1.7 Ursache bestätigen und alternative Erklärungen ausschließen

Eine plausible Hypothese ist noch keine bestätigte Ursache. Auch wenn eine Maßnahme scheinbar erfolgreich war, kann ein anderer gleichzeitig veränderter Zustand für die Wiederherstellung verantwortlich gewesen sein.

Deshalb gilt:

> Eine Ursache muss das Fehlerbild technisch erklären, zu Umfang und Zeitpunkt passen und durch Belege oder Gegenproben gestützt werden.

In komplexen IT-Systemen existiert häufig nicht nur eine einzelne Ursache. Meist wirken Auslöser, technische Fehler, Abhängigkeiten und begünstigende Bedingungen zusammen.

Ziel dieser Seite

Nach diesem Arbeitsschritt sollten:

- Symptom und Ursache getrennt sein,
- unmittelbare Ursache und Grundursache unterschieden sein,
- Auslöser und begünstigende Faktoren identifiziert sein,
- abhängige Systeme berücksichtigt sein,
- widersprechende Beobachtungen geprüft sein,
- alternative Erklärungen bewertet sein,
- die Ursache mit mehreren Belegen bestätigt sein,
- verbleibende Unsicherheiten dokumentiert sein.

---

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
[CHANGE]	Verändert Konfiguration oder Betriebszustand
[DISRUPTIV]	Kann Benutzer oder produktive Dienste beeinträchtigen

Kennzeichnung	Bedeutung
[SENSITIV]	Ausgabe kann vertrauliche Daten enthalten

2. Symptom, unmittelbare Ursache und Grundursache unterscheiden

Ebene	Bedeutung	Beispiel
Symptom	Sichtbare Auswirkung	Webanwendung liefert HTTP 503
Unmittelbare technische Ursache	Zustand, der das Symptom direkt erzeugt	Backend kann keine Datenbankverbindung aufbauen
Auslöser	Ereignis, das den Fehler aktiviert hat	Neue Konfiguration wurde bereitgestellt
Grundursache	Systemischer Grund, warum der Fehler möglich war	Pipeline prüfte den Datenbank-Hostnamen nicht
Begünstigender Faktor	Bedingung, die Entstehung oder Auswirkung verstärkte	Änderung wurde gleichzeitig auf allen Backends verteilt
Erkennungslücke	Grund für eine verspätete Erkennung	Monitoring prüfte nur, ob TCP 443 erreichbar war
Wiederherstellungsfaktor	Maßnahme, die den Betrieb wiederherstellte	Konfigurationsrollback
Präventionsmaßnahme	Maßnahme gegen eine Wiederholung	Validierung und schrittweise Bereitstellung einführen

3. Beispiel einer vollständigen Ursachenkette

Ebene	Feststellung
Symptom	Benutzer erhalten HTTP 503
Direkte Ursache	Webanwendung erreicht die Datenbank nicht
Technischer Mechanismus	Falscher Datenbank-Hostname in der Anwendungskonfiguration
Auslöser	Deployment um 14:31 Uhr
Grundursache	Deployment-Pipeline akzeptiert ungeprüfte Konfigurationswerte
Begünstigender Faktor	Änderung wurde auf allen Instanzen gleichzeitig ausgerollt
Erkennungslücke	Kein Ende-zu-Ende-Monitoring der Anmeldung
Wiederherstellung	Vorherige Konfiguration wurde wiederhergestellt
Prävention	Schema-Validierung, Staging-Test und schrittweiser Rollout

Die Aussage „Das Deployment war schuld“ wäre zu ungenau. Erst die vollständige Ursachenkette zeigt, was technisch fehlschlug und wodurch eine Wiederholung verhindert werden kann.

---

4. Nicht vorschnell nach einer einzigen Ursache suchen

Komplexe Störungen entstehen häufig durch eine Kombination mehrerer Bedingungen.

Beispiel

Ein einzelnes fehlerhaftes Backend führt nicht zwangsläufig zu einem Gesamtausfall. Der Ausfall entsteht möglicherweise erst, weil zusätzlich:

- der Load Balancer fehlerhafte Health Checks verwendet,
- alle Anfragen zum fehlerhaften Backend geleitet werden,
- die verbleibenden Backends überlastet werden,
- kein automatisches Failover erfolgt,
- Monitoring die wachsende Fehlerquote nicht erkennt.

In diesem Fall gibt es mehrere relevante Ursachen und beitragende Faktoren.

Zu dokumentierende Ursachenarten

- technische Ursache,
- Konfigurationsursache,
- Prozessursache,
- organisatorische Ursache,
- fehlende Schutzmaßnahme,
- fehlende Erkennung,
- fehlende oder unwirksame Wiederherstellung.

