7.15 Backupjob ist fehlgeschlagen

7.15.1 Ziel dieser Seite

Diese Seite beschreibt die systematische Diagnose, wenn ein Backupjob:

- nicht gestartet wurde;
- abgebrochen ist;
- nur teilweise erfolgreich war;
- kein verwendbares Recovery Point erzeugt hat;
- ein inkonsistentes Backup erstellt hat;
- das Zielrepository nicht erreichen konnte;
- wegen Snapshot-, VSS-, Netzwerk-, Kapazitäts- oder Berechtigungsfehlern gescheitert ist;
- als erfolgreich angezeigt wird, dessen Daten aber nicht wiederhergestellt werden können.

Ziele der Diagnose:

- den betroffenen Job und Sicherungszeitraum bestimmen;
- Scheduler-, Quell-, Snapshot-, Transport- und Zielfehler unterscheiden;
- Voll-, inkrementelle und differenzielle Sicherungsketten prüfen;
- Repository, Retention, Verschlüsselung und Unveränderbarkeit berücksichtigen;
- anwendungskonsistente Backups sicherstellen;
- die Wiederherstellbarkeit durch einen kontrollierten Restore-Test nachweisen;
- RPO und RTO bewerten;
- erneute Backupfehler verhindern.

Ein Backupjob gilt nicht allein deshalb als erfolgreich, weil ein Prozess mit Exit-Code `0` beendet wurde oder eine Sicherungsdatei vorhanden ist.

7.15.2 Sicherheitskennzeichnungen

Kennzeichnung	Bedeutung
LESEND	erfasst ausschließlich Zustände und Messwerte
NETZAKTIV	greift auf Quell- oder Zielsysteme zu
LASTERZEUGEND	kann Netzwerk, CPU, Arbeitsspeicher oder Storage stark belasten
ÄNDERND	verändert Job-, Repository- oder Sicherungszustände
LÖSCHEND	entfernt Recovery Points oder Repositorydaten
SENSITIV	kann Zugangsdaten, Schlüssel oder geschützte Daten betreffen
AUSFALLRISIKO	kann Sicherungen, Wiederherstellungen oder Produktivdienste unterbrechen

Grundregeln:

- Fehlgeschlagene Jobs, Protokolle und Statusdaten zuerst dokumentieren.
- VSS-Writer unmittelbar nach dem Fehler prüfen, bevor Dienste neu gestartet werden.
- Keine Recovery Points löschen, um kurzfristig Speicher freizugeben.
- Repositorys nicht ungeprüft entsperren.
- Retention oder Pruning nicht spontan verändern.
- Keine privaten Schlüssel, Repositorypasswörter oder Tokens in Diagnoseausgaben aufnehmen.
- Eine beschädigte Sicherungskette nicht durch manuelle Dateiänderungen „reparieren“.
- Produktive Datenbanken nicht durch unkoordinierte Dateikopien sichern.
- Restore-Tests nur in einem isolierten Ziel durchführen.
- Bei Ransomwareverdacht die Sicherungsumgebung nicht unkontrolliert verändern.

7.15.3 Ein Backup ist erst durch Wiederherstellung nachgewiesen

Ein technisch vollständiger Backupablauf besteht aus mehreren Nachweisen:

```
Job wurde geplant
→ Job wurde tatsächlich gestartet
→ Quelle wurde vollständig erfasst
→ konsistenter Sicherungszeitpunkt wurde erzeugt
→ Daten wurden übertragen
→ Daten wurden im Repository gespeichert
→ Metadaten und Katalog wurden geschrieben
→ Recovery Point wurde registriert
→ Integritätsprüfung war erfolgreich
→ Wiederherstellung wurde getestet
→ wiederhergestellte Anwendung funktioniert
```

Nicht ausreichend sind allein:

- „Job erfolgreich“;
- vorhandene Backupdatei;
- belegter Speicher im Repository;
- vorhandener Snapshot;
- erfolgreiche Datenübertragung;
- gültige Prüfsumme einer einzelnen Datei;
- erfolgreiches Einlesen des Katalogs.

7.15.4 Wichtige Begriffe

Begriff	Bedeutung
Vollbackup	enthält den vollständigen ausgewählten Datenbestand
inkrementelles Backup	enthält Änderungen seit der letzten relevanten Sicherung
differenzielles Backup	enthält Änderungen seit dem letzten Vollbackup
synthetisches Vollbackup	wird im Repository aus vorhandenen Sicherungsdaten erzeugt
Snapshot	zeitpunktbezogener Zustand eines Volumes oder Dateisystems
crash-konsistent	entspricht ungefähr dem Zustand nach einem unerwarteten Ausfall
anwendungskonsistent	Anwendung und Datenbank wurden für den Sicherungspunkt koordiniert
Recovery Point	für die Wiederherstellung registrierter Sicherungsstand
Repository	logischer Speicherbereich für Backups und Metadaten
Retention	Aufbewahrungsregel für Recovery Points
Immutability	zeitlich begrenzter Schutz vor Änderung oder Löschung
Air Gap	logische oder physische Trennung der Sicherung vom Produktivsystem
RPO	maximal tolerierbarer Datenverlust in Zeit
RTO	maximal tolerierbare Wiederherstellungsdauer
Backupfenster	vorgesehener Zeitraum für die Sicherung
Restore-Test	kontrollierter Nachweis der Wiederherstellbarkeit

Ein Storage-Snapshot auf demselben System ist allein kein vollständiger Schutz vor Ausfall, Fehlbedienung, Diebstahl oder Ransomware.

7.15.5 Phasen eines Backupjobs

Phase	Mögliche Fehler
Planung	falscher Zeitplan, deaktivierter Job, verpasster Start
Vorbereitung	Konfiguration, Lizenz, Agent oder Dienst fehlerhaft
Authentifizierung	Kennwort, Token, Zertifikat oder Schlüssel abgelaufen
Quellenprüfung	Pfad, Volume, VM, Datenbank oder Container fehlt
Snapshot	VSS-, LVM-, CSI-, Hypervisor- oder Storagefehler
Anwendungskonsistenz	Writer, Datenbank oder Quiescing schlägt fehl
Lesen	Berechtigung, Sperre, I/O-Fehler oder beschädigte Datei

Phase	Mögliche Fehler
Transport	DNS, Routing, VPN, Firewall, TLS oder Timeout
Schreiben	Repository voll, Quota, Schreibschutz oder Berechtigung
Deduplizierung	Index-, Cache- oder Metadatenfehler
Katalogisierung	Recovery Point wird nicht registriert
Verifikation	Prüfsumme oder Strukturprüfung schlägt fehl
Retention	alte Recovery Points werden nicht korrekt verarbeitet
Replikation	sekundäre oder externe Kopie wird nicht erstellt
Abschluss	Jobstatus, Benachrichtigung oder Monitoring ist falsch

Der erste sichtbare Fehler ist nicht immer die ursprüngliche Ursache. Ein später Repositoryfehler kann beispielsweise durch einen vorher fehlgeschlagenen Snapshot ausgelöst worden sein.

7.15.6 Fehlerklassen unterscheiden

Fehlerklasse	Typischer Befund
Job nicht gestartet	kein Lauf, kein Log oder nur Schedulerfehler
Job deaktiviert	Zeitplan vorhanden, Ausführung aber abgeschaltet
falscher Dienstkontext	manueller Start funktioniert, geplanter Lauf nicht
Agent nicht erreichbar	Backupserver erreicht Client oder Proxy nicht
Quellpfad fehlt	Volume, Freigabe, Mount oder PVC nicht vorhanden
Zugriff verweigert	Dienstkonto darf Quelle oder Ziel nicht lesen beziehungsweise schreiben
Snapshot fehlgeschlagen	VSS-, Hypervisor-, CSI- oder Storagefehler
Anwendung nicht konsistent	Datenbank-Writer oder Quiescing fehlerhaft
Repository voll	zu wenig Speicher, Quota oder Thin Pool voll
Repository nicht erreichbar	DNS-, Netzwerk-, TLS- oder Mountfehler
Repository gesperrt	paralleler Job oder verwaister Lock
Sicherungskette beschädigt	erforderliches Voll- oder Inkrementbackup fehlt
Retention fehlerhaft	alte Daten bleiben bestehen oder benötigte Punkte werden entfernt
Verschlüsselung fehlerhaft	Schlüssel, Passwort oder KMS nicht verfügbar
Prüfsummenfehler	Daten oder Repository beschädigt
Timeout	Job überschreitet Backupfenster oder Netzwerkgrenze
Teilbackup	einige Quellen oder Dateien wurden übersprungen

Fehlerklasse	Typischer Befund
Katalogfehler	Daten vorhanden, Recovery Point jedoch nicht nutzbar
Replikation fehlgeschlagen	Primärbackup vorhanden, Zweitkopie fehlt
Immutability-Konflikt	gewünschte Änderung oder Retention wird blockiert
Lizenz- oder Kapazitätslimit	geschützte Systeme oder Datenmenge überschreiten Produktlimit
Sicherheitsvorfall	Repository, Zugangsdaten oder Backupserver wurden manipuliert

---

7.15.7 Typische Symptome und Fehlermeldungen

```
Backup failed
Job failed
Partial success
Completed with warnings
Missed schedule
Access denied
Authentication failed
Permission denied
Repository unavailable
Repository is full
Quota exceeded
No space left on device
Snapshot creation failed
VSS writer failed
VSS_E_WRITERERROR
Unable to create snapshot
Volume not found
Source path not found
Connection timed out
Connection reset
TLS certificate verify failed
Repository locked
Stale lock
Checksum mismatch
Hash mismatch
Corrupt backup
Index damaged
```

```
Catalog update failed
Retention failed
Prune failed
Unable to decrypt
Encryption key not found
Object lock prevents deletion
Backup window exceeded
Incremental chain is broken
Recovery point unavailable
Restore validation failed
```

Die exakte Meldung einschließlich Fehlercode, Jobphase und Uhrzeit muss übernommen werden.

