Online Monitoring - Zabbix (Version 7 und 8)

Überwachen. Analysieren. Optimieren.

Zabbix ist die leistungsstarke, **skalierbare** und **flexible** Open-Source-Plattform für **umfassendes IT-Monitoring**. Ob Server, Netzwerke, Cloud-Umgebungen, Container oder Applikationen – **Zabbix** liefert Ihnen in Echtzeit wertvolle Einblicke in Ihre IT-Infrastruktur, erkennt Probleme, bevor sie auftreten, und hilft Ihnen, die **Performance und Stabilität** Ihrer Systeme zu optimieren.

Für meine Kunden stelle ich hier Tipps und Tricks, Anleitungen und Empfehlungen und News bereit.

Ansonsten: https://www.zabbix.com/

- Allgemein
  - Warum Zabbix?
- Einrichtung
  - Einfache Überwachung eines Webservers
  - Web-Überwachung mit Web Scenarios in Zabbix 7.0+
- Überwachung
  - Betriebskonzept Schwachstellen-Überwachung (Trivy)

Allgemein

Allgemein

Warum Zabbix?

Zentrale Überwachung für alle Systeme
Mit einer einzigen Lösung behalten Sie **Netzwerke, Server, virtuelle Maschinen, Cloud-Dienste, Anwendungen, IoT-Geräte und mehr** im Blick.

100% Open Source – Keine Lizenzkosten
Zabbix ist **komplett kostenlos**, ohne Einschränkungen oder versteckte Gebühren. Nutzen Sie eine **Enterprise-fähige Lösung**, ohne in teure Software investieren zu müssen.

Echtzeit-Alarmierung & Automatisierung
Erkennen und beheben Sie Probleme **proaktiv** mit flexiblen **Benachrichtigungen über E-Mail, SMS, Telegram, Slack und mehr**. Automatisieren Sie **Maßnahmen**, um Ausfälle zu verhindern.

Skalierbar für Unternehmen jeder Größe
Ob kleines Unternehmen oder globaler Konzern – Zabbix passt sich **dynamisch** an jede Infrastruktur an und kann **Millionen von Datenpunkten** verarbeiten.

Leistungsstarke Dashboards & Visualisierungen
Erstellen Sie **benutzerdefinierte Dashboards, Grafiken und Berichte**, um Ihre IT-Systeme übersichtlich zu analysieren.

Integrierbar mit führenden Technologien
Nahtlose Integration mit **Prometheus, Grafana, Kubernetes, Docker, AWS, Azure, VMware, SNMP, APIs und vielen weiteren Tools**.

Sicher & zuverlässig
Zabbix setzt auf **modernste Sicherheitsstandards**, Verschlüsselung und **stabile Performance**, um eine **sichere Überwachung** Ihrer IT-Umgebung zu gewährleisten.

Einrichtung

Einrichtung

Einfache Überwachung eines Webservers

Webserver-Überwachung (Alive & Ladezeit)

1. Host in Zabbix erstellen (falls noch nicht vorhanden)

Falls dein Webserver noch nicht als Host in Zabbix existiert, musst du ihn zuerst hinzufügen.

- Gehe zu → Data collection → Hosts
- Klicke auf Create host
- Gib die Host-Daten ein:
- Host name: Mein-Webserver
- Visible name: (optional, z. B. Webserver XYZ)
- Groups: Web-Server (oder eine passende Gruppe)
- Interfaces: - Add
  - Type: Agent
  - IP address: 123.123.123.123 (IP oder Domain deines Webservers)
  - Port: 10050
- Speichern (Add klicken)

2. Web-Szenario für Alive-Check & Ladezeit einrichten

Mit einem Web-Szenario kannst du prüfen, ob die Webseite erreichbar ist und wie lange sie zum Laden benötigt.

- Gehe zu → Data collection → Hosts
- Wähle deinen Webserver (Mein-Webserver)
- Wechsle zum Tab/Spalte Web
- Klicke auf Create web scenario
- Fülle die Basisinformationen aus:
- Name: Webserver-Check
- Update interval: 1m (Überprüfung jede Minute)
- Retries/Attemps: 1
- Agent: Zabbix
- **Web-Step hinzufügen (Alive-Check & Ladezeit)**
  - Reiter Steps add
- Klicke auf Add, um einen neuen Step hinzuzufügen.
  - Fülle die Felder aus:
  - Name: Homepage Check
  - URL: http://mein-webserver.de
  - Follow redirects: (Ja)
  - Timeout: 10s
  - Required string: (Optional, falls du nach einem bestimmten Text auf der Seite suchst)
- Speichern (Add klicken)

 Zabbix testet jetzt jede Minute, ob die Seite online ist und speichert die Ladezeit.

