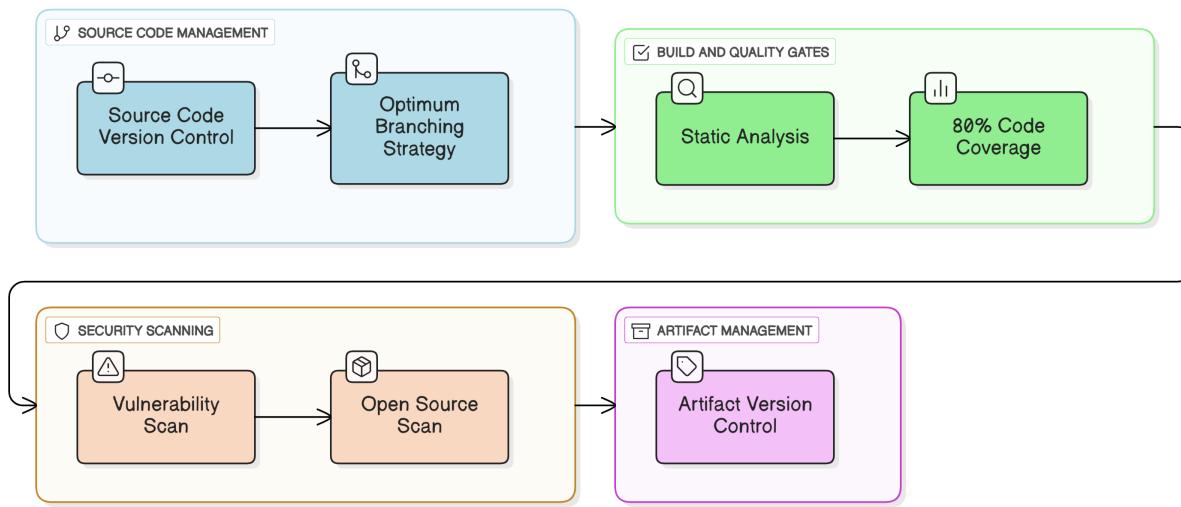


Figure 1: Software Development Pipeline Flow Chart showing the major stages of the software development lifecycle and their interactions.

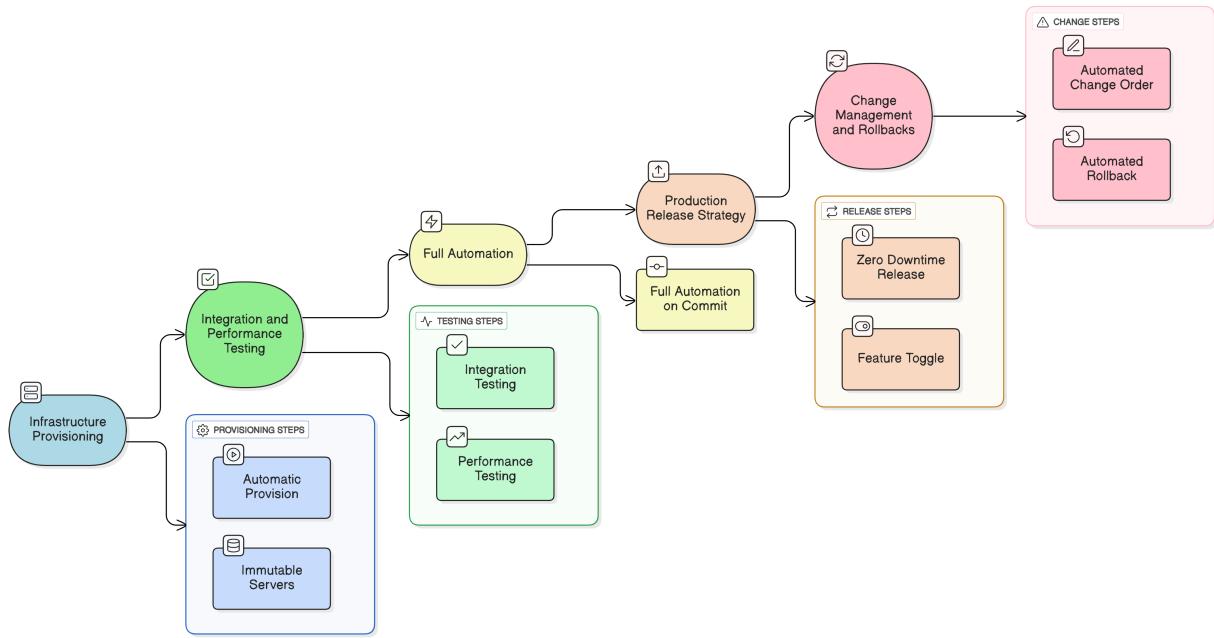


Figure 2: Infrastructure Provisioning and Release Flow highlighting automated infrastructure setup, testing, and release steps.

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

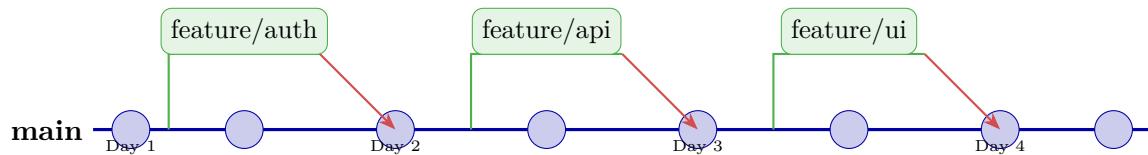
Organizations must select a branching strategy that aligns with their release cadence, team size, and deployment requirements.

