1.8 Lösung umsetzen, Rückfallplan anwenden und Funktion verifizieren

Nachdem die Ursache ausreichend bestätigt wurde, kann eine geeignete Lösung geplant und umgesetzt werden.

Dabei gilt:

> Eine Störung gilt nicht als behoben, nur weil eine Fehlermeldung verschwunden ist. Die ursprüngliche Funktion, ihre Abhängigkeiten und mögliche Nebenwirkungen müssen anschließend geprüft werden.

Jede technische Korrektur ist selbst eine Änderung und kann neue Fehler verursachen. Deshalb benötigt auch eine scheinbar einfache Maßnahme einen definierten Ausgangszustand, Erfolgskriterien und einen Rückfallplan.

Ziel dieser Seite

Nach diesem Arbeitsschritt sollten:

- Sofortmaßnahme und dauerhafte Lösung getrennt sein,
- Ziel und Umfang der Änderung definiert sein,
- Risiken und Abhängigkeiten bekannt sein,
- Konfiguration oder Ausgangszustand gesichert sein,
- ein überprüfbarer Rückfallplan vorliegen,
- Erfolgskriterien und Abbruchkriterien definiert sein,
- die Änderung möglichst schrittweise erfolgen,
- technische Funktion und Benutzerfunktion geprüft sein,
- Monitoring und Protokolle nach der Änderung kontrolliert sein,
- bei Fehlschlag kontrolliert zurückgefallen werden können.

---

1. Kennzeichnung der Befehle

Kennzeichnung	Bedeutung
`[RO]`	Read-only: liest Informationen aus
`[TEST]`	Führt eine aktive Prüfung aus
`[FILE]`	Erstellt oder überschreibt eine Datei
`[PRIV]`	Benötigt möglicherweise Administrator- oder Root-Rechte

Kennzeichnung	Bedeutung
[CHANGE]	Verändert Konfiguration oder Betriebszustand
[DISRUPTIV]	Kann Benutzer oder produktive Dienste beeinträchtigen
[SENSITIV]	Ausgabe kann vertrauliche Daten enthalten

---

2. Maßnahmenarten unterscheiden

Maßnahmenart	Zweck	Beispiel
Sofortmaßnahme	Auswirkungen schnell begrenzen	fehlerhaftes Backend aus dem Load Balancer nehmen
Workaround	Fehler umgehen, ohne Ursache zu entfernen	Benutzer verwendet vorübergehend anderen Server
Mitigation	Auswirkung oder Wahrscheinlichkeit reduzieren	Traffic auf funktionierende Instanzen begrenzen
Reparatur	unmittelbaren technischen Fehler beseitigen	fehlerhafte Konfiguration korrigieren
Dauerhafte Lösung	bestätigte Grundursache beseitigen	automatische Konfigurationsvalidierung einführen
Präventionsmaßnahme	Wiederholung oder Auswirkung verhindern	Canary-Rollout und Monitoring ergänzen
Rollback	vorherigen bekannten Zustand wiederherstellen	vorherige Anwendungsversion aktivieren
Roll-forward	Fehler durch eine neue Korrektur beheben	korrigierte Folgeversion bereitstellen

Ein Workaround kann den Betrieb wiederherstellen, ohne die eigentliche Ursache zu beseitigen. Er muss deshalb als temporär gekennzeichnet und nachverfolgt werden.

---

3. Wiederherstellung und Ursachenbehebung unterscheiden

Wiederherstellung

Ziel ist, die betroffene Funktion möglichst schnell wieder bereitzustellen.

Beispiele:

- Dienst neu starten,
- fehlerhaftes Backend deaktivieren,
- Failover auslösen,
- vorherige Konfiguration wiederherstellen,
- Ersatzgerät verwenden,
- Benutzer auf ein anderes System umleiten.

Ursachenbehebung

Ziel ist, das erneute Auftreten zu verhindern.

Beispiele:

- Konfigurationsvalidierung ergänzen,
- Speicherleck durch Softwarekorrektur beseitigen,
- Monitoring erweitern,
- fehlerhafte Automatisierung korrigieren,
- Redundanz verbessern,
- unklare Berechtigungsstruktur bereinigen.

Eine Störung kann bereits wiederhergestellt sein, während die dauerhafte Ursachenbehebung noch offen ist.

---

4. Vor jeder Änderung zu klärende Fragen

- Was genau soll verändert werden?
- Welche bestätigte Ursache wird damit behandelt?
- Welches System und welche Komponente sind betroffen?
- Welche Benutzer oder Dienste können beeinträchtigt werden?
- Welche Abhängigkeiten bestehen?
- Ist die Änderung dokumentiert und freigegeben?
- Ist der aktuelle Zustand gesichert?
- Existiert ein getesteter Rückfallweg?
- Welche Erfolgskriterien gelten?
- Welche Abbruchkriterien gelten?
- Wie lange darf die Änderung dauern?
- Wer führt die Änderung durch?
- Wer beobachtet Monitoring und Protokolle?
- Wer entscheidet über einen Rollback?
- Wie werden betroffene Personen informiert?

---

5. Änderungsumfang möglichst klein halten

Eine Korrektur sollte nur die Komponente verändern, die für die bestätigte Ursache relevant ist.

Ungeeignet

- mehrere Firewall-Regeln gleichzeitig ändern,
- Anwendung und Betriebssystem gleichzeitig aktualisieren,
- Dienstkonto und Berechtigungen gleichzeitig verändern,
- mehrere Backends gleichzeitig neu konfigurieren,
- gleichzeitig DNS, Proxy und Routing anpassen.