---

5. Eine Ursache gilt nicht allein durch zeitliche Nähe als bestätigt

Unzureichende Aussage

`Der Fehler begann nach dem Update, also war das Update die Ursache.`

Erforderliche Zusatzfragen

- Welche konkrete Komponente wurde verändert?
- Passt diese Komponente technisch zum Fehlerbild?
- Sind nur aktualisierte Systeme betroffen?
- Funktionieren identisch aktualisierte Vergleichssysteme?
- Gibt es passende Protokolleinträge?
- Verändert ein Rollback das Ergebnis?
- Könnte der Neustart während des Updates entscheidend gewesen sein?
- Wurden gleichzeitig weitere Änderungen ausgeführt?

- Trat der Fehler möglicherweise bereits vor dem Update auf?

---

6. Stärke von Belegen bewerten

Beleg	Aussagekraft
Vermutung ohne Messwert	Sehr gering
Einzelne Benutzerangabe	Gering
Zeitliche Nähe	Gering bis mittel
Passende Fehlermeldung	Mittel
Reproduzierbares Muster	Mittel bis hoch
Konfigurationsunterschied zum funktionierenden System	Hoch
Protokoll zeigt direkten technischen Fehler	Hoch
Paketaufzeichnung bestätigt Kommunikationsfehler	Hoch
Fehler verschwindet nach kontrollierter Entfernung der Ursache	Sehr hoch
Fehler erscheint in Testumgebung nach kontrollierter Wiederherstellung der Ursache erneut	Sehr hoch
Mehrere voneinander unabhängige Belege stimmen überein	Sehr hoch

Eine zuverlässige Bestätigung verwendet möglichst mehrere voneinander unabhängige Belege.

---

7. Kriterien für eine bestätigte Ursache

Eine Ursache sollte möglichst folgende Fragen beantworten:

Kriterium	Frage
Technischer Mechanismus	Wie erzeugt die Ursache das beobachtete Symptom?
Fehlerumfang	Warum sind genau diese Systeme oder Benutzer betroffen?
Zeitlicher Zusammenhang	Warum begann der Fehler zu diesem Zeitpunkt?
Reproduzierbarkeit	Lässt sich das Verhalten kontrolliert wiederholen?
Vergleich	Warum funktionieren nicht betroffene Systeme?
Gegenprobe	Verschwindet das Symptom ohne die vermutete Ursache?
Alternativen	Welche anderen Erklärungen wurden geprüft?
Widersprüche	Gibt es Fakten, die nicht zur Ursache passen?
Beweislage	Welche Logs, Messwerte oder Vergleiche bestätigen sie?

Kriterium	Frage
Wiederholungsschutz	Welche Maßnahme verhindert ein erneutes Auftreten?

Wenn eine vermutete Ursache den Umfang oder den technischen Mechanismus nicht erklären kann, ist sie wahrscheinlich unvollständig.

---

8. Die Gegenfrage stellen

Eine besonders wichtige Frage lautet:

> Was müsste beobachtet werden, wenn die vermutete Ursache nicht stimmt?

Beispiel

Hypothese:

`Eine Firewall-Regel blockiert TCP 443 aus VLAN 30.`

Mögliche Gegenbelege:

- Pakete aus VLAN 30 erreichen den Server nachweislich.
- Der Server sendet eine Antwort zurück.
- Der Fehler tritt auch innerhalb des Servernetzes auf.
- Ein lokaler Test auf dem Server schlägt ebenfalls fehl.
- Die Firewall protokolliert eine erlaubte Verbindung.
- Andere Dienste über denselben Netzwerkpfad sind ebenfalls betroffen.
- Der Fehler besteht nach Rücknahme der Firewall-Regel weiter.

Gegenbelege müssen genauso sorgfältig dokumentiert werden wie unterstützende Belege.

---

9. Kontrafaktische Prüfung

Eine kontrafaktische Prüfung untersucht, was ohne die vermutete Ursache geschehen würde.

Frage	Bedeutung
Wäre die Störung ohne diese Bedingung aufgetreten?	Prüft die Notwendigkeit der Ursache
Existiert dieselbe Bedingung auf funktionierenden Systemen?	Prüft, ob sie allein ausreichend ist
Verschwindet der Fehler, wenn die Bedingung entfernt wird?	Prüft den direkten Zusammenhang
Erscheint der Fehler in einer Testumgebung erneut?	Prüft Reproduzierbarkeit

Frage	Bedeutung
Erklärt die Ursache auch die nicht betroffenen Systeme?	Prüft den Fehlerumfang
Gibt es einen anderen Faktor mit derselben Wirkung?	Prüft alternative Erklärungen

Eine Bedingung kann notwendig, aber allein nicht ausreichend sein. Beispielsweise kann ein fehlerhaftes Backend erst zusammen mit einer fehlerhaften Load-Balancer-Konfiguration einen vollständigen Ausfall verursachen.

