4.14 Virtualisierungs- und Containerfehler

Bei virtualisierten Systemen befindet sich die Ursache einer Störung nicht zwangsläufig innerhalb des betroffenen Servers oder Containers. Der Fehler kann im Gastbetriebssystem, Hypervisor, Hostbetriebssystem, virtuellen Netzwerk, Storage, Managementsystem oder in einer Abhängigkeit außerhalb der Virtualisierungsplattform liegen.

Typische Auswirkungen sind:

- Eine virtuelle Maschine startet nicht.
- Eine VM läuft, ist aber nicht erreichbar.
- Ein Container beendet sich unmittelbar nach dem Start.
- Ein Container befindet sich in einer Neustartschleife.
- Die Anwendung ist nicht über den veröffentlichten Port erreichbar.
- Ein virtueller Datenträger ist voll oder nicht verfügbar.
- Snapshots wachsen unkontrolliert.
- Der Host besitzt zu wenig RAM, CPU oder Speicherplatz.
- Eine VM ist nach einer Migration oder Snapshot-Wiederherstellung inkonsistent.
- Virtuelle Netzwerkkarten sind mit dem falschen Netzwerk verbunden.
- Container können DNS-Namen oder andere Container nicht erreichen.
- Volume-Daten fehlen nach der Neuerstellung eines Containers.
- Das verwendete Image passt nicht zur Prozessorarchitektur.
- Integrationsdienste oder Gasterweiterungen sind fehlerhaft.
- Host und Gast verwenden konkurrierende Zeitquellen.

> **Wichtig:** Eine laufende VM, ein laufender Container oder ein Status `Up` beweist nur, dass der jeweilige Prozess gestartet wurde. Damit ist noch nicht bestätigt, dass Betriebssystem, Anwendung, Netzwerk, Storage und Abhängigkeiten funktionieren.

Kennzeichnung der Befehle

Kennzeichnung	Bedeutung
`[RO]`	Nur lesender Befehl
`[TEST]`	Führt eine aktive Prüfung aus
`[PRIV]`	Erhöhte Berechtigungen erforderlich
`[FILE]`	Liest Dateien, Images, Volumes oder Verzeichnisse
`[SENS]`	Ausgabe kann sensible Informationen enthalten
`[CHANGE]`	Verändert Konfiguration oder Systemzustand
`[DISRUPT]`	Kann den Betrieb unterbrechen

1. Fehlerbereich zuerst eindeutig bestimmen

Virtualisierungsfehler müssen schichtweise eingegrenzt werden.

```
Anwendung
    |
Gastbetriebssystem oder Container
    |
Virtuelle Netzwerkkarte und virtueller Datenträger
    |
Hypervisor beziehungsweise Container-Engine
    |
Hostbetriebssystem
    |
Physisches Netzwerk und Storage
    |
Hardware
```

Erste Prüffragen

- Ist nur eine Anwendung betroffen?
- Ist das gesamte Gastbetriebssystem betroffen?
- Sind mehrere VMs oder Container betroffen?
- Befinden sich die betroffenen Systeme auf demselben Host?
- Nutzen sie denselben virtuellen Switch?
- Nutzen sie dasselbe Storage-System?
- Trat der Fehler nach Migration, Neustart, Snapshot oder Update auf?
- Ist die Managementoberfläche erreichbar?
- Wird die VM beziehungsweise der Container als laufend angezeigt?
- Ist ein Konsolenzugriff möglich?
- Funktioniert die Kommunikation innerhalb des Gasts?
- Sind Hostressourcen ausgeschöpft?
- Wurde eine Konfiguration, ein Image oder ein Compose-Projekt verändert?

Erste Einordnung

Beobachtung	Wahrscheinlicher Fehlerbereich
Nur ein Dienst in einer VM gestört	Anwendung oder Gastbetriebssystem
Gesamte VM nicht erreichbar, Konsole funktioniert	Gastnetzwerk oder virtueller Switch
Gesamte VM reagiert auch über Konsole nicht	Gast, Ressourcen oder virtueller Datenträger

Beobachtung	Wahrscheinlicher Fehlerbereich
Mehrere VMs auf einem Host betroffen	Host, Hypervisor, Netzwerk oder Storage
VMs auf mehreren Hosts betroffen	Gemeinsames Netzwerk, Cluster oder Storage
Nur ein Container beendet sich	Anwendung, Image oder Containerkonfiguration
Alle Container reagieren nicht	Container-Engine, Host oder gemeinsame Ressource
Container läuft, veröffentlichter Port fehlt	Portfreigabe oder Netzwerkkonfiguration
Daten fehlen nach Neuerstellung	Volume oder Bind-Mount nicht korrekt eingebunden

2. VM, Container, Image und Host unterscheiden

Begriff	Bedeutung
Host	Physisches oder virtuelles System, das VMs oder Container ausführt
Hypervisor	Virtualisierungsschicht für virtuelle Maschinen
Virtuelle Maschine	Virtuelles Computersystem mit eigenem Betriebssystemkernel
Gastbetriebssystem	Betriebssystem innerhalb einer VM
Container-Engine	Verwaltet Images, Container, Netzwerke und Volumes
Image	Unveränderliche Vorlage für einen Container
Container	Laufende oder beendete Instanz eines Images
Volume	Von der Containerlebensdauer getrennter persistenter Speicher
Bind-Mount	Verzeichnis oder Datei des Hosts wird in den Container eingebunden
Snapshot beziehungsweise Checkpoint	Zustandsaufnahme einer VM oder eines Datenträgers
Virtueller Switch	Softwarebasierte Netzwerkverbindung für VMs
Portveröffentlichung	Zuordnung eines Hostports zu einem Containerport

Wichtige Unterschiede

- Ein VM-Snapshot ist keine unabhängige Datensicherung.
- Ein Container-Image enthält normalerweise keine später erzeugten Nutzdaten.
- Das Löschen eines Containers muss ein korrekt eingebundenes Volume nicht löschen.
- Daten im beschreibbaren Container-Layer können beim Entfernen des Containers verloren gehen.
- Ein `EXPOSE`-Eintrag im Image veröffentlicht keinen Hostport.

- Ein laufender Container ist nicht automatisch eine funktionsfähige Anwendung.
- Container teilen den Kernel des Hosts; VMs besitzen einen eigenen Gastkernel.

3. Hostzustand vor der Gastanalyse prüfen

Bevor eine VM oder ein Container verändert wird, muss der Hostzustand kontrolliert werden.

Prüfbereich	Typische Auswirkung
CPU-Auslastung	VMs und Container reagieren langsam
RAM-Auslastung	OOM-Ereignisse, Swapping oder VM-Startfehler
Speicherplatz	Images, Snapshots und virtuelle Datenträger können nicht wachsen
Inodes unter Linux	Containerdateien können nicht erstellt werden
Storage-Latenz	VMs frieren ein oder Datenbanken werden langsam
Netzwerk	Mehrere Gäste verlieren gleichzeitig die Verbindung
Zeit	Authentifizierung, Zertifikate und Cluster können fehlschlagen
Hypervisordienst	VMs können nicht verwaltet oder gestartet werden
Container-Engine	Containerstatus und Netzwerke sind nicht verfügbar

Windows-Host

```
[RO] Get-CimInstance Win32_OperatingSystem |
    Select-Object Caption, Version, BuildNumber,
        FreePhysicalMemory, TotalVisibleMemorySize
```

```
[RO] Get-Volume |
    Select-Object DriveLetter, FileSystemLabel,
        HealthStatus, SizeRemaining, Size
```

```
[RO] Get-Counter `
    '\Processor(_Total)\% Processor Time',
    '\Memory\Available MBytes'
```

Linux-Host

```
[RO] uptime
```

```
[RO] free -h
```

```
[RO] df -hT
```

```
[RO] df -i
```

```
[RO] lsblk -f
```

```
[RO] vmstat 1 5
```

macOS-Host

```
[RO] uptime
```

```
[RO] vm_stat
```

```
[RO] df -h
```

```
[RO] diskutil list
```

4. Hyper-V-Host und virtuelle Maschinen prüfen

Die Hyper-V-PowerShell-Befehle sind nur verfügbar, wenn die Hyper-V-Verwaltungstools installiert sind.