Besser

1. eine klar definierte Änderung,
2. definierter Funktionstest,
3. Beobachtung,
4. nächste Änderung nur bei Bedarf.

Kleine Änderungen lassen sich leichter:

- überprüfen,
- zuordnen,
- zurücknehmen,
- dokumentieren,
- auf Nebenwirkungen untersuchen.

---

6. Risiko der Änderung bewerten

Risikofaktor	Niedriges Risiko	Hohes Risiko
Umfang	einzelner Testclient	gesamte Produktion
Reversibilität	einfache Konfigurationsrücknahme	irreversible Datenmigration
Verfügbarkeit	redundantes System	einzelner kritischer Server
Erfahrung	dokumentierte Standardänderung	erstmalige unbekannte Änderung
Abhängigkeiten	wenige bekannte Abhängigkeiten	viele unbekannte Abhängigkeiten
Daten	keine Datenänderung	Schema- oder Datenänderung
Zugang	lokaler Konsolenzugang	nur entfernte Verbindung
Testbarkeit	funktionierende Testumgebung	nur Produktion verfügbar
Beobachtung	vollständiges Monitoring	kaum Messwerte vorhanden
Zeitdruck	geplantes Wartungsfenster	ungeplanter kritischer Ausfall

---

7. Änderungsplan erstellen

Feld	Eintrag
Ticket- oder Change-Nummer	
bestätigte Ursache	
geplante Änderung	
technisches Ziel	
betroffene Systeme	
betroffene Benutzer	

Feld	Eintrag
Abhängigkeiten	
erwartete Unterbrechung	
Risiko	Niedrig / Mittel / Hoch
ausführende Person	
beobachtende Person	
Wartungsfenster	
Sicherung des Ausgangszustands	
Erfolgskriterien	
Abbruchkriterien	
Rückfallplan	
Rückfallentscheidung durch	
Kommunikationsweg	
Beobachtungsdauer	

---

8. Backup und Rollback unterscheiden

Begriff	Bedeutung
Backup	Sicherung von Daten oder Konfiguration
Restore	Wiederherstellung aus einem Backup
Rollback	Rückkehr zum vorherigen bekannten Betriebszustand
Snapshot	Momentaufnahme eines bestimmten Systems oder Datenträgers
Export	Sicherung einer Konfiguration in einem unterstützten Format
Roll-forward	Korrektur durch eine nachfolgende Version oder Änderung

Ein vorhandenes Backup ist noch kein vollständiger Rückfallplan.

Ein Rückfallplan beantwortet zusätzlich:

- Welche Sicherung wird verwendet?
- Wie wird sie zurückgespielt?
- Welche Dienste müssen vorher beendet werden?
- Welche Abhängigkeiten sind betroffen?
- Wie lange dauert die Wiederherstellung?
- Wie wird der Erfolg des Rollbacks geprüft?
- Was geschieht mit Daten, die nach der Änderung entstanden sind?

- Wer darf den Rollback freigeben?
- Was geschieht, wenn auch der Rollback fehlschlägt?

---

9. Grenzen von Snapshots beachten

Ein Snapshot ersetzt kein reguläres Backup und ist nicht für jede Anwendung als Rückfallmethode geeignet.

Besondere Vorsicht gilt unter anderem bei:

- Datenbanken,
- Active Directory,
- verteilten Dateisystemen,
- Clustern,
- Replikationssystemen,
- Transaktionssystemen,
- verschlüsselten Volumes,
- Anwendungen mit externen Abhängigkeiten.

Vor der Nutzung eines Snapshots muss geprüft werden:

- Unterstützt der Hersteller diese Wiederherstellung?
- Ist der Snapshot anwendungskonsistent?
- Welche externen Systeme wurden seitdem verändert?
- Entstehen Replikations- oder Transaktionsprobleme?
- Gehen nach dem Snapshot erzeugte Daten verloren?
- Existiert zusätzlich ein unabhängiges Backup?

---

10. Ausgangskonfiguration sichern

Für produktive Systeme sollte möglichst die vom Hersteller vorgesehene Export-, Backup- oder Versionsverwaltungsfunktion verwendet werden.

Einfache Dateikopien sind nur für einzelne Konfigurationsdateien geeignet, wenn Berechtigungen, Besitzer, ACLs, erweiterte Attribute und Anwendungsanforderungen berücksichtigt werden.

Windows-Beispiel für eine einzelne Konfigurationsdatei

```
$backupFile = Join-Path "<Zielpfad>" "config-$(Get-Date -Format 'yyyyMMdd-HHmmss').bak"

Copy-Item "<Konfigurationsdatei>" $backupFile

Get-FileHash -Path "<Konfigurationsdatei>", $backupFile -Algorithm SHA256
```

Kennzeichnung:

- `Copy-Item`: `[FILE]`
- `Get-FileHash`: `[RO]`

Die Prüfsummen sollten bei einer unveränderten Kopie identisch sein.

Linux-Beispiel für eine einzelne Konfigurationsdatei

```
backup_file="<Zielpfad>/config-$(date +%Y%m%d-%H%M%S).bak"

cp --preserve=all -- "<Konfigurationsdatei>" "$backup_file"

sha256sum "<Konfigurationsdatei>" "$backup_file"
```

`cp --preserve=all` ist eine GNU-Option und steht nicht auf jedem Unix-System zur Verfügung.

