RELEASE MANAGEMENT PROCESS PROCEDURE

Document Reference: QMS-DEV-PRO-004

Version: 1.0

Classification: RESTRICTED - INTERNAL ENGINEERING ONLY

Role	Name & Signature	Position	Date
Prepared by			
Reviewed by			
Approved by			

TABLE OF CONTENTS

1. AMENDMENT RECORD

Reviewed at minimum annually or following a significant process failure, tooling change, or audit finding. All revisions are subject to the same approval requirements as the original document.

Date	Issue	Rev	Page	Subject	Revised By	Approved By

2. EXECUTIVE SUMMARY

This procedure defines the organisation's release management process: the controls, gates, roles, and tooling required to move a software change from a verified build to a production environment in a controlled and auditable manner.

It applies to all release types: feature releases, patch releases, infrastructure changes, and emergency fixes. It is technology-agnostic and is designed to work whether the organisation runs a quarterly release train or deploys to production multiple times per day.

This procedure establishes the mandatory quality and compliance architecture governing all release activity. Individual product teams shall map these requirements into their localized service engineering runbooks.

The procedure covers five areas:

- Release classification and planning: categorising changes, confirming readiness criteria, and scheduling deployments
- Environment promotion and testing strategy: the test phases required before each environment transition
- Deployment execution: pipeline mechanics, rollout strategy, and pre/post-deployment verification
- Rollback and emergency release: explicit procedures for failure recovery and out-of-band deployments
- Release metrics and continuous improvement: observable, tool-sourced measures of release health

Performance is tracked using metrics that CI/CD platforms, monitoring systems, and incident trackers already produce. Release frequency and deployment success rate are primary indicators. Change failure rate and time to restore are the failure-mode counterparts. All four are DORA core metrics and are collected as a natural output of the toolchain defined in Section 4.2.

3. GENERAL

3.1 Scope

This procedure governs all changes promoted to a shared environment (development integration, QA, staging, pre-production, or production). It covers:

- Application feature releases and minor updates
- Bug fix and security patch releases
- Infrastructure as Code (IaC) deployments: Terraform, Pulumi, Ansible, Crossplane, or approved equivalents.
- CI/CD pipeline configuration changes
- Database schema migrations and data configuration changes
- Cloud resource provisioning and configuration drift remediation
- Third-party dependency upgrades where they affect runtime behaviour
- Emergency and hotfix releases (see Section 4.6)

Changes that affect only local developer environments or feature branches not yet integrated are out of scope.

3.2 Purpose

- Ensure every production change has been tested, approved, and can be reversed within a defined time window
- Maintain a complete, auditable record of what was deployed, by whom, when, and why
- Reduce the mean time to recovery (MTTR) from deployment failures through pre-planned rollback procedures
- Provide consistent release quality across all technology stacks and team sizes
- Comply with ISO 9001:2015 change control requirements and ISO 27001:2022 change management controls
- Produce release metrics that are sufficient to identify degradation in deployment capability before it becomes a production incident pattern

3.3 Guiding Principles

- The pipeline is the release mechanism. Manual deployments to any shared environment are prohibited except under documented emergency procedures.
- Rollback Verification: All targeted deployment paths must have an associated, verified rollback procedure validated in a staging environment prior to production execution.
- Environment Parity: Configuration baselines between staging and production environments must remain synchronized; any detected drift must be remediated or explicitly approved prior to release.
- Release documentation is written before the deployment, not after. Release notes, runbooks, and rollback instructions are part of the release package.
- Release Velocity: Process configurations must optimize for small, frequent deployment payloads to minimize batch integration risks.

- Post-deployment monitoring is part of the release, not a separate activity. A release is not complete until the stabilisation window has elapsed without incident.

