5.4 Release Management und Deployment Management

> **Kurz erklärt**
>
> Release Management und Deployment Management sorgen dafür, dass neue oder geänderte Services, Anwendungen, Funktionen oder Infrastrukturkomponenten kontrolliert bereitgestellt werden.
>
> **Release Management** betrachtet vor allem, wann und in welcher Form eine Änderung für Benutzer oder Kunden verfügbar gemacht wird.
>
> **Deployment Management** betrachtet vor allem, wie technische Komponenten in eine Zielumgebung gebracht werden.
>
> Beide Bereiche hängen eng mit Change Enablement zusammen.

Warum Release und Deployment wichtig sind

Viele Änderungen werden nicht einzeln sichtbar.

Oft werden mehrere technische Änderungen gemeinsam bereitgestellt.

Beispiele:

- neue Anwendungsversion,
- Sicherheitsupdate,
- neue Funktion im Serviceportal,
- neue VPN-Client-Version,
- aktualisierte Serverkonfiguration,
- geänderte Schnittstelle,
- neues Monitoring,
- neue Self-Service-Funktion,
- aktualisiertes Berechtigungskonzept.

Wenn solche Änderungen unkoordiniert ausgerollt werden, können entstehen:

- Incidents nach Updates,
- Benutzerverwirrung,
- fehlende Dokumentation,
- inkompatible Versionen,
- unklare Verantwortlichkeiten,

- fehlende Rollback-Möglichkeiten,
- nicht informierter Service Desk,
- und unnötige Störungen im Betrieb.

Release und Deployment sorgen für geordnete Bereitstellung.

Release und Deployment unterscheiden

Begriff	Bedeutung
Release	Bündel neuer oder geänderter Funktionen, Services oder Komponenten, das bereitgestellt und für Nutzung freigegeben wird
Deployment	technische Überführung von Komponenten in eine Zielumgebung
Change	kontrollierte Änderung an einem Service oder unterstützenden Bestandteil
Rollout	Verteilung oder Einführung eines Releases bei Benutzern, Standorten oder Systemen
Rollback	Rücknahme einer Änderung auf einen vorherigen Zustand
Backout	geplanter Ausstieg aus einer Änderung, wenn Abbruchkriterien erfüllt sind

Ein Deployment kann technisch erfolgreich sein, obwohl das Release aus Benutzersicht noch nicht erfolgreich ist.

Beispiel:

Die neue Version wurde auf dem Server installiert.

Benutzer können aber eine wichtige Funktion nicht nutzen.

Technisch wurde deployed.

Das Release ist fachlich noch nicht erfolgreich.

Release Management

Release Management plant, koordiniert und steuert die Bereitstellung neuer oder geänderter Services und Funktionen.

Typische Fragen:

- Was wird veröffentlicht?
- Welche Funktionen oder Änderungen sind enthalten?

- Welche Benutzer oder Kunden sind betroffen?
- Wann wird das Release verfügbar?
- Welche Kommunikation ist notwendig?
- Welche Dokumentation muss aktualisiert werden?
- Welche Risiken bestehen?
- Welche Abhängigkeiten gibt es?
- Welche Changes gehören zum Release?
- Wie wird der Erfolg geprüft?

Release Management verbindet technische Bereitstellung mit Benutzer- und Serviceperspektive.

Deployment Management

Deployment Management sorgt dafür, dass technische Komponenten kontrolliert in eine Zielumgebung gebracht werden.

Typische Fragen:

- Welche Komponenten werden installiert oder geändert?
- In welche Umgebung wird deployed?
- Welche Version wird bereitgestellt?
- Welche Reihenfolge ist notwendig?
- Welche Abhängigkeiten bestehen?
- Welche Tests sind erforderlich?
- Wie wird der technische Zustand geprüft?
- Wie wird zurückgerollt?
- Welche Automatisierung wird genutzt?
- Welche Logs und Nachweise entstehen?

Deployment Management besitzt damit einen stärker technischen Fokus.