3. Trigger für Offline-Status (Alive-Check) erstellen

Falls der Webserver nicht mehr erreichbar ist, soll eine Alarmierung erfolgen.

- Gehe zu → Data collection → Hosts - Spalte Triggers
- Klicke auf Create trigger
- Fülle die Felder aus:
- Name: Webserver ist nicht erreichbar
- Expression: {Mein-Webserver:web.test.fail[Webserver-Check].last()}=1

- Severity: High
- Speichern (Add klicken)

□□ Jetzt löst der Trigger aus, wenn die Webseite nicht erreichbar ist.

4. Trigger für hohe Ladezeit erstellen

Falls die Ladezeit zu hoch ist, soll eine Warnung ausgelöst werden.

- Gehe zu → Data collection → Triggers
- Klicke auf Create trigger
- Fülle die Felder aus:
- Name: Webserver-Ladezeit zu hoch
- Expression: {Mein-Webserver:web.test.time[Webserver-Check, Homepage Check].last()}>3
- Severity: Warning
- Speichern (Add klicken)

□□ Jetzt bekommst du eine Warnung, wenn der Seitenaufbau länger als 3 Sekunden dauert.

5. (Optional) Benachrichtigung einrichten

Falls du eine E-Mail, Telegram- oder Slack-Benachrichtigung bekommen möchtest:

- Gehe zu → Alerts → Actions / Trigger actions
- Klicke auf Create action
- Fülle die Aktions-Details aus:
- Name: Webserver Down Alert
- Conditions:
- Trigger = Webserver ist nicht erreichbar
- Trigger = Webserver-Ladezeit zu hoch
- Operation - Operations ADD
- Sende Nachricht an Admins (z. B. per E-Mail oder Telegram)
- Speichern (Add klicken)

□□ Jetzt wirst du benachrichtigt, wenn der Webserver offline ist oder langsam lädt.

> □ Fazit

> ✔ Zabbix prüft automatisch alle 60 Sekunden, ob der Webserver erreichbar ist.

> ✔ Die Ladezeit wird gemessen und kann als Performance-Indikator dienen.

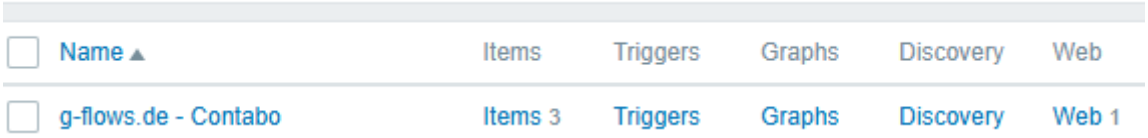
> ✔ Falls der Webserver offline ist oder zu langsam lädt, gibt es eine Alarmierung.

Einrichtung

Web-Überwachung mit Web Scenarios in Zabbix 7.0+

Anstatt HTTP-Agent oder externe Skripte zu nutzen, richten wir ein Web Scenario ein, das:

- ✔ Den HTTP-Statuscode überwacht (200 OK oder Fehlercodes 403, 500 usw.)
- ✔ Die Ladezeit (Response time) misst
- ✔ Die Download-Geschwindigkeit (Download speed) speichert
- ✔ Automatisch ausgeführt wird, ohne manuelle Anpassungen

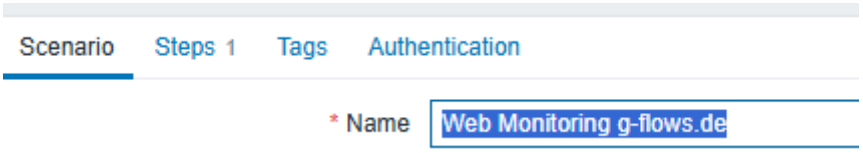


1️⃣ Web Scenario für "https://g-flows.de" erstellen

- Gehe zu Data collection → Hosts
- Wähle host (z.B. g-flows.de)
- Gehe zu Spalte Web → Create Web Scenario
  - Fülle folgende Werte aus:
  - Name: Web Monitoring g-flows.de
  - Update interval: 30s (Wie oft das Szenario laufen soll)
  - Retries/Attempts: 3 (Falls die Website nicht erreichbar ist, wird es erneut versucht)
  - Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) (Damit sich Zabbix wie ein echter Browser verhält) oder Chrome in neuester Variante
  - Variables: (Leer lassen)
  - Headers: (Leer lassen)
  - Required status codes: 200 (Damit Zabbix erwartet, dass die Seite mit HTTP 200 OK antwortet)
- ◫ Speichern!