3.4 Responsibility for Implementation

The Release Manager (Engineering Lead, Platform Lead, or Delivery Manager depending on team structure) is accountable for this procedure. Specific responsibilities:

- Release Manager: release scheduling, go/no-go decision authority, stakeholder communication, CAB coordination
- DevOps / Platform Engineer: pipeline configuration and maintenance, deployment execution, environment management, rollback tooling
- Engineering / Development Lead: code readiness sign-off, technical validation, post-deployment issue resolution
- QA Lead: test plan ownership, test execution coordination, release sign-off against defined exit criteria
- Security Engineer: security scan review, compliance validation for regulated releases
- Operations / SRE: production monitoring, on-call coverage during deployment window, incident response readiness

In smaller teams, multiple roles above may be held by the same person. The separation of the go/no-go decision from deployment execution is the only structural requirement that must be maintained regardless of team size.

3.5 References

- ISO 9001:2015, Quality Management Systems (clause 8.5, Production and service provision)
- ISO 27001:2022, Annex A control 8.32 (Change management)
- ITIL 4, Release Management and Deployment Management practices
- NIST SP 800-128, Guide for Security-Focused Configuration Management
- DORA State of DevOps Research: Deployment Frequency, Lead Time, Change Failure Rate, MTTR
- Internal: SDLC Procedure
- Internal: Code Review Procedure
- Internal: Change Management Procedure
- Internal: Incident Response Procedure
- Internal: Configuration Management Procedure
- Internal: Developer Guide / Engineering Handbook (where maintained)

3.6 Roles and Responsibilities

Role	Primary Responsibilities	Backup	Notes
Release Manager	Release planning and scheduling; go/no-go authority; CAB interface; stakeholder communication; release metrics review	Deputy Release Manager or Engineering Lead	The go/no-go decision must not be made by the person executing the deployment

DevOps / Platform Engineer	Pipeline maintenance and execution; environment parity enforcement; deployment automation; rollback tooling; post-deployment monitoring	Cross-trained team member	Responsible for pipeline health as a tracked KPI (see Section 4.2)
Engineering / Development Lead	Code freeze enforcement; release package review; technical sign-off; post-deployment defect resolution	Senior Developer	Must confirm that all items in the release have passed code review per QMS-DEV-PRO-003
QA Lead	Test strategy and execution coordination; exit criteria sign-off; regression coverage; performance and security test co-ordination	Senior QA Engineer	Test results are a mandatory release package component; releases do not proceed without them
Security Engineer	Security scan review; compliance checklist sign-off for regulated changes; vulnerability triage against release	Security team rotation	Required approver for any release touching auth, crypto, data handling, or network policy
Operations / SRE	Production monitoring during and after deployment; incident bridge ownership if deployment fails; on-call schedule coverage	Cross-trained team member	Must confirm monitoring and alerting are active before deployment window opens

3.7 Release Types and Classification

Release type determines the minimum lead time, approval requirements, and process steps. Teams must classify each release before planning begins. Misclassification that results in an under-controlled change is a non-conformance under this procedure.

Type	Description	Min Lead Time	Approval	Process Track
Major	Significant new functionality; architectural changes; breaking API changes; cross-system dependencies	2-4 weeks	Full CAB + Release Manager + Security	Full process, all phases mandatory
Minor	Non-breaking feature additions; improvements to existing functionality; non-critical dependency upgrades	1-2 weeks	Release Manager + QA Lead	Full process, phases may overlap in sprint cadence

Patch	Bug fixes; non-breaking dependency updates; performance improvements; configuration corrections	2-5 days	Engineering Lead + QA Lead	Abbreviated planning phase; all test phases required
Security Patch	Remediation of a confirmed vulnerability; CVE-driven dependency updates	4-24 hours (SLA-driven by severity)	Security Engineer + Release Manager	Security track: accelerated testing focused on vulnerability verification and regression
Emergency Fix	Production incident response; data integrity remediation; outage recovery	1-4 hours	Emergency CAB (synchronous) or Release Manager + Engineering Lead if CAB unavailable	Emergency track, see Section 4.6; full documentation within 24 hours post-deployment
Infrastructure	IaC changes; cloud resource provisioning; network policy changes; platform upgrades	3-7 days	Platform Lead + Security (if network or IAM)	Infrastructure track: IaC review and staging validation required; blast radius assessment mandatory