---

macOS-Beispiel für eine einzelne Konfigurationsdatei

```
backup_file="<Zielpfad>/config-$(date +%Y%m%d-%H%M%S).bak"

cp -p "<Konfigurationsdatei>" "$backup_file"

shasum -a 256 "<Konfigurationsdatei>" "$backup_file"
```

`cp -p` erhält wichtige Dateiattribute, garantiert aber nicht für jede Anwendung die vollständige Sicherung aller Metadaten. Für komplexe Anwendungen muss die vorgesehene Backupfunktion verwendet werden.

---

11. Konfiguration vor der Aktivierung validieren

Eine Konfigurationsdatei sollte möglichst vor einem Reload oder Neustart geprüft werden.

JSON unter Windows PowerShell prüfen

`[TEST] Get-Content "<Datei.json>" -Raw | ConvertFrom-Json | Out-Null`

Bei ungültigem JSON erzeugt PowerShell eine Fehlermeldung.

JSON unter Linux oder macOS mit Python prüfen

`[TEST] python3 -m json.tool "<Datei.json>" > /dev/null`

JSON mit jq prüfen, sofern installiert

`[TEST] jq empty "<Datei.json>"`

macOS-Property-List prüfen

`[TEST] plutil -lint "<Datei.plist>"`

systemd-Unit-Datei prüfen

`[TEST] systemd-analyze verify "<Datei.service>"`

NGINX-Konfiguration prüfen

`[TEST][PRIV] nginx -t`

Apache-Konfiguration prüfen

`[TEST][PRIV] apachectl configtest`

OpenSSH-Serverkonfiguration prüfen

`[TEST][PRIV] sshd -t`

> Die Werkzeuge prüfen unterschiedliche Aspekte. Ein erfolgreicher Syntax-Test bestätigt nicht automatisch, dass Netzwerkziele, Zugangsdaten, Zertifikate und abhängige Dienste funktionieren.

---

12. PowerShell-Änderung mit WhatIf vorprüfen

Einige PowerShell-Cmdlets unterstützen den allgemeinen Parameter `-WhatIf`.

Beispiel

`[TEST] Restart-Service -Name <Dienstname> -WhatIf`

PowerShell zeigt, welche Aktion vorgesehen ist, führt den Neustart aber nicht aus.

Wichtige Einschränkung

`-WhatIf`:

- simuliert nicht die Anwendung selbst,
- prüft nicht die spätere Funktionsfähigkeit,
- erkennt nicht jede Abhängigkeit,
- prüft nicht automatisch den Rückfallplan,

- funktioniert nur bei Befehlen, die `-WhatIf` unterstützen.

---

13. Dienständerungen unter Windows

Aktuellen Zustand prüfen

`[RO] Get-Service -Name <Dienstname>`

Abhängige Dienste prüfen

`[RO] Get-Service -Name <Dienstname> -DependentServices`

Benötigte Dienste prüfen

`[RO] Get-Service -Name <Dienstname> -RequiredServices`

Dienstkonfiguration prüfen

`[RO][SENSITIV] Get-CimInstance Win32_Service -Filter "Name='<Dienstname>'" | Select-Object Name, State, StartMode, StartName, PathName, ProcessId`

Neustart zunächst anzeigen

`[TEST] Restart-Service -Name <Dienstname> -WhatIf`

Dienst mit Bestätigungsabfrage neu starten

`[CHANGE][DISRUPTIV][PRIV] Restart-Service -Name <Dienstname> -Confirm`

Gestoppten Dienst starten

`[CHANGE][PRIV] Start-Service -Name <Dienstname> -Confirm`

Zustand danach prüfen

`[RO] Get-Service -Name <Dienstname>`

> `Restart-Service` sendet eine Stopp- und anschließend eine Startanforderung. War der Dienst bereits gestoppt, versucht das Cmdlet ihn zu starten. Vorher müssen abhängige Dienste und Auswirkungen geprüft werden.

---

14. Dienständerungen unter Linux

Aktuellen Zustand prüfen

`[RO] systemctl status <Dienstname>.service --no-pager`

Zustand kompakt prüfen

`[RO] systemctl is-active <Dienstname>.service`

Reload-Fähigkeit prüfen

`[RO] systemctl show <Dienstname>.service -p CanReload -p ActiveState -p SubState -p MainPID`

Konfiguration neu laden

`[CHANGE][PRIV] sudo systemctl reload <Dienstname>.service`

Ein Reload funktioniert nur, wenn der Dienst diese Funktion unterstützt.

Dienst neu starten

`[CHANGE][DISRUPTIV][PRIV] sudo systemctl restart <Dienstname>.service`

Nur dann neu starten, wenn der Dienst bereits läuft

`[CHANGE][DISRUPTIV][PRIV] sudo systemctl try-restart <Dienstname>.service`

Zustand danach prüfen

`[RO] systemctl status <Dienstname>.service --no-pager`

Neue Protokolle anzeigen

`[RO] journalctl -u <Dienstname>.service --since "5 minutes ago" --no-pager`

---

15. Dienständerungen unter macOS

macOS verwendet `launchd`. Für die Befehle muss der richtige Dienstkontext bekannt sein.