Release, Deployment und Change Enablement im Zusammenspiel

Ein möglicher Zusammenhang:

```
Änderungsbedarf entsteht
        ↓
Change Request wird erstellt
        ↓
Risiko und Auswirkung werden bewertet
        ↓
Release wird geplant
        ↓
```

```
Deployment wird vorbereitet
        ↓
Change wird genehmigt
        ↓
Deployment wird durchgeführt
        ↓
Release wird für Benutzer verfügbar
        ↓
Erfolg wird geprüft
        ↓
Dokumentation und Knowledge werden aktualisiert
```

Nicht jede Organisation trennt diese Schritte gleich stark.

Wichtig ist, dass technische Umsetzung, Freigabe und Kommunikation zusammenpassen.

Beispiel: Anwendungsversion

Eine Fachanwendung erhält eine neue Version.

Change Enablement

- bewertet Risiko,
- prüft Freigabe,
- legt Wartungsfenster fest,
- betrachtet Rollback und Kommunikation.

Release Management

- plant, welche Version wann veröffentlicht wird,
- klärt enthaltene Funktionen,
- informiert Benutzer und Service Desk,
- stellt Release Notes bereit.

Deployment Management

- installiert die neue Version,
- führt technische Prüfschritte aus,
- kontrolliert Logs und Monitoring,
- stellt bei Bedarf zurück.

Alle drei Bereiche müssen zusammenarbeiten.

Release-Paket

Ein Release-Paket kann alle Bestandteile enthalten, die für eine Bereitstellung notwendig sind.

Mögliche Inhalte:

- Anwendungsversion,
- Konfigurationsdateien,
- Datenbankskripte,
- Infrastrukturänderungen,
- Installationsanleitung,
- Testnachweise,
- Rollback-Anleitung,
- Release Notes,
- Benutzerinformation,
- aktualisierte Knowledge-Artikel,
- Monitoring-Anpassungen,
- Sicherheitsfreigaben,
- Change-Verknüpfungen.

Nicht jedes Release-Paket ist gleich umfangreich.

Der Umfang muss zum Risiko und zur Bedeutung des Releases passen.

Release-Plan

Ein Release-Plan beschreibt, wann und wie ein Release bereitgestellt wird.

Typische Inhalte:

- Release-Name oder Version,
- betroffene Services,
- enthaltene Changes,
- Zielumgebungen,
- geplanter Zeitraum,
- beteiligte Teams,
- Abhängigkeiten,
- Kommunikationsmaßnahmen,
- Teststrategie,
- Deployment-Schritte,
- Rollback-Plan,
- Erfolgskriterien,
- Verantwortlichkeiten.

Ein Release-Plan muss nicht kompliziert sein.

Er muss aber ausreichend klar sein, damit alle Beteiligten wissen, was passiert.

Release Notes

Release Notes beschreiben, was sich mit einem Release ändert.

Sie können enthalten:

- neue Funktionen,
- geänderte Funktionen,
- behobene Fehler,
- bekannte Einschränkungen,
- bekannte Fehler,
- Workarounds,
- betroffene Benutzergruppen,
- notwendige Benutzeraktionen,
- geänderte Systemanforderungen,
- Ansprechpartner oder Supportweg.

Release Notes sollten zielgruppengerecht sein.

Technische Teams benötigen andere Informationen als Endbenutzer.

Benutzerorientierte Release Notes

Ungeeignet:

> Backend-Komponente API-GW-PRD wurde auf Build 2026.08.14-rc3 aktualisiert.

Besser:

> Die Suche im Mitarbeiterportal wurde verbessert. Ergebnisse werden nun schneller angezeigt. Während des Updates kann das Portal am 14.08.2026 zwischen 20:00 und 20:30 Uhr kurzzeitig nicht verfügbar sein.

Benutzer müssen verstehen:

- Was ändert sich für mich?
- Wann passiert es?
- Muss ich etwas tun?
- Wo bekomme ich Hilfe?

Deployment-Plan

Ein Deployment-Plan beschreibt die technische Umsetzung.