4. PROCESS DETAILS

4.1 Objectives

- Every production deployment has a documented, approved release plan with a tested rollback procedure
- No artefact reaches production that was not built from a tagged, reviewed, signed commit via the CI/CD pipeline
- Release metrics are collected automatically from existing tooling and reviewed at the cadences defined in Section 6
- The time to restore service following a failed deployment is within the MTTR targets defined in Section 4.2
- All release activity is auditable: what was deployed, who authorised it, what tests were run, and what the outcome was

4.2 Key Performance Indicators

The following metrics are collected from CI/CD platforms, monitoring systems, and incident trackers as a natural output of the toolchain. Manual data collection for any of these metrics indicates a tooling gap that should be addressed. DORA core metrics are noted - they are included because they are measurable from existing tooling, not for framework alignment purposes.

KPI	What it measures	Target	How collected	Context
Deployment Frequency	Number of deployments to production per week or month	Varies by team maturity - target trend is increasing over time	CI/CD platform deployment events	Core DORA metric; low frequency indicates batch risk accumulation. Target: at least weekly for all teams; daily for mature teams.
Deployment Success Rate	Percentage of deployments that complete without requiring rollback or emergency fix within 1 hour of deployment	≥ 98%	CI/CD platform (deployments tagged as success/rollback/failed)	A rollback is not a failure of the deployment tooling; it is a failure of pre-deployment testing or readiness criteria.
Change Failure Rate (CFR)	Percentage of production deployments that result in a degraded service, incident, or rollback	< 5% (elite); < 10% (high performing)	Deployments correlated with P1/P2 incidents and rollback events in incident tracker	Core DORA metric; CFR above 15% indicates systemic testing or readiness gate failure.
Mean Time to Restore (MTTR)	Time from detection of a deployment-caused incident to full service restoration	< 1 hour (elite); < 1 day (high performing)	Incident tracker: incident open time to resolved time, filtered to deployment-caused incidents	Core DORA metric; measures rollback speed and incident response effectiveness combined.
Lead Time for Changes	Time from code commit to production deployment	< 1 hour (elite); < 1 day (high performing)	Git commit timestamp to production deployment timestamp via CI/CD	Core DORA metric; high lead time often caused by infrequent releases or long test cycles, not slow pipelines.
Rollback Rate	Percentage of deployments that required a rollback within the stabilisation window	Correlated with CFR limits	CI/CD platform: rollback events as a proportion of total deployments	Separate from CFR. A rollback is a controlled recovery; tracking it separately identifies whether rollback tooling is reliable.
Mean Time to Deploy (MTTD)	Elapsed time from pipeline trigger to deployment completion in production	< 30 minutes for application releases; < 60 minutes for infrastructure	CI/CD platform timestamps, pipeline start to environment-healthy confirmation	Tracks pipeline efficiency. Creeping MTTD indicates dependency bloat, test suite slowdown, or environment issues.

Rollback Execution Time	Time from rollback decision to previous version confirmed stable in production	< 15 minutes	Rollback event timestamps in CI/CD or runbook execution logs	Rollback SLA must be tested in staging before every major or infrastructure release. Untested rollbacks are a compliance gap.
Environment Parity Drift	Number of confirmed configuration differences between staging and production at time of release	Zero unresolved critical drift items at release gate	IaC diff tooling (Terraform plan, Ansible check, Conftest) run as part of staging validation	Configuration drift between staging and production is the most common cause of 'passes staging, fails production' incidents.
Release Documentation Completeness	Percentage of releases that ship with complete release notes, runbook, and rollback plan in the release package	Mandatory package validation check passed	Release package checklist (automated where possible (PR template, pipeline gate))	Incomplete documentation is a non-conformance. Post-incident reviews consistently find that missing runbooks extend MTTR.
Post-Deployment Incident Rate	Number of P1/P2 incidents per 100 deployments, attributed to the deployment	< 1 per 100 deployments	Incident tracker: deployment tag on incident records	The ultimate measure of release quality. Slow to accumulate signal but the most operationally meaningful metric.
Stabilisation Window Breach Rate	Percentage of releases where a monitoring alert fires within the defined stabilisation window	< 5%	Monitoring platform alert counts correlated with deployment events	High breach rate without corresponding incidents indicates over-sensitive alerting. High breach rate with incidents confirms insufficient pre-deployment testing.
CAB / Approval Cycle Time	Time from release package submission to all required approvals obtained	< 4 hours for standard; < 30 minutes for emergency	Approval workflow timestamps in change management system or PR platform	Bottleneck indicator. Consistently long approval cycles indicate process friction that creates pressure to skip controls.