Hyper-V-Hostinformationen anzeigen

```
[RO][PRIV] Get-VMHost |
    Select-Object Name, LogicalProcessorCount,
```

```
    MemoryCapacity, VirtualMachinePath,
    VirtualHardDiskPath
```

Alle virtuellen Maschinen anzeigen

```
[RO][PRIV] Get-VM |
  Select-Object Name, State, Status, CPUUsage,
    MemoryAssigned, Uptime, Version
```

Bestimmte VM anzeigen

```
[RO][PRIV] Get-VM -Name "<VM-NAME>" |
  Format-List *
```

Die vollständige Ausgabe kann sehr umfangreich sein.

Nur nicht normal laufende VMs anzeigen

```
[RO][PRIV] Get-VM |
  Where-Object {
    $_.State -ne 'Running' -or $_.Status -ne 'Operating normally'
  } |
  Select-Object Name, State, Status
```

Wichtige VM-Zustände

Zustand	Bedeutung
`Running`	VM ist eingeschaltet
`Off`	VM ist ausgeschaltet
`Paused`	Ausführung wurde angehalten
`Saved`	VM-Zustand wurde gespeichert
`Starting`	VM wird gestartet
`Stopping`	VM wird beendet
`Saving`	Zustand wird gespeichert
`Critical` im Status	Hyper-V meldet einen kritischen Zustand

> Ein Zustand `Running` sagt nichts darüber aus, ob das Gastbetriebssystem vollständig gestartet oder ein Dienst erreichbar ist.

5. Hyper-V-Arbeitsspeicher und CPU prüfen

VM-Arbeitsspeicher anzeigen

```
[RO][PRIV] Get-VMMemory -VMName "<VM-NAME>" |
    Select-Object VMName, DynamicMemoryEnabled,
        Startup, Minimum, Maximum, Buffer,
        Priority
```

Speicherzuweisung aller VMs anzeigen

```
[RO][PRIV] Get-VM |
    Select-Object Name, State,
        @{Name='MemoryAssigned_GB';Expression={
            [math]::Round($_.MemoryAssigned / 1GB, 2)
        }},
        CPUUsage, Uptime
```

Prozessorausweisung einer VM anzeigen

```
[RO][PRIV] Get-VMProcessor -VMName "<VM-NAME>" |
    Select-Object VMName, Count, Reserve,
        Maximum, RelativeWeight, CompatibilityForMigrationEnabled
```

Typische Ressourcenprobleme

- Der Host besitzt nicht genügend verfügbaren RAM zum Starten einer VM.
- Zu viele virtuelle CPUs wurden vergeben.
- Dynamischer Arbeitsspeicher erreicht sein Maximum.
- Gastbetriebssystem oder Anwendung benötigt mehr Startspeicher.
- Host führt gleichzeitig speicherintensive Sicherungs- oder Scanvorgänge aus.
- Mehrere VMs konkurrieren um dieselben physischen Ressourcen.
- NUMA- oder CPU-Kompatibilitätsanforderungen verhindern eine Migration.
- Ein Gast verbraucht RAM, ohne ihn an den Host zurückzugeben.

Zusätzlich innerhalb der VM prüfen

Windows-Gast:

```
[RO] Get-Counter `
    '\Processor(_Total)\% Processor Time',
    '\Memory\Available MBytes'
```

Linux-Gast:

```
[RO] free -h
```

```
[RO] vmstat 1 5
```

6. Hyper-V-Netzwerk untersuchen

Virtuelle Switches anzeigen

```
[RO][PRIV] Get-VMSwitch |
    Select-Object Name, SwitchType, NetAdapterInterfaceDescription
```

Virtuelle Netzwerkadapter einer VM anzeigen

```
[RO][PRIV] Get-VMNetworkAdapter -VMName "<VM-NAME>" |
    Select-Object VMName, Name, SwitchName,
        Status, MacAddress, IPAddresses
```

Alle VM-Netzwerkadapter anzeigen

```
[RO][PRIV] Get-VMNetworkAdapter -All |
    Select-Object VMName, Name, SwitchName,
        Status, MacAddress, IPAddresses
```

VLAN-Konfiguration anzeigen

```
[RO][PRIV] Get-VMNetworkAdapterVlan -VMName "<VM-NAME>"
```

Erweiterte Adapterfunktionen anzeigen

```
[RO][PRIV] Get-VMNetworkAdapterAdvancedFeature `
    -VMName "<VM-NAME>"
```

Typische Ursachen

- VM ist mit dem falschen virtuellen Switch verbunden.
- Switch wurde gelöscht oder umbenannt.
- VLAN-ID stimmt nicht.
- Physischer Hostadapter ist getrennt.
- MAC-Adresse wird durch Port-Security blockiert.
- Statische IP-Adresse im Gast ist falsch.
- DHCP ist im virtuellen Netzwerk nicht erreichbar.
- Eine Hostfirewall blockiert Management- oder Gastverkehr.
- Virtueller Switch besitzt den falschen Typ.

Hyper-V-Switchtypen

Typ	Verbindung
External	Verbindung zum physischen Netzwerk
Internal	Kommunikation zwischen Host und VMs
Private	Kommunikation nur zwischen VMs desselben Hosts

7. Hyper-V-Datenträger und Checkpoints prüfen

Virtuelle Festplatten einer VM anzeigen

```
[RO][PRIV] Get-VMHardDiskDrive -VMName "<VM-NAME>" |
    Select-Object VMName, ControllerType,
        ControllerNumber, ControllerLocation, Path
```

Informationen über eine bekannte VHD- oder VHDX-Datei

```
[RO][PRIV][SENS] Get-VHD -Path "<VHDX-PFAD>" |
    Select-Object Path, VhdFormat, VhdType,
        FileSize, Size, MinimumSize, ParentPath,
        Attached
```

Checkpoints einer VM anzeigen

```
[RO][PRIV] Get-VMSnapshot -VMName "<VM-NAME>" |
    Select-Object VMName, Name, SnapshotType,
        CreationTime, ParentSnapshotName
```

Checkpoints aller VMs anzeigen

```
[RO][PRIV] Get-VM |
    Get-VMSnapshot |
    Select-Object VMName, Name, SnapshotType,
        CreationTime
```

Speicherpfade und freien Speicher vergleichen

```
[RO] Get-Volume |
    Select-Object DriveLetter, FileSystemLabel,
        HealthStatus, SizeRemaining, Size
```

Typische Ursachen

- VHDX-Datei fehlt oder wurde verschoben.
- Berechtigungen auf dem Speicherpfad fehlen.
- Ein differenzierender Datenträger findet seinen Parent nicht.
- Snapshot- beziehungsweise AVHDX-Dateien wachsen stark.
- Das Hostvolume ist voll.
- Ein Cluster Shared Volume ist nicht verfügbar.
- SAN-, NAS- oder SMB-Speicher ist nicht erreichbar.
- VHDX ist bereits anderweitig eingebunden.
- Dateisystem oder physischer Datenträger meldet Fehler.