---

10. Ursachenbestätigung mit A/B-Vergleich

Zustand A	Zustand B	Mögliche Interpretation
Betroffenes System	Funktionierendes System	Unterschiede priorisieren
Fehlerhafte Konfiguration	Bekannte funktionierende Konfiguration	Konfigurationsunterschied prüfen
Benutzer A gestört	Benutzer B funktioniert	Konto oder Berechtigung prüfen
VLAN 30 gestört	VLAN 40 funktioniert	Netzwerkpfad oder ACL prüfen
Backend 1 funktioniert	Backend 2 gestört	Backend-spezifische Ursache
Anwendungsversion alt funktioniert	neue Version gestört	Versionsänderung priorisieren
Proxy verwendet	direkter Test funktioniert	Proxy oder Proxyrichtlinie prüfen
DNS-Auflösung verwendet	festes Ziel funktioniert	DNS oder Zielauswahl prüfen

Der Vergleich ist nur aussagekräftig, wenn alle nicht untersuchten Bedingungen möglichst gleich bleiben.

---

11. Betroffenes und funktionierendes System vergleichen

Zu vergleichende Eigenschaften

Bereich	Beispiele
Betriebssystem	Version, Build, Kernel
Updates	Patchstand und Installationszeit
Anwendung	Version, Module, Erweiterungen
Konfiguration	Inhalt, Prüfsumme, Änderungszeit
Dienste	Status, Starttyp, Dienstkonto
Netzwerk	IP, VLAN, Gateway, DNS, Proxy
Sicherheit	Firewallprofil, EDR, Zertifikate
Benutzer	Gruppen, Rollen, Richtlinien

Bereich	Beispiele
Dateien	Version, Besitzer, Rechte, ACL
Abhängigkeiten	Datenbank, API, Storage, DNS
Umgebung	Variablen, Pfade, Laufzeitversion
Ressourcen	CPU, RAM, Datenträger, Quota

---

12. Dateien und Konfigurationen unter Windows vergleichen

Dateiinhalte vergleichen

`[RO] Compare-Object (Get-Content "<Funktionierende-Datei>") (Get-Content "<Betroffene-Datei>")`

SHA-256-Prüfsumme bilden

`[RO] Get-FileHash "<Datei>" -Algorithm SHA256`

Dateiinformationen anzeigen

`[RO] Get-Item "<Datei>" | Select-Object FullName, Length, CreationTime, LastWriteTime`

Versionsinformationen einer ausführbaren Datei

`[RO] (Get-Item "<Dateipfad>").VersionInfo`

Berechtigungen anzeigen

`[RO][SENSITIV] Get-Acl "<Pfad>" | Format-List`

Installierte Hotfixes vergleichen

`[RO] Get-HotFix | Sort-Object InstalledOn -Descending`

---

13. Dateien und Konfigurationen unter Linux vergleichen

Dateien zeilenweise vergleichen

`[RO] diff -u "<Funktionierende-Datei>" "<Betroffene-Datei>"`

SHA-256-Prüfsumme bilden

`[RO] sha256sum "<Datei>"`

Dateiinformationen anzeigen

`[RO] stat "<Datei>"`

Dateityp bestimmen

`[RO] file "<Datei>"`

Berechtigungen und ACL anzeigen

`[RO][SENSITIV] getfacl "<Datei>"`

Debian- oder Ubuntu-Paketversion

`[RO] dpkg-query -W "<Paketname>"`

RHEL-, Rocky-, AlmaLinux- oder Fedora-Paketversion

`[RO] rpm -q "<Paketname>"`

---

14. Dateien und Konfigurationen unter macOS vergleichen

Dateien zeilenweise vergleichen

`[RO] diff -u "<Funktionierende-Datei>" "<Betroffene-Datei>"`

SHA-256-Prüfsumme bilden

`[RO] shasum -a 256 "<Datei>"`

Dateiinformationen anzeigen

`[RO] stat -x "<Datei>"`

Dateityp bestimmen

`[RO] file "<Datei>"`

Berechtigungen und ACL anzeigen

`[RO][SENSITIV] ls -lde "<Datei>"`

Anwendungsversion über Systeminformationen suchen

`[RO] system_profiler SPApplicationsDataType`

Bei umfangreichen Installationen kann `system_profiler SPApplicationsDataType` längere Zeit benötigen und eine große Ausgabe erzeugen.