4.3 Release Testing Strategy

All test phases must use the signed release artefact. Test results are components of the release package and must be present before the deployment gate is cleared. The QA Lead is responsible for maintaining and updating the test strategy for each release type.

Phase	Scope	Environment	Owner	Exit Criteria	Automation
Unit	Individual functions and components	Development / CI	Developer	100% pass rate; no new failures introduced	Automated; runs on every commit
Integration	Component interactions; service boundaries; database connections	CI / Test environment	QA + Developer	Zero critical defects; contract tests pass	Automated; runs on every PR to integration branch
System / E2E	End-to-end business flows; cross-service scenarios	Staging	QA Lead	All defined acceptance scenarios pass	Automated where possible; manual for complex user journeys
Performance / Load	Throughput, latency, and resource utilisation under defined load profiles	Staging (production-equivalent sizing required)	Performance / SRE	Meets all NFR thresholds defined in requirements; no regression from baseline	Run on every major/minor release; spot-check on patches
Security Scan	SAST, DAST, dependency vulnerability scan, IaC policy check	CI + Staging	Security Engineer	Zero Critical open findings; High findings triaged with accepted risk or remediation plan	Automated in CI; manual penetration test annually or for major releases in regulated environments
Regression	Verification that existing functionality is unaffected	Staging	QA Lead	Defined regression suite passes at 100%	Automated; scope adjusted based on change impact analysis
UAT	Business validation of acceptance criteria	UAT / Staging	Product Owner + Business stakeholders	Written sign-off from product owner	Required for major releases and any change to customer-facing workflows
Infrastructure Validation	IaC plan review; configuration drift check; resource provisioning verification	Staging	Platform Engineer	Zero configuration drift vs production baseline; IaC plan reviewed and approved	Mandatory for all infrastructure releases; Terraform plan or equivalent reviewed as code

4.4 Release Process Steps

Six phases apply to all standard release types. Section 4.6 defines the emergency release track. Phase completion is confirmed by the responsible role signing off the gate criteria. Gate sign-off is recorded in the release record - verbal confirmation alone is not sufficient.

1. Release Planning

Steps	• Assign release classification per Section 3.7 before any other planning activity. • Define release scope: list all tickets, PRs, and configuration changes included. Scope must be locked before the planning gate; additions after lock require re-assessment. • Conduct impact assessment: identify affected services, dependent systems, data stores, and external integrations. • Perform blast radius analysis: what is the worst-case impact if the release causes a production incident? • Define rollback strategy: for each change in scope, confirm the rollback mechanism (pipeline rollback, feature flag toggle, database migration reversal, DNS failover, etc.). • Confirm environment availability and maintenance window with operations. • Identify blackout periods: confirm no deployment conflicts with business-critical periods (financial close, peak traffic windows, contracted SLA periods). • Raise change request per QMS-CHG-PRO-001 and obtain required approvals for the release type. • Publish release schedule to stakeholders with go/no-go decision timeline.
Gate	Release plan approved; scope locked; rollback strategy documented; change request approved; maintenance window confirmed.
Outputs	Release plan, impact assessment, blast radius assessment, rollback strategy, change request record.