Systemdienst prüfen

`[RO][PRIV] launchctl print system/<Dienstlabel>`

Benutzerdienst prüfen

`[RO] launchctl print gui/$(id -u)/<Dienstlabel>`

Systemdienst beenden und neu starten

`[CHANGE][DISRUPTIV][PRIV] sudo launchctl kickstart -k system/<Dienstlabel>`

Benutzerdienst beenden und neu starten

`[CHANGE][DISRUPTIV] launchctl kickstart -k gui/$(id -u)/<Dienstlabel>`

Die Option `-k` beendet eine bereits laufende Instanz, bevor sie neu gestartet wird.

Protokolle danach prüfen

`[RO] log show --last 5m --predicate 'process == "<Prozessname>"' --style compact`

> Vor dem Einsatz muss die lokale Dokumentation mit `man launchctl` geprüft werden. System-, Benutzer- und GUI-Dienste verwenden unterschiedliche Domains. Ein falscher Dienstkontext kann dazu führen, dass nicht der erwartete Dienst angesprochen wird.

---

16. Reload, Restart und Reboot unterscheiden

Maßnahme	Wirkung	Typisches Risiko
Reload	Konfiguration wird im laufenden Prozess neu geladen	Niedrig bis mittel
Restart	Prozess wird beendet und neu gestartet	Mittel bis hoch
Reboot	gesamtes Betriebssystem wird neu gestartet	Hoch
Failover	Betrieb wechselt auf anderes System	Mittel bis hoch
Rollback	vorheriger Zustand wird wiederhergestellt	Abhängig von Daten und Anwendung

Sinnvolle Reihenfolge

1. anwendungseigene Konfigurationsprüfung,
2. Reload, wenn unterstützt,
3. gezielter Dienstneustart,
4. Failover oder Rollback,
5. vollständiger Systemneustart nur mit technischer Begründung.

Ein vollständiger Neustart sollte nicht als Standardlösung verwendet werden, wenn ein einzelner Dienst gezielt behandelt werden kann.

---

17. Änderungen schrittweise ausrollen

Verfahren	Beschreibung
Einzelner Testclient	Änderung zunächst auf einem ausgewählten Client
Einzelne Instanz	Änderung auf einem Server innerhalb einer redundanten Gruppe

Verfahren	Beschreibung
Canary	kleiner Anteil produktiver Systeme oder Anfragen
Rolling Deployment	Instanzen werden nacheinander geändert
Blue-Green	neue und alte Umgebung existieren parallel
Staged Rollout	Änderung wird in definierten Stufen erweitert
Wartungsgruppe	ausgewählte Benutzer oder Systeme werden zuerst umgestellt

Beispiel für einen stufenweisen Rollout

1. Testumgebung,
2. einzelne produktionsnahe Instanz,
3. ausgewählte Benutzer,
4. kleiner Produktionsanteil,
5. Monitoring auswerten,
6. weitere Instanzen,
7. vollständiger Rollout.

Nach jeder Stufe muss ein Entscheidungspunkt bestehen:

- fortsetzen,
- beobachten,
- stoppen,
- zurückfallen.

---

18. Canary und Kontrollgruppe vergleichen

Messwert	Kontrollgruppe	Canary	Bewertung
Fehlerrate	0,5 %	0,6 %	ähnlich
Antwortzeit	220 ms	225 ms	ähnlich
CPU-Auslastung	42 %	44 %	ähnlich
HTTP 500	2	3	beobachten
erfolgreiche Anmeldungen	99,5 %	99,4 %	ähnlich
neue Protokollfehler	0	0	unauffällig

Die Bewertung darf nicht nur auf dem Gesamtsystem erfolgen. Ein kleiner Canary kann in aggregierten Messwerten unsichtbar bleiben. Canary und Kontrollgruppe müssen getrennt ausgewertet werden.

---

19. Abbruchkriterien vor dem Rollout festlegen

Mögliche Abbruchkriterien:

- Dienst startet nicht,
- Health Check schlägt fehl,
- Benutzerfunktion funktioniert nicht,
- Fehlerquote steigt über den freigegebenen Grenzwert,
- Antwortzeit verschlechtert sich erheblich,
- Datenintegritätsfehler tritt auf,
- unerwartete Systeme sind betroffen,
- Administrationszugang geht verloren,
- Monitoring meldet kritische Fehler,
- Rückfallmöglichkeit ist nicht mehr gewährleistet,
- Wartungsfenster wird überschritten.

Schwellenwerte müssen aus SLA, SLO, Monitoring-Baseline oder dokumentierten Anforderungen stammen und dürfen nicht frei erfunden werden.

---

20. Technische Verifikation in mehreren Ebenen

Ebene	Zu prüfende Frage
Prozess	Läuft der erwartete Prozess?
Dienst	Meldet der Dienst einen gesunden Zustand?
Listener	Lauscht der erwartete Port?
Netzwerk	Ist das Ziel vom vorgesehenen Client erreichbar?
TLS	Funktionieren Zertifikat und Handshake?
Protokoll	Liefert der Dienst eine gültige Antwort?
Authentifizierung	Kann sich der Benutzer anmelden?
Autorisierung	Darf der Benutzer die vorgesehene Funktion verwenden?
Anwendung	Funktioniert die konkrete Aktion?
Daten	Werden Daten korrekt gelesen und geschrieben?
Abhängigkeiten	Funktionieren Datenbank, Storage, DNS und APIs?
Monitoring	Wird das System korrekt als gesund erkannt?
Protokolle	Entstehen keine neuen relevanten Fehler?
Redundanz	Sind verbleibende Instanzen und Failover weiterhin verfügbar?