Mögliche Inhalte:

- Zielumgebung,
- Version,
- betroffene Komponenten,
- Reihenfolge der Schritte,
- Verantwortliche,
- benötigte Berechtigungen,
- benötigte Dateien oder Images,
- Vorbereitungsaufgaben,
- technische Prüfschritte,
- Monitoring,
- Abbruchkriterien,
- Rollback-Schritte,
- Dokumentationshinweise.

Bei einfachen Deployments kann der Plan kurz sein.

Bei kritischen Deployments muss er detaillierter sein.

Umgebungen

Viele Organisationen nutzen mehrere Umgebungen.

Beispiele:

- Entwicklung,
- Test,
- Integration,
- Abnahme,
- Staging,
- Produktion.

Ziel ist, Änderungen vor der Produktion zu prüfen.

Typische Fragen:

- Entspricht die Testumgebung der Produktion ausreichend?
- Sind Testdaten geeignet?
- Sind Schnittstellen realistisch abgebildet?
- Sind Berechtigungen vergleichbar?

- Wurden Konfigurationen korrekt übernommen?
- Sind Unterschiede zwischen Umgebungen dokumentiert?

Ein Test in einer völlig abweichenden Umgebung gibt nur begrenzte Sicherheit.

Test und Abnahme

Vor einem Release sollten geeignete Tests durchgeführt werden.

Mögliche Tests:

- Funktionstest,
- Integrationstest,
- Sicherheitstest,
- Performance-Test,
- Regressionstest,
- Benutzerakzeptanztest,
- Smoke Test,
- Wiederherstellungstest,
- Monitoringtest.

Der Testumfang hängt vom Risiko ab.

Ein kritisches Release benötigt mehr Prüfung als eine kleine, risikoarme Änderung.

Regressionstest

Ein Regressionstest prüft, ob bestehende Funktionen nach einer Änderung weiterhin funktionieren.

Beispiel:

Eine neue Exportfunktion wird eingebaut.

Der Regressionstest prüft zusätzlich:

- Anmeldung,
- Suche,
- Speichern,
- bestehender Export,
- Berechtigungen,
- Schnittstellen.

Dadurch wird verhindert, dass eine neue Funktion alte Funktionen unbeabsichtigt beschädigt.

Smoke Test nach Deployment

Ein Smoke Test ist eine kurze Grundprüfung direkt nach dem Deployment.

Beispiele:

- Service startet,
- Anmeldung funktioniert,
- Startseite lädt,
- Datenbankverbindung steht,
- Schnittstelle antwortet,
- Monitoring zeigt keine kritischen Alarme,
- zentrale Benutzeraktion funktioniert.

Ein Smoke Test ersetzt keine vollständige Abnahme.

Er hilft aber, grobe Fehler sofort zu erkennen.

Deployment-Methoden

Organisationen können verschiedene Deployment-Methoden nutzen.

Beispiele:

Methode	Kurzbeschreibung
Big Bang	Bereitstellung für alle Benutzer gleichzeitig
phasenweise Bereitstellung	Einführung in mehreren Gruppen oder Standorten
Pilot	begrenzte Einführung bei kleiner Benutzergruppe
Rolling Deployment	schrittweise Aktualisierung einzelner Systeme
Blue-Green Deployment	Wechsel zwischen zwei vorbereiteten Umgebungen
Canary Deployment	neue Version zunächst für kleinen Teil der Benutzer
Feature Toggle	Funktion wird technisch bereitgestellt, aber gezielt aktiviert oder deaktiviert

Diese Methoden sind Praxisbeispiele.

ITIL schreibt keine bestimmte technische Deployment-Methode für alle Organisationen vor.

Big-Bang-Deployment

Beim Big-Bang-Deployment wird eine Änderung für alle betroffenen Benutzer oder Systeme gleichzeitig bereitgestellt.

Vorteile:

- einfacher Zeitplan,
- keine parallelen Versionen,
- schnelle vollständige Einführung.

Risiken:

- Fehler betreffen sofort alle Benutzer,
- hoher Kommunikationsbedarf,
- Rollback kann kritisch sein,
- Supportspitzen möglich.