> AVHDX-Dateien und Snapshotketten dürfen nicht manuell im Dateisystem gelöscht oder umbenannt werden. Dadurch kann die virtuelle Festplattenkette unbrauchbar werden.

8. Hyper-V-Integrationsdienste und Ereignisprotokolle prüfen

Integrationsdienste einer VM anzeigen

```
[RO][PRIV] Get-VMIntegrationService -VMName "<VM-NAME>" |
    Select-Object Name, Enabled, PrimaryStatusDescription,
        SecondaryStatusDescription
```

Hyper-V-VMMS-Ereignisse

```
[RO][SENS] Get-WinEvent -LogName `
    'Microsoft-Windows-Hyper-V-VMMS-Admin' `
    -ErrorAction SilentlyContinue |
Select-Object -First 100 TimeCreated, LevelDisplayName, Id, Message
```

Hyper-V-Worker-Ereignisse

```
[RO][SENS] Get-WinEvent -LogName `
    'Microsoft-Windows-Hyper-V-Worker-Admin' `
    -ErrorAction SilentlyContinue |
Select-Object -First 100 TimeCreated, LevelDisplayName, Id, Message
```

Nur Ereignisse der letzten 24 Stunden

```
[RO][SENS] Get-WinEvent -FilterHashtable @{
    LogName   = 'Microsoft-Windows-Hyper-V-VMMS-Admin'
    StartTime = (Get-Date).AddHours(-24)
} -ErrorAction SilentlyContinue |
Select-Object TimeCreated, LevelDisplayName, Id, Message
```

Typische Meldungsbereiche

- Startfehler
- Arbeitsspeicher konnte nicht reserviert werden
- Virtuelle Festplatte nicht gefunden
- Zugriff auf Speicherpfad verweigert
- Virtueller Switch nicht vorhanden
- Checkpoint konnte nicht erstellt oder zusammengeführt werden
- Migration fehlgeschlagen
- Konfigurationsversion nicht unterstützt
- Integrationsdienst nicht verfügbar

9. KVM und libvirt unter Linux prüfen

Alle libvirt-VMs anzeigen

```
[RO][PRIV] sudo virsh list --all
```

Status einer VM

```
[RO][PRIV] sudo virsh domstate "<VM-NAME>"
```

Ausführliche VM-Informationen

```
[RO][PRIV][SENS] sudo virsh dominfo "<VM-NAME>"
```

Virtuelle Netzwerkadapter

```
[RO][PRIV][SENS] sudo virsh domiflist "<VM-NAME>"
```

Virtuelle Datenträger

```
[RO][PRIV][SENS] sudo virsh domblklist "<VM-NAME>" --details
```

VM-Ressourcenstatistik

```
[TEST][PRIV][SENS] sudo virsh domstats "<VM-NAME>"
```

Snapshots anzeigen

```
[RO][PRIV] sudo virsh snapshot-list "<VM-NAME>"
```

Virtuelle Netzwerke anzeigen

```
[RO][PRIV] sudo virsh net-list --all
```

Storage-Pools anzeigen

```
[RO][PRIV] sudo virsh pool-list --all
```

Storage-Volumes eines Pools anzeigen

```
[RO][PRIV][SENS] sudo virsh vol-list "<POOL-NAME>"
```

> `virsh` kann je nach Verbindung auf eine lokale oder entfernte libvirt-Instanz zugreifen. Vor Änderungen muss geprüft werden, welche Verbindung und welcher Host tatsächlich verwendet werden.

10. KVM-, QEMU- und libvirt-Dienste untersuchen

Je nach Distribution und libvirt-Version können unterschiedliche Dienste verwendet werden.

Klassischer libvirt-Dienst

```
[RO] systemctl status libvirtd --no-pager
```

Modularer QEMU-Dienst

```
[RO] systemctl status virtqemud --no-pager
```

Laufende QEMU-Prozesse anzeigen

```
[RO][SENS] ps -ef | grep '[q]emu-system'
```

libvirt-Protokolle

```
[RO][PRIV][SENS] sudo journalctl -u libvirtd \
    --since "-24 hours" --no-pager
```

virtqemud-Protokolle

```
[RO][PRIV][SENS] sudo journalctl -u virtqemud \
    --since "-24 hours" --no-pager
```

Kernelunterstützung prüfen

```
[RO] lsmod | grep -E '^kvm|kvm_'
```

CPU-Virtualisierungsmerkmale anzeigen

```
[RO] lscpu |
    grep -E 'Virtualization|Hypervisor|Architecture'
```

KVM-Gerät prüfen

```
[RO][FILE] ls -l /dev/kvm
```

Typische Ursachen

- Virtualisierung ist in BIOS beziehungsweise UEFI deaktiviert.
- `/dev/kvm` fehlt oder besitzt falsche Berechtigungen.
- libvirt-Dienst ist nicht aktiv.
- VM-Konfiguration verweist auf fehlende Datenträger.
- Storage-Pool ist nicht aktiv.
- Virtuelles Netzwerk ist nicht aktiv.
- AppArmor oder SELinux blockiert den Zugriff.
- QEMU-Version unterstützt eine Geräteoption nicht.
- CPU-Modell ist nach einer Migration nicht kompatibel.
- Host besitzt nicht genügend RAM.

11. VMware und andere Hypervisoren systematisch eingrenzen

Die genauen Befehle hängen von Produkt, Version und Managementplattform ab. Ohne bestätigte Plattform sollten keine produktspezifischen Befehle übernommen werden.

Zu prüfende Bereiche

- Hostverbindungsstatus
- VM-Energiezustand
- Hostressourcen
- Datastore-Kapazität
- Snapshotstatus
- Virtuelle Switches und Portgruppen
- VLAN-Zuordnung
- Virtuelle Netzwerkkarten
- Virtuelle Datenträger
- VMware Tools beziehungsweise Gasterweiterungen
- HA- und Clusterstatus
- vMotion- beziehungsweise Migrationsereignisse
- Alarme und Aufgabenverlauf
- Lizenz- und Zertifikatsstatus
- Managementserver-Verbindung

Typische Fehler

Beobachtung	Mögliche Ursache
VM ist „orphaned“ oder „inaccessible“	Konfigurationsdatei oder Datastore nicht erreichbar
Snapshot kann nicht erstellt werden	Datastore voll oder Snapshotkette fehlerhaft
Migration scheitert	CPU, Netzwerk, Storage oder Kompatibilität
VM-Netzwerk fehlt	Portgruppe oder virtueller Switch falsch
Konsole funktioniert, Netzwerk nicht	Gastnetzwerk, VLAN oder Portgruppe
Mehrere Hosts nicht verwaltbar	Managementserver, Zertifikat oder Netzwerk
HA startet VM nicht	Clusterkapazität, Admission Control oder Storage
Tools-Status veraltet	Gasterweiterungen fehlen oder sind inkompatibel

> Änderungen an Datastore-Dateien dürfen nur mit dokumentierten Herstellerverfahren durchgeführt werden.

12. Docker-Client und Docker-Daemon unterscheiden

Docker besteht mindestens aus:

- Docker-Client
- Docker-Daemon beziehungsweise Engine
- Container Runtime
- Images
- Container
- Netzwerke
- Volumes

Der Client kann installiert sein, obwohl der Daemon nicht erreichbar ist.