---

21. Windows-Verifikation

Dienststatus

[RO] Get-Service -Name <Dienstname>

Prozess-ID des Dienstes

[RO] Get-CimInstance Win32_Service -Filter "Name='<Dienstname>'" | Select-Object Name, State, ProcessId

TCP-Listener

[RO][PRIV] Get-NetTCPConnection -State Listen -LocalPort <Port>

Remote-Porttest

[TEST] Test-NetConnection <Servername> -Port <Port> -InformationLevel Detailed

HTTP- oder HTTPS-Test

[TEST] curl.exe -sS -o NUL -w "HTTP=%{http_code} Ziel=%{remote_ip} Gesamt=%{time_total}\n" --connect-timeout 5 --max-time 15 https://<Servername>/

Neue Systemereignisse

[RO] Get-WinEvent -FilterHashtable @{LogName="System"; StartTime=(Get-Date).AddMinutes(-10)} | Select-Object TimeCreated, Id, ProviderName, LevelDisplayName, Message

Neue Anwendungsereignisse

[RO] Get-WinEvent -FilterHashtable @{LogName="Application"; StartTime=(Get-Date).AddMinutes(-10)} | Select-Object TimeCreated, Id, ProviderName, LevelDisplayName, Message

---

22. Linux-Verifikation

Dienststatus

[RO] systemctl status <Dienstname>.service --no-pager

Nur aktiven Zustand abfragen

[RO] systemctl is-active <Dienstname>.service

Hauptprozess-ID

[RO] systemctl show <Dienstname>.service -p MainPID

TCP-Listener

[RO][PRIV] ss -lntp | grep ":<Port>"

Remote-Porttest

[TEST] nc -vz -w 3 <Servername> <Port>

HTTP- oder HTTPS-Test

`[TEST] curl -sS -o /dev/null -w "HTTP=%{http_code} Ziel=%{remote_ip} Gesamt=%{time_total}\n" --connect-timeout 5 --max-time 15 https://<Servername>/`

Neue Dienstprotokolle

`[RO] journalctl -u <Dienstname>.service --since "10 minutes ago" --no-pager`

Neue Fehler des Systems

`[RO] journalctl --since "10 minutes ago" -p err --no-pager`

---

23. macOS-Verifikation

Systemdienststatus

`[RO][PRIV] launchctl print system/<Dienstlabel>`

Benutzerdienststatus

`[RO] launchctl print gui/$(id -u)/<Dienstlabel>`

TCP-Listener

`[RO][PRIV] lsof -nP -iTCP:<Port> -sTCP:LISTEN`

Remote-Porttest

`[TEST] nc -vz -w 3 <Servername> <Port>`

HTTP- oder HTTPS-Test

`[TEST] curl -sS -o /dev/null -w "HTTP=%{http_code} Ziel=%{remote_ip} Gesamt=%{time_total}\n" --connect-timeout 5 --max-time 15 https://<Servername>/`

Neue Prozessprotokolle

`[RO] log show --last 10m --predicate 'process == "<Prozessname>"' --style compact`

Neue Fehler und Faults

`[RO] log show --last 10m --predicate 'messageType == error OR messageType == fault' --style compact`

---

24. Ursprüngliche Benutzerfunktion prüfen

Eine technische Verifikation reicht nicht aus. Abschließend muss die ursprünglich gestörte Funktion getestet werden.

Beispiele

- Benutzer kann sich anmelden,
- Datei kann geöffnet werden,
- Datei kann gespeichert werden,
- E-Mail kann gesendet und empfangen werden,
- Druckauftrag wird vollständig ausgegeben,
- VPN-Verbindung wird aufgebaut,
- Anwendung kann Datenbankdaten lesen,
- Anwendung kann Daten speichern,
- Backupjob endet erfolgreich,
- Client erhält korrekte DHCP-Optionen,
- Gruppenrichtlinie wird angewendet,
- API liefert erwartete Daten.

Der Test sollte möglichst:

- vom ursprünglichen Quellsystem,
- mit dem ursprünglichen Benutzerkontext,
- über denselben Netzwerkpfad,
- mit derselben Zieladresse,
- mit derselben Aktion

durchgeführt werden.

---

25. Positive und negative Tests durchführen

Positiver Test

Prüft, ob eine erlaubte Funktion erfolgreich ist.

Beispiel:

`Ein berechtigter Benutzer kann eine Datei lesen.`

Negativer Test

Prüft, ob eine nicht erlaubte Funktion weiterhin verhindert wird.

Beispiel:

`Ein unberechtigter Benutzer kann die Datei weiterhin nicht lesen.`

Weitere negative Tests

- geschlossener Port bleibt geschlossen,
- nicht autorisierter Benutzer bleibt abgewiesen,

- deaktiviertes Konto kann sich nicht anmelden,
- abgelaufenes Token wird nicht akzeptiert,
- unerlaubtes VLAN erhält keinen Zugriff,
- alte verwundbare Schnittstelle bleibt deaktiviert.

Eine Lösung ist fehlerhaft, wenn sie die Funktion wiederherstellt, dabei aber Sicherheitskontrollen unbeabsichtigt entfernt.

---

26. Regressionstests durchführen

Ein Regressionstest prüft, ob bisher funktionierende Bereiche nach der Änderung weiterhin funktionieren.