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

```

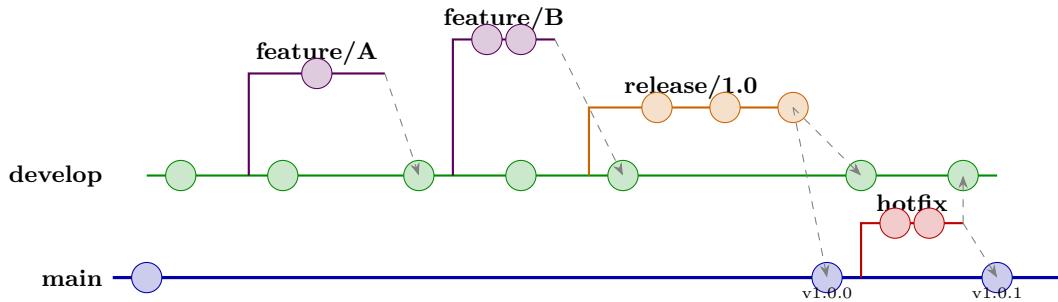
1 name: Trunk-Based CI/CD
2
3 on:
4   push:
5     branches: [main]
6   pull_request:
7     branches: [main]
8
9 jobs:
10   validate:
11     runs-on: ubuntu-latest
12     steps:
13       - uses: actions/checkout@v4
14         with:
15           fetch-depth: 0
16
17       - name: Check Branch Age
18         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)))
46   message: "Potential path traversal vulnerability"
47   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*.yml'
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 # Snyk 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: Snyk Configuration (.snyk)

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	Approved for all use
Weak Copyleft	LGPL, MPL	Approved with attribution
Strong Copyleft	GPL, AGPL	Restricted - requires review
Proprietary	Commercial licenses	Restricted - requires procurement
Unknown	No license specified	Blocked - 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

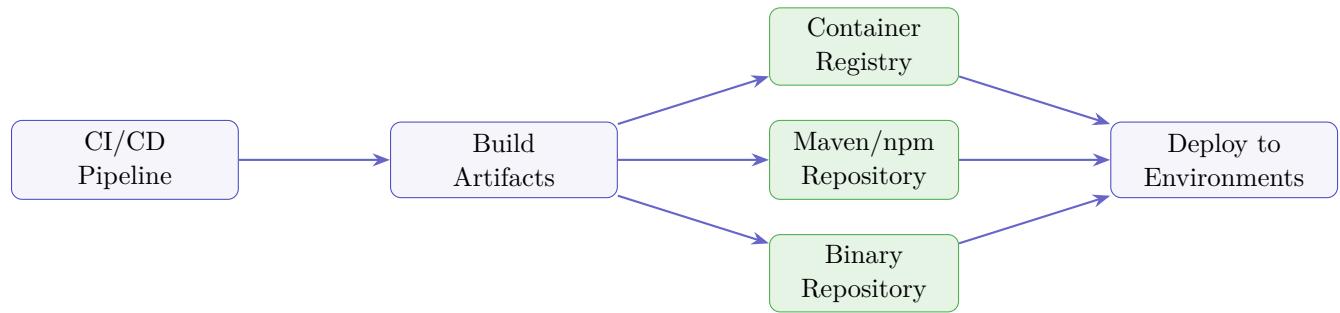
5 Artifact Management

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"}, {"type": "breaking", "release": "major"}, {"type": "docs", "release": false}, {"type": "style", "release": false}, {"type": "refactor", "release": "patch"}, {"type": "test", "release": false}, {"type": "ci", "release": false}
11            ],
12        }],
13        "@semantic-release/release-notes-generator",
14        "@semantic-release/changelog",
15        ["@semantic-release/npm", {
16            "npmPublish": true
17        }],
18        ["@semantic-release/git", {
19            "assets": ["package.json", "CHANGELOG.md"], "message": "chore(release): ${nextRelease.version}"
20        }],
21        "@semantic-release/github"
22    ],
23 }
24 }
```

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 ==
48             'refs/heads/main' }}
```

```
48         # Build timestamp
49         type=raw,value={{date 'YYYYMMDD-HHmmss'}}
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       env:
71         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             ${{ env.REGISTRY }}/{{ env.IMAGE_NAME }}@${{ needs.build.outputs.digest }}
28
29       - name: Generate SLSA Provenance
30         uses:
31           - slsa-framework/slsa-github-generator/.github/workflows/generator_content
32         with:
33           image: ${{ env.REGISTRY }}/{{ env.IMAGE_NAME }}
34           digest: ${{ needs.build.outputs.digest }}
35
36   verify:
37     needs: sign
38     runs-on: ubuntu-latest
39     steps:
40       - name: Verify image signature
41         run: |
42           cosign verify \
43             --certificate-identity-regexp ".*@mycompany.com" \
44             --certificate-oidc-issuer \
45               https://token.actions.githubusercontent.com \
${{ 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 \
${{ env.REGISTERY }}/{{ env.IMAGE_NAME }}@${{{
              ↳ 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            ${{ steps.plan.outputs.stdout }}
95            ';
96            github.rest.issues.createComment({
97              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-${var.version}-{{timestamp}}"
16       instance_type = "t3.medium"
17       region        = var.aws_region
18
19       source_ami_filter {
20           filters = {
21               name      =
22                   ↳ "ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"
23               root-device-type = "ebs"
24               virtualization-type = "hvm"
25           }
26           owners      = ["099720109477"]    # Canonical
27           most_recent = true
28       }
29
30       ssh_username = "ubuntu"
31
32       launch_block_device_mappings {
33           device_name      = "/dev/sda1"
34           volume_size      = 30
35           volume_type      = "gp3"
36           encrypted        = true
37           delete_on_termination = true
38       }
39
40       tags = {
41           Name      = "myapp-${var.version}"
42           Version   = var.version
43           BuildDate = timestamp()
44           GitCommit  = var.git_commit
45           Environment = var.environment
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 }
```

```
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)


```
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            --participant 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   thresholds: {