2️⃣ Schritte für das Web-Szenario (Web Steps) hinzufügen

Öffne das gerade erstellte Web Monitoring g-flows.de



- Gehe zu Steps → Add
- Fülle folgende Werte aus:
- Name: Check Homepage
- URL: https://g-flows.de
- Timeout: 10s
- Required string: (Leer lassen oder Welcome falls ein spezifischer Text auf der Seite vorhanden ist)
- Required status codes: 200
- Retrieve mode: Body (Falls du nur den Body checken willst)
- ◫ Speichern!

3️⃣ Automatisch erfasste Web Monitoring Items

Sobald das Web Scenario gespeichert ist, erstellt Zabbix automatisch folgende Überwachungswerte:

□□ Allgemeine Werte für das gesamte Web-Szenario

- web.test.fail[g-flows.de,Web Monitoring g-flows.de] → Gibt die Nummer des fehlgeschlagenen Schritts zurück (falls Fehler)
- web.test.error[g-flows.de,Web Monitoring g-flows.de] → Letzte Fehlermeldung

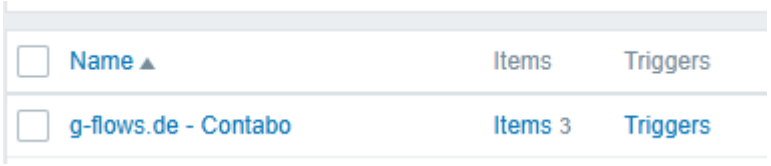
□□ Werte für den Schritt "Check Homepage"

- web.test.time[g-flows.de,Web Monitoring g-flows.de,Check Homepage] → Ladezeit der Seite (Response time)
- web.test.rspcode[g-flows.de,Web Monitoring g-flows.de,Check Homepage] → HTTP-Statuscode (z. B. 200, 404, 500)
- web.test.in[g-flows.de,Web Monitoring g-flows.de,Check Homepage] → Eingehende Download-Geschwindigkeit

□□ Diese Werte sind jetzt automatisch verfügbar!

4□ Trigger für Alarme einrichten

4.1 Alarm: Webserver antwortet nicht (200 OK fehlt)



Gehe zu Triggers → Create trigger

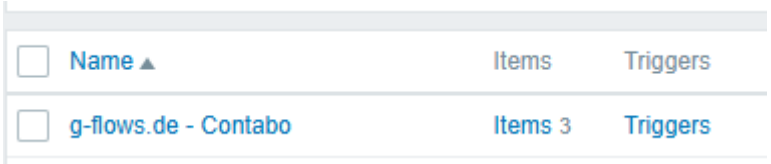
Fülle folgende Werte aus:

- Name: Webserver g-flows.de nicht erreichbar
- Severity: High
- Expression:blast(/g-flows.de/web.test.rspcode[g-flows.de,Web Monitoring g-flows.de,Check Homepage])<>200

□□ Speichern!

4.2 Alarm: Webserver lädt zu langsam (> 2s)

Gehe zu Triggers → Create trigger



Fülle folgende Werte aus:

Name: Webserver g-flows.de lädt zu langsam

Severity: Warning

Expression: last(/g-flows.de/web.test.time[g-flows.de,Web Monitoring g-flows.de,Check Homepage])>2

□□ Speichern!

Fazit – Warum ist Web Monitoring die beste Lösung?

- ✔ Web Scenarios sind speziell für Webseiten-Monitoring optimiert
- ✔ Kein kompliziertes Preprocessing oder externe Skripte nötig!
- ✔ Erfasst HTTP-Status, Ladezeit und Download-Speed automatisch
- ✔ Funktioniert auch mit Weiterleitungen und verschlüsselten HTTPS-Seiten
- ✔ Trigger für langsame Ladezeiten oder Fehler sind einfach konfigurierbar
- ✔ Mehrere Webseiten-Checks in einem einzigen Web Scenario möglich

➡ Fazit: Diese Methode ist die beste & einfachste Lösung für dein Zabbix Web Monitoring!

Überwachung

Überwachung

Betriebskonzept Schwachstellen-Überwachung (Trivy)

Kunde	[Kunde]
Stand	[Datum] · Trivy 0.76.1 · Skript security-scan.sh v6
Verantwortlich	[Name, Rolle]
Geltungsbereich	[Liste der Server oder Verweis auf Inventar]

1. Zweck

Jeder Server wird jede Nacht auf bekannte Sicherheitslücken in seinen Anwendungen geprüft. Gemeldet wird nur, wo gehandelt werden kann: wenn der Hersteller eine korrigierte Fassung bereitstellt, oder wenn eine Lücke so lange offen ist, dass über die Anwendung selbst entschieden werden muss.