Client- und Serverversion anzeigen

```
[RO] docker version
```

Wenn nur der Clientbereich erscheint und der Serverbereich einen Fehler meldet, kann der Docker-Daemon nicht erreicht werden.

Ausführliche Docker-Informationen

```
[RO][SENS] docker info
```

Die Ausgabe kann enthalten:

- Hostname
- Betriebssystem
- Kernelversion
- Storage-Treiber
- Anzahl der Container und Images
- Netzwerk-Plugins
- Registryinformationen
- Sicherheitsoptionen
- Docker-Root-Verzeichnis
- Proxyinformationen

Docker-Kontext anzeigen

```
[RO][SENS] docker context show
```

Alle Docker-Kontexte anzeigen

```
[RO][SENS] docker context ls
```

> Ein falscher Docker-Kontext kann dazu führen, dass Befehle gegen einen anderen Host oder Docker Desktop statt gegen die erwartete Engine ausgeführt werden.

13. Docker-Dienst unter Linux prüfen

Docker-Dienststatus

```
[RO] systemctl status docker --no-pager
```

Container-Runtime prüfen

```
[RO] systemctl status containerd --no-pager
```

Docker-Protokolle der letzten 24 Stunden

```
[RO][PRIV][SENS] sudo journalctl -u docker \
  --since "-24 hours" --no-pager
```

containerd-Protokolle

```
[RO][PRIV][SENS] sudo journalctl -u containerd \
  --since "-24 hours" --no-pager
```

Docker-Socket prüfen

```
[RO][FILE] ls -l /var/run/docker.sock
```

Typische Fehlermeldungen

Meldung	Mögliche Ursache
`Cannot connect to the Docker daemon`	Daemon gestoppt, falscher Socket oder falscher Kontext

Meldung	Mögliche Ursache
`permission denied` am Socket	Benutzer besitzt keine erforderliche Berechtigung
`no space left on device`	Speicherplatz oder Inodes ausgeschöpft
`address already in use`	Hostport bereits belegt
`network not found`	Konfiguriertes Docker-Netzwerk fehlt
`volume not found`	Extern erwartetes Volume fehlt
`manifest unknown`	Image oder Tag existiert nicht
`no matching manifest`	Architektur oder Plattform wird nicht angeboten
`pull access denied`	Registry, Anmeldung oder Berechtigung
`toomanyrequests`	Registry-Limit erreicht

14. Containerstatus und Exit-Code prüfen

Laufende Container anzeigen

```
[RO][SENS] docker ps
```

Alle Container einschließlich beendeter Container

```
[RO][SENS] docker ps -a
```

Kompakte Zustandsübersicht

```
[RO][SENS] docker ps -a \
    --format 'table {{.Names}}\t{{.Image}}\t{{.Status}}\t{{.Ports}}'
```

Zustand eines Containers detailliert anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
    --format '{{json .State}}'
```

Wichtige Zustandsinformationen gezielt anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
    --format 'Status={{.State.Status}} ExitCode={{.State.ExitCode}} OOMKilled={{.State.OOMKilled}}
Error={{.State.Error}} StartedAt={{.State.StartedAt}} FinishedAt={{.State.FinishedAt}}'
```

Neustartzähler anzeigen

```
[RO] docker inspect "<CONTAINER>" \
    --format 'RestartCount={{.RestartCount}}'
```

Typische Exit-Codes

Exit-Code	Typische Bedeutung
`0`	Prozess wurde erfolgreich beendet
`1`	Allgemeiner Anwendungsfehler
`2`	Häufig falsche Argumente oder Anwendungsfehler
`125`	Docker konnte den Container nicht starten
`126`	Befehl gefunden, aber nicht ausführbar
`127`	Befehl nicht gefunden
`137`	Prozess erhielt häufig `SIGKILL`; OOM oder erzwungenes Beenden möglich
`139`	Segmentation Fault möglich
`143`	Prozess erhielt normalerweise `SIGTERM`

> Ein Exit-Code beschreibt zunächst nur das Ende des Hauptprozesses. Die genaue Ursache muss mit Containerprotokoll, Docker-Ereignissen und Hostmeldungen bestätigt werden.

15. Containerprotokolle sicher auswerten

Letzte 100 Protokollzeilen

```
[RO][SENS] docker logs --tail 100 "<CONTAINER>"
```

Protokolle mit Zeitstempeln

```
[RO][SENS] docker logs --timestamps --tail 100 "<CONTAINER>"
```

Protokolle seit einem bestimmten Zeitraum

```
[RO][SENS] docker logs --since 1h "<CONTAINER>"
```

Protokolle fortlaufend beobachten

```
[TEST][SENS] docker logs --follow --tail 100 "<CONTAINER>"
```

Die fortlaufende Ausgabe wird mit Strg + C beendet. Dadurch wird normalerweise nicht der Container beendet, sondern nur die lokale Anzeige.

Wichtige Einschränkungen

- `docker logs` funktioniert abhängig vom verwendeten Logging-Treiber.
- Anwendungen können zusätzlich in Dateien, Datenbanken oder externe Systeme protokollieren.
- Protokolle können Passwörter, Tokens, URLs, Benutzernamen oder personenbezogene Daten enthalten.
- Unbegrenzte Logabfragen können sehr große Datenmengen ausgeben.
- Fehlende Ausgabe bedeutet nicht automatisch, dass kein Fehler vorhanden ist.
- Ein Container kann bereits vor Initialisierung der Anwendungsprotokollierung beendet werden.

16. Healthcheck und Anwendungszustand prüfen

Ein Docker-Healthcheck bewertet einen im Image oder Container definierten Test.

Healthcheckstatus anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format '{{json .State.Health}}'
```

Kompakte Anzeige

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format 'Status={{.State.Status}} Health={{if .State.Health}}{{.State.Health.Status}}{{else}}nicht
```

```
definiert{{end}}'
```

Healthcheck-Konfiguration anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format '{{json .Config.Healthcheck}}'
```

Mögliche Health-Zustände

Zustand	Bedeutung
`starting`	Startphase oder noch nicht genügend Prüfungen
`healthy`	Definierter Test war erfolgreich
`unhealthy`	Definierter Test ist wiederholt fehlgeschlagen
Nicht vorhanden	Kein Docker-Healthcheck definiert

Wichtige Bewertung

- `healthy` bestätigt nur den definierten Test.
- Ein zu einfacher Test kann echte Fehler übersehen.
- Ein zu strenger Test kann einen funktionierenden Dienst als fehlerhaft markieren.
- Der Test kann von Werkzeugen abhängen, die im Image fehlen.
- Ein Healthcheck kann intern funktionieren, obwohl der Dienst extern nicht erreichbar ist.
- Ein Container kann `Up` sein und trotzdem `unhealthy`.

17. Containerressourcen und OOM-Fehler untersuchen

Aktuelle Ressourcennutzung einmalig anzeigen

```
[TEST][SENS] docker stats --no-stream
```

Nur einen Container anzeigen

```
[TEST][SENS] docker stats --no-stream "<CONTAINER>"
```

Konfigurierte Ressourcenlimits anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
    --format 'Memory={{.HostConfig.Memory}} MemorySwap={{.HostConfig.MemorySwap}}
NanoCPUs={{.HostConfig.NanoCpus}} PidsLimit={{.HostConfig.PidsLimit}}'
```

Prüfen, ob der Container wegen Speichermangel beendet wurde

```
[RO] docker inspect "<CONTAINER>" \
    --format 'OOMKilled={{.State.OOMKilled}} ExitCode={{.State.ExitCode}}'
```

Linux-Kernelmeldungen zu OOM

```
[RO][PRIV][SENS] sudo journalctl -k -b --no-pager |
    grep -Ei 'out of memory|oom-kill|killed process'
```

Typische Ursachen

- Container besitzt ein zu niedriges Speicherlimit.
- Host besitzt keinen freien Arbeitsspeicher.
- Anwendung besitzt ein Speicherleck.
- Java-, Node.js- oder Datenbankkonfiguration ignoriert die Containergrenze.
- Zu viele Prozesse erreichen das PID-Limit.
- Hohe I/O-Wartezeit wird als CPU- oder Anwendungsproblem fehlinterpretiert.
- Ein Container konkurriert ohne Limits mit anderen Workloads.