7.15.8 Sofortmaßnahmen

1. Fehlermeldung vollständig dokumentieren.
2. Job-ID, Jobname und Sicherungsprodukt erfassen.
3. Start- und Endzeit bestimmen.
4. Letzten erfolgreichen Recovery Point feststellen.
5. Betroffene Quellen und Ziele bestimmen.
6. RPO-Verletzung berechnen.
7. Aktuelle Logs und Ereignisse sichern.
8. Snapshot- oder VSS-Zustand prüfen.
9. Repository nicht bereinigen oder entsperren.
10. Prüfen, ob noch ein Job aktiv ist.
11. Kapazität und Quota erfassen.
12. Bei Sicherheitsverdacht Backupserver und Repository schützen.
13. Ursache nachweisen, bevor der Job erneut gestartet wird.
14. Nach der Korrektur einen kontrollierten neuen Lauf durchführen.
15. Wiederherstellbarkeit des neuen Recovery Points testen.

Wiederholte Neustarts eines Backupjobs können:

- zusätzliche Snapshots erzeugen;
- Repositorylocks verursachen;
- Backupfenster überschreiten;
- Bandbreite und Storage belasten;
- fehlerhafte inkrementelle Ketten verlängern;
- verwertbare Beweise überschreiben.

7.15.9 RPO und aktuelle Schutzlücke bestimmen

Beispiel:

```
Letzter erfolgreicher Recovery Point:
01.08.2026 22:00 Uhr

Aktueller Zeitpunkt:
02.08.2026 10:00 Uhr

Aktuelle Schutzlücke:
12 Stunden

Vorgesehenes RPO:
4 Stunden

Ergebnis:
RPO ist bereits verletzt.
```

Zu dokumentieren sind:

- letzter erfolgreicher Job;
- letzter erfolgreich verifizierter Recovery Point;
- letzter extern replizierter Recovery Point;
- letzter Restore-Test;
- aktueller Zeitpunkt;
- Änderungsrate der Produktivdaten;
- maximal tolerierbarer Datenverlust;
- Priorität des betroffenen Systems.

Ein erfolgreicher Job nach der Störung beseitigt nicht rückwirkend die bereits entstandene RPO-Verletzung.

7.15.10 Job und Scheduler unter Windows prüfen

Geplanten Task anzeigen:

LESEND

```
Get-ScheduledTask `
 -TaskName "<Taskname>"
```

Laufzeitinformationen:

```
Get-ScheduledTask `
  -TaskName "<Taskname>" |
Get-ScheduledTaskInfo
```

Wichtige Felder:

```
State
LastRunTime
LastTaskResult
NextRunTime
NumberOfMissedRuns
```

Detaillierte Abfrage mit `schtasks`:

```
schtasks /query /tn "<Taskpfad-und-Name>" /v /fo LIST /hresult
```

Zu prüfen sind:

- Task aktiviert;
- richtiger Trigger;
- letzte Laufzeit;
- nächster Lauf;
- verpasste Läufe;
- letzter Rückgabecode;
- ausführendes Konto;
- „Unabhängig von der Benutzeranmeldung ausführen“;
- „Mit höchsten Privilegien ausführen“;
- Recht „Anmelden als Stapelverarbeitungsauftrag“;
- Arbeitsverzeichnis;
- Argumente;
- Netzwerkverfügbarkeit;
- Ablauf des Kennworts;
- geändertes Dienstkonto;
- Laufzeitbegrenzung;
- parallele Ausführung;
- Verhalten nach verpasstem Start.

Ein gemapptes Netzlaufwerk der interaktiven Benutzersitzung steht einem geplanten Task nicht automatisch zur Verfügung. Für geplante Jobs sollte der tatsächlich erreichbare UNC-Pfad verwendet werden.

7.15.11 Windows-Task-Scheduler-Ereignisse prüfen

Vorhandene Ereignisse:

```
Get-WinEvent `
  -LogName "Microsoft-Windows-TaskScheduler/Operational" `
  -MaxEvents 100 |
Select-Object `
  TimeCreated,
  Id,
  LevelDisplayName,
  Message
```

Das Operational-Protokoll kann deaktiviert sein. Eine Aktivierung ist eine Konfigurationsänderung und muss dokumentiert werden.

Zu korrelieren sind:

- Triggerzeit;
- Start der Aktion;
- Prozess-ID;
- Rückgabecode;
- Abbruch;
- Zeitüberschreitung;
- Berechtigungsfehler;
- fehlender Benutzerkontext;
- mehrfacher Jobstart;
- Beendigung durch Task Scheduler.

7.15.12 Systemd-Timer unter Linux prüfen

Timerübersicht:

LESEND

```
systemctl list-timers --all
```

Bestimmten Timer prüfen:

```
systemctl \
  status backup.timer \
```

```
  --no-pager
```

Zugehörigen Dienst prüfen:

```
systemctl \
  status backup.service \
  --no-pager
```

Letzten Exit-Code erfassen:

```
systemctl show \
  backup.service \
  -p Result \
  -p ExecMainCode \
  -p ExecMainStatus \
  -p ActiveEnterTimestamp \
  -p ActiveExitTimestamp
```

Protokolle:

```
journalctl \
  -u backup.timer \
  -u backup.service \
  --since "-24 hours"
```

Zu prüfen sind:

- Timer aktiviert;
- Timer tatsächlich geladen;
- letzter und nächster Lauf;
- `OnCalendar`;
- `Persistent`;
- zufällige Verzögerung;
- zugehörige Service-Unit;
- Exit-Code;
- Timeout;
- Benutzerkontext;
- Umgebungsvariablen;
- Arbeitsverzeichnis;
- Netzwerkabhängigkeiten;
- Secretbereitstellung;

- paralleler Lauf;
- System war zum geplanten Zeitpunkt ausgeschaltet.

7.15.13 Cronjobs prüfen

Cronjobs des aktuellen Benutzers:

```
crontab -l
```

Cronjobs von `root`:

```
sudo crontab -l
```

Systemweite Konfigurationen können abhängig von der Distribution unter anderem liegen in:

```
/etc/crontab
/etc/cron.d/
/etc/cron.hourly/
/etc/cron.daily/
/etc/cron.weekly/
/etc/cron.monthly/
```

Zu prüfen sind:

- richtiger Benutzer;
- korrekte Zeitangabe;
- Zeitzone;
- vollständige Programmpfade;
- reduzierte `PATH`-Variable;
- Arbeitsverzeichnis;
- Shell;
- Standardausgabe und Fehlerausgabe;
- Zugangsdaten;
- gemountete Dateisysteme;
- parallele Prozesse;
- Exit-Code;
- Job wurde durch einen anderen Lauf blockiert;
- Server war ausgeschaltet.

Ein Skript, das in einer interaktiven Shell funktioniert, muss nicht unter Cron funktionieren.

7.15.14 Quelle und Sicherungsumfang prüfen

Zu erfassen sind:

- vorgesehene Quellen;
- tatsächlich eingeschlossene Quellen;
- Ausschlüsse;
- Dateisystemgrenzen;
- Mountpoints;
- Volumes;
- virtuelle Maschinen;
- Container-Volumes;
- Datenbanken;
- Systemzustand;
- Anwendungsdaten;
- Konfigurationen;
- Secrets;
- Zertifikate;
- Berechtigungen und ACLs.

Linux-Mounts:

```
findmnt
```

```
lsblk \
  -o NAME,TYPE,FSTYPE,SIZE,FSAVAIL,FSUSE%,MOUNTPOINTS
```

Windows-Volumes:

```
Get-Volume
```

Datei- oder Verzeichnisexistenz:

```
Test-Path `
  -LiteralPath "<Quellpfad>"
```

Mögliche Fehler:

- Mount fehlt und leeres Verzeichnis wird gesichert;
- Laufwerksbuchstabe hat sich geändert;
- neue Partition wurde nicht in den Job aufgenommen;
- Ausschlussregel ist zu breit;

- symbolischer Link wird nicht verfolgt;
- Dateisystemgrenze wird nicht überschritten;
- Containerdaten liegen in einem anderen Volume;
- neuer Datenbankpfad fehlt;
- verschlüsseltes Volume ist nicht entsperrt;
- Quelldateien werden durch eine Anwendung gesperrt;
- Dienstkonto darf neue Verzeichnisse nicht lesen.

Ein erfolgreicher Job kann trotzdem unvollständig sein, wenn der Sicherungsumfang falsch definiert ist.

7.15.15 Kapazität, Inodes und Quotas prüfen

Windows

```
Get-Volume |
  Select-Object `
    DriveLetter,
    FileSystemLabel,
    FileSystem,
    Size,
    SizeRemaining,
    HealthStatus
```

Linux

```
df -hT
```

```
df -i
```

macOS

```
df -h
```

Zu prüfen sind:

- freier Speicher auf der Quelle;
- temporärer Snapshotbereich;
- Cache- und Stagingbereich;
- Repositorykapazität;

- Dateisystem-Inodes;
- Benutzer- oder Verzeichnisquota;
- Object-Storage-Quota;
- Cloudkontingent;
- Thin Pool;
- Deduplizierungsmetadaten;
- Katalogdatenbank;
- Logvolume;
- sekundäres Replikationsziel.

