Patterns & Anti-Patterns bei der Automatisierung mit Ansible

... kannste so machen, ist dann halt ***!

Henning Rohde

AmaliX GmbH
Ideen & Lösungen für DevOps-Teams & ITinfra-Projekte

Rant

- Ansible ist ein wirres Konglomerat
  - YAML/JSON
  - Jinja2
  - Python
- Anti-Features
  - diverse Variablen-Quellen
  - diverse Gültigkeitsbereiche
  - komplexe, inkongruente Variablen-Präzedenz und Vererbung
  - kein Unset, keine Konstanten
  - kein Schutz vor Namens-Kollisionen
  - unsaubere Differenzierung zw. Defaults, Variablen, Fakten und Parametern
  - unsaubere Typisierung
  - aufwändige Validierung

TL;DR

Good Practices for Ansible - GPA

1. Introduction

Ansible is simple, flexible, and powerful. Like any powerful tool, there are many ways to use it, some better than others.

This document aims to gather good practices from the field of Ansible practitioners at Red Hat, consultants, developers, and others. And thus it strives to give any Red Hat employee, partner or customer (or any Ansible user) a guideline from which to start in good conditions their automation journey.

Those are opinionated guidelines based on the experience of many people. They are not meant to be followed blindly if they don't fit the reader's specific use case, organization or needs; there is a reason why they are called *good* and not *best* practices.

The reader of this document is expected to have working practice of Ansible. If they are new to Ansible, the Getting started section of the official Ansible documentation is a better place to start.

This document is split in six main sections. Each section covers a different aspect of automation using Ansible (and in a broader term the whole Red Hat Ansible Automation Platform, including Ansible Tower):

1. structures: we need to know what to use for which purpose before we can delve into the details, this section explains this.
2. roles: as we recommend to use roles to host the most actual Ansible code, this is also where we'll cover the more low level aspects of code (tasks, variables, etc…).
3. collections
4. playbooks

who's me?

- Freelancer seit 2009
- gestartet als klassischer SysAdmin für KMU
- aktuelle Rollen
  - (Betriebs-)DevOps'ler für Infrastruktur & Automatisierung
  - Infrastruktur-Architekt
  - Analyse & Konzept für RZ-Migrationen
- Gründungsmitglied Hamburger Freelancer-Stammtisch 2011
- jenseits der IT
  - stolzer Vater
  - SchiKo auf dem Chaos-Kongreß
  - Speaker und Volunteer auf Fach- und „freien" Konferenzen

Mein Weg zu Ansible

- Projekt 2016:
  - großer Mobilfunker
  - vergammelte IT einer Akquisition
  - Outsourcing sollte es „retten“
  - größere Unix-Farm: Solaris 8..10, RHEL3..6
  - RHEL5+6: Konfig-Mgmt mit Puppet
  - Solaris + Rest jedoch hemdsärmlich-händisch administriert
  - erste Amtshandlung: Konfig-Mgmt auf 100% der Server
    - Roadblock: Ruby für Puppet-Agent auf dem altem Eisen
    - Lösung: Ansible mit Daten-Import aus Puppet/hieradata

Was ist Ansible?

- Automatisierung, Config-Management
- Infrastruct. as code, Config as code
- Methoden der Software-Entwicklung
  - release early, often
  - agile
  - Architur + Design -> Wartbarkeit
  - git-ops
  - DevOps -> Zusammenarbeit von Entwicklern und Admins

- Designziele
  - minimalistisch
  - sicher
  - zuverlässig
  - leicht erlernbar

Patterns

- gute und bewährte Problemlösungsansätze
- verknüpft mit einem konkreten Kontext
- standardisiert in Form von Abläufen & Strukturen

- logisch klarer und einfacher Aufbau
- Zweck: Wiederverwendung
- mit hohem Wiedererkennungswert (auch/insbes.) durch andere

- Ende 70'er: (Bau-)Architektur-Theorie, in röm. Tradition
- Ende 80'er: analoge Anwendung auf Software-Architektur
- 94: GoF-book

- Beispiele: 3-Tier-Architektur, Iterator, Exception handling

Abgrenzung Anti-Pattern

- Antithese zu Best-Practice: so besser/gefälligst nicht!
- Musterbeispiele für Ineffizienz und Dysfunktionalität
- erlauben ggfs. Rückschlüsse auf ”üblicheÜrsachen
- mit Wiedererkennungswert, bestenfalls prognostisch

Anti-Pattern

- Secrets im Code
- Rückgriff auf Shell trotz spezif. Modul
- keine Rollen verwendet
- zu mächtige Rollen
- --extra-vars als notwendiger Userinput
- endlose Filterketten

Herausforderungen Ansible für Netzwerker

- brownfield
- pets, never cattle!
- no downtimes, never!

- restricted shell
- Privilegien-Separation
- no downtimes, never!
- diametral andere Philosophie vs. Server

Danke

- meiner Frau und unseren Räuberkindern
- meinen Kunden
- den Kolleg*innen
- Michael, Jens und allen anderen für die Orga-Arbeit