Geeignet eher dann, wenn:

- Risiko überschaubar ist,
- Testqualität hoch ist,
- Rollback möglich ist,
- und alle Benutzer gleichzeitig wechseln müssen.

Phasenweise Bereitstellung

Bei einer phasenweisen Bereitstellung erfolgt der Rollout schrittweise.

Beispiele:

- zuerst IT-Team,
- dann Pilotgruppe,
- dann ein Standort,
- dann alle Standorte,
- dann externe Benutzer.

Vorteile:

- Fehler werden früher erkannt,
- Auswirkungen bleiben zunächst begrenzt,
- Support kann Erfahrungen sammeln,
- Kommunikation kann angepasst werden.

Risiken:

- mehrere Versionen parallel,
- längere Gesamtdauer,
- zusätzlicher Koordinationsaufwand.

Pilot

Ein Pilot testet ein Release mit einer begrenzten Benutzergruppe.

Geeignet für:

- neue Software,
- neue Geräte,
- neue Prozesse,
- größere Updates,
- Änderungen mit Benutzerwirkung.

Ein Pilot sollte klare Kriterien besitzen:

- Wer nimmt teil?
- Welche Funktionen werden getestet?
- Wie wird Feedback gesammelt?
- Wann ist der Pilot erfolgreich?
- Wann wird gestoppt?
- Was passiert nach dem Pilot?

Ein Pilot ohne Auswertung bringt wenig Nutzen.

Blue-Green Deployment

Beim Blue-Green Deployment existieren zwei Umgebungen.

Beispiel:

- Blau = aktuelle produktive Umgebung,
- Grün = neue vorbereitete Umgebung.

Nach Tests wird der Verkehr auf die neue Umgebung umgeschaltet.

Vorteile:

- schneller Wechsel möglich,
- Rollback kann einfacher sein,
- neue Umgebung kann vorbereitet werden.

Risiken:

- höhere Infrastrukturkosten,
- Datenbank- und Datenkonsistenz beachten,
- Umschaltung muss sauber getestet sein,
- nicht für jede Architektur geeignet.

Canary Deployment

Beim Canary Deployment erhält zuerst nur ein kleiner Teil der Benutzer die neue Version.

Wenn keine Probleme auftreten, wird der Anteil erhöht.

Vorteile:

- Risiko wird begrenzt,
- echte Nutzung liefert frühe Signale,
- Fehler betreffen zunächst wenige Benutzer.

Risiken:

- Monitoring muss gut sein,
- Benutzergruppen müssen steuerbar sein,
- parallele Versionen können komplex sein,
- Support muss wissen, wer welche Version nutzt.

Feature Toggles

Feature Toggles ermöglichen, Funktionen unabhängig vom technischen Deployment zu aktivieren oder zu deaktivieren.

Beispiel:

Die neue Funktion ist bereits deployed, aber nur für eine Pilotgruppe aktiviert.

Vorteile:

- kontrollierte Freigabe,
- schnelles Deaktivieren bei Problemen,
- Trennung von Deployment und Release möglich.

Risiken:

- zusätzliche Komplexität,
- alte Schalter werden vergessen,
- Tests werden schwieriger,
- falsche Aktivierung kann Incidents erzeugen.

Feature Toggles benötigen klare Verwaltung und Dokumentation.

Deployment-Automatisierung

Deployments werden häufig automatisiert.

Beispiele:

- Softwareverteilung,

- CI/CD-Pipeline,
- Skripte,
- Container-Deployment,
- Infrastruktur als Code,
- Konfigurationsmanagement,
- Paketverteilung,
- Cloud-Automatisierung.

Vorteile:

- reproduzierbar,
- schneller,
- weniger manuelle Fehler,
- besser protokollierbar,
- leichter wiederholbar.

Risiken:

- falsche Automatisierung skaliert Fehler schnell,
- Berechtigungen können zu weit sein,
- Rollback muss ebenfalls berücksichtigt werden,
- Tests müssen zuverlässig sein,
- Pipeline-Ausfälle können Deployments blockieren.