---

15. Prüfsummen richtig interpretieren

Ergebnis	Bedeutung

Prüfsummen identisch	Dateien sind zum Prüfzeitpunkt binär identisch
Prüfsummen unterschiedlich	Mindestens ein Byte unterscheidet sich
Dateiinhalt gleich, Metadaten anders	Besitzer, Rechte oder Zeitstempel können abweichen
Anwendungsversion gleich, Prüfsumme anders	anderer Build, beschädigte Datei oder Modifikation möglich
Prüfsumme gleich, Verhalten anders	Ursache liegt wahrscheinlich außerhalb der Datei

Eine identische Prüfsumme beweist nicht, dass die gesamte Umgebung identisch ist.

---

16. Dienstabhängigkeiten unter Windows prüfen

Benötigte Dienste anzeigen

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

Abhängige Dienste anzeigen

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

Dienstkonfiguration anzeigen

`[RO] sc.exe qc <Dienstname>`

Erweiterte Dienstinformationen

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

Dienstereignisse anzeigen

`[RO] Get-WinEvent -FilterHashtable @{LogName="System"; ProviderName="Service Control Manager"; StartTime=(Get-Date).AddHours(-1)} | Select-Object TimeCreated, Id, Message`

Listener eines Dienstes prüfen

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

Prozess zum Listener bestimmen

`[RO] Get-Process -Id <OwningProcess>`

---

17. Dienstabhängigkeiten unter Linux prüfen

Abhängigkeiten eines Dienstes anzeigen

`[RO] systemctl list-dependencies <Dienstname>.service --all`

Dienste anzeigen, die vom untersuchten Dienst abhängen

`[RO] systemctl list-dependencies <Dienstname>.service --reverse --all`

Deklarierte Abhängigkeiten und Reihenfolge

`[RO] systemctl show <Dienstname>.service -p Requires -p Wants -p After -p Before`

Effektive Unit-Konfiguration anzeigen

`[RO] systemctl cat <Dienstname>.service`

Dienstzustand anzeigen

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

Dienstprotokoll anzeigen

`[RO] journalctl -u <Dienstname>.service --since "1 hour ago" --no-pager`

Listener prüfen

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

Prozess untersuchen

`[RO] ps -p <PID> -o pid,ppid,user,lstart,stat,%cpu,%mem,command`

---

18. Dienstzustand unter macOS prüfen

macOS verwendet `launchd`. Die richtige Dienst-Domain hängt davon ab, ob es sich um einen System-, Benutzer- oder GUI-Dienst handelt.

Geladene Dienste anzeigen

`[RO] launchctl list`

Systemdienst untersuchen

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

Benutzerdienst untersuchen

`[RO] launchctl print gui/<Benutzer-ID>/<Dienstlabel>`

Die Benutzer-ID kann mit folgendem Befehl ermittelt werden:

`[RO] id -u`

Property-List anzeigen

`[RO] plutil -p "<Pfad-zur-plist-Datei>"`

Listener prüfen

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

Prozess untersuchen

`[RO] ps -p <PID> -o pid,ppid,user,lstart,state,%cpu,%mem,command`

`launchd` bildet Abhängigkeiten nicht in derselben Weise wie systemd ab. Deshalb müssen zusätzlich verwendete Sockets, Dateien, Netzwerkziele und Anwendungsprotokolle geprüft werden.

---

19. Technische Abhängigkeiten vollständig betrachten

Ein sichtbarer Dienst kann von vielen weiteren Komponenten abhängen.

Komponente	Mögliche Abhängigkeit
Client	DNS, Proxy, Zertifikat, Richtlinie
DNS	Zone, Forwarder, Netzwerk, Root-Server
Webserver	Zertifikat, Dateisystem, Backend
Reverse Proxy	Upstream, Health Check, DNS
Anwendung	Datenbank, Cache, Queue, API
Datenbank	Storage, Speicher, Replikation
Authentifizierung	Active Directory, LDAP, RADIUS, MFA
Zertifikat	CA, Zwischenzertifikat, Uhrzeit, Sperrprüfung
Dateifreigabe	DNS, Kerberos, Berechtigung, Storage
Container	Netzwerk, Volume, Secret, Image
Virtuelle Maschine	Hypervisor, Datastore, virtuelles Netzwerk
Cloud-Dienst	IAM, Sicherheitsgruppe, Region, Provider
Backup	Agent, Repository, Netzwerk, Dienstkonto
Monitoring	Agent, Collector, Zeit, Datenbank