Ein Backup kann trotz freiem Repository fehlschlagen, wenn der lokale Snapshot-, Cache-, Katalog- oder Temp-Bereich voll ist.

7.15.16 Windows Server Backup prüfen

Status eines laufenden Backups:

LESEND

```
wbadmin get status
```

Vorhandene Backupversionen:

```
wbadmin get versions
```

Verfügbare Datenträger:

```
wbadmin get disks
```

Vorhandene Windows-Server-Backup-Dienste:

```
Get-Service `
  -Name wbengine,VSS,swprv `
  -ErrorAction SilentlyContinue
```

Backup- und VSS-Ereignisse:

```
Get-WinEvent `
  -FilterHashtable @{
    LogName = "Application"
```

```
    StartTime = (Get-Date).AddHours(-12)
  } |
Where-Object {
  $_.ProviderName -in @(
    "Microsoft-Windows-Backup",
    "VSS"
  )
} |
Select-Object `
  TimeCreated,
  ProviderName,
  Id,
  LevelDisplayName,
  Message
```

Zu prüfen sind:

- vorhandene Recovery Points;
- Jobstatus;
- Sicherungsziel;
- Quellvolumes;
- System-State-Komponenten;
- VSS-Fehler;
- Zielkapazität;
- Zugriff auf Netzwerkziel;
- Berechtigungen;
- Datenträgerstatus;
- Katalog;
- Ausschlüsse;
- parallele Sicherungssoftware.

`wbadmin start backup` startet einen neuen Sicherungsvorgang und ist kein rein diagnostischer Befehl.

7.15.17 VSS systematisch prüfen

VSS-Writer unmittelbar nach dem Fehler prüfen:

LESEND

```
vssadmin list writers
```

VSS-Provider:

```
vssadmin list providers
```

Shadow-Copy-Speicher:

```
vssadmin list shadowstorage
```

Vorhandene Shadow Copies:

```
vssadmin list shadows
```

Ein gesunder Writer zeigt gewöhnlich:

```
State: Stable
Last error: No error
```

Zu prüfen sind:

- Name des fehlgeschlagenen Writers;
- Writer-ID;
- Writer-Instanz-ID;
- Writerzustand;
- letzter Fehler;
- betroffene Anwendung;
- VSS-Provider;
- Shadow-Copy-Speicher;
- freier Speicher;
- parallele Backupsoftware;
- Zeitüberschreitung;
- Systemlast;
- Anwendung mit eigenem Writer;
- Datenbank- oder Hypervisorintegration.

Ein einzelner fehlerhafter Writer kann den gesamten anwendungskonsistenten Backupjob stoppen.

Der pauschale Neustart aller VSS-Dienste ist keine erste Diagnosemaßnahme. Zuerst muss der betroffene Writer dokumentiert und dem zugehörigen Dienst oder Produkt zugeordnet werden.

7.15.18 Snapshot und Anwendungskonsistenz unterscheiden

Mögliche Sicherungsarten:

Sicherungsart	Eigenschaften
Dateikopie im laufenden Betrieb	kann bei aktiven Datenbanken inkonsistent sein
crash-konsistenter Snapshot	entspricht ungefähr einem ungeplanten Ausfall
anwendungskonsistenter Snapshot	Anwendung bestätigt konsistenten Zustand
logisches Datenbankbackup	exportiert Daten über Datenbankwerkzeuge
physisches Datenbankbackup	sichert Datenbankdateien mit vorgesehenem Verfahren
Hypervisor-Snapshot	sichert VM-Zustand, ersetzt aber nicht automatisch ein Backup
Storage-Snapshot	liegt häufig auf demselben Storage und benötigt zusätzliche Kopie

Zu prüfen sind:

- Anwendung unterstützt Snapshotkoordination;
- Writer oder Hook wurde ausgeführt;
- Datenbank wurde gequiesced;
- Transaktionslogs sind enthalten;
- WAL-, Redo- oder Binärlogs sind verfügbar;
- Snapshot wurde korrekt aufgelöst;
- Hypervisor-Snapshot wurde konsolidiert;
- Storage-Snapshot wurde repliziert;
- mehrere Volumes wurden gleichzeitig konsistent erfasst;
- Restoreverfahren ist dokumentiert.

7.15.19 Netzwerkpfad zum Repository prüfen

Zu bestimmen sind:

- Repositoryhostname;
- Ziel-IP-Adresse;
- Port;
- Protokoll;
- Proxy;
- VPN;
- Routing;
- Firewall;
- DNS;
- TLS;
- Netzwerkfreigabe;
- Objektstorage-Endpunkt;
- Backup-Gateway oder Proxyserver.

Windows

```
Resolve-DnsName `
  -Name "<Repository-FQDN>"
```

```
Test-NetConnection `
  -ComputerName "<Repository-FQDN>" `
  -Port <Port>
```

SMB-Verbindungen:

```
Get-SmbConnection
```

Linux und macOS

```
getent hosts <Repository-FQDN>
```

```
findmnt
```

HTTPS-Endpunkt:

```
curl \
  -v \
  https://<Repository-FQDN>/
```

TLS-Test:

```
openssl s_client \
  -connect <Repository-FQDN>:<Port> \
  -servername <Repository-FQDN> \
  </dev/null
```

Zu prüfen sind:

- korrekte DNS-Antwort;
- IPv4- und IPv6-Pfad;
- Portfreigabe;
- TLS-Zertifikat;
- Proxyregel;

- MTU;
- Paketverlust;
- Timeout;
- SMB-, NFS-, SFTP- oder API-Erreichbarkeit;
- Verbindung aus dem Dienstkontext;
- Bandbreite während des Backupfensters.

7.15.20 Dienstkonto, Zugangsdaten und Secrets prüfen

Zu erfassen sind:

- ausführendes Konto;
- Quellberechtigungen;
- Zielberechtigungen;
- Kennwortablauf;
- Kontosperre;
- Zertifikatsablauf;
- SSH-Schlüssel;
- API-Token;
- Cloudrolle;
- Secretpfad;
- KMS-Zugriff;
- Service-Principal;
- MFA- oder Conditional-Access-Auswirkung;
- Recht zur Anmeldung als Stapelverarbeitungsauftrag.

Typische Fehler:

- Kennwort wurde geändert;
- gespeicherte Zugangsdaten sind veraltet;
- Token ist abgelaufen;
- Zertifikat wurde erneuert, Bindung aber nicht;
- Dienst läuft unter anderem Konto;
- interaktiver Benutzer besitzt Rechte, Dienstkonto nicht;
- Netzwerkfreigabe ist nur als Laufwerksbuchstabe eingebunden;
- SSH-Hostkey hat sich geändert;
- Secret wurde nicht in den Container eingebunden;
- KMS ist nicht erreichbar;
- Rollenberechtigung wurde entzogen;
- Repository erlaubt Lesen, aber kein Schreiben;
- Retention benötigt Löschrecht, Backupkonto besitzt es nicht;
- Immutability verhindert absichtlich eine Löschung.

Zugangsdaten dürfen nicht zur Diagnose in Befehlszeilen, Protokolle oder Tickets geschrieben werden.

7.15.21 Repositoryzustand prüfen

Zu prüfen sind:

- Erreichbarkeit;
- Schreibbarkeit;
- Kapazität;
- Quota;
- Dateisystemzustand;
- Repositorydatenbank;
- Index;
- Katalog;
- Locks;
- Cache;
- Deduplizierungsmetadaten;
- Prüfsummen;
- Immutability;
- Object Lock;
- Retention;
- parallele Jobs;
- Wartungsjob;
- Replikation;
- verwendete Verschlüsselung;
- erforderlicher Schlüssel.

Ein Repositorylock kann verursacht werden durch:

- aktiven Backupjob;
- aktiven Restore;
- Integritätsprüfung;
- Pruning;
- Retention;
- Replikation;
- abgestürzten Prozess;
- Netzwerkunterbrechung;
- verwaisten Lock.

Ein Lock darf erst entfernt werden, wenn zweifelsfrei feststeht, dass kein zugehöriger Prozess mehr arbeitet.

7.15.22 Sicherungskette prüfen

Beispiel einer inkrementellen Kette:

```
Vollbackup
→ Inkrement 1
→ Inkrement 2
→ Inkrement 3
```

Für die Wiederherstellung können alle benötigten Kettenglieder erforderlich sein.

Beispiel einer differenziellen Sicherung:

```
Vollbackup
→ aktuelles Differenzialbackup
```

Zu prüfen sind:

- zugehöriges Vollbackup vorhanden;
- alle benötigten Inkremente vorhanden;
- Metadaten und Katalog konsistent;
- keine Datei manuell entfernt;
- Repositoryreplikation vollständig;
- synthetisches Vollbackup erfolgreich;
- Aufbewahrungsregel hat keine Abhängigkeit verletzt;
- erforderliche Datenbanklogs vorhanden;
- Verschlüsselungsschlüssel für alle Generationen vorhanden;
- Softwareversion kann den Recovery Point lesen.

Eine vorhandene letzte Inkrementdatei ist ohne die erforderliche Kette möglicherweise nicht wiederherstellbar.