Geänderte Komponente	Mögliche Regressionstests
Firewall	andere erlaubte und gesperrte Verbindungen
DNS	interne und externe Namensauflösung
DHCP	neue und bestehende Clients
Webserver	Anmeldung, Upload, Download, API
Datenbank	Lesen, Schreiben, Replikation, Backup
Dateiserver	Lesen, Schreiben, ACL, Freigaben
Active Directory	Anmeldung, Gruppenrichtlinie, Replikation
Proxy	erlaubte Seiten, Ausnahmen, Authentifizierung
Zertifikat	Browser, API, mobile Clients, alte Clients
Load Balancer	alle Backends, Health Checks, Sitzungen
Backup	Sicherung und testweise Wiederherstellung

---

27. Persistenz der Lösung prüfen

Manche Korrekturen funktionieren nur bis zum nächsten:

- Neustart,
- Dienstneustart,
- DHCP-Lease-Wechsel,
- Gruppenrichtlinien-Refresh,
- Konfigurationsdeployment,
- Container-Neustart,
- Failover,
- Update,
- Passwort- oder Zertifikatswechsel.

Zu prüfen ist:

- Wurde die richtige permanente Konfiguration geändert?
- Wird die Datei durch Automatisierung überschrieben?
- Stammt die Konfiguration aus einer zentralen Quelle?
- Ist die Änderung in Versionsverwaltung übernommen?
- Muss ein Deploymenttemplate angepasst werden?
- Existieren mehrere Konfigurationskopien?
- Wird beim Neustart eine alte Konfiguration geladen?

Ein Neustart zur Persistenzprüfung darf nur geplant durchgeführt werden, wenn er technisch erforderlich und freigegeben ist.

---

28. Monitoring nach der Änderung

Nach einer erfolgreichen Funktionsprüfung muss das System weiter beobachtet werden.

Zu beobachtende Werte

- Verfügbarkeit,
- Fehlerquote,
- Antwortzeit,
- CPU-Auslastung,
- Speichernutzung,
- Datenträgerbelegung,
- I/O-Wartezeit,
- Netzwerkfehler,
- Verbindungsanzahl,
- Warteschlangen,
- Datenbankverbindungen,
- Replikationszustand,
- Zertifikatsfehler,
- Anmeldefehler,
- Benutzerbeschwerden.

Die Beobachtungsdauer richtet sich nach dem Fehlerbild.

Fehlerart	Geeignete Beobachtung
dauerhafter Startfehler	unmittelbare Tests nach Start
sporadischer Fehler	mehrere typische Nutzungszyklen
nächtlicher Job	mindestens nächster geplanter Lauf
DHCP-Problem	mindestens ein Lease-Erneuerungszyklus
Backupfehler	vollständiger Sicherungs- und Prüfzyklus

Fehlerart	Geeignete Beobachtung
Lastproblem	typischer Lastzeitraum
Zertifikatserneuerung	vollständige Kette und betroffene Clients
Replikationsproblem	vorgesehener Replikationszyklus

---

29. Rollback-Auslöser

Ein Rollback sollte ausgelöst werden, wenn:

- definiertes Erfolgskriterium nicht erreicht wird,
- ursprünglicher Fehler bestehen bleibt,
- neue kritische Fehler entstehen,
- unerwartete Systeme betroffen sind,
- Sicherheitskontrolle nicht mehr funktioniert,
- Datenintegrität gefährdet ist,
- Administrationszugang verloren zu gehen droht,
- Abbruchkriterium erreicht wird,
- Wartungsfenster überschritten wird,
- weitere sichere Diagnose nicht möglich ist.

Die Entscheidung darf nicht erst improvisiert werden, nachdem ein Problem entstanden ist.

---

30. Kontrollierter Rollback-Ablauf

1. weiteren Rollout stoppen,
2. zuständige Personen informieren,
3. aktuellen Fehlerzustand sichern,
4. Rückfallentscheidung dokumentieren,
5. freigegebenen Rückfallplan ausführen,
6. Dienst- und Systemstatus prüfen,
7. ursprüngliche Funktion testen,
8. negative Tests durchführen,
9. Protokolle und Monitoring kontrollieren,
10. Ergebnis dokumentieren,
11. weitere Untersuchung planen.

> Auch ein Rollback ist eine Änderung und kann fehlschlagen.

---

31. Rollback oder Roll-forward wählen

Situation	Rollback eher geeignet	Roll-forward eher geeignet
vorherige Version bekannt funktionsfähig	Ja	Möglich
Konfiguration einfach reversibel	Ja	Möglich
Datenbankschema bereits verändert	Riskant	Häufig besser
neue Daten nicht rückwärtskompatibel	Riskant	Häufig besser
kritische Sicherheitslücke würde wieder geöffnet	Eher ungeeignet	Besser
Fehlerkorrektur ist klein und geprüft	Möglich	Häufig besser
vorheriger Zustand nicht eindeutig	Riskant	Möglicherweise besser
Rollback bereits getestet	Ja	Nicht zwingend
Zeit bis zur Folgekorrektur sehr lang	Häufig besser	Eher ungeeignet
Produktionsauswirkung steigt schnell	schneller sicherer Weg entscheidend	schneller sicherer Weg entscheidend

Ein Rollback darf keine bekannte Sicherheitslücke oder Dateninkonsistenz unbewertet wieder einführen.