45     // Response time thresholds
46     'http_req_duration': ['p(95)<500', 'p(99)<1000'],
47   }
48 }
```

```

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-*/* /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 | ${{ steps.analyze.outputs.requests_per_second }} RPS |
91           ↪ >100 | ${{ steps.analyze.outputs.passed == 'true' &&
92             ↪ '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

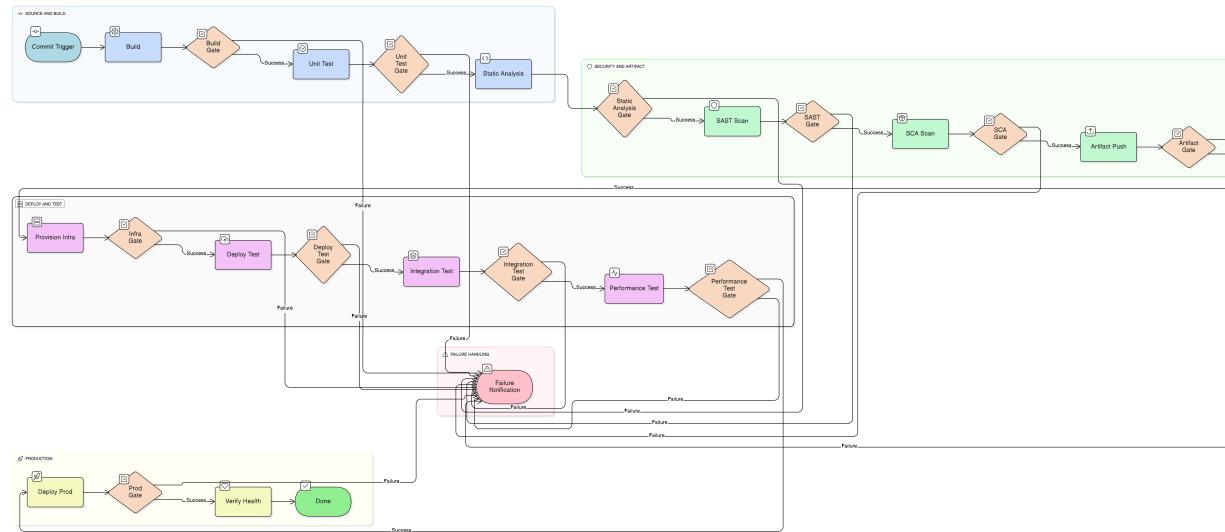


Figure 3: End-to-End CI/CD Pipeline Flow illustrating the complete path from code commit through deployment and verification.

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         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         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    stage('Deployment') {
147        steps {
148            container('aws-lambda') {
149                sh """
150                    aws lambda update-function-code \
151                        --function-name ${FUNCTION_NAME} \
152                        --zip-file fileb://lambda.zip
153
154                .
155            }
156        }
157    }
158
159    stage('Post-Deployment') {
160        steps {
161            container('node') {
162                sh """
163                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
164                        --header "Content-Type: application/json" \
165                        --data '{"files": ["*"]}'
166
167                .
168            }
169        }
170    }
171
172    stage('Final Report') {
173        steps {
174            container('node') {
175                sh """
176                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
177                        --header "Content-Type: application/json" \
178                        --data '{"files": ["*"]}'
179
180                .
181            }
182        }
183    }
184
185    stage('Success') {
186        steps {
187            container('node') {
188                sh """
189                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
190                        --header "Content-Type: application/json" \
191                        --data '{"files": ["*"]}'
192
193                .
194            }
195        }
196    }
197
198    stage('Failure') {
199        steps {
200            container('node') {
201                sh """
202                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
203                        --header "Content-Type: application/json" \
204                        --data '{"files": ["*"]}'
205
206                .
207            }
208        }
209    }
210
211    stage('Cleanup') {
212        steps {
213            container('node') {
214                sh """
215                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
216                        --header "Content-Type: application/json" \
217                        --data '{"files": ["*"]}'
218
219                .
220            }
221        }
222    }
223
224    stage('Final Report') {
225        steps {
226            container('node') {
227                sh """
228                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
229                        --header "Content-Type: application/json" \
230                        --data '{"files": ["*"]}'
231
232                .
233            }
234        }
235    }
236
237    stage('Success') {
238        steps {
239            container('node') {
240                sh """
241                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
242                        --header "Content-Type: application/json" \
243                        --data '{"files": ["*"]}'
244
245                .
246            }
247        }
248    }
249
250    stage('Failure') {
251        steps {
252            container('node') {
253                sh """
254                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
255                        --header "Content-Type: application/json" \
256                        --data '{"files": ["*"]}'
257
258                .
259            }
260        }
261    }
262
263    stage('Cleanup') {
264        steps {
265            container('node') {
266                sh """
267                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
268                        --header "Content-Type: application/json" \
269                        --data '{"files": ["*"]}'
270
271                .
