

## **DOAG 2021 Online Konferenz vom 16.November bis 18.November 2021**

Lifecycle Management ohne „Pack“ und auch für die SE

# **DATENBANKWARTUNG UND PATCH MANAGEMENT MIT ANSIBLE AUTOMATISIEREN**

# Über uns

**Memet Omer**  
IG Metall – IT



**Memet.Omer@igmetall.de**

**Gunther Pippèrr**

Der IT-Macher GmbH & GPI Consult



**gunther@pipperr.de**

**Mein Blog**

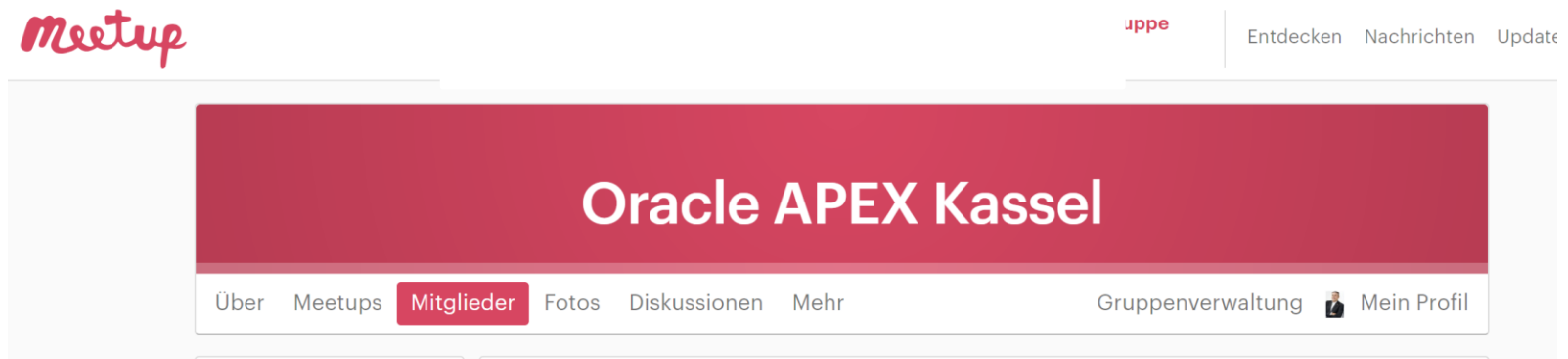
**<https://www.pipperr.de/dokuwiki/>**

# APEX Meetup Gruppe Kassel-Göttingen

Raum für Veranstaltung in Kassel oder Göttingen gesucht, können Sie uns unterstützen?

## Mitglieder gesucht!

<https://www.meetup.com/de-DE/Oracle-APEX-Kassel/>



# Agenda

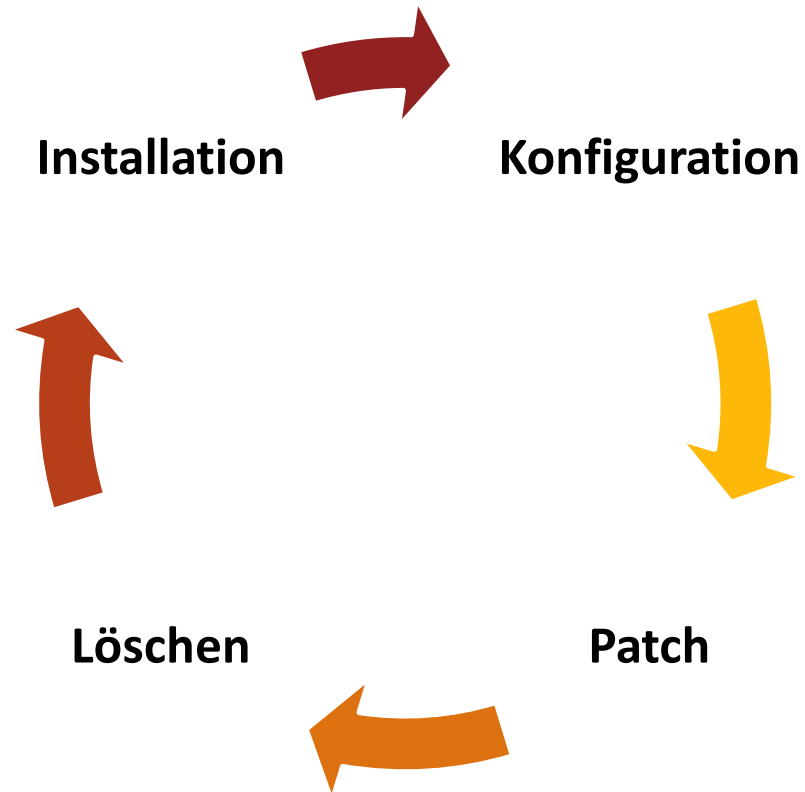
- 1 **Die Ausgangslage / Intention / Szenarien**
- 2 Basis Knowhow Ansible / Ansible Tower
- 3 Ein Playbook entwickeln / Eine integrative Umgebung erstellen
- 4 Projekt Erfahrungsbericht
- 5 Fazit