7.15.23 Retention, Pruning und Immutability prüfen

Retention kann unter anderem festlegen:

- tägliche Recovery Points;
- wöchentliche Recovery Points;
- monatliche Recovery Points;
- jährliche Recovery Points;
- Mindestalter;
- maximale Anzahl;
- gesetzliche Aufbewahrung;
- Grandfather-Father-Son-Schema;
- Aufbewahrung nach Tags oder Richtlinien.

Zu prüfen sind:

- tatsächliche Retention;
- geplante Retention;
- letzte erfolgreiche Bereinigung;
- verfügbare Kapazität;
- unveränderbare Recovery Points;
- Legal Hold;
- Object Lock;
- abhängige inkrementelle Sicherungen;
- Replikationsverzug;
- Zeitunterschiede;
- konkurrierende Wartungsjobs.

Immutability kann erklären, warum Speicher nicht freigegeben werden kann. Sie darf nicht deaktiviert werden, um einen Kapazitätsfehler kurzfristig zu umgehen.

7.15.24 Integrität und Prüfsummen unterscheiden

Eine Integritätsprüfung kann bestätigen:

- Repositorystruktur lesbar;
- Index konsistent;
- Datenblöcke vorhanden;
- Prüfsummen stimmen;
- Katalog verweist auf vorhandene Daten;
- Verschlüsselung und Entschlüsselung funktionieren.

Sie bestätigt nicht automatisch:

- Sicherungsumfang vollständig;
- Anwendungskonsistenz;
- richtige Version;
- vorhandene Berechtigungen;
- funktionierenden Systemstart;
- erfülltes RPO;
- erfülltes RTO;
- vollständige Wiederherstellungsdokumentation.

Deshalb sind sowohl technische Integritätsprüfungen als auch reale Restore-Tests erforderlich.

7.15.25 Restic-Repository prüfen

Die folgenden Befehle verwenden die bereits sicher konfigurierte Repository- und Kennwortbereitstellung. Kennwörter dürfen nicht direkt in den Befehl geschrieben werden.

Snapshots anzeigen:

LESEND

```
restic snapshots
```

Repositorystatistik:

```
restic stats
```

Locks anzeigen:

```
restic list locks
```

Repositorystruktur prüfen:

LESEND, LASTERZEUGEND

```
restic check
```

Repositorydaten vollständig lesen:

LESEND, SEHR LASTERZEUGEND

```
restic check --read-data
```

Ein `--read-data`-Lauf kann das gesamte Repository lesen und muss in das Betriebs- und Backupfenster passen.

Retention vorab simulieren:

LESEND

```
restic forget \
  --dry-run \
  <vorgesehene-Retention-Optionen>
```

Wichtige Regeln:

- `forget` verändert Snapshotreferenzen;
- `prune` entfernt nicht mehr referenzierte Daten;
- `prune` kann lange dauern;

- während Wartungsvorgängen kann das Repository gesperrt sein;
- ein Lock darf nicht automatisch mit `unlock` entfernt werden;
- nach Pruning ist eine Integritätsprüfung sinnvoll;
- beschädigte Snapshots dürfen nicht ohne vorherige Sicherung und Ursachenanalyse repariert werden.

7.15.26 Rsync-Job prüfen

Version:

```
rsync --version
```

Trockenlauf:

LESEND, NETZAKTIV, möglicherweise LASTERZEUGEND

```
rsync \
  --dry-run \
  --itemize-changes \
  <Quelle> \
  <Ziel>
```

Unmittelbar nach dem Job den Exit-Code ausgeben:

```
printf '%s\n' "$?"
```

Wichtige Rsync-Exit-Codes:

Exit-Code	Bedeutung
0	erfolgreich
1	Syntax- oder Verwendungsfehler
2	Protokollinkompatibilität
3	Fehler bei Auswahl von Ein- oder Ausgabedateien
5	Client-Server-Protokoll konnte nicht gestartet werden
10	Socket-I/O-Fehler
11	Datei-I/O-Fehler
12	Fehler im Rsync-Protokolldatenstrom
23	teilweise Übertragung wegen Fehler

Exit-Code	Bedeutung
24	teilweise Übertragung, weil Quelldateien verschwanden
30	Timeout bei Datenübertragung
35	Timeout beim Warten auf Daemonverbindung

Zu prüfen sind:

- vollständige Quelle und Ziel;
- SSH- oder Daemonverbindung;
- Berechtigungen;
- Eigentümer und Gruppen;
- ACLs;
- erweiterte Attribute;
- Hardlinks;
- Sparse Files;
- symbolische Links;
- Dateisystemgrenzen;
- Ausschlüsse;
- teilweise Übertragungen;
- gelöschte Quelldateien;
- Protokollausgabe;
- Exit-Code;
- verwendete Optionen.

`rsync --delete` kann Daten am Ziel löschen. Eine Synchronisation mit `--delete` ist allein kein versioniertes Backup.

7.15.27 macOS und Time Machine prüfen

Time-Machine-Status:

LESEND

```
tmutil status
```

Konfigurierte Ziele:

```
tmutil destinationinfo
```

Letztes Backup:

```
tmutil latestbackup
```

Vorhandene Backups:

```
tmutil listbackups
```

Aktuelle Time-Machine-Protokolle:

```
log show \
  --predicate 'process == "backupd"' \
  --last 6h \
  --style compact
```

Speicherzustand:

```
df -h
```

Backupvolume:

```
diskutil info "<Backupvolume>"
```

Zu prüfen sind:

- Backupziel angeschlossen;
- Netzwerkziel erreichbar;
- Volume eingebunden;
- freier Speicher auf Quelle und Ziel;
- Verschlüsselungskennwort verfügbar;
- VPN oder Sicherheitssoftware beeinflusst Verbindung;
- Backupvolume unterstützt;
- lokale Snapshots;
- beschädigtes Backup;
- häufig geänderte große Dateien;
- Synchronisationsdienst blockiert Dateien;
- Mac ist entsperrt;
- Stromversorgung;
- Ruhezustand;
- aktuelle macOS-Version.

Netzwerkbasierte Time-Machine-Backups können über das Time-Machine-Menü mit gedrückter Wahltaste und **„Backups überprüfen“** verifiziert werden.

Das Löschen oder Neuformatieren des Backupziels ist keine erste Diagnosemaßnahme.

7.15.28 Docker-Backups prüfen

Containerübersicht:

```
docker ps -a
```

Volumes:

```
docker volume ls
```

Bestimmtes Volume:

```
docker volume inspect \
  <Volumename>
```

Containermounts:

```
docker inspect \
  --format '{{json .Mounts}}' \
  <Containername>
```

Zu unterscheiden sind:

- Containerimage;
- beschreibbarer Container-Layer;
- benanntes Volume;
- anonymes Volume;
- Bind-Mount;
- externe Datenbank;
- Docker-Compose-Konfiguration;
- Secrets;
- Umgebungsvariablen;
- Zertifikate;
- Netzwerk- und Proxykonfiguration.

Wichtige Regeln:

- `docker export` sichert keine eingebundenen Volumes.
- `docker commit` sichert keine Daten in eingebundenen Volumes.
- Daten in Volumes müssen separat gesichert werden.
- Eine Kopie eines aktiven Datenbankvolumes ist nicht automatisch anwendungskonsistent.
- Compose-Datei, Umgebungsvariablen und Secrets müssen getrennt und sicher berücksichtigt werden.
- Direktes Manipulieren der Daten unter dem Docker-Datenverzeichnis ist nicht unterstützt und kann Daten beschädigen.
- Vor dem Backup einer Datenbank muss deren vorgesehenes Sicherungsverfahren verwendet werden.

Ein erfolgreicher Export des Containers beweist nicht, dass die persistenten Anwendungsdaten enthalten sind.

7.15.29 Kubernetes-Backupjobs prüfen

CronJobs:

```
kubectl get cronjobs \
  --all-namespaces
```

Jobs:

```
kubectl get jobs \
  --all-namespaces
```

Bestimmten Job untersuchen:

```
kubectl describe job \
  <Job> \
  -n <Namespace>
```

Podlogs:

```
kubectl logs \
  job/<Job> \
  -n <Namespace>
```

Ereignisse:

```
kubectl get events \
  --all-namespaces \
  --sort-by=.metadata.creationTimestamp
```

PVCs:

```
kubectl get pvc \
  --all-namespaces
```

VolumeSnapshots:

```
kubectl get volumesnapshot \
  --all-namespaces
```

VolumeSnapshotContents:

```
kubectl get volumesnapshotcontent
```

Zu prüfen sind:

- CronJob ausgesetzt;
- letzter Zeitplan;
- fehlgeschlagene Jobs;
- Pod-Exit-Code;
- BackoffLimit;
- aktive Deadline;
- ServiceAccount;
- RBAC;
- Secret;
- Repositoryzugriff;
- PVC-Mount;
- CSI-Treiber;
- VolumeSnapshotClass;
- `readyToUse`;
- Snapshotfehler;
- Node-Druck;
- Ephemeral Storage;
- Netzwerkpolicy;
- externe Datenbank;
- S3- oder Object-Storage-Zugang;
- Aufbewahrung alter Jobs.

Ein Kubernetes-Backup muss je nach Wiederherstellungsziel berücksichtigen:

- Kubernetes-Objekte;
- Namespaces;
- Deployments und StatefulSets;
- Services und Ingress;
- ConfigMaps;
- Secrets;
- CRDs und benutzerdefinierte Ressourcen;
- Persistent Volumes;
- externe Datenbanken;
- Images und Registryabhängigkeiten;
- Clusterkonfiguration;
- gegebenenfalls etcd.