272            }
273        }
274    }
275
276    stage('Final Report') {
277        steps {
278            container('node') {
279                sh """
280                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
281                        --header "Content-Type: application/json" \
282                        --data '{"files": ["*"]}'
283
284                .
285            }
286        }
287    }
288
289    stage('Success') {
290        steps {
291            container('node') {
292                sh """
293                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
294                        --header "Content-Type: application/json" \
295                        --data '{"files": ["*"]}'
296
297                .
298            }
299        }
300    }
301
302    stage('Failure') {
303        steps {
304            container('node') {
305                sh """
306                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
307                        --header "Content-Type: application/json" \
308                        --data '{"files": ["*"]}'
309
310                .
311            }
312        }
313    }
314
315    stage('Cleanup') {
316        steps {
317            container('node') {
318                sh """
319                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
320                        --header "Content-Type: application/json" \
321                        --data '{"files": ["*"]}'
322
323                .
324            }
325        }
326    }
327
328    stage('Final Report') {
329        steps {
330            container('node') {
331                sh """
332                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
333                        --header "Content-Type: application/json" \
334                        --data '{"files": ["*"]}'
335
336                .
337            }
338        }
339    }
340
341    stage('Success') {
342        steps {
343            container('node') {
344                sh """
345                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
346                        --header "Content-Type: application/json" \
347                        --data '{"files": ["*"]}'
348
349                .
350            }
351        }
352    }
353
354    stage('Failure') {
355        steps {
356            container('node') {
357                sh """
358                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
359                        --header "Content-Type: application/json" \
360                        --data '{"files": ["*"]}'
361
362                .
363            }
364        }
365    }
366
367    stage('Cleanup') {
368        steps {
369            container('node') {
370                sh """
371                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
372                        --header "Content-Type: application/json" \
373                        --data '{"files": ["*"]}'
374
375                .
376            }
377        }
378    }
379
380    stage('Final Report') {
381        steps {
382            container('node') {
383                sh """
384                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
385                        --header "Content-Type: application/json" \
386                        --data '{"files": ["*"]}'
387
388                .
389            }
390        }
391    }
392
393    stage('Success') {
394        steps {
395            container('node') {
396                sh """
397                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
398                        --header "Content-Type: application/json" \
399                        --data '{"files": ["*"]}'
400
401                .
402            }
403        }
404    }
405
406    stage('Failure') {
407        steps {
408            container('node') {
409                sh """
410                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
411                        --header "Content-Type: application/json" \
412                        --data '{"files": ["*"]}'
413
414                .
415            }
416        }
417    }
418
419    stage('Cleanup') {
420        steps {
421            container('node') {
422                sh """
423                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
424                        --header "Content-Type: application/json" \
425                        --data '{"files": ["*"]}'
426
427                .
428            }
429        }
430    }
431
432    stage('Final Report') {
433        steps {
434            container('node') {
435                sh """
436                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
437                        --header "Content-Type: application/json" \
438                        --data '{"files": ["*"]}'
439
440                .
441            }
442        }
443    }
444
445    stage('Success') {
446        steps {
447            container('node') {
448                sh """
449                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
450                        --header "Content-Type: application/json" \
451                        --data '{"files": ["*"]}'
452
453                .
454            }
455        }
456    }
457
458    stage('Failure') {
459        steps {
460            container('node') {
461                sh """
462                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
463                        --header "Content-Type: application/json" \
464                        --data '{"files": ["*"]}'
465
466                .
467            }
468        }
469    }
470
471    stage('Cleanup') {
472        steps {
473            container('node') {
474                sh """
475                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
476                        --header "Content-Type: application/json" \
477                        --data '{"files": ["*"]}'
478
479                .
480            }
481        }
482    }
483
484    stage('Final Report') {
485        steps {
486            container('node') {
487                sh """
488                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
489                        --header "Content-Type: application/json" \
490                        --data '{"files": ["*"]}'
491
492                .
493            }
494        }
495    }
496
497    stage('Success') {
498        steps {
499            container('node') {
500                sh """
501                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
502                        --header "Content-Type: application/json" \
503                        --data '{"files": ["*"]}'
504
505                .
506            }
507        }
508    }
509
510    stage('Failure') {
511        steps {
512            container('node') {
513                sh """
514                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
515                        --header "Content-Type: application/json" \
516                        --data '{"files": ["*"]}'
517
518                .
519            }
520        }
521    }
522
523    stage('Cleanup') {
524        steps {
525            container('node') {
526                sh """
527                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
528                        --header "Content-Type: application/json" \
529                        --data '{"files": ["*"]}'
530
531                .
532            }
533        }
534    }
535
536    stage('Final Report') {
537        steps {
538            container('node') {
539                sh """
540                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
541                        --header "Content-Type: application/json" \
542                        --data '{"files": ["*"]}'
543
544                .
545            }
546        }
547    }
548
549    stage('Success') {
550        steps {
551            container('node') {
552                sh """
553                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
554                        --header "Content-Type: application/json" \
555                        --data '{"files": ["*"]}'
556
557                .
558            }
559        }
560    }
561
562    stage('Failure') {
563        steps {
564            container('node') {
565                sh """
566                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
567                        --header "Content-Type: application/json" \
568                        --data '{"files": ["*"]}'
569
570                .
571            }
572        }
573    }
574
575    stage('Cleanup') {
576        steps {
577            container('node') {
578                sh """
579                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
580                        --header "Content-Type: application/json" \
581                        --data '{"files": ["*"]}'
582
583                .
584            }
585        }
586    }
587
588    stage('Final Report') {
589        steps {
590            container('node') {
591                sh """
592                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
593                        --header "Content-Type: application/json" \
594                        --data '{"files": ["*"]}'
595
596                .