---

32. Rollback einfacher Konfigurationsdateien

Eine Konfigurationsdatei darf nur dann direkt zurückkopiert werden, wenn:

- die Sicherung eindeutig zugeordnet ist,
- die Prüfsumme geprüft wurde,
- keine neuen relevanten Änderungen verloren gehen,
- Dateibesitzer und Berechtigungen bekannt sind,
- keine Datenmigration betroffen ist,
- die Anwendung dieses Verfahren unterstützt,
- der Dienst anschließend korrekt validiert werden kann.

Windows-Beispiel

`[CHANGE][PRIV] Copy-Item "<Sicherungsdatei>" "<Konfigurationsdatei>" -Force`

Linux-Beispiel

`[CHANGE][PRIV] sudo cp --preserve=all -- "<Sicherungsdatei>" "<Konfigurationsdatei>"`

macOS-Beispiel

`[CHANGE][PRIV] sudo cp -p "<Sicherungsdatei>" "<Konfigurationsdatei>"`

> Diese Befehle sind keine universelle Rollbacklösung. Datenbanken, Verzeichnisdienste, Cluster, Containerplattformen und komplexe Anwendungen benötigen die vom Hersteller vorgesehene Wiederherstellungsmethode.

Nach dem Zurückkopieren müssen mindestens geprüft werden:

- Inhalt,
- Prüfsumme,
- Besitzer,
- Gruppe,
- Berechtigungen,
- ACL,
- Syntax,
- Dienststart,
- Funktion.

---

33. Kommunikation während der Umsetzung

Zeitpunkt	Information
Vor Beginn	Umfang, erwartete Auswirkung und Zeitraum
Zu Beginn	Änderung wurde gestartet
Währenddessen	relevante Abweichungen oder Verzögerungen
Bei Abbruch	Grund und eingeleitete Rückfallmaßnahme
Nach Erfolg	Funktion wiederhergestellt und Beobachtung läuft
Nach Rollback	vorheriger Zustand wiederhergestellt
Nach Abschluss	Ergebnis, offene Punkte und Präventionsmaßnahmen

Technische Kommunikation sollte sachlich sein und zwischen folgenden Zuständen unterscheiden:

- Störung erkannt,
- Ursache vermutet,
- Ursache bestätigt,
- Workaround aktiv,
- Dienst wiederhergestellt,
- dauerhafte Lösung umgesetzt,
- Beobachtung läuft,
- Störung abgeschlossen.

---

34. Vollständiges Praxisbeispiel

Störung

Ein Backend liefert HTTP `503`, weil die Anwendung einen falschen Datenbank-Hostnamen verwendet.

Bestätigte Ursache

Die neue Konfigurationsdatei enthält `db-prdo.example` statt `db-prod.example`.

Geplante Lösung

- Konfiguration korrigieren,
- Syntax validieren,
- Änderung zunächst auf einem Backend anwenden,
- Dienst neu laden oder neu starten,
- technische und fachliche Funktion prüfen,
- danach weitere Instanzen ändern.

Rückfallplan

- gesicherte vorherige Konfiguration zurückspielen,
- Dienst erneut starten,
- Backend aus Load-Balancer-Rotation entfernen, falls es weiterhin fehlerhaft ist.

Erfolgskriterien

- Dienststatus aktiv,
- TCP-Listener vorhanden,
- Datenbankverbindung erfolgreich,
- HTTP-Health-Check erfolgreich,
- Anmeldung funktioniert,
- keine neuen Fehler im Protokoll,
- Fehlerrate entspricht der Kontrollgruppe.

Abbruchkriterien

- Dienst startet nicht,
- Datenbankverbindung bleibt gestört,
- neue Fehler treten auf,
- Antwortzeit verschlechtert sich deutlich,
- Benutzerfunktion schlägt fehl.

Ablauf

Schritt	Maßnahme
1	Ausgangszustand und Protokolle sichern

Schritt	Maßnahme
2	aktuelle Konfiguration sichern und Prüfsumme bilden
3	korrigierte Datei in Testumgebung validieren
4	ein Backend aus der Rotation nehmen
5	Konfiguration auf diesem Backend ändern
6	Konfigurationsprüfung ausführen
7	Dienst kontrolliert neu laden oder starten
8	Listener und Protokolle prüfen
9	Datenbankfunktion prüfen
10	HTTP- und Anmeldetest durchführen
11	Backend mit Kontrollgruppe vergleichen
12	Backend wieder in Rotation nehmen
13	Monitoring beobachten
14	Änderung schrittweise auf weitere Instanzen ausrollen

35. Verifikationsprotokoll

Feld	Vor Änderung	Nach Änderung	Nach Beobachtung
Zeitpunkt			
Dienststatus			
Prozess-ID			
Listener			
Ziel-IP			
HTTP-Status			
Antwortzeit			
Authentifizierung			
Benutzerfunktion			
Datenbankverbindung			
Fehler im Protokoll			
Monitoringstatus			
Fehlerrate			
Nebenwirkungen			

36. Änderungs- und Rollbackprotokoll

Feld	Eintrag
Ticket- oder Change-Nummer	
bestätigte Ursache	
umgesetzte Änderung	
Beginn	
Ende	
ausführende Person	
betroffene Systeme	
Sicherungsdatei oder Backup	
Prüfsumme	
Konfigurationsprüfung	Bestanden / Nicht bestanden
Dienstneustart erforderlich	Ja / Nein
Erfolgskriterien erreicht	Ja / Nein / Teilweise
Abbruchkriterium erreicht	Ja / Nein
Rollback durchgeführt	Ja / Nein
Rollback-Ergebnis	
Benutzerfunktion geprüft	Ja / Nein
Regressionstests durchgeführt	
Beobachtungsdauer	
neue Fehler	
verbleibende Risiken	
offene Maßnahmen	