Manuelles Deployment

Manuelle Deployments können notwendig sein, wenn:

- Umgebung sehr speziell ist,
- Automatisierung noch nicht vorhanden ist,
- Änderung einmalig ist,
- Risiko eine bewusste Schritt-für-Schritt-Kontrolle erfordert.

Risiken manueller Deployments:

- Tippfehler,
- vergessene Schritte,
- unterschiedliche Durchführung,
- schlechte Reproduzierbarkeit,
- fehlende Protokollierung.

Runbooks und Checklisten reduzieren diese Risiken.

Release-Kalender

Ein Release-Kalender zeigt geplante Releases und wichtige Change-Zeitpunkte.

Er hilft bei:

- Koordination,
- Vermeidung von Überschneidungen,
- Kommunikation,
- Ressourcenplanung,
- Wartungsfenstern,
- Change Freeze,
- Lieferantenabstimmung,
- Bereitschaftsplanung.

Ein Release-Kalender sollte für relevante Teams sichtbar sein.

Change Freeze und Release Freeze

Ein Freeze ist ein Zeitraum, in dem Releases oder Changes eingeschränkt werden.

Beispiele:

- Jahresabschluss,
- Feiertagsgeschäft,
- Produktionshochlauf,
- Auditphase,
- Prüfungszeitraum,
- große Migration.

Ein Freeze bedeutet nicht zwingend, dass gar nichts geändert werden darf.

Sicherheitsupdates oder Emergency Changes können trotzdem notwendig sein.

Dann muss besonders bewusst entschieden und dokumentiert werden.

Kommunikation vor einem Release

Vor einem Release sollten betroffene Gruppen informiert werden.

Mögliche Inhalte:

- was sich ändert,
- wann es sich ändert,
- welche Services betroffen sind,
- ob Ausfallzeiten entstehen,
- welche neuen Funktionen verfügbar sind,
- welche bekannten Einschränkungen bestehen,

- was Benutzer tun müssen,
- wo Hilfe verfügbar ist,
- wann die nächste Information folgt.

Der Service Desk muss besonders gut vorbereitet sein.

Service Desk vorbereiten

Der Service Desk benötigt vor einem Release:

- Release Notes,
- bekannte Fehler,
- Workarounds,
- Supportskripte,
- Eskalationswege,
- betroffene Benutzergruppen,
- geplante Ausfallzeiten,
- Ansprechpartner,
- Statusinformationen,
- Knowledge-Artikel.

Ohne Vorbereitung steigen nach einem Release unnötige Tickets und Rückfragen.

Monitoring während und nach Deployment

Während und nach einem Deployment sollte Monitoring aktiv beobachtet werden.

Zu prüfen sind:

- Verfügbarkeit,
- Antwortzeiten,
- Fehlerraten,
- CPU und Arbeitsspeicher,
- Datenbankverbindungen,
- Schnittstellen,
- Warteschlangen,
- Logfehler,
- Sicherheitsmeldungen,
- Benutzeranmeldungen,
- Ticketaufkommen.

Nach kritischen Releases kann eine verstärkte Beobachtungsphase sinnvoll sein.

Rollback und Roll Forward

Bei Problemen nach einem Deployment gibt es zwei grundsätzliche Richtungen.

Ansatz	Bedeutung
Rollback	Rückkehr zum vorherigen Zustand
Roll Forward	Vorwärtskorrektur durch neue Änderung oder Hotfix

Rollback ist nicht immer möglich.

Roll Forward kann sinnvoll sein, wenn:

- Daten bereits migriert wurden,
- Rückkehr zu riskant wäre,
- Fehler schnell korrigierbar ist,
- neue Version nur kleine Korrektur benötigt.

Beide Wege benötigen Entscheidung, Dokumentation und Risikobewertung.

Release-Erfolg prüfen

Nach einem Release sollte geprüft werden:

- Wurde das Release wie geplant bereitgestellt?
- Funktionieren Kernservices?
- Sind Benutzer arbeitsfähig?
- Sind Incidents nach dem Release entstanden?
- Wurden erwartete Funktionen geliefert?
- Sind bekannte Fehler dokumentiert?
- Wurde der Service Desk informiert?
- Sind Monitoring und Logs unauffällig?
- Muss Knowledge aktualisiert werden?
- Wurde der erwartete Nutzen erreicht?