597            }
598        }
599    }
600
601    stage('Success') {
602        steps {
603            container('node') {
604                sh """
605                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
606                        --header "Content-Type: application/json" \
607                        --data '{"files": ["*"]}'
608
609                .
610            }
611        }
612    }
613
614    stage('Failure') {
615        steps {
616            container('node') {
617                sh """
618                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
619                        --header "Content-Type: application/json" \
620                        --data '{"files": ["*"]}'
621
622                .
623            }
624        }
625    }
626
627    stage('Cleanup') {
628        steps {
629            container('node') {
630                sh """
631                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
632                        --header "Content-Type: application/json" \
633                        --data '{"files": ["*"]}'
634
635                .
636            }
637        }
638    }
639
640    stage('Final Report') {
641        steps {
642            container('node') {
643                sh """
644                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
645                        --header "Content-Type: application/json" \
646                        --data '{"files": ["*"]}'
647
648                .
649            }
650        }
651    }
652
653    stage('Success') {
654        steps {
655            container('node') {
656                sh """
657                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
658                        --header "Content-Type: application/json" \
659                        --data '{"files": ["*"]}'
660
661                .
662            }
663        }
664    }
665
666    stage('Failure') {
667        steps {
668            container('node') {
669                sh """
670                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
671                        --header "Content-Type: application/json" \
672                        --data '{"files": ["*"]}'
673
674                .
675            }
676        }
677    }
678
679    stage('Cleanup') {
680        steps {
681            container('node') {
682                sh """
683                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
684                        --header "Content-Type: application/json" \
685                        --data '{"files": ["*"]}'
686
687                .
688            }
689        }
690    }
691
692    stage('Final Report') {
693        steps {
694            container('node') {
695                sh """
696                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
697                        --header "Content-Type: application/json" \
698                        --data '{"files": ["*"]}'
699
700                .
701            }
702        }
703    }
704
705    stage('Success') {
706        steps {
707            container('node') {
708                sh """
709                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
710                        --header "Content-Type: application/json" \
711                        --data '{"files": ["*"]}'
712
713                .
714            }
715        }
716    }
717
718    stage('Failure') {
719        steps {
720            container('node') {
721                sh """
722                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
723                        --header "Content-Type: application/json" \
724                        --data '{"files": ["*"]}'
725
726                .
727            }
728        }
729    }
730
731    stage('Cleanup') {
732        steps {
733            container('node') {
734                sh """
735                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
736                        --header "Content-Type: application/json" \
737                        --data '{"files": ["*"]}'
738
739                .
740            }
741        }
742    }
743
744    stage('Final Report') {
745        steps {
746            container('node') {
747                sh """
748                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
749                        --header "Content-Type: application/json" \
750                        --data '{"files": ["*"]}'
751
752                .
753            }
754        }
755    }
756
757    stage('Success') {
758        steps {
759            container('node') {
760                sh """
761                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
762                        --header "Content-Type: application/json" \
763                        --data '{"files": ["*"]}'
764
765                .
766            }
767        }
768    }
769
770    stage('Failure') {
771        steps {
772            container('node') {
773                sh """
774                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
775                        --header "Content-Type: application/json" \
776                        --data '{"files": ["*"]}'
777
778                .
779            }
780        }
781    }
782
783    stage('Cleanup') {
784        steps {
785            container('node') {
786                sh """
787                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
788                        --header "Content-Type: application/json" \
789                        --data '{"files": ["*"]}'
790
791                .
792            }
793        }
794    }
795
796    stage('Final Report') {
797        steps {
798            container('node') {
799                sh """
800                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
801                        --header "Content-Type: application/json" \
802                        --data '{"files": ["*"]}'
803
804                .
805            }
806        }
807    }
808
809    stage('Success') {
810        steps {
811            container('node') {
812                sh """
813                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
814                        --header "Content-Type: application/json" \
815                        --data '{"files": ["*"]}'
816
817                .
818            }
819        }
820    }
821
822    stage('Failure') {
823        steps {
824            container('node') {
825                sh """
826                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
827                        --header "Content-Type: application/json" \
828                        --data '{"files": ["*"]}'
829
830                .
831            }
832        }
833    }
834
835    stage('Cleanup') {
836        steps {
837            container('node') {
838                sh """
839                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
840                        --header "Content-Type: application/json" \
841                        --data '{"files": ["*"]}'
842
843                .
844            }
845        }
846    }
847
848    stage('Final Report') {
849        steps {
850            container('node') {
851                sh """
852                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
853                        --header "Content-Type: application/json" \
854                        --data '{"files": ["*"]}'
855
856                .
857            }
858        }
859    }
860
861    stage('Success') {
862        steps {
863            container('node') {
864                sh """
865                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
866                        --header "Content-Type: application/json" \
867                        --data '{"files": ["*"]}'
868
869                .
870            }
871        }
872    }
873
874    stage('Failure') {
875        steps {
876            container('node') {
877                sh """
878                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
879                        --header "Content-Type: application/json" \
880                        --data '{"files": ["*"]}'
881
882                .
883            }
884        }
885    }
886
887    stage('Cleanup') {
888        steps {
889            container('node') {
890                sh """
891                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
892                        --header "Content-Type: application/json" \
893                        --data '{"files": ["*"]}'
894
895                .
896            }
897        }
898    }
899
900    stage('Final Report') {
901        steps {
902            container('node') {
903                sh """
904                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
905                        --header "Content-Type: application/json" \
906                        --data '{"files": ["*"]}'
907
908                .