Ein CSI-VolumeSnapshot ist nicht automatisch anwendungskonsistent und nicht automatisch außerhalb des ursprünglichen Storagebackends geschützt.

7.15.30 Datenbanken korrekt sichern

Aktive Datenbanken dürfen nicht wie gewöhnliche statische Dateien behandelt werden.

Zu unterscheiden sind:

- logisches Backup;
- physisches Backup;
- anwendungskonsistenter Snapshot;
- Transaktionslogs;
- Point-in-Time Recovery;
- Replikation;
- dateibasierte Offlinekopie.

Zu prüfen sind:

- Datenbankverbindung;
- Backupbenutzer;
- Berechtigung;
- Lock- und Transaktionszustand;
- verfügbare WAL-, Redo- oder Binärlogs;
- Logarchivierung;
- Backupmodus;
- Datenbankversion;
- Verschlüsselung;
- Komprimierung;
- freier Speicher;
- Konsistenzprüfung;

- Restoreversion;
- Abhängigkeit zu vorherigen Backups.

7.15.31 PostgreSQL-Backups prüfen

PostgreSQL unterstützt unterschiedliche Verfahren:

- `pg_dump` für logische Sicherungen;
- `pg_dumpall` für clusterweite logische Objekte;
- `pg_basebackup` für physische Basissicherungen;
- WAL-Archivierung für Point-in-Time Recovery;
- `pg_verifybackup` zur Prüfung eines von `pg_basebackup` erzeugten Backups.

Basissicherung prüfen:

LESEND, LASTERZEUGEND

```
pg_verifybackup \
  <Backupverzeichnis>
```

Zu prüfen sind:

- Backupmanifest;
- erforderliche WAL-Dateien;
- Replikationsberechtigung;
- `pg_hba.conf`;
- erreichbarer PostgreSQL-Port;
- `max_wal_senders`;
- Archivierungsbefehl;
- WAL-Ziel;
- freier Speicher;
- Backup vom Primary oder Standby;
- Version der Werkzeuge;
- inkrementelle Abhängigkeiten;
- Point-in-Time-Recovery-Konfiguration.

`pg_verifybackup` prüft die Struktur und Integrität einer geeigneten Basissicherung, ersetzt aber keinen vollständigen Wiederherstellungstest.

7.15.32 NAS-, SAN- und Storage-Snapshots prüfen

Zu erfassen sind:

- Storage Pool;

- Volume;
- LUN;
- Dateisystem;
- Snapshot;
- Replikation;
- Zielsystem;
- Retention;
- Thin Provisioning;
- Quota;
- Kompression;
- Deduplizierung;
- Immutable Snapshot;
- Offsitekopie.

Mögliche Fehler:

- Pool voll;
- Volume voll;
- Snapshotreserve voll;
- Thin Pool voll;
- Metadatenbereich voll;
- Snapshotlimit erreicht;
- Replikationsziel nicht erreichbar;
- Zertifikat abgelaufen;
- Storagekonto gesperrt;
- LUN nicht sichtbar;
- Multipathfehler;
- Snapshotabhängigkeit;
- Storagejob überschneidet sich mit Backupjob;
- Snapshot wurde erstellt, aber nicht extern kopiert;
- Quiescing der Anwendung fehlte.

Ein Snapshot auf demselben NAS oder SAN schützt nicht vor dem vollständigen Ausfall oder der Kompromittierung dieses Systems.

7.15.33 Cloud- und Object-Storage-Ziele prüfen

Zu prüfen sind:

- Bucket oder Container;
- Region;
- Endpunkt;
- DNS;
- TLS-Zertifikat;
- Proxy;
- API-Token;

- Zugriffsschlüssel;
- Rolle;
- Schreibberechtigung;
- List-, Read- und Delete-Rechte;
- Object Lock;
- Retention;
- Legal Hold;
- Quota;
- Kosten- oder Kontolimit;
- Rate Limit;
- Request Timeout;
- Multipart Upload;
- unvollständige Uploads;
- KMS-Schlüssel;
- Verschlüsselung;
- Uhrzeit des Clients;
- Providerstatus.

Ein Konto kann Schreibrechte besitzen, aber wegen fehlender List-, Read- oder Delete-Rechte bei Verifikation oder Retention scheitern.

7.15.34 Leistung, Backupfenster und Timeouts prüfen

Zu messen sind:

- gelesene Datenmenge;
- geschriebene Datenmenge;
- Änderungsrate;
- Deduplizierungsrate;
- Kompressionsrate;
- Durchsatz;
- Latenz;
- Paketverlust;
- CPU-Auslastung;
- Arbeitsspeicher;
- I/O-Wait;
- Repositorylatenz;
- Snapshotdauer;
- Gesamtdauer;
- verbleibendes Backupfenster.

Mögliche Ursachen:

- Produktivlast;
- parallele Backupjobs;
- Virenscanner;

- Storage-Scrub;
- Rebuild;
- Snapshotkonsolidierung;
- WAN-Limit;
- hohe Latenz;
- Paketverlust;
- MTU-Problem;
- viele kleine Dateien;
- nicht erreichbarer erster Zielserver;
- zu kleiner Cache;
- langsame Deduplizierungsdatenbank;
- Wartungsjob im Repository;
- Cloud-API-Drosselung;
- neue große Datenmenge;
- geänderte Komprimierung oder Verschlüsselung.

Ein Job, der das Backupfenster überschreitet, kann vom Scheduler oder Produkt beendet werden, obwohl Quelle und Ziel grundsätzlich funktionieren.

7.15.35 Ransomware- und Manipulationsverdacht

Warnsignale:

- mehrere Backupjobs gleichzeitig fehlerhaft;
- Backupkonto unerwartet gesperrt;
- Recovery Points fehlen;
- Retention wurde verändert;
- Immutability wurde deaktiviert;
- Repository enthält ungewöhnliche Löschvorgänge;
- ungewöhnlich viele Dateien wurden geändert;
- Datenmenge oder Deduplizierungsrate verändert sich stark;
- Backups enthalten bereits verschlüsselte Produktivdaten;
- Backupserver zeigt unbekannte Anmeldungen;
- Logs oder Auditdaten fehlen;
- Verschlüsselungsschlüssel wurden verändert;
- Repository ist plötzlich nicht erreichbar;
- ungewöhnliche API-Aufrufe;
- Sekundärkopie oder Offlinekopie fehlt ebenfalls.

Bei begründetem Verdacht:

1. Incident-Response-Prozess aktivieren.
2. Beweise sichern.
3. Repository vor weiteren Veränderungen schützen.
4. Immutability und Offlinekopien prüfen.
5. Backupzugangsdaten als möglicherweise kompromittiert behandeln.

6. Keine normalen Retention- oder Pruningvorgänge starten.
7. Wiederherstellungspunkt vor dem vermuteten Vorfall bestimmen.
8. Restore ausschließlich in isolierter Umgebung testen.
9. Produktivwiederherstellung erst nach Freigabe durchführen.

7.15.36 Hypothese und Gegenbeweis formulieren

Beispiel:

```
Hypothese:
Der geplante Backupjob verwendet ein Dienstkonto,
dessen Kennwort geändert wurde. Deshalb kann der Task
nicht im unbeaufsichtigten Kontext starten.

Erwarteter Befund:
Der Task Scheduler zeigt einen Anmelde- oder
Authentifizierungsfehler.
Der manuelle Start unter einem anderen Konto funktioniert.
Es existiert kein neues Backuplog des eigentlichen Jobs.

Gegenbeweis:
Der Task startet nachweislich unter dem vorgesehenen Konto
und erreicht die erste Phase des Backupjobs.

Testmethode:
Taskstatus, Task-Scheduler-Ereignisse, ausführendes Konto
und Zeitpunkt des letzten Kennwortwechsels vergleichen.

Erfolgskriterium:
Schedulerfehler und Dienstkontoänderung sind zeitlich
und technisch eindeutig zugeordnet.
```

Eine belastbare Hypothese enthält:

- betroffene Jobphase;
- vermutete Ursache;
- erwarteten Befund;
- Gegenbeweis;
- Testmethode;
- Risiko;
- Erfolgskriterium.