> Ein höheres Speicherlimit behebt kein Speicherleck. Zuerst müssen Verbrauch, Wachstum und Anwendungsprotokolle untersucht werden.

18. Docker-Prozesse und Hauptprozess prüfen

Prozesse im Container anzeigen

```
[RO][SENS] docker top "<CONTAINER>"
```

Konfigurierten Startbefehl anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format 'Entrypoint={{json .Config.Entrypoint}} Cmd={{json .Config.Cmd}}'
```

Arbeitsverzeichnis und Benutzer anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format 'User={{.Config.User}} WorkingDir={{.Config.WorkingDir}}'
```

Typische Ursachen eines sofortigen Containerendes

- Hauptprozess beendet sich regulär.
- Startbefehl oder Entrypoint ist falsch.
- Binärdatei oder Skript fehlt.
- Datei besitzt keine Ausführungsberechtigung.
- Konfigurationsdatei fehlt.
- Umgebungsvariable ist nicht gesetzt.
- Abhängiger Dienst ist noch nicht erreichbar.
- Datenbankmigration schlägt fehl.
- Architektur des Images passt nicht zum Host.
- Anwendung läuft als falscher Benutzer.
- Gemountete Datei verdeckt eine Datei aus dem Image.

> Ein Container bleibt nur so lange aktiv, wie sein Hauptprozess läuft. Das erfolgreiche Erstellen des Containers bedeutet nicht, dass dieser Prozess dauerhaft läuft.

19. Docker-Portveröffentlichungen und Listener prüfen

Portzuordnungen aller Container

```
[RO][SENS] docker ps \
  --format 'table {{.Names}}\t{{.Ports}}'
```

Portzuordnung eines Containers

```
[RO][SENS] docker port "<CONTAINER>"
```

Detaillierte Portkonfiguration

```
[RO][SENS] docker inspect "<CONTAINER>" \
    --format '{{json .NetworkSettings.Ports}}'
```

Listener auf dem Linux-Host

```
[RO][PRIV][SENS] sudo ss -lntup
```

Listener auf Windows

```
[RO][PRIV][SENS] Get-NetTCPConnection -State Listen |
    Sort-Object LocalPort |
    Select-Object LocalAddress, LocalPort, OwningProcess
```

Listener auf macOS

```
[RO][PRIV][SENS] sudo lsof -nP -iTCP -sTCP:LISTEN
```

Wichtige Unterscheidung

```
Client
  |
  ▼
Host-IP:Hostport
  |
Portveröffentlichung
  ▼
Container-IP:Containerport
  |
  ▼
Anwendungslistener
```

Ein Fehler kann an jeder Stelle liegen.

Typische Ursachen

- Kein Hostport wurde veröffentlicht.

- Falscher Containerport wurde eingetragen.
- Anwendung lauscht auf einem anderen Port.
- Anwendung bindet nur an `127.0.0.1` im Container.
- Hostport wird bereits verwendet.
- Hostfirewall blockiert den Port.
- Reverse Proxy zeigt auf den falschen Hostnamen oder Port.
- Anwendung startet erst nach längerer Initialisierung.
- IPv4- und IPv6-Bindung werden verwechselt.

20. Docker-Netzwerke und DNS untersuchen

Docker-Netzwerke anzeigen

```
[RO][SENS] docker network ls
```

Netzwerkdetails anzeigen

```
[RO][SENS] docker network inspect "<NETZWERK>"
```

Die Ausgabe kann Container-IP-Adressen, Netzwerknamen und weitere interne Informationen enthalten.

Netzwerke eines Containers anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format '{{json .NetworkSettings.Networks}}'
```

DNS-Konfiguration im Container anzeigen

```
[RO][SENS] docker exec "<CONTAINER>" cat /etc/resolv.conf
```

Der Befehl funktioniert nur, wenn der Container läuft und `cat` im Image vorhanden ist.

Namensauflösung im Container testen

```
[TEST][SENS] docker exec "<CONTAINER>" \
  getent hosts "<ZIELNAME>"
```

`getent` ist nicht in jedem Container-Image vorhanden.

TCP-Verbindung aus einem Container testen

```
[TEST][SENS] docker exec "<CONTAINER>" \
    curl -v --connect-timeout 5 "http://<ZIEL>:<PORT>/"
```

`curl` ist nicht in jedem Image installiert. Fehlende Diagnosewerkzeuge dürfen nicht unkontrolliert in einen Produktivcontainer installiert werden.

Typische Ursachen

- Container befinden sich in unterschiedlichen Netzwerken.
- Falscher DNS-Name wird verwendet.
- Dienstname stimmt nicht mit Compose-Konfiguration überein.
- Anwendung nutzt `localhost` für einen anderen Container.
- Netzwerk wurde nach Containererstellung geändert.
- DNS-Server des Hosts ist nicht erreichbar.
- Überschneidende IP-Netze verursachen falsche Routen.
- VPN kollidiert mit dem Docker-Adressbereich.
- Firewall oder Forwardingregel blockiert den Verkehr.

> `localhost` innerhalb eines Containers bezeichnet normalerweise den Container selbst und nicht den Docker-Host oder einen anderen Container.

21. Docker-Volumes und Bind-Mounts untersuchen

Volumes anzeigen

```
[RO][SENS] docker volume ls
```

Volumeinformationen anzeigen

```
[RO][SENS] docker volume inspect "<VOLUME>"
```

Mounts eines Containers anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
    --format '{{json .Mounts}}'
```

Kompakte Mountübersicht

```
[RO][SENS] docker inspect "<CONTAINER>" \
    --format '{{range .Mounts}}{{println .Type .Source "->" .Destination "RW=" .RW}}{{end}}'
```

Speicherverbrauch der Docker-Objekte

```
[TEST][SENS] docker system df
```

Ausführlich:

```
[TEST][SENS] docker system df -v
```

Typische Ursachen fehlender Daten

- Falsches Volume wurde eingebunden.
- Volume wurde unter einem neuen Projektnamen erstellt.
- Bind-Mount verweist auf einen falschen Hostpfad.
- Ein leerer Hostpfad verdeckt vorhandene Image-Daten.
- Anwendung schreibt außerhalb des vorgesehenen Mountpoints.
- Container läuft als Benutzer ohne Schreibberechtigung.
- Volume wurde versehentlich entfernt.
- NFS-, SMB- oder Netzwerkspeicher ist nicht verfügbar.
- SELinux verhindert den Zugriff.
- Datenträger des Docker-Hosts ist voll.
- Bei Docker Desktop wurde die Dateifreigabe oder virtuelle Storage-Schicht gestört.

> “ `docker volume prune` und `docker system prune` dürfen nicht als allgemeine Fehlerbehebung ausgeführt werden. Sie können nicht verwendete, aber weiterhin benötigte Daten oder Images entfernen.