---

20. Abhängigkeitskette dokumentieren

Beispiel: Anmeldung an einer Webanwendung

Reihenfolge	Komponente	Erforderliche Funktion

1	Client	Netzwerkverbindung vorhanden
2	DNS	Hostname wird korrekt aufgelöst
3	Firewall	TCP 443 wird erlaubt
4	Load Balancer	funktionsfähiges Backend wird gewählt
5	Webserver	TLS und HTTP funktionieren
6	Anwendung	Anmeldeanfrage wird verarbeitet
7	Identitätsdienst	Benutzer wird authentifiziert
8	Datenbank	Benutzer- und Sitzungsdaten sind verfügbar
9	Anwendung	Rolle wird geprüft
10	Browser	Sitzungscookie wird gespeichert

Ein Fehler in einer späteren Abhängigkeit kann nach außen wie ein allgemeiner „Webserverfehler“ wirken.

---

21. Abhängigkeit gezielt umgehen oder isolieren

Eine Komponente kann für einen Test kontrolliert aus dem Pfad genommen werden. Dies darf keine dauerhafte oder unautorisierte Umgehung von Sicherheitskontrollen werden.

DNS für einen einzelnen curl-Test umgehen

`[TEST] curl --resolve "server.example:443:<IP-Adresse>" -v https://server.example/`

Dadurch bleiben Hostname, HTTP-Host-Header, SNI und Zertifikatsprüfung erhalten.

Proxy für einen einzelnen curl-Test umgehen

`[TEST] curl --noproxy "*" -v https://server.example/`

> Das Umgehen eines Unternehmensproxys kann gegen Sicherheitsrichtlinien verstoßen. Der Test darf nur durchgeführt werden, wenn direkte Verbindungen erlaubt und ausdrücklich autorisiert sind.

Anwendungsebene gegen Transportebene

Windows:

`[TEST] Test-NetConnection server.example -Port 443`

Linux und macOS:

`[TEST] nc -vz -w 3 server.example 443`

Anschließend:

`[TEST] curl -v https://server.example/`

TCP-Test	HTTP-Test	Wahrscheinlicher Bereich
erfolgreich	erfolgreich	grundlegender Dienstpfad funktioniert
erfolgreich	fehlgeschlagen	TLS, HTTP, Anwendung oder Authentifizierung
fehlgeschlagen	nicht möglich	Netzwerk, Firewall oder Listener
wechselnd	wechselnd	Load Balancer, Backend oder instabiler Pfad

---

22. Alternative Erklärungen bei typischen Beobachtungen

Beobachtung	Mögliche alternative Erklärungen
Neustart löst Fehler	Cache, Speicherleck, Prozesszustand, Verbindung, Timing
Dienstneustart löst Fehler	Abhängigkeit wurde neu verbunden, Port freigegeben, Konfiguration neu geladen
DNS-Cache-Leerung löst Fehler	TTL wäre gleichzeitig abgelaufen, DNS-Eintrag wurde parallel geändert
Firewall-Änderung löst Fehler	Stateful Session wurde erneuert, NAT oder Route änderte sich
Update-Rollback löst Fehler	Rollback enthielt ebenfalls Neustart oder Konfigurationsrücksetzung
Kabelwechsel löst Fehler	Port wurde neu ausgehandelt, Switchport oder DHCP-Lease änderte sich
Anmeldung mit Testkonto funktioniert	Konto, Rolle, Profil, MFA oder Lizenz unterscheiden sich
Zugriff über IP funktioniert	DNS möglich, aber HTTPS über IP wurde nicht vollständig geprüft
Port ist offen	Anwendung kann intern trotzdem fehlerhaft sein
Dienststatus ist „Running“	Prozess kann hängen oder Abhängigkeit nicht erreichen
Ein Test ist erfolgreich	sporadischer Fehler kann weiterhin bestehen
Fehler verschwindet von selbst	Last, Lease, TTL, Token oder Providerzustand änderte sich

---

23. Warum „Nach Maßnahme funktioniert es“ nicht immer genügt

Zwischen Maßnahme und erfolgreichem Test können weitere Veränderungen stattgefunden haben:

- Cacheeintrag ist abgelaufen,
- DHCP-Lease wurde erneuert,
- DNS-Eintrag wurde repliziert,
- Benutzerkonto wurde entsperrt,
- Token wurde erneuert,
- Providerstörung wurde behoben,
- Failover ist erfolgt,
- geplante Aufgabe wurde beendet,
- Last ist zurückgegangen,
- anderer Backend-Server wurde ausgewählt.

Deshalb müssen Zeitpunkt, technische Wirkung und Kontrolltest zusammen betrachtet werden.

---

24. Gegenprobe durchführen

Eine Gegenprobe testet, ob das Ergebnis wirklich von der vermuteten Ursache abhängt.