Ein Release ist nicht erfolgreich, nur weil Dateien deployed wurden.

Es muss aus Service- und Benutzersicht funktionieren.

Post-Deployment Review

Ein Post-Deployment Review kann nach wichtigen Deployments sinnvoll sein.

Fragen:

- Was lief gut?
- Was lief anders als geplant?

- Waren Tests ausreichend?
- War Kommunikation ausreichend?
- Gab es Incidents?
- War Rollback realistisch?
- Waren Abhängigkeiten bekannt?
- War die Automatisierung zuverlässig?
- Muss das Runbook verbessert werden?
- Welche Lessons Learned gibt es?

Nicht jedes kleine Deployment benötigt ein ausführliches Review.

Bei kritischen oder fehlgeschlagenen Deployments ist es wichtig.

Release Management und Knowledge Management

Nach einem Release müssen Wissensbestände aktualisiert werden.

Mögliche Inhalte:

- Benutzeranleitungen,
- Service-Desk-Artikel,
- Known Errors,
- Workarounds,
- Runbooks,
- FAQ,
- Screenshots,
- Menüpfade,
- Installationsanleitungen,
- Monitoringhinweise,
- Eskalationswege.

Veraltete Knowledge-Artikel erzeugen nach Releases häufig neue Tickets.

Release Management und Configuration Management

Nach Releases und Deployments müssen Configuration Items aktuell bleiben.

Mögliche Änderungen:

- neue Version,
- neue Komponente,
- geänderte Schnittstelle,
- geänderte Abhängigkeit,
- neuer Server,
- neues Zertifikat,

- geänderte Konfiguration,
- neuer Owner,
- geänderter Lieferant.

Wenn die CMDB veraltet bleibt, werden spätere Incidents, Problems und Changes schwieriger.

Release Management und Incident Management

Nach Releases können Incidents entstehen.

Deshalb sollte Incident Management vorbereitet sein.

Wichtig:

- Release-Zeitpunkt kennen,
- bekannte Änderungen kennen,
- betroffene Services erkennen,
- typische Symptome kennen,
- Known Errors nutzen,
- Eskalationswege kennen,
- Release-Team erreichbar haben.

Ein Anstieg von Incidents nach einem Release kann auf Probleme in Test, Kommunikation oder Deployment hinweisen.

Release Management und Problem Management

Wenn nach einem Release wiederkehrende oder schwere Incidents entstehen, kann ein Problem Record notwendig sein.

Problem Management prüft dann:

- War das Release Ursache oder Auslöser?
- Welche Funktion ist betroffen?
- Warum wurde der Fehler im Test nicht erkannt?
- Welche Abhängigkeit war unbekannt?
- Welche Benutzergruppe ist betroffen?
- Gibt es einen Workaround?
- Ist Rollback oder Roll Forward sinnvoll?
- Welche dauerhafte Lösung wird benötigt?

Release Management und Change Enablement

Change Enablement sorgt dafür, dass Releases und Deployments kontrolliert umgesetzt werden.

Zu klären ist:

- Welche Changes gehören zum Release?
- Welche Freigaben sind notwendig?
- Welche Risiken bestehen?
- Sind Wartungsfenster abgestimmt?
- Gibt es parallele Changes?
- Gibt es ein Rollback?
- Ist Kommunikation vorbereitet?
- Wurde der Erfolg geprüft?

Release und Deployment sollten nicht an Change Enablement vorbei erfolgen.

Praxisbeispiel: VPN-Client-Rollout

Ausgangslage

Eine alte VPN-Client-Version verursacht wiederkehrende Incidents.

Release

Neue VPN-Client-Version wird bereitgestellt.

Deployment

- Pilotgruppe erhält neue Version,
- Logs und Verbindungsstabilität werden geprüft,
- Service Desk erhält Workaround und Eskalationshinweise,
- anschließend phasenweiser Rollout an alle Notebooks.

Erfolgskontrolle

VPN-Incidents gehen zurück.