2. Release Preparation

Steps	• Enforce code freeze at the agreed cut-off time. Any change added after freeze requires release manager sign-off and scope re-assessment. • Assemble the release package: all code changes, IaC changes, database migration scripts, configuration deltas, and environment variable changes. • Verify all items in the release package have passed code review per QMS-DEV-PRO-003. • Tag the release in version control. The release tag is the authoritative reference for what is being deployed. • Build the release artefact from the tagged commit via the CI pipeline. No manual builds for production artefacts. • Sign the release artefact (cosign, Notary, or equivalent) and store in the artefact registry. • Prepare release documentation: release notes, deployment runbook, rollback runbook, and monitoring runbook.

	• Prepare and validate test environments to match production configuration. Run IaC diff to confirm parity. • Notify operations and on-call team of the deployment window and escalation path.
Gate	Release package assembled from tagged commit; all items reviewed; artefact signed; documentation complete; environments confirmed as production-equivalent.
Outputs	Signed release artefact, release tag, release notes, deployment runbook, rollback runbook.

3. Testing and Validation

Steps	• Execute the test phases defined in Section 4.3 in the sequence specified. No phase may be skipped without Release Manager approval and documented justification. • All test execution must use the signed release artefact, not a separate build. • Record all test results in the test management system. Pass/fail status and execution timestamp are mandatory fields. • Any Critical or High defect discovered during testing requires triage before release proceeds. Resolution options: fix and re-test (requires new artefact), accepted risk with security engineer sign-off (High only), or release scope reduction. • Performance test results are compared against the established baseline. Regressions above 10% on any defined SLO require triage. • UAT sign-off must be written (email or ticketing system comment is acceptable; verbal only is not). • Confirm security scan results are within the accepted thresholds defined in the security checklist. • Complete the release readiness checklist; all items must be checked by the responsible role.
Gate	All mandatory test phases complete; no open Critical defects; High defects triaged; UAT signed off; security scan within threshold; readiness checklist complete.
Outputs	Test execution records, defect log, performance comparison report, UAT sign-off, security scan report, completed readiness checklist.

4. Deployment

Steps	• Confirm all pre-deployment checks pass: monitoring active, alerting configured, on-call team confirmed, rollback tested in staging, change request in approved state. • Execute deployment via the CI/CD pipeline. Manual deployment steps require written justification in the release record. • Apply the defined rollout strategy per release type and risk classification: Blue/green: route traffic to new environment after smoke tests pass; keep old environment warm for the stabilisation window. Canary: route a defined percentage of traffic (e.g. 5%, then 25%, then 100%) with a hold period and metric check at each stage. Feature flag: deploy to 100% of infrastructure but gate user access via flag; increment exposure incrementally.

	Rolling: update instances in batches; confirm health of each batch before proceeding. • Run smoke tests immediately after each deployment stage. Smoke tests must cover all critical user-facing paths. • Monitor the defined KPIs (error rate, latency, saturation) in real time during deployment. • Do not leave the deployment unattended during the stabilisation window. The deploying engineer or an on-call engineer must be actively monitoring. • If any rollback trigger condition is met (see Section 4.7), execute the rollback procedure immediately. Do not attempt to fix forward during the initial stabilisation window.
Gate	Pre-deployment checks passed; pipeline execution logged; smoke tests pass; no rollback trigger conditions met at end of deployment.
Outputs	Deployment pipeline log, smoke test results, real-time monitoring snapshot at deployment completion.

5. Post-Deployment Validation and Stabilisation

Steps	• Monitor production for the defined stabilisation window from deployment completion: Low-risk changes (patch, minor config): 30 minutes minimum. Standard feature releases (minor): 2 hours minimum. Major releases or infrastructure changes: 24 hours minimum, with active monitoring for first 4 hours. • Compare production metrics against pre-deployment baseline for: error rate, p50/p95/p99 latency, throughput, saturation, and any release-specific SLOs. • Confirm no new error types introduced in application logs. • For database migrations: verify row counts, index health, and query performance on affected tables. • Obtain post-deployment sign-off from the QA Lead and Operations lead. • If all criteria are met at end of stabilisation window, close the deployment record in the change management system. • If monitoring alerts fire during stabilisation: follow rollback procedure if service degradation is confirmed; escalate to incident management per QMS-INC-PRO-001.
Gate	Stabilisation window elapsed; all production metrics within SLO; no new error patterns; post-deployment sign-offs obtained.
Outputs	Post-deployment monitoring report, sign-offs, closed change record.