7.15.37 Kontrollierte Maßnahmen

Maßnahme	Voraussetzung	Risiko
Zeitplan korrigieren	falscher oder deaktivierter Trigger bestätigt	doppelter oder unerwarteter Lauf
Dienstkonto korrigieren	Konto- oder Anmeldefehler bestätigt	Zugriff auf Quelle und Ziel verändert sich
Zugangsdaten erneuern	abgelaufenes Secret bestätigt	falsches Secret unterbricht weitere Jobs
Quellpfad korrigieren	falscher Pfad bestätigt	falsche Daten könnten gesichert werden
Ausschlussregel korrigieren	fehlende Daten nachgewiesen	Backupmenge steigt
VSS-Writer-Ursache beheben	fehlerhafter Writer dokumentiert	Anwendungsdienst kann unterbrochen werden
Snapshotbereich erweitern	Kapazitätsfehler bestätigt	Storagebelegung steigt
Repositorykapazität erweitern	Repository voll bestätigt	falsche Ebene bleibt wirkungslos
Netzwerkregel korrigieren	blockierter Pfad bestätigt	Sicherheitsgrenze verändert sich
TLS-Zertifikat erneuern	Zertifikatsfehler bestätigt	falsche Kette oder Bindung
verwaisten Lock entfernen	kein aktiver Prozess zweifelsfrei bestätigt	Repositorybeschädigung bei Fehlentscheidung
Retention korrigieren	fehlerhafte Richtlinie bestätigt	Recovery Points können verloren gehen
Pruning ausführen	Retention geprüft und Wartungsfenster vorhanden	hohe Last und Repositorylock
beschädigte Kette neu aufbauen	Kettenschaden bestätigt	zusätzlicher Speicher und längeres Fenster
Vollbackup starten	inkrementelle Basis unbrauchbar	hohe Last und lange Laufzeit
Agent aktualisieren	Kompatibilitätsfehler bestätigt	Neustart oder Versionskonflikt
Anwendungshook korrigieren	Konsistenzfehler bestätigt	Anwendung kann beeinflusst werden
Backup erneut starten	Ursache behoben und Kapazität vorhanden	erneute Last und Snapshotbildung

Vor jeder Maßnahme sind zu dokumentieren:

- Job-ID;
- letzter erfolgreicher Recovery Point;
- aktuelle Schutzlücke;
- Repositoryzustand;
- laufende Prozesse;
- erwartete Datenmenge;

- Risiko;
- Freigabe;
- Rückweg;
- Erfolgskriterium.

7.15.38 Maßnahmen, die nicht spontan ausgeführt werden dürfen

Recovery Points manuell aus dem Repository löschen

Repositoryverzeichnisse umbenennen

Repositorylock ungeprüft entfernen

Retention pauschal verkürzen

Immutability deaktivieren

Object Lock umgehen

Pruning während eines aktiven Backups starten

alle Snapshots löschen

VSS-Dienste vor der Beweissicherung neu starten

Datenbankdateien im laufenden Betrieb kopieren

inkrementelle Kettenglieder einzeln löschen

Backupkatalog ohne Herstellervorgabe neu erstellen

Repositorydaten manuell verändern

Verschlüsselungsschlüssel ersetzen

verlorenes Repositorykennwort „zurücksetzen“

Backupziel neu formatieren

Docker-Volume ungeprüft als einfache Dateikopie sichern

Kubernetes-PVC löschen und neu erstellen

Restore direkt über Produktivdaten ausführen

beschädigtes Backup als einzigen Recovery Point behalten

7.15.39 Vollständiger Diagnoseablauf

1. **Fehlermeldung vollständig aufnehmen**
   Wortlaut, Fehlercode, Job-ID und Zeitpunkt dokumentieren.
2. **Job und Sicherungsprodukt bestimmen**
   Plan, Richtlinie, Agent und Server erfassen.
3. **Auswirkung bestimmen**
   Einzelne Quelle, Jobgruppe, Repository oder gesamte Umgebung unterscheiden.
4. **Letzten erfolgreichen Recovery Point bestimmen**
   Nicht nur letzten erfolgreichen Prozesslauf prüfen.
5. **RPO-Verletzung berechnen**
   Aktuelle Schutzlücke mit Sollwert vergleichen.

6. **Jobphase bestimmen**
   Planung, Snapshot, Lesen, Transport, Schreiben oder Abschluss unterscheiden.
7. **Scheduler prüfen**
   Trigger, Konto, letzte Ausführung und Exit-Code erfassen.
8. **Backupdienste und Agenten prüfen**
   Status und Version dokumentieren.
9. **Logs sichern**
   Job-, Betriebssystem-, Anwendungs- und Repositorylogs korrelieren.
10. **Quelle vollständig bestimmen**
    Pfade, Volumes, VMs, Container und Datenbanken erfassen.
11. **Ausschlüsse prüfen**
    Neue oder versehentlich ausgeschlossene Daten erkennen.
12. **Quellberechtigungen prüfen**
    Tatsächlichen Dienstkontext verwenden.
13. **Quellkapazität prüfen**
    Freien Speicher, Inodes und Temp-Bereich untersuchen.
14. **Snapshotzustand prüfen**
    VSS, Hypervisor, CSI oder Storage-Snapshot unterscheiden.
15. **Anwendungskonsistenz prüfen**
    Writer, Hook oder Datenbankverfahren auswerten.
16. **Ziel und Repository bestimmen**
    Host, Pfad, Port, Protokoll und Backend erfassen.
17. **Netzwerkpfad prüfen**
    DNS, Routing, Firewall, VPN, Proxy und TLS untersuchen.
18. **Zielberechtigungen prüfen**
    Schreiben, Lesen, Listen und gegebenenfalls Löschen unterscheiden.
19. **Repositorykapazität prüfen**
    Speicher, Quota, Thin Pool, Katalog und Cache berücksichtigen.
20. **Repositorylock prüfen**
    Aktive und verwaiste Locks unterscheiden.
21. **Parallele Jobs prüfen**
    Backup, Restore, Replication, Retention und Pruning erfassen.
22. **Sicherungskette prüfen**
    Vollbackup und erforderliche Folgebackups zuordnen.
23. **Retention und Immutability prüfen**
    Aufbewahrung, Object Lock und Legal Hold berücksichtigen.
24. **Verschlüsselung prüfen**
    Schlüssel, Kennwort, KMS und Zertifikate erfassen.
25. **Integrität prüfen**
    Herstellervorgesehene Repositoryprüfung verwenden.
26. **Performance prüfen**
    Dauer, Durchsatz, Latenz, Last und Backupfenster vergleichen.
27. **Sicherheitsverdacht bewerten**
    Manipulation, ungewöhnliche Änderungen und fehlende Recovery Points prüfen.
28. **Hypothese und Gegenbeweis formulieren**
    Ursache vor der Änderung messbar festlegen.

29. **Eine kontrollierte Maßnahme durchführen**
    Risiko, Rückweg und Freigabe beachten.
30. **Job kontrolliert neu starten**
    Doppelstarts und parallele Wartung ausschließen.
31. **Jobphasen überwachen**
    Snapshot, Übertragung und Abschluss beobachten.
32. **Exit-Code und Warnungen auswerten**
    Teilerfolg nicht als Vollerfolg behandeln.
33. **Neuen Recovery Point bestätigen**
    Zeit, Umfang und Repository registrieren.
34. **Integritätsprüfung durchführen**
    Katalog und Datenblöcke prüfen.
35. **Isolierten Restore-Test durchführen**
    Repräsentative Daten oder Anwendung wiederherstellen.
36. **Wiederhergestellte Daten validieren**
    Inhalt, Metadaten, Rechte und Anwendung prüfen.
37. **RPO und RTO bewerten**
    Tatsächliche Werte dokumentieren.
38. **Sekundärkopie prüfen**
    Offsite-, Offline- oder immutable Kopie bestätigen.
39. **Temporäre Änderungen zurücknehmen**
    Debuglogging, Testfreigaben und Ausnahmen entfernen.
40. **Ursache dokumentieren**
    Technischen Nachweis festhalten.
41. **Prävention festlegen**
    Monitoring, Kapazität, Retention oder Jobdesign verbessern.

7.15.40 Befundmatrix

Befund	Mögliche Einordnung	Nächster Nachweis
kein Joblog vorhanden	Scheduler oder Prozessstart fehlerhaft	Scheduler-Ereignisse
Job startet manuell, geplant aber nicht	Dienstkonto, Umgebung oder Rechte	Ausführungskontext vergleichen
Task meldet Erfolg, kein Recovery Point	Skript wertet Exit-Code falsch aus	Unterprozess und Repository prüfen
Quelle fehlt	Mount, Volume oder Pfad nicht verfügbar	Mount- und Volumezustand
nur neue Verzeichnisse fehlen	Ausschluss oder Berechtigung	Jobumfang und ACLs
VSS-Writer fehlgeschlagen	Anwendungskonsistenz fehlerhaft	`vssadmin list writers`
Snapshot erstellt, Job scheitert später	Transport, Repository oder Katalog	nächste Jobphase
Repository voll	Kapazität, Quota oder Retention	Belegung aller Repositoryebenen
Repository hat Platz, Cache ist voll	lokaler Stagingbereich erschöpft	Quellhost und Proxy prüfen

Befund	Mögliche Einordnung	Nächster Nachweis
nur geplanter Netzwerkjob scheitert	gemapptes Laufwerk fehlt	UNC-Pfad und Dienstkonto
Zugriff verweigert	Berechtigung oder geändertes Konto	Quell- und Zielrechte
Authentifizierung fehlgeschlagen	Kennwort, Token oder Zertifikat	Ablauf und Dienstkontext
Repository locked	aktiver oder verwaister Prozess	Prozesse und Jobhistorie
Inkrementbackup fehlt	Kette unterbrochen	Vollbackup und Folgepunkte
Retention schlägt fehl	Immutability oder fehlendes Löschrecht	Richtlinie und Object Lock
Pruning läuft sehr lange	Repositorygröße oder Wartungsüberschneidung	Prozess und I/O-Last
Rsync Exit-Code `23`	teilweise Übertragung	vollständige Fehlerausgabe
Rsync Exit-Code `24`	Quelldateien verschwanden	aktive Anwendung und Dateityp
Dockerexport erfolgreich, Daten fehlen	Volume nicht enthalten	Mounts und Volumes
Kubernetes-Snapshot nicht `readyToUse`	CSI- oder Storagefehler	Snapshotereignisse
Datenbankbackup vorhanden, Restore scheitert	Inkonsistenz oder fehlende Logs	Datenbankverifikation
Backup erfolgreich, Replikation fehlgeschlagen	Zweitkopie fehlt	Replikationsjob und Ziel
Backupdauer plötzlich erhöht	Datenwachstum, Last oder Netzwerk	Baselinevergleich
viele Dateien wurden neu gesichert	Verschlüsselung, Zeitstempel oder Anwendung	Änderungsursache
Prüfsummenfehler	Repository- oder Storagebeschädigung	Integritätsprüfung und Hardware
ältere Recovery Points fehlen unerwartet	Retentionfehler oder Manipulation	Audit- und Repositorylogs
Restore funktioniert, Anwendung nicht	Abhängigkeiten oder Konsistenz fehlen	vollständiger Anwendungstest

---

7.15.41 Typische Diagnosefehler

- Nur die letzte Fehlermeldung betrachten.
- Jobstatus „Warning“ als Erfolg behandeln.
- Exit-Code eines Unterprozesses nicht übernehmen.
- Letzten Job mit letztem Recovery Point gleichsetzen.
- RPO-Verletzung nicht berechnen.
- Scheduler und Backupsoftware nicht getrennt prüfen.
- Manuellen und geplanten Benutzerkontext gleichsetzen.
- Gemappte Laufwerke in unbeaufsichtigten Jobs verwenden.