2. Was geprüft wird

Bereich	Geprüft durch
Laufende Container (Docker)	Trivy, inklusive Prüfung beim Hersteller
Anwendungen in `/opt` und `/var/www`	Trivy (installierte Bibliotheken, z. B. PHP, Node, Java)
Pakete des Betriebssystems	Automatische Sicherheitsupdates und Wazuh
Nicht geprüft	Verzeichnisse mit „backup“ im Namen, Nextcloud-Benutzerdaten (werden automatisch erkannt)

3. Ablauf

1. Jede Nacht um 2:30 Uhr lädt Trivy die aktuelle Schwachstellen-Datenbank.
2. Container und Verzeichnisse werden ohne Netzverbindung und nur lesend geprüft.
3. Bei Containern mit kritischen Lücken prüft Trivy, ob das aktuelle Image des Herstellers unter derselben Bezeichnung die Lücken behebt.
4. Die Zahlen gehen an Zabbix (Alarme), die Einzelheiten an Wazuh (jede Lücke einzeln).

4. Alarmregel und Reaktion

Meldung	Bedeutung	Reaktion
HIGH	Der Hersteller hat ein neueres Image, das kritische Lücken schließt.	Image neu laden, nur diesen Dienst neu starten, Funktion prüfen. Frist: [Frist]
WARNING	Eine kritische Lücke ist seit 30 Tagen offen, ohne Fix vom Hersteller.	Entscheiden: Risiko tragen, Zugang einschränken oder Anwendung ersetzen. Entscheidung dokumentieren.

Information	Alle übrigen Funde, z. B. Lücken ohne Fix vom Hersteller.	Keine. Sichtbar in Wazuh und in der Liste auf dem Server.
WARNING	Scan mit Fehlern, oder seit 26 Stunden kein Scan.	Ursache im Journal des Servers prüfen (`journalctl -u trivy-manual` bzw. Cron-Lauf).

Infrastruktur-Container (z. B. Datenbanken, Reverse Proxy), die nachts automatisch aktualisiert werden, lösen keine Alarme aus.

5. Absicherung des Scanners selbst

- Trivy läuft in einer festen Version: als Container per Digest, ohne Docker per Programm mit kontrollierter Prüfsumme. Ein Update von Trivy gibt es nur über eine Freigabe.
- Prüfungen laufen ohne Netz und nur lesend, der Scanner erhält keinen Zugriff auf Docker selbst.
- Keine Nutzungsdaten an den Hersteller (Telemetrie abgeschaltet), kein Durchsuchen nach Passwörtern oder Schlüsseln.

6. Änderungen und Freigabe

1. Verteilt wird ausschließlich per Ansible (`playbooks/setup_security_monitoring.yml`), nie von Hand auf einzelnen Servern.
2. Jede Änderung wird zuerst in einem Testzweig auf zwei Servern geprüft.
3. Erst nach bestandener Prüfung kommt sie in den freigegebenen Stand und wird gruppenweise verteilt.
4. Das Playbook bricht ab, wenn Voraussetzungen fehlen, und läuft nur mit gezielter Server-Auswahl.

7. Voraussetzungen je Server

- Wazuh-Agent installiert (auf dem Wazuh-Server: Manager)
- Docker oder Internetzugang zu GitHub für die Installation des Programms
- Mindestens 3 GB frei für die Datenbanken von Trivy
- Template „Template Security Trivy“ in Zabbix verknüpft

8. Grenzen

- Für Anwendungen ohne Docker, für selbst gebaute Images und für Images mit fest eingetragenem Digest kann Trivy nicht prüfen, ob der Hersteller einen Fix hat. Dort gibt es keinen HIGH-Alarm, nur die Warnung nach 30 Tagen.
- Anwendungen außerhalb von `/opt` und `/var/www` werden nicht geprüft. [Hier abweichende Installationsorte des Kunden eintragen]
- Öffentliche Registries begrenzen Abrufe ohne Anmeldung. Ist die Grenze erreicht, meldet der Scan einen Fehler, die Herstellerprüfung fehlt dann für diese Nacht.

9. Kurzreferenz auf dem Server

Datei	Inhalt
`/var/log/trivy-actionable-critical.list`	Lücken mit Fix vom Hersteller (HIGH)
`/var/log/trivy-stale-critical.list`	Lücken seit 30 Tagen ohne Fix (WARNING)
`/var/log/trivy-vendor-check.list`	Stand der Herstellerprüfung je Container
`/var/log/trivy-scans.json`	Rohdaten für Wazuh
`/usr/local/bin/security-scan.sh`	Scan-Skript (nur über Ansible ändern)

Scan von Hand starten, unabhängig von der SSH-Verbindung: `systemd-run --unit=trivy-manual --collect /usr/local/bin/security-scan.sh`