monfetch

Server-Monitoring im r/unixporn-Style

emily

August 2021

Probleme

- Server-Monitoring meist sehr komplex & aufwändig

Probleme

- Server-Monitoring meist sehr komplex & aufwändig
- Erfordert viele Ressourcen:
  - Datenbankcluster
  - Data Sources
  - Datenaufbereiter
  - etc.

Probleme

- Server-Monitoring meist sehr komplex & aufwändig
- Erfordert viele Ressourcen:
  - Datenbankcluster
  - Data Sources
  - Datenaufbereiter
  - etc.
- Meist Cloud-gebunden

Probleme

- Server-Monitoring meist sehr komplex & aufwändig
- Erfordert viele Ressourcen:
  - Datenbankcluster
  - Data Sources
  - Datenaufbereiter
  - etc.
- Meist Cloud-gebunden
- Datensammler meist sehr schwergewichtig

Probleme

- Server-Monitoring meist sehr komplex & aufwändig
- Erfordert viele Ressourcen:
  - Datenbankcluster
  - Data Sources
  - Datenaufbereiter
  - etc.
- Meist Cloud-gebunden
- Datensammler meist sehr schwergewichtig
- Für Enterprise gut, für jedoch Homelab massiver Overkill

Was unterscheidet monfetch davon?

- Sehr simple Architektur
  - Eigener Server/Client: Wochenendsprojekt

Was unterscheidet monfetch davon?

- Sehr simple Architektur
  - Eigener Server/Client: Wochenendsprojekt
- Keine Datenbank benötigt

Was unterscheidet monfetch davon?

- Sehr simple Architektur
  - Eigener Server/Client: Wochenendsprojekt
- Keine Datenbank benötigt
- Keine Cloud-Anbindung

Was unterscheidet monfetch davon?

- Sehr simple Architektur
  - Eigener Server/Client: Wochenendsprojekt
- Keine Datenbank benötigt
- Keine Cloud-Anbindung
- Datensammler (Agent) sehr leichtgewichtig

Was unterscheidet monfetch davon?

- Sehr simple Architektur
  - Eigener Server/Client: Wochenendsprojekt
- Keine Datenbank benötigt
- Keine Cloud-Anbindung
- Datensammler (Agent) sehr leichtgewichtig
- Bonus: Server auch sehr leichtgewichtig

Weitere Vorteile

- Generierte Seite sehr klein
  - HTML: 25 Zeilen
  - CSS: 89 Zeilen (nicht minified)
  - kein Javascript

Weitere Vorteile

- Generierte Seite sehr klein
  - HTML: 25 Zeilen
  - CSS: 89 Zeilen (nicht minified)
  - kein Javascript
- Sehr übersichtlich
  - Daten im fetch-Stil, daher sehr leicht lesbar für *NIX-Nutzende

Weitere Vorteile

- Generierte Seite sehr klein
  - HTML: 25 Zeilen
  - CSS: 89 Zeilen (nicht minified)
  - kein Javascript
- Sehr übersichtlich
  - Daten im fetch-Stil, daher sehr leicht lesbar für *NIX-Nutzende
- Freie Software: AGPL-3.0-or-later

Überblick Architektur

- Server in Python3 (Flask), Agent in Shell

Überblick Architektur

- Server in Python3 (Flask), Agent in Shell
- Agent sendet Daten mit cURL per POST

Überblick Architektur

- Server in Python3 (Flask), Agent in Shell
- Agent sendet Daten mit cURL per POST
- Daten größtenteils aus procfs/sysfs

Überblick Architektur

- Server in Python3 (Flask), Agent in Shell
- Agent sendet Daten mit cURL per POST
- Daten größtenteils aus procfs/sysfs
- Authentifizierung per Shared Secret in Request Path

Erweiterungen Architektur

- WSGI-Server: uWSGI
  - Handlet Anfragen
  - Leichtgewichtig

Erweiterungen Architektur

- WSGI-Server: uWSGI
  - Handlet Anfragen
  - Leichtgewichtig
- systemd
  - Supervision
  - Nicht sonderlich leichtgewichtig, aber einfach ersetzbar

Erweiterungen Architektur

- WSGI-Server: uWSGI
  - Handlet Anfragen
  - Leichtgewichtig
- systemd
  - Supervision
  - Nicht sonderlich leichtgewichtig, aber einfach ersetzbar
- git
  - Versionsverwaltung
  - Macht Aktualisierung einfach
  - Recht leichtgewichtig

Aufbau Demonstration

- Echtes, laufendes Beispiel

Aufbau Demonstration

- Echtes, laufendes Beispiel
- Installation Server mit uWSGI

Aufbau Demonstration

- Echtes, laufendes Beispiel
- Installation Server mit uWSGI
- Installation Client

Aufbau Demonstration

- Echtes, laufendes Beispiel
- Installation Server mit uWSGI
- Installation Client
- Aufsetzen von systemd-Services

Schritte nach der Demonstration

- Den uWSGI-Server reverse-proxien

Schritte nach der Demonstration

- Den uWSGI-Server reverse-proxien
- HTTPS einrichten (Schutz gegen MITM-Attacken)

Schritte nach der Demonstration

- Den uWSGI-Server reverse-proxien
- HTTPS einrichten (Schutz gegen MITM-Attacken)
- Alles außer / blocken (Schutz gegen Passwort-Brute-Force)

Schritte nach der Demonstration

- Den uWSGI-Server reverse-proxien
- HTTPS einrichten (Schutz gegen MITM-Attacken)
- Alles außer / blocken (Schutz gegen Passwort-Brute-Force)
- Aktuell halten per `git stash && git pull && git stash pop`

Ende

Gibt es noch Fragen?