Known Error wird aktualisiert.

Knowledge-Artikel wird angepasst.

Praxisbeispiel: Mitarbeiterportal

Ausgangslage

Das Mitarbeiterportal erhält neue Funktionen.

Release Management

- Release Notes erstellen,
- Benutzerinformation vorbereiten,
- Fachbereich einbinden,
- Service Desk briefen.

Deployment Management

- neue Version im Wartungsfenster installieren,
- Datenbankskripte ausführen,
- Smoke Test durchführen,
- Monitoring prüfen.

Nachbereitung

Incidents nach Release beobachten und Knowledge-Artikel aktualisieren.

Praxisbeispiel: Fehlgeschlagenes Deployment

Situation

Eine neue Anwendungsversion wird erfolgreich installiert.

Problem

Nach dem Deployment funktioniert der PDF-Export nicht mehr.

Maßnahmen

- Incident erstellen,
- Fachbereich informieren,
- Rollback oder Hotfix bewerten,
- Problem Record bei wiederkehrendem Fehler prüfen,
- Testplan erweitern,
- Release Notes und Known Errors aktualisieren.

Lerneffekt

Der Testplan muss künftig auch Exportfunktionen abdecken.

Typische Fehler

Fehler 1

Release und Deployment werden gleichgesetzt.

Fehler 2

Technisches Deployment gelingt, aber Benutzer sind nicht vorbereitet.

Fehler 3

Service Desk erhält keine Release-Informationen.

Fehler 4

Release Notes fehlen oder sind zu technisch.

Fehler 5

Tests decken wichtige Geschäftsprozesse nicht ab.

Fehler 6

Rollback wird nicht realistisch geplant.

Fehler 7

Mehrere Releases kollidieren zeitlich.

Fehler 8

Monitoring wird nach Deployment nicht beobachtet.

Fehler 9

Known Errors und Workarounds werden nicht dokumentiert.

Fehler 10

CMDB und Knowledge Base bleiben nach Release veraltet.

Fehler 11

Fehlgeschlagene Deployments werden nicht ausgewertet.

Fehler 12

Automatisierung wird genutzt, aber nicht überwacht.

Checkliste Release Management

- [ ] Release-Inhalt ist beschrieben
- [ ] betroffene Services sind bekannt
- [ ] betroffene Benutzergruppen sind bekannt
- [ ] enthaltene Changes sind verknüpft
- [ ] Risiken sind bewertet
- [ ] Release Notes sind vorbereitet
- [ ] Service Desk ist informiert
- [ ] Kommunikationsplan ist vorhanden
- [ ] Testnachweise liegen vor
- [ ] Known Errors sind dokumentiert
- [ ] Erfolgskriterien sind definiert
- [ ] Nachbeobachtung ist geplant

Checkliste Deployment Management

- [ ] Zielumgebung ist bekannt
- [ ] Version ist eindeutig
- [ ] Komponenten sind vollständig
- [ ] Deployment-Schritte sind dokumentiert
- [ ] Berechtigungen sind vorhanden
- [ ] Backups oder Sicherungen sind geprüft
- [ ] Rollback oder Roll Forward ist bewertet
- [ ] Abbruchkriterien sind definiert
- [ ] Monitoring ist aktiv
- [ ] Smoke Test ist definiert
- [ ] Ergebnis wird dokumentiert
- [ ] Abweichungen werden erfasst

Checkliste Kommunikation

- [ ] Benutzerinformation vorbereitet
- [ ] Service Desk informiert
- [ ] Fachbereiche informiert
- [ ] Management informiert, falls relevant
- [ ] Lieferanten eingebunden, falls relevant
- [ ] Ausfallzeit klar benannt
- [ ] neue Funktionen verständlich erklärt
- [ ] bekannte Einschränkungen genannt
- [ ] Supportweg beschrieben
- [ ] Abschlussmeldung vorgesehen

Checkliste Nachbereitung

- [ ] Release-Erfolg geprüft
- [ ] Incidents nach Release ausgewertet
- [ ] Monitoring geprüft
- [ ] Logs geprüft
- [ ] Fachbereichsfeedback eingeholt
- [ ] Knowledge Base aktualisiert
- [ ] Runbooks aktualisiert
- [ ] CMDB aktualisiert
- [ ] Known Errors aktualisiert
- [ ] Problem Record erstellt, falls erforderlich
- [ ] Lessons Learned dokumentiert

Bedeutung für Fachinformatiker für Systemintegration

Fachinformatiker sind häufig direkt an Deployments beteiligt.