Sourcecode unter [https://github.com/gpipperr/DOAG\\_2021\\_ansible\\_oracle\\_patch\\_management](https://github.com/gpipperr/DOAG_2021_ansible_oracle_patch_management)

# Intention - LifeCycle Management

- LifeCycle Management



**ORACLE®**  
**ENTERPRISE MANAGER**



Oracle  
Automation  
Manager

# Was bietet uns Oracle?

---

- Oracle Enterprise Manager und das LifeCycle Management Pack
  - Zusammenfassung der früher separat erhältlichen Oracle Datenbank Management Packs:
    - Oracle Change Management Pack
    - Oracle Configuration Management Pack
    - Oracle Provisioning Pack



Nur für Oracle EE Datenbank Umgebung, die mit dem Oracle Enterprise Manager verwaltet wird

# Das LifeCycle Management Pack

---

Optimal für eine EE Edition Umgebung für die, die schon alles haben!

## KEY BENEFITS

- Non-intrusive agentless discovery of Servers on the network
- Integrated workflow promoting discovered servers to managed
- 360 degree view of assets in data center
- Automation to provision, patch and upgrade Oracle Database, RAC and underlying infrastructure
- My Oracle Support integration providing patch recommendation, pre-deployment analysis, rollout and reporting
- Deployment procedures that minimize downtime and enforce segregation of duties
- Automation for database schema and data deployment across instances
- Impact analysis of application upgrade due to database customizations
- Configuration comparison and search
- Topology view for impact and root cause analysis
- Compliance Standards for Oracle best practices and industry compliance requirements and reporting






## RELATED PRODUCTS

Oracle Database Lifecycle Management Pack provides maximum benefits when used with the following packs:

- Oracle Diagnostics Pack
- Oracle Tuning Pack
- Oracle Data Masking Pack
- Secure Test Data Management Pack
- Cloud Management Pack for Database

Aus der Oracle Dokumentation

# Warum dann Ansible verwenden? (1)

- Haben die den Oracle Enterprise Manager im Einsatz? 
- Können Sie sich min. 4-8 Stunden jeden Tag um dem OEM kümmern? 
- Haben Sie alle Problemchen mit dem OEM Agent gelöst und kennen Sie schon persönlich alle Support Mitarbeiter beim Vornamen? 
- Haben Sie nur EE Datenbanken im Unternehmen? 
- Haben Sie auch alle Packs, in allen Datenbanken lizenziert? 
- **Dann => Verwenden Sie das Lifecycle Management Pack!**



# Alternative – Ansible

---

- Open-Source Automatisierungs-Werkzeug
  - Seit Februar 2012
- Orchestrierung / allgemeinen Konfiguration und Administration von Unix/Linux und MS Windows System
- Agentenlose Architektur
  - Zugriff über SSH / WINRM

# Ansible als Alternative

## Vorteil

- OpenSource
- Frei erweiterbar mit eigenen Skripten
- Kann alle Aufgaben rund um Netzwerk/Server/Datenbank-Wartung erfüllen
- Viele Libraries erleichterten den Einstieg
- Viele Beispiele und gute Dokumentation im Netz mit reger Community