Geeignete Gegenproben

- gleicher Test vor und nach der Korrektur,
- betroffenes und funktionierendes System vergleichen,
- Ursache in Testumgebung entfernen,
- vorherige Konfiguration kontrolliert wiederherstellen,
- bestimmtes Backend direkt testen,
- Funktion mit und ohne Proxy vergleichen,
- Funktion aus unterschiedlichen VLANs vergleichen,
- Benutzer und Client getrennt wechseln.

Nicht geeignete Gegenprobe

Eine produktive Fehlkonfiguration wird absichtlich erneut aktiviert, obwohl dadurch ein weiterer Ausfall entstehen könnte.

Eine erneute Fehlererzeugung gehört normalerweise in:

- Testumgebung,
- Staging,
- Labor,
- isolierte Kopie,
- geplantes Wartungsfenster mit Freigabe.

---

25. Die Fünf-Warum-Methode vorsichtig einsetzen

Die Fünf-Warum-Methode fragt wiederholt nach dem technischen oder organisatorischen Grund eines Fehlers.

Beispiel

Frage	Antwort
Warum war die Anwendung nicht erreichbar?	Der Dienst konnte nicht starten
Warum konnte der Dienst nicht starten?	Die Konfigurationsdatei war ungültig
Warum war die Konfigurationsdatei ungültig?	Ein Deployment setzte einen falschen Portwert
Warum wurde der falsche Wert akzeptiert?	Es gab keine Schema-Validierung
Warum fehlte die Validierung?	Sie war im Deploymentprozess nicht vorgesehen

Mögliche Grundursache

`Der Deploymentprozess validiert Konfigurationswerte nicht vor der produktiven Bereitstellung.`

Mögliche Prävention

- Schema-Validierung,
- automatisierter Konfigurationstest,
- Staging,
- schrittweiser Rollout,
- automatische Rücknahme bei fehlgeschlagenem Health Check.

---

26. Grenzen der Fünf-Warum-Methode

Die Methode darf nicht dazu führen, dass eine komplexe Störung künstlich auf eine einzelne lineare Ursache reduziert wird.

Mögliche Probleme

- mehrere unabhängige Ursachen werden übersehen,
- gewünschte Antwort wird durch die Fragestellung vorgegeben,
- Untersuchung endet bei „menschlichem Fehler“,
- technische und organisatorische Faktoren werden vermischt,
- fehlende Belege werden durch Vermutungen ersetzt.

Für komplexe Störungen ist eine Ursachenmatrix häufig besser.

---

27. Ursachenmatrix

Faktor	Vorhanden	Notwendig	Allein ausreichend	Belegt	Bewertung

Falscher Datenbank-Hostname	Ja	Ja	Nein	Konfigurationsvergleich	Direkte Ursache
Keine Konfigurationsvalidierung	Ja	Nein	Nein	Pipeline geprüft	Grundursache
Gleichzeitiger Rollout	Ja	Nein	Nein	Deploymentprotokoll	Verstärkte Auswirkung
Kein Ende-zu-Ende-Monitoring	Ja	Nein	Nein	Monitoringkonfiguration	Erkennungslücke
Hohe Benutzerlast	Nein	Nein	Nein	Monitoring	Widerlegt
Datenbankausfall	Nein	Nein	Nein	Datenbankmonitoring	Widerlegt

---

28. Menschliche Handlung nicht automatisch als Grundursache verwenden

Ungeeignet

`Der Administrator hat die falsche Adresse eingetragen.`

Diese Aussage beendet die Untersuchung zu früh.

Weiterführende Fragen

- Warum konnte ein ungültiger Wert gespeichert werden?
- Gab es eine technische Validierung?
- War die Benutzeroberfläche eindeutig?
- Wurde die Änderung automatisch geprüft?
- Existierte ein Vier-Augen-Prinzip?
- War die Dokumentation aktuell?
- War ausreichend Zeit für die Änderung vorgesehen?
- Konnte der Rollout schrittweise erfolgen?
- Gab es eine automatische Rückfallmöglichkeit?
- Warum erkannte das Monitoring den Fehler nicht?

Eine sachliche Ursachenanalyse untersucht, wie Systeme und Prozesse verbessert werden können, statt einzelne Personen zu beschuldigen.