6. Release Closure

Steps	• Publish final release notes to all relevant stakeholders and documentation systems. • Update system documentation: architecture diagrams, API documentation, runbooks, and the configuration management system.

	• Record release metrics in the release tracking system: deployment time, success/rollback, post-deployment incident count, lead time. • Conduct a release retrospective for major releases. For minor and patch releases, capture lessons learned as a brief comment on the release record. • Any process gaps identified during the release are raised as improvement items per Section 6. • Archive the release package, all test evidence, approval records, and deployment logs per Section 4.8. • Release Manager signs off the release record.
Gate	Release notes published; documentation updated; metrics recorded; release record signed off.
Outputs	Closed release record, updated documentation, release metrics entry, retrospective notes (major releases).

4.5 Rollback Procedure

A rollback procedure must be defined, documented, and tested in staging before any major or infrastructure release. For minor and patch releases, the pipeline rollback mechanism (redeploy previous artefact) is the minimum acceptable procedure.

Rollback Trigger Conditions

Initiate rollback immediately if any of the following are observed within the stabilisation window:

- Error rate exceeds the pre-deployment baseline by more than 10% sustained for 5 minutes
- p95 latency exceeds SLO threshold
- Health check endpoint returns non-200 for more than 2 consecutive checks
- Data integrity check fails post-migration
- Any P1 incident is raised and attributed to the deployment
- On-call engineer makes a rollback call based on operational judgement; this is always a valid trigger and is not subject to committee approval

Rollback Execution

- The on-call engineer or deploying engineer initiates rollback without waiting for additional approvals once trigger conditions are met.
- Rollback is executed via the CI/CD pipeline using the previous signed artefact. Manual rollback steps are acceptable only if the pipeline rollback mechanism has failed.
- Target: previous version confirmed stable in production within 15 minutes of rollback initiation.
- Once stable, raise an incident record, and link it to the deployment record.
- A post-deployment review is conducted before the next attempt. The release is not re-attempted until root cause is identified.

4.6 Emergency Release Procedure

Emergency releases bypass the standard planning and preparation phases but do not bypass automated checks, artefact signing, or post-deployment verification. All steps skipped during the emergency track must be completed retrospectively within 24 hours.

Stage	Emergency Action	Standard Step Deferred
Initiation	Apply EMERGENCY classification; assemble on-call engineer, engineering lead, and security engineer (if security-related) on an incident bridge; confirm the fix scope is the minimum necessary to restore service	Full planning phase and impact assessment - completed retrospectively within 24 hours
Assessment	Quick impact and blast radius assessment on the bridge call (target < 15 minutes); identify rollback option; obtain verbal Emergency CAB approval or Release Manager + Engineering Lead co-approval	Formal change request - raised retrospectively within 4 hours of deployment
Testing	Run mandatory automated CI validation checks; execute critical-path verification and target component regression testing	Full test suite, UAT, and performance testing - retrospective review of skipped tests within 24 hours
Deployment	Deploy via pipeline using the standard deployment mechanism; enhanced monitoring active from deployment start; rollback ready and confirmed before deployment begins	Staged rollout strategy - emergency deployments go direct to 100% with immediate rollback readiness
Validation	Monitor for 30 minutes minimum post-deployment; confirm resolution of the triggering incident; confirm no new error patterns	Extended stabilisation window - continued monitoring for 24 hours post-deployment
Documentation	Create emergency release record immediately; log approval decisions with timestamps; record metric snapshot at deployment	Full release notes and runbook - completed within 24 hours; retrospective review within 5 business days

Emergency releases that occur more than twice in any rolling 30-day period for the same service are a process non-conformance and require a root cause review at the next engineering review.