22. Docker-Images und Architektur prüfen

Lokale Images anzeigen

```
[RO][SENS] docker image ls
```

Imageinformationen anzeigen

```
[RO][SENS] docker image inspect "<IMAGE>:<TAG>"
```

Architektur eines lokalen Images

```
[RO] docker image inspect "<IMAGE>:<TAG>" \
    --format 'OS={{.Os}} Architecture={{.Architecture}}'
```

Hostarchitektur

Linux:

```
[RO] uname -m
```

macOS:

```
[RO] uname -m
```

Windows:

```
[RO] Get-CimInstance Win32_OperatingSystem |
    Select-Object OSArchitecture
```

Typische Fehlermeldungen

Meldung	Mögliche Ursache
`no matching manifest`	Image unterstützt die Hostplattform nicht
`exec format error`	Binärdatei besitzt falsche Architektur oder fehlerhaften Interpreter
`manifest unknown`	Tag oder Image existiert nicht
`pull access denied`	Privates Image oder falscher Name
`unauthorized`	Registryanmeldung oder Berechtigung fehlt
`certificate signed by unknown authority`	Registryzertifikat wird nicht vertraut

Meldung	Mögliche Ursache
`unexpected EOF`	Download oder Netzwerkverbindung abgebrochen

Multi-Architektur-Images

Ein Image-Tag kann verschiedene plattformspezifische Manifestvarianten enthalten. Deshalb muss geprüft werden, ob genau die benötigte Kombination aus Betriebssystem und Architektur angeboten wird.

23. Umgebungsvariablen und Geheimnisse prüfen

Konfigurierte Containerumgebung anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format '{{range .Config.Env}}{{println .}}{{end}}'
```

> Diese Ausgabe kann Passwörter, Tokens, API-Schlüssel und Datenbankzugänge enthalten. Sie darf nicht unredigiert in Tickets, Dokumentationen oder externe Systeme kopiert werden.

Nur Namen der Umgebungsvariablen anzeigen

```
[RO][SENS] docker inspect "<CONTAINER>" \
  --format '{{range .Config.Env}}{{println .}}{{end}}' |
  sed 's/=.*$/=<REDACTED>/'
```

Typische Fehler

- Pflichtvariable fehlt.
- Variablenname ist falsch geschrieben.
- Wert enthält unbeabsichtigte Leerzeichen.
- Sonderzeichen wurden durch Shell oder Compose interpretiert.
- Datenbankhostname zeigt auf `localhost`.
- Portnummer ist falsch.
- Geheimnis wurde rotiert, Container aber nicht neu erstellt.
- `.env`-Datei wurde aus einem anderen Verzeichnis geladen.
- Compose-Projekt verwendet eine andere Konfigurationsdatei.
- Variable ist im Image festgelegt und wird unerwartet überschrieben.

Sicherheitsregel

Geheimnisse sollten nicht dauerhaft als Klartext in:

- Screenshots
- Tickets
- BookStack-Seiten
- Chatverläufen
- öffentlichen Repositories
- ungeschützten Diagnoseberichten

gespeichert werden.

24. Docker-Compose-Projekte untersuchen

Je nach Installation wird das aktuelle Plugin mit `docker compose` verwendet. Ältere Systeme können noch das separate Werkzeug `docker-compose` besitzen.

Projektstatus anzeigen

```
[RO][SENS] docker compose ps -a
```

Compose-Prozesse anzeigen

```
[RO][SENS] docker compose top
```

Letzte Protokollzeilen aller Dienste

```
[RO][SENS] docker compose logs --tail 100
```

Protokoll eines Dienstes

```
[RO][SENS] docker compose logs --tail 100 "<DIENST>"
```

Zusammengeführte Konfiguration prüfen

```
[RO][SENS] docker compose config
```

> `docker compose config` kann aufgelöste Umgebungsvariablen und sensible Werte enthalten. Die Ausgabe muss vor Weitergabe redigiert werden.

Dienste der Konfiguration anzeigen

```
[RO] docker compose config --services
```

Images der Konfiguration anzeigen

```
[RO][SENS] docker compose config --images
```

Typische Compose-Fehler

- Befehl wird im falschen Verzeichnis ausgeführt.
- Falsche Compose-Datei wird verwendet.
- Projektname hat sich geändert.
- Externes Netzwerk oder Volume fehlt.
- `.env`-Datei fehlt.
- Variable wurde nicht ersetzt.
- Abhängiger Dienst ist gestartet, aber noch nicht betriebsbereit.
- Port ist bereits belegt.
- Bind-Mount-Pfad existiert auf dem Host nicht.
- Image-Tag zeigt inzwischen auf eine andere Version.
- Mehrere Compose-Dateien überschreiben sich unerwartet.

25. Docker-Ereignisse und Neustartschleifen untersuchen

Ereignisse der letzten Stunde

```
[RO][SENS] docker events --since 1h
```

Der Befehl wartet nach Ausgabe vorhandener Ereignisse auf neue Ereignisse und wird mit `Strg` + `C` beendet.

Ereignisse eines Containers

```
[RO][SENS] docker events \
  --since 1h \
```

```
    --filter container="<CONTAINER>"
```

Nur bestimmte Ereignistypen

```
[RO][SENS] docker events \
    --since 1h \
    --filter container="<CONTAINER>" \
    --filter event=die
```

Neustartrichtlinie anzeigen

```
[RO] docker inspect "<CONTAINER>" \
    --format '{{json .HostConfig.RestartPolicy}}'
```

Typische Neustartrichtlinien

Richtlinie	Verhalten
`no`	Kein automatischer Neustart
`always`	Container wird grundsätzlich neu gestartet
`unless-stopped`	Neustart, außer er wurde ausdrücklich beendet
`on-failure`	Neustart bei fehlerhaftem Exit-Code

Eine Neustartrichtlinie kann die eigentliche Störung verdecken. Der Container erscheint wiederholt kurz als laufend, obwohl der Hauptprozess ständig abstürzt.

Prüfreihenfolge bei Neustartschleifen

1. Neustartzähler erfassen.
2. Exit-Code und OOM-Status prüfen.
3. Protokolle mit Zeitstempeln lesen.
4. Startbefehl und Umgebungsvariablen kontrollieren.
5. Mounts und Berechtigungen prüfen.
6. Abhängige Dienste untersuchen.
7. Hostereignisse und Speicherzustand prüfen.
8. Erst danach über Änderungen an der Neustartrichtlinie entscheiden.

26. Docker Desktop unter Windows und macOS eingrenzen

Docker Desktop führt Linux-Container normalerweise innerhalb einer verwalteten Virtualisierungsumgebung aus. Dadurch existieren zusätzliche Schichten:

```
Docker-Client
    |
Docker Desktop
    |
Verwaltete Linux-VM beziehungsweise Virtualisierungsschicht
    |
Docker Engine
    |
Container
```

Plattformübergreifende Prüfungen

```
[RO] docker version
```

```
[RO][SENS] docker info
```

```
[RO][SENS] docker context ls
```

```
[RO][SENS] docker ps -a
```

```
[TEST][SENS] docker system df
```

Zusätzlich unter Windows prüfen

- Läuft Docker Desktop?
- Wird der erwartete Docker-Kontext verwendet?
- Funktioniert WSL 2, falls dieses Backend verwendet wird?
- Besitzt die virtuelle Docker-Festplatte freien Speicher?
- Sind Unternehmensrichtlinien oder Sicherheitssoftware beteiligt?

WSL-Status anzeigen