37. Häufige Fehler bei der Lösungsumsetzung

Fehler	Folge	Besseres Vorgehen
Keine Sicherung	Ausgangszustand kann nicht wiederhergestellt werden	Konfiguration oder Backup vorher sichern
Backup mit Rollback verwechselt	Wiederherstellung ist nicht geplant	vollständigen Rückfallablauf dokumentieren
Mehrere Änderungen gleichzeitig	Wirkung nicht zuordenbar	kleine einzelne Änderungen
Sofortiger vollständiger Rollout	großer möglicher Ausfall	Canary oder stufenweiser Rollout

Fehler	Folge	Besseres Vorgehen
Nur Dienststatus geprüft	Benutzerfunktion kann weiterhin gestört sein	Ende-zu-Ende-Test durchführen
Nur positiver Test	Sicherheitskontrolle kann unbemerkt fehlen	negativen Test ergänzen
Keine Abbruchkriterien	zu lange an fehlerhafter Änderung festgehalten	Grenzwerte vorher definieren
Rollback erst im Fehlerfall geplant	zusätzliche Ausfallzeit	Rollback vor Beginn vorbereiten
Snapshot als einziges Backup	Wiederherstellung möglicherweise ungeeignet	anwendungsgerechtes Backup verwenden
Neustart als Standardlösung	unnötige Unterbrechung und Informationsverlust	kleinste geeignete Maßnahme wählen
Monitoring sofort beendet	sporadischer Fehler bleibt unentdeckt	angemessene Beobachtungsdauer
Workaround als dauerhafte Lösung behandelt	Grundursache bleibt bestehen	Folgeaufgabe mit Verantwortlichkeit
Sicherheitsupdate zurückgenommen	bekannte Schwachstelle erneut aktiv	Sicherheitsrisiko bewerten und Roll-forward prüfen
Dokumentation erst später erstellt	genaue Schritte und Zeiten gehen verloren	während der Änderung dokumentieren

Kurzcheckliste

- [ ] Ursache ausreichend bestätigt
- [ ] Sofortmaßnahme und dauerhafte Lösung getrennt
- [ ] Änderungsumfang eindeutig definiert
- [ ] Auswirkungen und Abhängigkeiten geprüft
- [ ] Risiko bewertet
- [ ] Freigabe eingeholt
- [ ] Wartungsfenster geklärt
- [ ] Ausgangszustand dokumentiert
- [ ] Konfiguration oder Daten gesichert
- [ ] Prüfsummen dokumentiert
- [ ] Rückfallplan vorhanden
- [ ] Rückfallplan technisch durchführbar
- [ ] Erfolgskriterien festgelegt
- [ ] Abbruchkriterien festgelegt
- [ ] Monitoring vorbereitet

- [ ] Kommunikation vorbereitet
- [ ] Konfiguration vor Aktivierung validiert
- [ ] Änderung möglichst klein gehalten
- [ ] Änderung zunächst begrenzt ausgerollt
- [ ] Dienststatus geprüft
- [ ] Prozess und Listener geprüft
- [ ] Netzwerk- und Protokolltest durchgeführt
- [ ] ursprüngliche Benutzerfunktion geprüft
- [ ] negative Sicherheitstests durchgeführt
- [ ] Regressionstests durchgeführt
- [ ] Protokolle nach der Änderung geprüft
- [ ] Monitoring nach der Änderung beobachtet
- [ ] Persistenz der Lösung berücksichtigt
- [ ] Rollback bei Erreichen eines Abbruchkriteriums durchgeführt
- [ ] Ergebnis und offene Maßnahmen dokumentiert

Ergebnis dieses Arbeitsschrittes

Am Ende dieses Schrittes ist die bestätigte Ursache durch eine kontrollierte und dokumentierte Änderung behandelt.

Die Lösung gilt erst als erfolgreich, wenn:

- technische Prüfungen bestanden sind,
- die ursprüngliche Benutzerfunktion funktioniert,
- abhängige Funktionen weiterhin funktionieren,
- Sicherheitskontrollen weiterhin wirksam sind,
- keine neuen relevanten Fehler auftreten,
- Monitoring einen stabilen Zustand zeigt,
- die Änderung dauerhaft bestehen bleibt,
- Rückfall- oder Folgemaßnahmen dokumentiert sind.

Nächste Seite:
1.9 Störung dokumentieren, abschließen und Wiederholung verhindern

---

Offizielle Hersteller-, Standard- und Projektdokumentation

- Microsoft Learn – Restart-Service
- Microsoft Learn – Start-Service
- Microsoft Learn – Stop-Service

- Microsoft Learn – Get-Service
- Microsoft Learn – Test-NetConnection
- Microsoft Learn – Get-WinEvent
- Microsoft Learn – Get-FileHash
- freedesktop.org – systemctl
- freedesktop.org – systemd-analyze
- freedesktop.org – journalctl
- Google SRE Workbook – Canarying Releases
- Google SRE Workbook – Configuration Design and Best Practices
- Apple Support – Scriptverwaltung mit launchd
- curl – Offizielle Befehlsreferenz