---

29. Bestätigungsgrade verwenden

Grad	Bedeutung
Bestätigt	Direkter Mechanismus, mehrere Belege und Gegenprobe vorhanden

Grad	Bedeutung
Sehr wahrscheinlich	Mechanismus und mehrere Belege vorhanden, Gegenprobe nicht möglich
Wahrscheinlich	Plausibel und teilweise belegt
Möglich	Technisch denkbar, aber kaum belegt
Unklar	Belege reichen nicht für eine Bewertung
Unwahrscheinlich	Mehrere Fakten widersprechen
Widerlegt	Definierte Vorhersage wurde durch Gegenbeleg widerlegt

Wenn eine Gegenprobe aus Sicherheits- oder Produktionsgründen nicht möglich ist, sollte die Ursache als „sehr wahrscheinlich“ und nicht automatisch als vollständig bestätigt dokumentiert werden.

---

30. Wann die Ursachensuche beendet werden kann

Die Untersuchung kann beendet oder in eine spätere Nachanalyse überführt werden, wenn:

- der Dienst stabil wiederhergestellt ist,
- der technische Mechanismus nachvollzogen wurde,
- Umfang und Zeitpunkt erklärt sind,
- alternative wahrscheinliche Ursachen geprüft wurden,
- ausreichend starke Belege vorhanden sind,
- verbleibende Unsicherheiten dokumentiert sind,
- Präventionsmaßnahmen abgeleitet werden können,
- weiteres Testen ein unverhältnismäßiges Risiko erzeugen würde.

Eine Untersuchung sollte nicht endlos fortgeführt werden. Der notwendige Beweisgrad hängt von Auswirkung und Risiko der Störung ab.

Störung	Erforderlicher Beweisgrad
Einzelner unkritischer Arbeitsplatz	technische Plausibilität und dokumentierter Funktionstest
Wiederkehrende Unternehmensstörung	mehrere Belege und Gegenprobe
Kritischer Produktionsausfall	ausführliche Ursachenanalyse
Datenverlust	hohe Beweissicherheit und Managementbeteiligung
Sicherheitsvorfall	Incident Response und möglicherweise Forensik
Rechtlich relevanter Vorfall	dokumentierte Beweismittelkette und Fachstellen

---

31. Vollständiges Praxisbeispiel

Störung

Seit 14:31 Uhr liefert `https://server.example` abwechselnd HTTP `200` und HTTP `503`.

Fehlerumfang

- alle Benutzer betroffen,
- ungefähr jede zweite Anfrage schlägt fehl,
- DNS liefert zwei Zieladressen,
- TCP und TLS funktionieren zu beiden Adressen.

Hypothesen

Nr.	Hypothese
H1	DNS liefert eine nicht vorgesehene IP-Adresse
H2	Eines der Backends ist fehlerhaft
H3	Load Balancer verteilt Anfragen falsch
H4	Datenbank ist vollständig ausgefallen
H5	Clientnetzwerk ist instabil

Prüfungen

- Beide DNS-Adressen gehören zur vorgesehenen Umgebung.
- Backend 1 liefert reproduzierbar HTTP `200`.
- Backend 2 liefert reproduzierbar HTTP `503`.
- Datenbankmonitoring zeigt keinen allgemeinen Ausfall.
- Backend 2 protokolliert einen nicht auflösbaren Datenbank-Hostnamen.
- Konfigurationsvergleich zeigt einen Tippfehler.
- Der Tippfehler wurde mit dem Deployment um 14:31 Uhr eingeführt.
- Die Pipeline akzeptierte den Wert ohne Prüfung.
- Der Load Balancer markierte Backend 2 weiterhin als gesund, weil der Health Check nur eine statische Seite prüfte.

Bewertung der Hypothesen

Hypothese	Bewertung	Begründung
H1	Widerlegt	Beide DNS-Adressen sind vorgesehen
H2	Bestätigt	Fehler ist reproduzierbar an Backend 2 gebunden
H3	Teilweise bestätigt	Health Check erkennt den Backendfehler nicht
H4	Widerlegt	Datenbank und Backend 1 funktionieren
H5	Widerlegt	TCP- und TLS-Verbindungen sind stabil

Ursachenkette

Ebene	Feststellung
Symptom	ungefähr jede zweite Anfrage liefert HTTP 503
Direkte Ursache	Backend 2 erreicht die Datenbank nicht
Technischer Fehler	falscher Datenbank-Hostname
Auslöser	Deployment um 14:31 Uhr
Grundursache	fehlende Validierung der Konfiguration
Begünstigender Faktor	fehlerhafter Health Check
Erkennungslücke	Monitoring prüft keine Datenbankfunktion
Sofortmaßnahme	Backend 2 aus Rotation nehmen oder Konfiguration zurücksetzen
Prävention	Validierung und anwendungsnahen Health Check einführen