Im Alltag bedeutet das:

- Versionen sauber dokumentieren,
- Deployment-Schritte nachvollziehbar ausführen,
- Abhängigkeiten prüfen,
- Tests durchführen,
- Monitoring beobachten,
- Rollback vorbereiten,
- Service Desk informieren,

- Knowledge-Artikel aktualisieren,
- und Fehler nach Releases strukturiert auswerten.

Gute technische Bereitstellung bedeutet nicht nur:

> Installation erfolgreich.

Sondern:

> Service funktioniert, Benutzer sind arbeitsfähig, Support ist vorbereitet und Dokumentation stimmt.

Zusammenfassung

> Release-Inhalt festlegen
> ↓
> Changes, Risiken und Abhängigkeiten prüfen
> ↓
> Deployment vorbereiten
> ↓
> Service Desk und Benutzer informieren
> ↓
> technische Bereitstellung durchführen
> ↓
> Smoke Test und Monitoring prüfen
> ↓
> Release aus Benutzersicht bewerten
> ↓
> Incidents und Feedback beobachten
> ↓
> Knowledge, CMDB und Runbooks aktualisieren
> ↓
> Lessons Learned übernehmen

Merksätze

>

> Release Management betrachtet die Bereitstellung aus Service- und Benutzersicht.

> Deployment Management betrachtet die technische Überführung in eine Zielumgebung.

> Ein erfolgreiches Deployment ist nicht automatisch ein erfolgreiches Release.

> Service Desk und Benutzer müssen auf wichtige Releases vorbereitet sein.

> Rollback und Kommunikation gehören vor dem Deployment geplant.

> Nach einem Release müssen Knowledge, CMDB und Monitoring geprüft werden.

Verwandte Seiten

- 5.1 Change Enablement – Ziele, Begriffe und Abgrenzung
- 5.2 Change-Typen und Risikobewertung
- 5.3 Genehmigung, Planung und Umsetzung von Changes
- 5.5 Change-Erfolg messen und Continual Improvement
- Incident Management
- Problem Management
- Knowledge Management
- Service Configuration Management

Quellen und Versionsstand

Offizielle Grundlagen

- PeopleCert – ITIL Practice Guide: Release Management
- PeopleCert – ITIL Practice Guide: Deployment Management
- PeopleCert – ITIL Practice Guide: Change Enablement
- PeopleCert – ITIL Practice Guide: Incident Management

- PeopleCert – ITIL Practice Guide: Problem Management
- PeopleCert – ITIL Practice Guide: Knowledge Management
- ITIL Foundation – Version 5

Einordnung

Die dargestellten:

- Deployment-Methoden,
- Checklisten,
- Release-Plan-Inhalte,
- Kommunikationshinweise,
- Praxisbeispiele,
- und Nachbereitungsfragen

sind herstellerneutrale Praxisempfehlungen.

ITIL schreibt keine universelle:

- Deployment-Methode,
- Release-Kalender-Struktur,
- Release-Notes-Vorlage,
- Testtiefe,
- Rollout-Strategie,
- oder Toolauswahl

für alle Organisationen vor.

Die konkrete Umsetzung muss an:

- Services,
- Risiken,
- Benutzergruppen,
- technische Architektur,
- Change-Modell,
- Deployment-Werkzeuge,
- Sicherheitsanforderungen,
- Lieferanten,
- und organisatorische Fähigkeiten

angepasst werden.

Behandelter Framework-Stand: ITIL Version 5
Zusätzlich berücksichtigt: aktuelle ITIL-4-Practice-Guidance
Fachlicher Stand: August 2026