4.7 Process Map

The end-to-end state diagram is provided as a separate artefact in the link below. The map shows all six standard phases, the emergency release track, rollback transitions, stabilisation window, and integration points with Change Management and Incident Response procedures.

Release Management Process Map

4.8 Retained Documentation

Document Type	Content	Retention	Location	Access
Release Plans	Scope, impact assessment, blast radius, rollback strategy, schedule	3 years	Release Management System	Team
Release Packages	Tagged artefact reference, change manifest, signed build record	Product lifetime	Artefact Registry / Version Control	Team
Test Evidence	Test execution records, defect log, performance reports, UAT sign-off, security scans	2 years	Test Management System	Team
Approval and Sign-off Records	CAB approval, go/no-go decisions, UAT sign-off, post-deployment sign-off	3 years	QMS / Change Management System	Management
Deployment Logs	Pipeline execution logs, smoke test results, rollback events	3 years	CI/CD Platform	Team
Post-Deployment Monitoring Reports	Metric snapshots, alert records, stabilisation window outcomes	2 years	Monitoring Platform / Release Record	Team
Incident Reports (deployment-caused)	Incident timeline, root cause, deployment linkage, MTTR	5 years	Incident Management System	Management
Release Metrics Records	Per-release KPI values; monthly and quarterly aggregates	3 years	Analytics / Reporting Platform	Management
Emergency Release Records	Emergency approval, abbreviated test evidence, post-deployment review	5 years	QMS / Incident Management System	Management

5. RISKS AND OPPORTUNITIES

5.1 Risk Register

Risks are reviewed quarterly as part of the continuous improvement cycle. The Release Manager owns the risk register and is responsible for confirming that mitigations are in place and effective. Risk status is a standing agenda item at the quarterly engineering review.

Category	Risk	Mitigation
Technical	Environment configuration drift between staging and production causing 'works in staging, fails in production' failures	IaC enforcement for all environment configuration; automated drift detection run before every deployment; IaC diff is a required pre-deployment gate item
Technical	Pipeline failure blocking deployment during a maintenance window	Pipeline health tracked as a KPI; redundant pipeline configuration for critical services; documented manual deployment procedure as a last resort with Release Manager approval
Technical	Database migration failure causing data integrity issues or extended rollback time	Migration scripts tested in staging against a production data snapshot; rollback script required alongside every forward migration; migration run time benchmarked before production window
Technical	Artefact integrity - deploying an unverified or tampered build	Mandatory artefact signing (cosign or equivalent); pipeline configured to reject unsigned artefacts at every environment boundary; artefact registry access-controlled
Process	Scope creep after code freeze introducing unreviewed changes	Freeze enforced via branch protection; post-freeze additions require Release Manager sign-off and documented justification; frozen scope is the authoritative release boundary
Process	Go/no-go pressure from business stakeholders overriding test failures	Go/no-go authority is the Release Manager's alone; business stakeholders are informed, not decision-makers; any override requires written acceptance of risk by the authorising executive
Process	Emergency releases becoming the default path for normal changes	Emergency release frequency tracked as a KPI; more than two emergencies in 30 days for one service triggers mandatory root cause review; emergency track requires explicit classification approval
Process	Incomplete rollback procedures discovered during a production incident	Rollback tested in staging before every major and infrastructure release; rollback test recorded in release package; rollback SLA (< 15 minutes) verified during staging test
Security	Vulnerability introduced in a dependency bundled in the release artefact	SCA scan run as part of CI and as a pre-release gate; Critical CVEs block release; High CVEs require Security Engineer triage; SBOM generated per release