---

32. Dokumentationsvorlage für die Ursachenkette

Feld	Eintrag
Ticketnummer	
Symptom	
Fehlerumfang	
Beginn der Störung	
Direkte technische Ursache	
Technischer Mechanismus	
Auslöser	
Grundursache	
Begünstigende Faktoren	
Erkennungslücken	
Wiederherstellungsfaktoren	
Unterstützende Belege	
Widersprechende Belege	
Geprüfte Alternativen	
Durchgeführte Gegenprobe	
Bestätigungsgrad	

Feld	Eintrag
Verbleibende Unsicherheiten	
Sofortmaßnahme	
Präventionsmaßnahme	

33. Vorlage für alternative Erklärungen

Nr.	Alternative Erklärung	Erwartete Beobachtung	Tatsächliche Beobachtung	Beleg	Status
A1					Offen
A2					Offen
A3					Offen
A4					Offen

34. Prüffragen zur Grundursache

- Erklärt die Ursache alle beobachteten Symptome?
- Erklärt sie, warum bestimmte Systeme nicht betroffen waren?
- Erklärt sie den Beginn der Störung?
- Ist der technische Mechanismus nachvollziehbar?
- Gibt es direkte technische Belege?
- Wurde die Ursache mit einem Vergleichssystem geprüft?
- Wurde eine Gegenprobe durchgeführt?
- Welche alternativen Erklärungen wurden ausgeschlossen?
- Welche Beobachtungen widersprechen der Ursache?
- Existiert dieselbe Bedingung auf funktionierenden Systemen?
- Kann die Ursache in einer Testumgebung reproduziert werden?
- Ist die genannte Ursache nur ein menschlicher Fehler?
- Welche technische Schutzmaßnahme hätte den Fehler verhindert?
- Welche Monitoringfunktion hätte den Fehler früher erkannt?
- Sind mehrere Ursachen oder begünstigende Faktoren vorhanden?

Kurzcheckliste

- [ ] Symptom und Ursache getrennt
- [ ] Direkte technische Ursache bestimmt
- [ ] Technischen Mechanismus beschrieben
- [ ] Auslöser bestimmt
- [ ] Grundursache untersucht

- [ ] Begünstigende Faktoren berücksichtigt
- [ ] Erkennungslücken berücksichtigt
- [ ] Abhängigkeiten geprüft
- [ ] Betroffenes und funktionierendes System verglichen
- [ ] Konfigurationsunterschiede geprüft
- [ ] Datei- und Versionsunterschiede geprüft
- [ ] Dienstabhängigkeiten geprüft
- [ ] Unterstützende Belege dokumentiert
- [ ] Widersprechende Belege dokumentiert
- [ ] Alternative Erklärungen aufgestellt
- [ ] Alternative Erklärungen geprüft
- [ ] Gegenprobe durchgeführt oder begründet ausgelassen
- [ ] Zeitliche Korrelation nicht mit Ursache verwechselt
- [ ] Wiederherstellungsmaßnahme nicht automatisch als Ursache interpretiert
- [ ] Mehrere mögliche Ursachen berücksichtigt
- [ ] Menschliche Handlung nicht vorschnell als Grundursache verwendet
- [ ] Bestätigungsgrad dokumentiert
- [ ] Verbleibende Unsicherheiten dokumentiert
- [ ] Präventionsmaßnahmen ableitbar

Ergebnis dieses Arbeitsschrittes

Am Ende dieses Schrittes liegt eine nachvollziehbare Ursachenkette vor. Sie unterscheidet:

- sichtbares Symptom,
- unmittelbare technische Ursache,
- technischen Mechanismus,
- auslösendes Ereignis,
- Grundursache,
- begünstigende Faktoren,
- Erkennungslücken,
- Wiederherstellungsmaßnahmen,
- mögliche Präventionsmaßnahmen.

Eine Ursache gilt nicht allein deshalb als bestätigt, weil der Fehler nach einer Änderung verschwunden ist. Sie muss zum Fehlerbild passen, durch technische Belege gestützt und gegen alternative Erklärungen geprüft worden sein.

Nächste Seite:
1.8 Lösung umsetzen, Rückfallplan anwenden und Funktion verifizieren

---

Offizielle Hersteller-, Standard- und Projektdokumentation

- NIST – SP 800-61 Revision 3: Incident Response Recommendations and Considerations
- Google SRE – Postmortem Culture
- Google SRE Workbook – Postmortem Practices
- Microsoft Learn – Get-Service
- Microsoft Learn – Compare-Object
- Microsoft Learn – Get-FileHash
- Microsoft Learn – Get-WinEvent
- freedesktop.org – systemctl
- freedesktop.org – systemd-Unit-Abhängigkeiten
- freedesktop.org – journalctl
- curl – Offizielle Befehlsreferenz
- Apple Support – Protokollmeldungen in der Konsole anzeigen