909            }
910        }
911    }
912
913    stage('Success') {
914        steps {
915            container('node') {
916                sh """
917                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
918                        --header "Content-Type: application/json" \
919                        --data '{"files": ["*"]}'
920
921                .
922            }
923        }
924    }
925
926    stage('Failure') {
927        steps {
928            container('node') {
929                sh """
930                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
931                        --header "Content-Type: application/json" \
932                        --data '{"files": ["*"]}'
933
934                .
935            }
936        }
937    }
938
939    stage('Cleanup') {
940        steps {
941            container('node') {
942                sh """
943                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
944                        --header "Content-Type: application/json" \
945                        --data '{"files": ["*"]}'
946
947                .
948            }
949        }
950    }
951
952    stage('Final Report') {
953        steps {
954            container('node') {
955                sh """
956                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
957                        --header "Content-Type: application/json" \
958                        --data '{"files": ["*"]}'
959
960                .
961            }
962        }
963    }
964
965    stage('Success') {
966        steps {
967            container('node') {
968                sh """
969                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
970                        --header "Content-Type: application/json" \
971                        --data '{"files": ["*"]}'
972
973                .
974            }
975        }
976    }
977
978    stage('Failure') {
979        steps {
980            container('node') {
981                sh """
982                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
983                        --header "Content-Type: application/json" \
984                        --data '{"files": ["*"]}'
985
986                .
987            }
988        }
989    }
990
991    stage('Cleanup') {
992        steps {
993            container('node') {
994                sh """
995                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
996                        --header "Content-Type: application/json" \
997                        --data '{"files": ["*"]}'
998
999                .
1000            }
1001        }
1002    }
1003
1004    stage('Final Report') {
1005        steps {
1006            container('node') {
1007                sh """
1008                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1009                        --header "Content-Type: application/json" \
1010                        --data '{"files": ["*"]}'
1011
1012                .
1013            }
1014        }
1015    }
1016
1017    stage('Success') {
1018        steps {
1019            container('node') {
1020                sh """
1021                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1022                        --header "Content-Type: application/json" \
1023                        --data '{"files": ["*"]}'
1024
1025                .
1026            }
1027        }
1028    }
1029
1030    stage('Failure') {
1031        steps {
1032            container('node') {
1033                sh """
1034                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1035                        --header "Content-Type: application/json" \
1036                        --data '{"files": ["*"]}'
1037
1038                .
1039            }
1040        }
1041    }
1042
1043    stage('Cleanup') {
1044        steps {
1045            container('node') {
1046                sh """
1047                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1048                        --header "Content-Type: application/json" \
1049                        --data '{"files": ["*"]}'
1050
1051                .
1052            }
1053        }
1054    }
1055
1056    stage('Final Report') {
1057        steps {
1058            container('node') {
1059                sh """
1060                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1061                        --header "Content-Type: application/json" \
1062                        --data '{"files": ["*"]}'
1063
1064                .
1065            }
1066        }
1067    }
1068
1069    stage('Success') {
1070        steps {
1071            container('node') {
1072                sh """
1073                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1074                        --header "Content-Type: application/json" \
1075                        --data '{"files": ["*"]}'
1076
1077                .
1078            }
1079        }
1080    }
1081
1082    stage('Failure') {
1083        steps {
1084            container('node') {
1085                sh """
1086                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1087                        --header "Content-Type: application/json" \
1088                        --data '{"files": ["*"]}'
1089
1090                .
1091            }
1092        }
1093    }
1094
1095    stage('Cleanup') {
1096        steps {
1097            container('node') {
1098                sh """
1099                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1100                        --header "Content-Type: application/json" \
1101                        --data '{"files": ["*"]}'
1102
1103                .
1104            }
1105        }
1106    }
1107
1108    stage('Final Report') {
1109        steps {
1110            container('node') {
1111                sh """
1112                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1113                        --header "Content-Type: application/json" \
1114                        --data '{"files": ["*"]}'
1115
1116                .
1117            }
1118        }
1119    }
1120
1121    stage('Success') {
1122        steps {
1123            container('node') {
1124                sh """
1125                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1126                        --header "Content-Type: application/json" \
1127                        --data '{"files": ["*"]}'
1128
1129                .
1130            }
1131        }
1132    }
1133
1134    stage('Failure') {
1135        steps {
1136            container('node') {
1137                sh """
1138                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1139                        --header "Content-Type: application/json" \
1140                        --data '{"files": ["*"]}'
1141
1142                .
1143            }
1144        }
1145    }
1146
1147    stage('Cleanup') {
1148        steps {
1149            container('node') {
1150                sh """
1151                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1152                        --header "Content-Type: application/json" \
1153                        --data '{"files": ["*"]}'
1154
1155                .
1156            }
1157        }
1158    }
1159
1160    stage('Final Report') {
1161        steps {
1162            container('node') {
1163                sh """
1164                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1165                        --header "Content-Type: application/json" \
1166                        --data '{"files": ["*"]}'
1167
1168                .
1169            }
1170        }
1171    }
1172
1173    stage('Success') {
1174        steps {
1175            container('node') {
1176                sh """
1177                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1178                        --header "Content-Type: application/json" \
1179                        --data '{"files": ["*"]}'
1180
1181                .
1182            }
1183        }
1184    }
1185
1186    stage('Failure') {
1187        steps {
1188            container('node') {
1189                sh """
1190                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1191                        --header "Content-Type: application/json" \
1192                        --data '{"files": ["*"]}'
1193
1194                .
