Container - wo anfangen?!

Unterschiedliche Ansätze für den Betrieb von Software am Rande der Wolke, entsprechende Anwendungsfälle, Entwicklungsressourcen und Skalierbarkeit

INSYS icom, Michael Kress

Fertige Produkte

Container wie **Apps auf dem Smartphone**. Partner, Open-Source-Community oder INSYS haben voll funktionsfähige Container für einen speziellen Zweck erstellt, die die Benutzer einfach herunterladen, konfigurieren und verwenden können.

Beispiel: icom Data Suite (iDS)

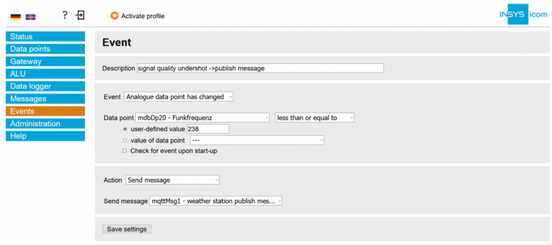

Figure 1:Konfigurieren statt Programmieren

Beispiel: Mirasoft AnyViz

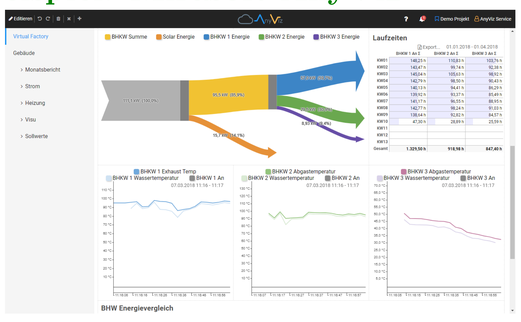

Figure 2:Ähnlich to iDS, aber komplett Cloud-basierend

Pros and Kontras

- Konfigurieren statt Programmieren
- Support vom Hersteller
- Eventuelle Lizenzkosten
- Eventuell schwierig zu erweitern

Empfehlung

Ideal für **nicht-Programmierer.**

Bekannte Linux Distributionen

Linux Distributionen wie beim **Raspberry Pi**. Minimalistische Versionen von Standard-Linux-Distributionen sind oft gut bekannt. Es sind gängige Paketmanager enthalten (wie apt-get, apk), so dass zusätzliche Software wie auf einem normalen Linux-PC installiert werden kann.

Beispiel: Debian Linux

Figure 3:Debian ist die Basis vieler bekannter Distributionen.

Beispiel: Alpine Linux

Figure 4:Alpine Linux versucht so schlank wie möglich zu bleiben.

Pros and Kontras

- Gut bekannte Packet-Manager
- Unglaublich viel verfügbare Software
- Sehr guter Community-Support
- Abhängig von Entscheidungen der Distri
- Container kann sehr schnell sehr groß werden
- Updates nach langer Zeit sind schwierig

Empfehlung

Ideal für **schnelle PoC** (Proof of Concept).

Basic Container

Script-Sprachen erlauben Programmierung ohne (Cross-) Compiler.
Es gibt Container mit installierten Programmiersprachen wie Python oder NodeJS. Benutzer können sich in einen solchen Container einloggen und sofort ihre Programme schreiben. Alternativ wird ein Programm auf einem PC geschrieben und einfach in den Container kopiert.

Beispiel: Python Scripte

Dieses kleine Skript abonniert ein Topic eines MQTT-Brokers und visualisiert die Werte via einfachem HTTP-Server.

```
1  import paho.mqtt.client as mqtt
2  import http.server
3  import socketserver
4
5  def on_connect(client, userdata, flags, rc):
6      client.subscribe("machine/temp")
7
8  def on_message(client, userdata, msg):
9      with open("index.html", "w", encoding='utf-8') as file:
10         file.write(f"<html><body>Temperature:
             {msg.payload}</body></html>")
11
12 mqtt_client = mqtt.Client()
13 mqtt_client.on_connect = on_connect
14 mqtt_client.on_message = on_message
15 mqtt_client.connect_async("mqtt.broker.de", 8889)
16 mqtt_client.loop_start()
17
18 Handler = http.server.SimpleHTTPRequestHandler
19 with socketserver.TCPServer(("", 8080), Handler) as httpd:
20     httpd.serve_forever()
```

Pros and Kontras

- Minimale Umgebung ergibt kleine Container
- Programmierung wie auf dem PC
- Unabhängig von der Architektur
- Nachhaltig durch leichte Wartbarkeit
- Programmieren im Container ohne SDK
- Abhängig vom Ersteller des Basis-Containers
- Funktionserweiterungen evtl. schwierig

Empfehlung

Ideal für **Skripte** um einfache Aufgaben zu lösen.

Eigenentwicklungen

Es gibt Build-Skripte und ein SDK (Software Development Kit) zur **Erstellung eigener Container**. Für Experten gedacht: Verwenden der präferierten Programmiersprache und verfügbarer Open- und Closed-Source. Der Container kann sogar nur aus einer einzigen Binärdatei bestehen (statisch gelinkt)! Sammlung nützlicher Links: https://m3-container.net/#scripts

Basis Container als Template

Es gibt Build-Skripte auf github, die als Vorlagen für eigene Container dienen. Geeignet für alle Programmiersprachen, u.a. C/C++, go oder C#.

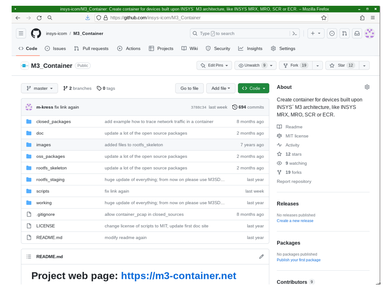

Figure 5:Skripte auf github in Kombination mit dem SDK

Pros and Kontras

- Kontrolle über alles
- Minimale Containergröße
- Integration in eigene CI/CD-Pipeline möglich
- 100% reproduzierbare Builds
- Minimale Angriffsfläche für Hackerangriffe
- Minimale externe Abhängigkeiten
- Erfordert tiefere Kenntisse
- Erforder initial mehr Entwicklungszeit

Empfehlung

Ideal für **professionelle Entwickler**, optimal für Massen-Roll-Out.