ARA ansible-reporting

Wer schon mal mit `ansible`, `ansible-playbook`. `ansible-console` gearbeitet hat kennt die Ausgabe auf der Konsole oder in einem Logfile.

Tritt ein Fehler auf, wirft Ansible oft einen riesigen JSON-Block aus, der escapte Newlines (\n) enthält. Das ist meist schwer zu lesen.

Am Ende jedes `ansible-playbook` Laufs steht der `PLAY RECAP`.

`unreachable > 0 Host nicht erreichbar` Ist noch relativ einfach zu finden

`failed > 0` Mindestens ein Task ist fehlgeschlagen. Bei welchen Task. `rescued > 0` Ein block/rescue hat einen Fehler aufgefangen. Wo und welchen?

Jetzt geht die Suche los!

Entweder in der Console oder im Log.

Das Kann schon eine *Herausforderung* sein.

ARA ist ein Analyse-, Protokollierungs- und Reporting-System.

Es hat folgende Aufgaben:

- Sammeln (Collect): Die Ergebnisse eines bereits laufenden oder abgeschlossenen Ansible-Runs speichern.
- Archivieren (Archive): Alle Daten in einer durchsuchbaren Datenbank speichern.
- Visualisieren (Visualize): Die Ergebnisse über eine Web-UI darstellen

ARA biete mit Records auch die Möglichkeit benutzerdefinierte, nicht-technische Metadaten hinzuzufügen. Somit kann ein Playbook-Lauf einideutig mit der Ursache verknüpft werden (Ticket-Nummer, Chane Request) oder/und gibst die commit-id der Konffirguration mit an.

ARA biete sich auch ann wenn `ansible-pull` oder `ansible-rulebook` verwendet werden, da es eine zentrale Sammelstelle der Ausgaben von `ansible-pull` und `ansible-rulebook` sein kann und sich somit eignet in ARA nach Läufen und Ereignissen zu suchen.

In dem Vortrag wir ARA vorgestellt.

- Installation und Erstkonfiguration von ARA
- kennenlernen der ARA Webui und der ARA Console
- zeigen von ARA Records, Labels
- Wie kann ARA bei `ansible-pull`, `ansible-rulebook` genutzt werden

Aufbau Vortrag (geplant)

Vortrag: ARA – Ansible Run Analysis

Zielgruppe: Ansible-Administratoren & DevOps Engineers

1. Einleitung & Problemstellung

Der "Schmerz" ohne ARA:

- Unübersichtliche Konsolenausgaben (stdout).
- Mühsame Suche in Logfiles nach Fehlern in hunderten Tasks.
- Das Problem dezentraler Läufe (ansible-pull).

Was ist ARA?

- Definition: Ein Open-Source "Black-Box-Rekorder" für Ansible.
- Die drei Säulen: Collect (Sammeln), Archive (Speichern), Visualize (Anzeigen).

2. Architektur & Funktionsweise

Wie ARA arbeitet:

- Ansible Callback Plugin: Klinkt sich in den Prozess ein.
- ARA API: Empfängt die Daten.
- Datenbank: SQLite (Standard) oder PostgreSQL/MySQL.
- Web-Interface: Dashboard zur Auswertung.
- Szenarien: Lokal (SQLite) vs. Zentral (API-Server).

3. Installation & Erstkonfiguration

Installation:

- Server-Seite: pip install ara[server] oder via Docker-Container.
- Client-Seite (wo Ansible läuft): pip install ara.

Konfiguration:

- Einbinden des Callback-Plugins

4. Nutzung der WebUI & CLI

- Live-Demo WebUI:
  - Dashboard: Übersicht über die letzten Läufe (Erfolgsquote).
  - Detailansicht: Drilldown von Playbooks über Plays bis hin zu einzelnen Tasks.
  - Task-Details: Anzeige von stdout, stderr und dem vollständigen JSON-Response.
  - Filter-Funktion: Suche nach fehlgeschlagenen Hosts oder spezifischen Modulen.
    - Playbook auf Fehler laufen lassen und Fehler finden

ARA CLI:

Abfragen von Statistiken direkt auf dem Terminal.(Anwendungsideen)

5. Profi-Features:

- ara_record.
- ARA Labels

6. Spezial-Szenarien: Pull & Rulebook

Ansible-Pull

Ansible-Rulebook

7. Fazit & Q&A

Zusammenfassung

Offene Fragerunde