Security	Unauthorised or undocumented change reaching production	Branch protection and required pipeline execution enforced at infrastructure level; deployment logs are immutable; any deployment not matching a release record is an audit incident
Operational	Key person dependency - release blocked because the Release Manager or on-call engineer is unavailable	Named deputy for every role; on-call schedule maintained with at least two contacts per tier; process documented sufficiently for any trained engineer to execute
Business	Deployment during a business blackout period causing service disruption	Blackout calendar maintained and checked during release planning; automated gate in release checklist; override requires VP-level approval and documented business justification

5.2 Opportunities

Area	Opportunity	Action
Deployment Automation	Eliminate remaining manual deployment steps	Audit current pipeline for manual steps; automate each with a defined test; target zero manual steps for all tier-1 services within one roadmap cycle
Progressive Delivery	Reduce blast radius of all releases through feature flags and canary deployments	Adopt LaunchDarkly, Unleash, or equivalent; define rollout percentages and metric gates; make canary the default strategy for all major releases
Release Train Cadence	Reduce batch size and deployment risk by increasing release frequency	Move from monthly/quarterly release trains to weekly or bi-weekly; measure CFR and MTTR through the transition; small frequent releases consistently outperform large infrequent ones
Automated Release Notes	Reduce documentation effort and eliminate gaps	Enforce conventional commits; auto-generate release notes from commit history and linked tickets via tooling (Release Please, semantic-release, or equivalent)
SBOM Generation	Improve supply chain visibility and compliance posture	Generate an SBOM (Software Bill of Materials) per release artefact using Syft or equivalent; attach to release record; required for SOC 2, FedRAMP, and increasing numbers of enterprise procurement requirements
Release Metrics Dashboard	Make release health visible without manual reporting	Instrument CI/CD platform and monitoring tools to auto-publish KPIs to a team dashboard; review in weekly team standup; anomaly detection on key metrics

6. CONTINUOUS IMPROVEMENT

6.1 Improvement Cadences

Cadence	Trigger / Frequency	What is reviewed	Required Output
Post-Release Review	After every major release; optional for minor and patch	Deployment timeline; metric performance vs baseline; any incidents or rollbacks; gate effectiveness	Release retrospective notes; action items logged in project tracker
Weekly Metrics Review	Weekly	Deployment frequency; success rate; MTTD; any rollback or emergency release events	Status report; escalation of any metric breaching threshold
Monthly KPI Review	Monthly	All KPIs in Section 4.2; trend analysis; emergency release frequency; CFR and MTTR trajectory	KPI report; named action items for any metric outside target; owner assigned
Quarterly Engineering Review	Quarterly	Full KPI review; risk register update; process improvement backlog; tooling evaluation; blackout calendar review	Updated risk register; process change decisions; procedure amendment if required
Annual Procedure Review	Annually (minimum)	Full document review; compliance alignment (ISO, ITIL); team structure changes; regulatory changes	Updated procedure version; amendment record entry; re-approval cycle
Post-Incident Review	After any P1/P2 incident attributed to a deployment	Deployment timeline; rollback effectiveness; gap in pre-deployment testing or gates; MTTR vs target	Root-Cause Analysis (RCA) Report; specific procedural action items; updated runbooks if applicable

6.2 Non-Conformance Handling

The following conditions are classified as non-conformances under this procedure and require a documented corrective action within 5 business days:

- A production deployment executed without a corresponding approved release record
- An artefact deployed to production that was not built from a tagged, reviewed commit via the CI/CD pipeline
- A mandatory test phase skipped without Release Manager approval and documented justification
- A rollback procedure that did not achieve the 15-minute SLO during a production incident
- Emergency release frequency exceeding two events in 30 days for a single service
- Release documentation (notes, runbook, rollback plan) absent from the release package at deployment

Non-conformances are logged in the QMS non-conformance register and reviewed at the next quarterly engineering review. Repeat non-conformances on the same item trigger a formal corrective action process.

6.3 Document Control

This document is the authoritative version of the release management procedure. It supersedes all previous versions. Previous versions are archived in the document management system with their amendment records.

The procedure is reviewed whenever: a post-incident review identifies a process gap; a metric trend indicates a systemic issue; tooling changes affect a defined process step; or the annual review cycle is due.