1195            }
1196        }
1197    }
1198
1199    stage('Cleanup') {
1200        steps {
1201            container('node') {
1202                sh """
1203                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1204                        --header "Content-Type: application/json" \
1205                        --data '{"files": ["*"]}'
1206
1207                .
1208            }
1209        }
1210    }
1211
1212    stage('Final Report') {
1213        steps {
1214            container('node') {
1215                sh """
1216                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1217                        --header "Content-Type: application/json" \
1218                        --data '{"files": ["*"]}'
1219
1220                .
1221            }
1222        }
1223    }
1224
1225    stage('Success') {
1226        steps {
1227            container('node') {
1228                sh """
1229                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1230                        --header "Content-Type: application/json" \
1231                        --data '{"files": ["*"]}'
1232
1233                .
1234            }
1235        }
1236    }
1237
1238    stage('Failure') {
1239        steps {
1240            container('node') {
1241                sh """
1242                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1243                        --header "Content-Type: application/json" \
1244                        --data '{"files": ["*"]}'
1245
1246                .
1247            }
1248        }
1249    }
1250
1251    stage('Cleanup') {
1252        steps {
1253            container('node') {
1254                sh """
1255                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1256                        --header "Content-Type: application/json" \
1257                        --data '{"files": ["*"]}'
1258
1259                .
1260            }
1261        }
1262    }
1263
1264    stage('Final Report') {
1265        steps {
1266            container('node') {
1267                sh """
1268                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1269                        --header "Content-Type: application/json" \
1270                        --data '{"files": ["*"]}'
1271
1272                .
1273            }
1274        }
1275    }
1276
1277    stage('Success') {
1278        steps {
1279            container('node') {
1280                sh """
1281                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1282                        --header "Content-Type: application/json" \
1283                        --data '{"files": ["*"]}'
1284
1285                .
1286            }
1287        }
1288    }
1289
1290    stage('Failure') {
1291        steps {
1292            container('node') {
1293                sh """
1294                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1295                        --header "Content-Type: application/json" \
1296                        --data '{"files": ["*"]}'
1297
1298                .
1299            }
1300        }
1301    }
1302
1303    stage('Cleanup') {
1304        steps {
1305            container('node') {
1306                sh """
1307                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1308                        --header "Content-Type: application/json" \
1309                        --data '{"files": ["*"]}'
1310
1311                .
1312            }
1313        }
1314    }
1315
1316    stage('Final Report') {
1317        steps {
1318            container('node') {
1319                sh """
1320                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1321                        --header "Content-Type: application/json" \
1322                        --data '{"files": ["*"]}'
1323
1324                .
1325            }
1326        }
1327    }
1328
1329    stage('Success') {
1330        steps {
1331            container('node') {
1332                sh """
1333                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1334                        --header "Content-Type: application/json" \
1335                        --data '{"files": ["*"]}'
1336
1337                .
1338            }
1339        }
1340    }
1341
1342    stage('Failure') {
1343        steps {
1344            container('node') {
1345                sh """
1346                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1347                        --header "Content-Type: application/json" \
1348                        --data '{"files": ["*"]}'
1349
1350                .
1351            }
1352        }
1353    }
1354
1355    stage('Cleanup') {
1356        steps {
1357            container('node') {
1358                sh """
1359                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1360                        --header "Content-Type: application/json" \
1361                        --data '{"files": ["*"]}'
1362
1363                .
1364            }
1365        }
1366    }
1367
1368    stage('Final Report') {
1369        steps {
1370            container('node') {
1371                sh """
1372                    curl -X POST https://api.cloudflare.com/client/v4/zones/${ZONE_ID}/purge_cache \
1373                        --header "Content-Type: application/json"
```

```
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     }
194 }
```

```
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     post {
233         always {
234             cleanWs()
235         }
236     }
237 }
```

```
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 }
```

```
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
37              ↪ }}/api/sn_chg_rest/change/standard" \
38          -H "Authorization: Bearer ${{ secrets.SERVICENOW_TOKEN }}" \
39          -H "Content-Type: application/json" \
40          -d '{
41              "short_description": "${{ steps.details.outputs.summary
42                  ↪ }}",
43              "description": "Automated deployment from CI/CD
44                  ↪ pipeline\n\nVersion: ${{ inputs.version
45                  ↪ }}\nEnvironment: ${{ inputs.environment }}\nCommit:
46                  ↪ ${{ github.sha }}\nPipeline: ${{ github.server_url
47                  ↪ }}/${{ github.repository }}/actions/runs/${{ 
48                  ↪ github.run_id }}",
49              "category": "Software",
50              "service_offering": "Application Services",
51              "assignment_group": "DevOps Team",
52              "requested_by": "${{ github.actor }}",
53              "start_date": "${{ steps.details.outputs.start_time }}",
54              "end_date": "${{ steps.details.outputs.end_time }}",
55              "risk": "low",
56              "impact": "3",
57              "u_change_type": "standard",
58              "u_rollback_plan": "Automated rollback via ArgoCD",
59              "cmdb_ci": "myapp-${{ inputs.environment }}"
60          }')
61
62
63      CHANGE_NUMBER=$(echo $RESPONSE | jq -r '.result.number')
64      echo "change_number=$CHANGE_NUMBER" >> $GITHUB_OUTPUT
65      echo "Change order created: $CHANGE_NUMBER"
66
67
68      - name: Attach Evidence
69      run: |
70          # Attach test results
71          curl -X POST \
72              "${{ secrets.SERVICENOW_INSTANCE
73                  ↪ }}/api/now/attachment/file" \
74              -H "Authorization: Bearer ${{ secrets.SERVICENOW_TOKEN }}" \
75              -H "Content-Type: multipart/form-data" \
76              -F "table_name=change_request" \
77              -F "table_sys_id=${{ steps.create.outputs.change_sys_id }}" \
78                  ↪ \
79              -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:
8       ↢ deployments
9     notifications.argoproj.io/subscribe.on-sync-failed.slack:
10       ↢ deployments
11 spec:
12   project: production
13   source:
14     repoURL: https://github.com/mycompany/myapp-manifests
15     targetRevision: HEAD
16     path: environments/production
17
18   destination:
19     server: https://kubernetes.default.svc
20     namespace: production
21
22   syncPolicy:
23     automated:
24       prune: true
25       selfHeal: true
26       allowEmpty: false
27
28   syncOptions:
29     - CreateNamespace=true
30     - PrunePropagationPolicy=foreground
31     - PruneLast=true
32
33   retry:
34     limit: 3
35     backoff:
36       duration: 5s
37       factor: 2
38       maxDuration: 3m
39
40
41   # Rollback configuration
42   revisionHistoryLimit: 10
43
44 ---
45   apiVersion: argoproj.io/v1alpha1
46   kind: ApplicationSet
47   metadata:
48     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:
47           ↪ ${{ inputs.reason ||
48             github.event.client_payload.reason }}\nTarget: ${{
49               ↪ steps.target.outputs.version }}",
50   "justification": "${{ inputs.reason ||
51     ↪ github.event.client_payload.reason }}",
52   "risk": "high",
53   "impact": "2"
54 )
55 echo "change_number=$(echo $RESPONSE | jq -r
56   ↪ '.result.number')" >> $GITHUB_OUTPUT
57
58 - name: Execute Rollback
59   run: |
60     # Rollback using ArgoCD
61     argo cd app rollback myapp-production ${{
62       ↪ steps.target.outputs.version }} \
63       --server ${{ secrets.ARGOCD_SERVER }} \
64       --auth-token ${{ secrets.ARGOCD_TOKEN }} \
65
66 - name: Wait for Rollback
67   run: |
68     argo cd app wait myapp-production \
69       --health \
70       --timeout 600 \
71       --server ${{ secrets.ARGOCD_SERVER }} \
72       --auth-token ${{ secrets.ARGOCD_TOKEN }} \
73
74 - name: Verify Rollback
75   run: |
76     # Run smoke tests
77     npm run test:smoke -- --url ${{ secrets.PRODUCTION_URL }} \
78
79 - name: Update Change Order
80   run: |
81     curl -X PATCH \
82       "${{ secrets.SERVICENOW_INSTANCE
83         ↪ }}/api/now/table/change_request/${{
84           ↪ steps.change.outputs.change_sys_id }}" \
85       -H "Authorization: Bearer ${{ secrets.SERVICENOW_TOKEN }}" \
86       -H "Content-Type: application/json" \
87       -d '{
88         "state": "closed",
89         "close_code": "successful",
90         "close_notes": "Rollback completed successfully.
91           ↪ Application restored to version ${{
92             ↪ steps.target.outputs.version }}"
93       }'
94
95 - 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 |  
48         ↪ humanizePercentage }}"  
49  
50     - alert: SlowLeadTime  
51         expr: cicd:lead_time:avg_1d > 24  
52         for: 2h  
53         labels:  
54             severity: warning  
55         annotations:  
56             summary: "Lead time exceeds 24 hours"  
57             description: "Average lead time is {{ $value |  
58                 ↪ 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
3   '@opentelemetry/auto-instrumentations-node';
4 import { OTLPTraceExporter } from
5   '@opentelemetry/exporter-trace-otlp-grpc';
6 import { OTLPMetricExporter } from
7   '@opentelemetry/exporter-metrics-otlp-grpc';
8 import { PeriodicExportingMetricReader } from
9   '@opentelemetry/sdk-metrics';
10 import { Resource } from '@opentelemetry/resources';
11 import { SemanticResourceAttributes } from
12   '@opentelemetry/semantic-conventions';
13
14 const resource = new Resource({
15   [SemanticResourceAttributes.SERVICE_NAME]: process.env.SERVICE_NAME
16     || 'myapp',
17   [SemanticResourceAttributes.SERVICE_VERSION]: process.env.VERSION ||
18     'unknown',
19   [SemanticResourceAttributes.DEPLOYMENT_ENVIRONMENT]:
20     process.env.ENVIRONMENT || 'development',
21 });
22
23 const sdk = new NodeSDK({
24   resource,
25
26   traceExporter: new OTLPTraceExporter({
27     url: process.env.OTEL_EXPORTER_OTLP_ENDPOINT ||
28       'http://otel-collector:4317',
29   }),
30
31   metricReader: new PeriodicExportingMetricReader({
32     exporter: new OTLPMetricExporter({
33       url: process.env.OTEL_EXPORTER_OTLP_ENDPOINT ||
34         'http://otel-collector:4317',
35     }),
36     exportIntervalMillis: 60000,
37   }),
38
39   instrumentations: [
40     getNodeAutoInstrumentations({
41       '@opentelemetry/instrumentation-fs': { enabled: false },
42       '@opentelemetry/instrumentation-http': {
43         requestHook: (span, request) => {
44           span.setAttribute('http.request.id',
45             request.headers['x-request-id']);
46         },
47       },
48     },
49   ],
50
51 });

```

```
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