- Fehlende Mounts übersehen.
- Nur Repositorykapazität prüfen.
- Quell-, Cache-, Snapshot- und Katalogspeicher ignorieren.
- Inodes nicht prüfen.
- VSS-Writer erst nach einem Neustart kontrollieren.
- Alle VSS-Dienste pauschal neu starten.
- Snapshot mit vollständigem Backup gleichsetzen.
- Crash-Konsistenz mit Anwendungskonsistenz verwechseln.
- Aktive Datenbankdateien unkoordiniert kopieren.
- Dockerexport als Volumebackup betrachten.
- Kubernetes-Objekte oder externe Datenbanken vergessen.
- PVC-Größe mit freiem Speicher gleichsetzen.
- Inkrementelle Abhängigkeiten ignorieren.
- Recovery Points manuell aus dem Repository löschen.
- Repositorylock automatisch entfernen.
- Retention unter Zeitdruck verkürzen.
- Immutability als Fehler statt Schutzfunktion behandeln.
- Prüfsummenprüfung mit Restore-Test gleichsetzen.
- Verschlüsselungsschlüssel nicht in den Wiederherstellungsplan aufnehmen.
- Backup und Replikation gleichsetzen.
- Nur lokale Kopie prüfen.
- Keine isolierte Wiederherstellung testen.
- Nur einzelne Datei, aber nicht Anwendung testen.
- Wiederherstellungsdauer nicht messen.
- Backupfehler bei möglichem Ransomwarevorfall normal behandeln.
- Temporäres Debuglogging aktiv lassen.
- Ursache nach erfolgreichem Wiederholungslauf nicht dokumentieren.

7.15.42 Verifikation und Restore-Test

Nach der Maßnahme müssen mindestens folgende Punkte geprüft werden:

- Scheduler ist aktiviert;
- nächster Lauf ist korrekt;
- Dienstkonto ist gültig;
- Job startet im unbeaufsichtigten Kontext;
- alle vorgesehenen Quellen sind enthalten;
- Ausschlüsse entsprechen dem Sollzustand;
- Snapshot wurde erfolgreich erstellt;
- Anwendungskonsistenz wurde bestätigt;
- Daten wurden vollständig übertragen;
- Repository besitzt ausreichend Kapazität;
- Recovery Point wurde registriert;
- Job enthält keine unbeachteten Warnungen;
- Sicherungskette ist vollständig;

- Integritätsprüfung ist erfolgreich;
- Verschlüsselungsschlüssel ist verfügbar;
- sekundäre Kopie wurde erstellt;
- Immutability ist aktiv, sofern vorgesehen;
- repräsentative Dateien lassen sich wiederherstellen;
- Dateiinhalte stimmen;
- Dateigrößen stimmen;
- Zeitstempel stimmen, sofern erforderlich;
- Eigentümer und Berechtigungen stimmen;
- ACLs und erweiterte Attribute stimmen;
- Datenbank lässt sich öffnen;
- Anwendung startet mit den wiederhergestellten Daten;
- Abhängigkeiten und Konfigurationen sind vorhanden;
- tatsächliches RPO wurde bestimmt;
- tatsächliches RTO wurde gemessen;
- Restore-Dokumentation ist vollständig;
- temporäre Testdaten wurden kontrolliert entfernt.

Ein Restore-Test sollte nicht die produktiven Originaldaten überschreiben.

7.15.43 Stufen eines Restore-Tests

Stufe	Nachweis
Katalogtest	Recovery Point wird angezeigt
Lesetest	Sicherungsdaten können gelesen werden
Integritätstest	Prüfsummen und Repositorystruktur stimmen
Datei-Restore	ausgewählte Dateien werden isoliert wiederhergestellt
Metadatenprüfung	Rechte, Eigentümer und Zeitstempel stimmen
Datenbank-Restore	Datenbank wird konsistent wiederhergestellt
Anwendungs-Restore	Anwendung startet und arbeitet
System-Restore	VM, Host oder Dienst wird vollständig wiederhergestellt
DR-Test	definierte Notfallumgebung wird innerhalb des RTO aufgebaut

Je kritischer das System ist, desto höher muss die regelmäßig getestete Restore-Stufe sein.

7.15.44 Prävention und Monitoring

Zu überwachen sind:

- letzter Jobstart;

- letzter erfolgreicher Job;
- letzter Recovery Point;
- letzter verifizierter Recovery Point;
- letzter externer Recovery Point;
- letzter Restore-Test;
- Alter des Restore-Tests;
- Jobdauer;
- übertragene Datenmenge;
- Änderungsrate;
- Durchsatz;
- Warnungen;
- Teilfehler;
- Repositorykapazität;
- Snapshotkapazität;
- Cachekapazität;
- Inodes;
- Quotas;
- Repositorylocks;
- Retention;
- Pruning;
- Immutability;
- Replikation;
- Verschlüsselungsschlüssel;
- Zertifikats- und Tokenablauf;
- RPO;
- RTO.

Ein geeignetes Monitoring alarmiert nicht nur bei `Failed`, sondern auch bei:

- Job wurde nicht gestartet;
- kein neuer Recovery Point;
- Job erfolgreich, aber Datenmenge ungewöhnlich klein;
- Jobdauer ungewöhnlich kurz;
- Jobdauer ungewöhnlich lang;
- nur Teilbackup;
- letzte Verifikation zu alt;
- letzter Restore-Test zu alt;
- sekundäre Kopie fehlt;
- Immutability nicht aktiv;
- Repository wächst unerwartet;
- Retention wurde verändert.

7.15.45 Empfohlene Schutzstruktur

Eine robuste Sicherungsstrategie berücksichtigt mehrere Fehlerdomänen:

```
Produktivdaten
→ primäres Backuprepository
→ getrennte Zweitkopie
→ unveränderbare oder offline Kopie
→ regelmäßig getestete Wiederherstellung
```

Zu vermeiden ist:

```
Produktivdaten
und
einzige Sicherung
auf demselben Host,
demselben Storage Pool
oder mit denselben kompromittierbaren Zugangsdaten
```

Backups sollten entsprechend Schutzbedarf:

- verschlüsselt;
- getrennt;
- unveränderbar;
- überwacht;
- regelmäßig geprüft;
- regelmäßig wiederhergestellt;
- gegen unberechtigte Löschung geschützt sein.

7.15.46 Dokumentationsvorlage

```
Störung:
<exakte Fehlermeldung>

Zeitpunkt:
<Datum und Uhrzeit>

Sicherungsprodukt:
<Produkt und Version>

Jobname:
<Name>
```

Job-ID:

<ID>

Betroffene Quelle:

<System, Volume, Pfad, VM, Container oder Datenbank>

Sicherungsziel:

<Repository, Freigabe, Bucket oder Storage>

Geplanter Start:

<Zeitpunkt>

Tatsächlicher Start:

<Zeitpunkt oder nicht gestartet>

Endzeit:

<Zeitpunkt>

Jobphase:

<Planung, Snapshot, Lesen, Transport, Schreiben,

Katalog, Retention oder Replikation>

Exit-Code:

<Wert>

Letzter erfolgreicher Recovery Point:

<Datum und Uhrzeit>

Letzter verifizierter Recovery Point:

<Datum und Uhrzeit>

Aktuelle Schutzlücke:

<Zeit>

Vorgesehenes RPO:

<Zeit>

Vorgesehenes RTO:

<Zeit>

Schedulerzustand:

<Befund>

Dienstkonto:

<Konto ohne Kennwort>

Quellzustand:

<Pfad, Volume, Mount und Berechtigungen>

Snapshotzustand:

<VSS, Hypervisor, CSI oder Storage>

Anwendungskonsistenz:

<Befund>

Netzwerkpfad:

<DNS, Ziel-IP, Port und Protokoll>

Repositoryzustand:

<Kapazität, Quota, Lock und Integrität>

Sicherungskette:

<Vollbackup und Abhängigkeiten>

Retention und Immutability:

<Befund>

Verschlüsselung:

<Verfahren und Schlüsselverfügbarkeit>

Nachgewiesene Ursache:

<technischer Befund>

Gegenbeweis ausgeschlossen durch:

<Test und Ergebnis>

Durchgeführte Maßnahme:

<genaue Änderung>

```
Risiko und Rückweg:
<Beschreibung>

Neuer Recovery Point:
<Datum, Uhrzeit und ID>

Integritätsprüfung:
<Ergebnis>

Restore-Test:
<Ziel, Umfang und Ergebnis>

Tatsächliches RPO:
<Wert>

Tatsächliches RTO:
<Wert>

Prävention:
<Monitoring, Kapazität, Retention oder Prozessänderung>
```

7.15.47 Checkliste

- [ ] exakte Fehlermeldung dokumentiert
- [ ] Fehlercode dokumentiert
- [ ] Jobname erfasst
- [ ] Job-ID erfasst
- [ ] Sicherungsprodukt und Version erfasst
- [ ] Start- und Endzeit erfasst
- [ ] betroffene Jobphase bestimmt
- [ ] letzten erfolgreichen Job bestimmt
- [ ] letzten Recovery Point bestimmt
- [ ] letzten verifizierten Recovery Point bestimmt
- [ ] aktuelle Schutzlücke berechnet
- [ ] RPO bewertet
- [ ] RTO berücksichtigt

- [ ] Scheduler geprüft
- [ ] Trigger geprüft
- [ ] verpasste Läufe geprüft
- [ ] Dienstkonto geprüft
- [ ] Kennwortablauf geprüft
- [ ] Tokenablauf geprüft
- [ ] Zertifikatsablauf geprüft
- [ ] Recht zur geplanten Anmeldung geprüft
- [ ] Arbeitsverzeichnis geprüft
- [ ] Umgebungsvariablen geprüft
- [ ] Exit-Code geprüft
- [ ] Standardausgabe geprüft
- [ ] Fehlerausgabe geprüft
- [ ] Backupdienste geprüft
- [ ] Agentstatus geprüft
- [ ] Agentversion geprüft
- [ ] vollständige Quelle erfasst
- [ ] Quellpfade geprüft
- [ ] Volumes geprüft
- [ ] Mountpoints geprüft
- [ ] Ausschlüsse geprüft
- [ ] Quellberechtigungen geprüft
- [ ] freien Speicher der Quelle geprüft
- [ ] Inodes geprüft
- [ ] Cache- und Temp-Bereich geprüft
- [ ] Snapshotbereich geprüft
- [ ] VSS-Writer geprüft
- [ ] VSS-Provider geprüft
- [ ] Anwendungskonsistenz geprüft
- [ ] Hypervisor-Snapshot geprüft
- [ ] Storage-Snapshot geprüft
- [ ] Repositoryhostname bestimmt
- [ ] DNS geprüft
- [ ] Ziel-IP bestimmt

- [ ] Port geprüft
- [ ] Firewall geprüft
- [ ] VPN geprüft
- [ ] Proxy geprüft
- [ ] TLS geprüft
- [ ] Repositorypfad geprüft
- [ ] Zielberechtigungen geprüft
- [ ] Repositorykapazität geprüft
- [ ] Repositoryquota geprüft
- [ ] Thin Pool geprüft
- [ ] Repositorylock geprüft
- [ ] parallele Jobs geprüft
- [ ] Restorevorgänge geprüft
- [ ] Retentionjob geprüft
- [ ] Pruning geprüft
- [ ] Replikation geprüft
- [ ] Vollbackup bestimmt
- [ ] inkrementelle Kette geprüft
- [ ] differenzielle Abhängigkeit geprüft
- [ ] Katalog geprüft
- [ ] Repositoryindex geprüft
- [ ] Prüfsummen geprüft
- [ ] Verschlüsselungsschlüssel geprüft
- [ ] KMS-Erreichbarkeit geprüft
- [ ] Immutability geprüft
- [ ] Object Lock geprüft
- [ ] sekundäre Kopie geprüft
- [ ] Offsitekopie geprüft
- [ ] Offlinekopie geprüft
- [ ] Docker-Volumes bei Bedarf geprüft
- [ ] Kubernetes-Objekte bei Bedarf geprüft
- [ ] PVCs bei Bedarf geprüft
- [ ] VolumeSnapshots bei Bedarf geprüft
- [ ] Datenbankverfahren geprüft

- [ ] Transaktionslogs geprüft
- [ ] WAL-, Redo- oder Binärlogs geprüft
- [ ] Jobdauer mit Baseline verglichen
- [ ] Datenmenge mit Baseline verglichen
- [ ] Durchsatz geprüft
- [ ] Backupfenster geprüft
- [ ] Sicherheitsverdacht bewertet
- [ ] Hypothese formuliert
- [ ] Gegenbeweis festgelegt
- [ ] Risiko und Rückweg dokumentiert
- [ ] nur eine kontrollierte Maßnahme durchgeführt
- [ ] neuen Job überwacht
- [ ] neuen Recovery Point bestätigt
- [ ] Integritätsprüfung durchgeführt
- [ ] isolierten Restore-Test durchgeführt
- [ ] Dateiinhalte geprüft
- [ ] Metadaten und Berechtigungen geprüft
- [ ] Datenbank geöffnet
- [ ] Anwendung getestet
- [ ] tatsächliches RPO dokumentiert
- [ ] tatsächliches RTO gemessen
- [ ] temporäre Änderungen zurückgenommen
- [ ] Ursache dokumentiert
- [ ] Präventionsmaßnahme festgelegt

7.15.48 Schnellreferenz

Aufgabe	Befehl
Windows-Task	`Get-ScheduledTask -TaskName "<Taskname>"`
Windows-Task-Laufzeit	`Get-ScheduledTask -TaskName "<Taskname>" | Get-ScheduledTaskInfo`
detaillierte Taskabfrage	`schtasks /query /tn "<Task>" /v /fo LIST /hresult`
Windows-Backupstatus	`wbadmin get status`
Windows-Backupversionen	`wbadmin get versions`
Windows-Backupdatenträger	`wbadmin get disks`

Aufgabe	Befehl
VSS-Writer	`vssadmin list writers`
VSS-Provider	`vssadmin list providers`
VSS-Speicher	`vssadmin list shadowstorage`
Windows-Volumes	`Get-Volume`
Systemd-Timer	`systemctl list-timers --all`
Backup-Service	`systemctl status backup.service`
Backup-Service-Logs	`journalctl -u backup.service --since "-24 hours"`
Benutzer-Cronjobs	`crontab -l`
Root-Cronjobs	`sudo crontab -l`
Linux-Speicher	`df -hT`
Linux-Inodes	`df -i`
Linux-Mounts	`findmnt`
Rsync-Trockenlauf	`rsync --dry-run --itemize-changes <Quelle> <Ziel>`
Restic-Snapshots	`restic snapshots`
Restic-Statistik	`restic stats`
Restic-Locks	`restic list locks`
Restic-Prüfung	`restic check`
Time-Machine-Status	`tmutil status`
Time-Machine-Ziel	`tmutil destinationinfo`
letztes Time-Machine-Backup	`tmutil latestbackup`
Docker-Volumes	`docker volume ls`
Docker-Volume untersuchen	`docker volume inspect <Volume>`
Kubernetes-CronJobs	`kubectl get cronjobs --all-namespaces`
Kubernetes-Jobs	`kubectl get jobs --all-namespaces`
Kubernetes-PVCs	`kubectl get pvc --all-namespaces`
Kubernetes-Snapshots	`kubectl get volumesnapshot --all-namespaces`
PostgreSQL-Basissicherung prüfen	`pg_verifybackup <Backupverzeichnis>`

7.15.49 Quellen

Offizielle Microsoft-Dokumentation

- Microsoft Learn – Windows Server Backup Command Reference

- Microsoft Learn – wbadmin start backup
- Microsoft Learn – Volume Shadow Copy Service
- Microsoft Learn – Backup fails because of a VSS Writer
- Microsoft Learn – No VSS writers are listed
- Microsoft Learn – Get-ScheduledTaskInfo
- Microsoft Learn – schtasks
- Microsoft Learn – schtasks query

Offizielle Linux- und Projektdokumentation

- systemd – systemctl
- systemd – systemd.timer
- systemd – journalctl
- Rsync – Official man page
- Restic – Backing up
- Restic – Restoring
- Restic – Removing backup snapshots
- Restic – Working with repositories
- Restic – Troubleshooting

Offizielle Docker-Dokumentation

- Docker Docs – Volumes
- Docker Docs – Storage
- Docker Docs – docker container export
- Docker Docs – Back up and restore Docker Desktop data

Offizielle Kubernetes-Dokumentation

- Kubernetes – Volume Snapshots
- Kubernetes – Persistent Volumes
- Kubernetes – Operating etcd clusters

Offizielle PostgreSQL-Dokumentation

- PostgreSQL – Backup and Restore
- PostgreSQL – pg_basebackup
- PostgreSQL – pg_verifybackup

- PostgreSQL – Continuous Archiving and Point-in-Time Recovery

Offizielle Apple-Dokumentation

- Apple – Back up your Mac with Time Machine
- Apple – Time Machine troubleshooting
- Apple – If a Time Machine backup fails
- Apple – Verify your backup disk
- Apple – Restore items backed up with Time Machine

Offizielle Sicherheitsempfehlungen

- BSI – Datensicherungskonzept CON.3
- BSI – Top 10 Ransomware-Maßnahmen
- CISA – StopRansomware Guide

Für diese Seite wurden keine Community-Berichte, Hersteller-Social-Media-Aussagen oder eigenen Laborergebnisse als Nachweis verwendet.