```
[RO] wsl --status
```

WSL-Distributionen anzeigen

```
[RO][SENS] wsl --list --verbose
```

Zusätzlich unter macOS prüfen

- Läuft Docker Desktop?
- Ist genügend freier Speicher auf dem Mac vorhanden?
- Wird der erwartete Kontext verwendet?
- Passt das Image zu `arm64` oder `amd64`?
- Sind Bind-Mount-Pfade verfügbar?
- Blockiert eine Sicherheits- oder Netzwerksoftware die Verbindung?
- Ist die virtuelle Docker-Disk ausgelastet?

> Das Beenden oder Zurücksetzen von Docker Desktop kann alle laufenden Container unterbrechen. Ein Factory Reset kann lokale Images, Container, Netzwerke und andere Daten entfernen und darf nicht als allgemeine Diagnosemaßnahme verwendet werden.

27. Podman-Container unter Linux untersuchen

Podman besitzt viele Docker-ähnliche Befehle, verwendet aber kein identisches Betriebsmodell. Rootless- und Rootful-Container besitzen getrennte Ansichten.

Container anzeigen

```
[RO][SENS] podman ps -a
```

Podman-Systeminformationen

```
[RO][SENS] podman info
```

Containerdetails

```
[RO][SENS] podman inspect "<CONTAINER>"
```

Protokolle

```
[RO][SENS] podman logs --tail 100 "<CONTAINER>"
```

Ressourcennutzung

```
[TEST][SENS] podman stats --no-stream
```

Netzwerke

```
[RO][SENS] podman network ls
```

Volumes

```
[RO][SENS] podman volume ls
```

Speicherverbrauch

```
[TEST][SENS] podman system df
```

Wichtige Besonderheit

Ein Container, der als normaler Benutzer erstellt wurde, erscheint normalerweise nicht automatisch in der Root-Ansicht:

```
[RO][SENS] podman ps -a
```

und:

```
[RO][PRIV][SENS] sudo podman ps -a
```

können deshalb unterschiedliche Container anzeigen.

28. Snapshots, Checkpoints und Sicherungen richtig bewerten

Snapshots und Checkpoints speichern abhängig von der Plattform:

- Datenträgeränderungen

- VM-Konfiguration
- gegebenenfalls Arbeitsspeicherzustand
- Abhängigkeiten zu Parent-Dateien

Sie sind für kurzfristige Rückfallpunkte geeignet, ersetzen aber keine unabhängige Sicherung.

Risiken lang bestehender Snapshots

- Stark wachsender Speicherverbrauch
- Zusätzliche I/O-Latenz
- Lange Zusammenführung
- Komplexe Snapshotketten
- Ausfall bei fehlender Parent-Datei
- Datastore oder Hostvolume läuft voll
- Anwendungskonsistenz ist nicht garantiert
- Sicherungssoftware wird beeinträchtigt

Vor einer Snapshotaktion prüfen

- Wer hat den Snapshot erstellt?
- Warum wurde er erstellt?
- Wie alt ist er?
- Wie groß ist er?
- Wird er von einer Sicherung verwendet?
- Ist eine Zusammenführung aktiv?
- Ist genügend Speicher für die Zusammenführung vorhanden?
- Ist die Anwendung im Snapshot konsistent?
- Existiert eine unabhängige Sicherung?

> Snapshotdateien niemals manuell löschen. Erstellung, Zusammenführung und Entfernung müssen über das vorgesehene Managementwerkzeug erfolgen.

29. Zeitfehler in virtuellen Systemen untersuchen

Eine VM kann Zeit aus mehreren Quellen erhalten:

- Hypervisor
- virtuelle Hardwareuhr
- Gastbetriebssystem-Zeitdienst
- Active Directory
- externer NTP-Server

Windows-Gast

```
[RO] w32tm /query /source
```

```
[RO] w32tm /query /status
```

Linux-Gast

```
[RO] timedatectl status
```

```
[RO] chronyc tracking
```

falls chrony verwendet wird.

Typische Ursachen

- Hypervisor und NTP korrigieren gleichzeitig.
- Hostzeit ist falsch.
- VM wurde aus einem alten Snapshot gestartet.
- VM war lange pausiert.
- Migration verursachte einen Zeitsprung.
- Virtueller Domänencontroller verwendet eine falsche Quelle.
- Gasterweiterungen setzen die Zeit zurück.

Bei Authentifizierungs-, Zertifikats- und Protokollproblemen sollte die Zeit frühzeitig geprüft werden.

30. Kritische Änderungen und gefährliche Schnelllösungen

Folgende Maßnahmen dürfen nicht unkontrolliert durchgeführt werden:

Maßnahme	Risiko
VM hart ausschalten	Datenverlust und Dateisystemfehler
Container mit `kill` beenden	Anwendung erhält keine reguläre Beendigungszeit
Docker Factory Reset	Verlust lokaler Docker-Daten
`docker system prune`	Entfernt nicht verwendete Docker-Objekte
`docker volume prune`	Kann persistente Daten entfernen

Maßnahme	Risiko
Snapshotdateien manuell löschen	Virtuelle Datenträgerkette kann unbrauchbar werden
VHDX oder QCOW2 während des Betriebs kopieren	Kopie kann inkonsistent sein
VM auf Snapshot zurücksetzen	Neuere Daten und Zustände gehen verloren
Hypervisordienst neu starten	Mehrere VMs können betroffen sein
Netzwerkbrücke oder Switch neu erstellen	Alle verbundenen Gäste können getrennt werden
Storage aushängen	VMs und Container verlieren Datenträgerzugriff
Rechte auf Docker-Socket weit öffnen	Praktisch administrative Kontrolle über den Host
Secrets unredigiert exportieren	Zugangsdaten werden offengelegt

Vor einer Änderung erforderlich

- Exakter Zielhost
- Exakte VM beziehungsweise exakter Container
- Aktueller Zustand
- Abhängige Dienste
- Sicherung
- Wartungsfenster
- Rückfallplan
- Verantwortliche Freigabe
- Funktionsprüfung nach der Änderung

31. Sichere Reihenfolge der Fehleranalyse

1. Betroffene Anwendung, VM oder Container eindeutig bestimmen.
2. Umfang der Störung feststellen.
3. Host, Hypervisor und Managementzugang prüfen.
4. Hostressourcen und Speicherplatz kontrollieren.
5. VM- beziehungsweise Containerstatus und Startzeit erfassen.
6. Exit-Code, Healthcheck und Ereignisse prüfen.
7. Protokolle zum Fehlerzeitpunkt auswerten.
8. Virtuelles Netzwerk und Portzuordnung untersuchen.
9. Virtuelle Datenträger, Volumes und Mounts kontrollieren.
10. Abhängige Dienste und Namensauflösung prüfen.
11. Ressourcenlimits und OOM-Ereignisse untersuchen.
12. Image-, Versions- und Architekturkompatibilität prüfen.
13. Snapshot-, Migrations- und Änderungsverlauf berücksichtigen.
14. Sicherung und Rückfallplan verifizieren.
15. Erst danach eine Änderung oder einen kontrollierten Neustart durchführen.
16. Anwendung aus interner und externer Sicht testen.

17. Host, VM, Container und Monitoring erneut kontrollieren.
18. Ursache und Maßnahme dokumentieren.

32. Schnelle Befehlsübersicht

Aufgabe	Hyper-V	KVM/libvirt	Docker auf Windows, Linux und macOS
Hostinformationen	`[RO][PRIV] Get-VMHost`	Hostbefehle wie `free`, `df`, `uptime`	`[RO][SENS] docker info`
Alle Gäste	`[RO][PRIV] Get-VM`	`[RO][PRIV] sudo virsh list --all`	`[RO][SENS] docker ps -a`
Gaststatus	`[RO][PRIV] Get-VM -Name "<VM>"`	`[RO][PRIV] sudo virsh domstate "<VM>"`	`[RO] docker inspect "<CONTAINER>" --format '{{json .State}}'`
Ressourcen	VM-, Host- und Performance-Counter	`[TEST][PRIV] sudo virsh domstats "<VM>"`	`[TEST] docker stats --no-stream`
Netzwerke	`[RO][PRIV] Get-VMSwitch`	`[RO][PRIV] sudo virsh net-list --all`	`[RO] docker network ls`
Gastnetzadapter	`[RO][PRIV] Get-VMNetworkAdapter -VMName "<VM>"`	`[RO][PRIV] sudo virsh domiflist "<VM>"`	`[RO] docker inspect "<CONTAINER>" --format '{{json .NetworkSettings.Networks}}'`
Datenträger	`[RO][PRIV] Get-VMHardDiskDrive -VMName "<VM>"`	`[RO][PRIV] sudo virsh domblklist "<VM>" --details`	`[RO] docker inspect "<CONTAINER>" --format '{{json .Mounts}}'`
Snapshots	`[RO][PRIV] Get-VMSnapshot -VMName "<VM>"`	`[RO][PRIV] sudo virsh snapshot-list "<VM>"`	Nicht direkt vergleichbar
Protokolle	Hyper-V-Ereignisprotokolle	Journal und VM-Protokolle	`[RO][SENS] docker logs --tail 100 "<CONTAINER>"`
Ereignisse	Windows-Ereignisanzeige	Journal	`[RO][SENS] docker events --since 1h`
Ports	Gast und virtuellen Switch prüfen	Gast und Bridge prüfen	`[RO] docker port "<CONTAINER>"`
Volumes	VHD/VHDX und Hostvolume	Storage-Pools und Volumes	`[RO] docker volume ls`
Speicherverbrauch	Hostvolume und VHDX-Größe	`df`, Pools und Images	`[TEST] docker system df -v`
Architektur	Host- und Gastarchitektur	`[RO] uname -m`	`docker image inspect ... --format '{{.Architecture}}'`
Healthcheck	Gast- beziehungsweise anwendungsspezifisch	Gast- beziehungsweise anwendungsspezifisch	`docker inspect ... --format '{{json .State.Health}}'`

33. Entscheidungsmatrix

Befund	Wahrscheinliche Ursache	Nächster Schritt
VM läuft, Konsole funktioniert, Netzwerk nicht	Gastnetzwerk, Switch oder VLAN	Virtuelle und interne Netzwerkkonfiguration prüfen
VM startet nicht, Host-RAM knapp	Ressourcenproblem	Host- und VM-Speicherzuweisung prüfen
Mehrere VMs frieren ein	Host oder gemeinsames Storage	Hostlast und Storage-Latenz prüfen
Snapshot wächst stark	Viele Schreibänderungen oder alter Snapshot	Zweck, Größe und Zusammenführung planen
Container endet mit Code 127	Startbefehl nicht gefunden	Entrypoint, Cmd und Image prüfen
Container endet mit Code 137	OOM oder erzwungenes Beenden möglich	OOM-Status und Hostprotokolle prüfen
Container läuft, Anwendung nicht erreichbar	Listener, Port oder Healthcheck	Containerlistener und Portzuordnung prüfen
Port kann nicht gebunden werden	Hostport bereits belegt	Hostlistener und Compose-Konfiguration prüfen
Container findet Datenbank nicht	DNS, Netzwerk oder Hostname	Gemeinsames Netzwerk und Dienstnamen prüfen
Daten nach Neuerstellung weg	Fehlendes oder falsches Volume	Mountkonfiguration und vorhandene Volumes prüfen
Image startet auf ARM nicht	Plattform nicht unterstützt	Imagearchitektur und Manifest prüfen
Docker-Client findet Daemon nicht	Dienst, Socket oder Kontext	`docker version`, Dienst und Kontext prüfen
Nur Root sieht Podman-Container	Rootful- und Rootless-Ansicht	Richtigen Benutzerkontext verwenden
Host hat Platz, Docker meldet voll	Docker-Disk, Inodes oder Storage-Layer	`docker system df`, `df` und Inodes prüfen
Nach Snapshot stimmen Anmeldungen nicht	Zeitabweichung	Gastzeit und Zeitquelle prüfen

34. Dokumentationsvorlage

Ticket:

Datum und Uhrzeit:

Bearbeiter:

Betroffene Plattform:

Hypervisor oder Container-Engine:

Version:

Host:

Hostbetriebssystem:

Physischer oder virtueller Host:

Cluster:

Managementsystem:

Betroffene VM oder Container:

VM- beziehungsweise Container-ID:

Image und Tag:

Gastbetriebssystem:

Architektur:

Aktueller Zustand:

Startzeit:

Uptime:

Exit-Code:

Healthcheck:

Neustartzähler:

Umfang der Störung:

Originale Fehlermeldung:

Fehlerzeitpunkt:

Letzte funktionierende Nutzung:

Letzte Änderung:

Update, Snapshot oder Migration:

Host-CPU:

Host-RAM:

Hostspeicher:

Inode-Belegung:

Storage-Zustand:

Virtueller Datenträger:

Snapshots oder Checkpoints:

Virtueller Switch oder Netzwerk:

VLAN:

IP-Adresse:

```
DNS:
Portveröffentlichung:
Hostlistener:
Containerlistener:
Firewall:

Volumes und Mounts:
Berechtigungen:
Umgebungsvariablen geprüft:
Sensible Werte redigiert:
Abhängige Dienste:
Zeitquelle:

Relevante Hostprotokolle:
Hypervisorprotokolle:
Gastprotokolle:
Containerprotokolle:
OOM-Ereignisse:

Vermutete Ursache:
Durchgeführte Tests:
Sicherung geprüft:
Wartungsfenster:
Durchgeführte Änderung:
Rückfallplan:

Status nach der Maßnahme:
Interne Funktionsprüfung:
Externe Funktionsprüfung:
Monitoring:
Weiterführende Maßnahmen:
```

Merksatz

> Bei Virtualisierungs- und Containerfehlern wird immer von außen nach innen geprüft: Host, Virtualisierungsschicht, Netzwerk und Storage, Gast beziehungsweise Container und erst danach die Anwendung. Ein Status `Running`

> oder `Up` ist noch kein Funktionsnachweis.

Quellen und weiterführende Dokumentation

- Microsoft Learn – Hyper-V mit PowerShell verwalten
- Microsoft Learn – Hyper-V-PowerShell-Referenz
- Microsoft Learn – Hyper-V-Ereignisprotokolle
- libvirt – virsh-Dokumentation
- libvirt – Architektur
- QEMU – System Emulation
- Docker Docs – Diagnose des Docker-Daemons
- Docker Docs – docker inspect
- Docker Docs – docker logs
- Docker Docs – docker stats
- Docker Docs – docker events
- Docker Docs – Docker-Netzwerke
- Docker Docs – Docker-Volumes
- Docker Docs – Compose-Dateireferenz
- Docker Docs – Multi-Platform-Images
- Docker Docs – Docker Desktop Troubleshoot
- Podman-Dokumentation