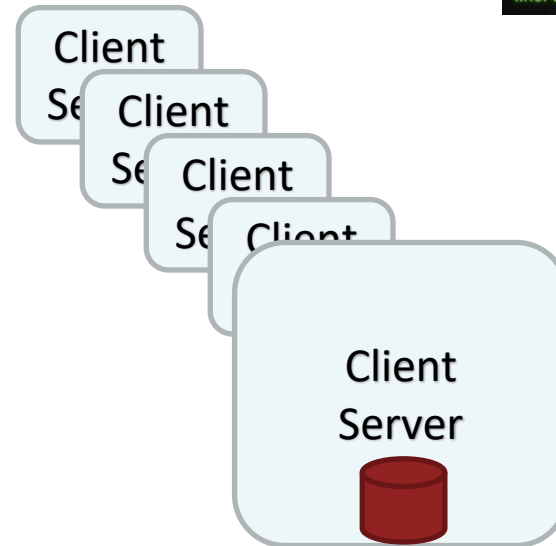
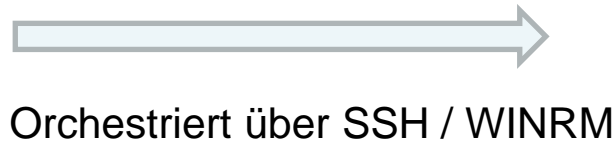
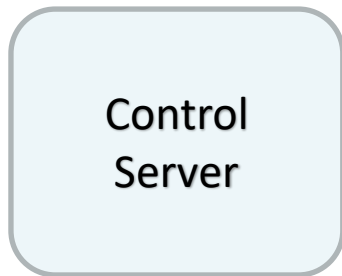
## Nachteil

- Einarbeitungsaufwand mit vielen kleinen Detailproblemen
- Komplexe Skripte entstehen mit hohem Knowhow Bedarf für die Wartung und Entwicklung
- Mitarbeiter müssen sich intensiv einarbeiten und gerne auch intensiv programmieren wollen
- Ein eigenes Software Projekt entsteht mit allem was dazu gehört
- Test ist schwierig, nur eingeschränkte „dry Run“ Funktionalität

# Ansible Architektur

## ■ Grundprinzip

Agentless SSH-based



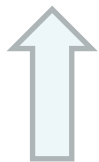
**Oracle Linux Automation Engine**

**Anisble** - Skript Library (Python)



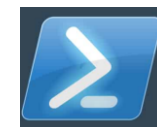
**AWX / Tower** - Web Interface zur Verwaltung – Password Safe – Job Steuerung

**Oracle Linux Automation Manager**



REST API zum Management

Windows Remote Management



# Wie erfolgt die Ausführung?

- Voraussetzung: Python ab 2.5 ist auf dem Ziel installiert
- Module werden auf das Ziel kopiert ( Verzeichnis `local_tmp` in `ansible.cfg`)
- Ablauf:
  - Anlegen lokales Verzeichnis: `$HOME/.ansible/tmp/ansible-tmp-xxx/ansible-cache`
  - Module kopieren
  - Module ausführen und Ergebnis als Json Output zurückgeben
  - Temporäre Daten wieder löschen `ansible-tmp-xxx/ansible-cache`

# Ansible bereitstellen

## ■ Oracle 8 :

- Repository installieren

```
dnf install oracle-epel-release-el8.x86_64
```

- Ansible Packages installieren

```
dnf install ansible ansible-doc vim-ansible
```

- Ansible User auf Control Sever und Client Server anlegen
  - Key Exchange konfigurieren
- Basis Verzeichnis für die Playbooks anlegen
- Client Server im Inventory Datei hinterlegen
- Zugriff testen

```
ansible ORA_SERVER -a "/bin/uname -a" -k
```

Siehe [https://www.pipperr.de/dokuwiki/doku.php?id=linux:oracle\\_linux\\_8\\_ansible](https://www.pipperr.de/dokuwiki/doku.php?id=linux:oracle_linux_8_ansible)

# Oracle Linux **Automation Manager** bereitstellen

- “Oracle Ansible Tower”

Seit September 2021



```
dnf config-manager --add-repo  
http://yum.oracle.com/repo/OracleLinux/OL8/automation/x86_64
```

```
dnf install ol-automation-manager
```

Dann konfigurieren => Siehe Blog Eintrag:

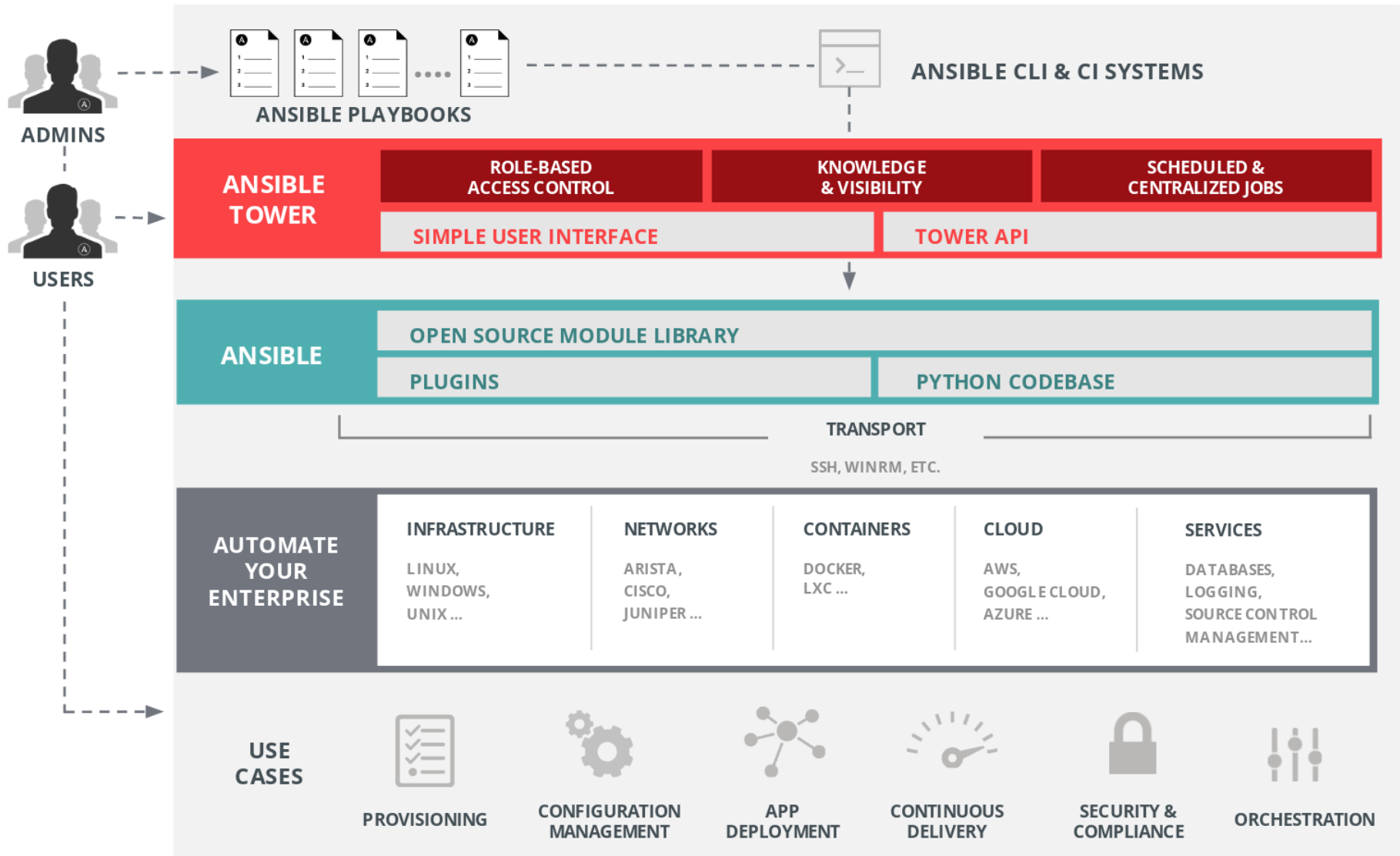
[https://www.pipperr.de/dokuwiki/doku.php?id=linux:oracle\\_linux\\_8\\_ansible\\_tower](https://www.pipperr.de/dokuwiki/doku.php?id=linux:oracle_linux_8_ansible_tower)

# Ansible Tower - Oracle Linux Automation Manager

---

- Weboberfläche um Ansible Playbooks (Templates) zu verwalten und auszuführen
  - Templates sind in Projekten organisiert
  - Alle zu wartenden Systeme werden in Inventories registriert und können in Gruppen organisiert werden
- Job Steuerung mit Jobs Logs
- Zugriffkontrolle
- Password Safe
- Visuelles Dashboard

# Übersicht Ansible Tower



Quelle: <https://www.redhat.com/en/blog/take-ansible-and-jenkins-integration-next-level-cicd-ansible-tower>



# Ansible Tower / AWX

## Vorteile

- Git Integration
- Password Safe
- Job Steuerung für wiederholende Aufgaben

## Nachteile

- Kein “out of the Box” Inventarisierungs-System im eigentlichen Sinne
- Sandbox Konzept in komplexeren Ansible Szenarien (wie weitere Module über Collections nachladen) teilweise schwer zu debuggen

- Wie bist du auf Ansible gekommen?
  - Wir haben Tools gesucht um die wiederkehrenden gleichen Tätigkeiten zu automatisieren.
  - Hauptsächlich wollte ich die Installation der Oracle CPUs automatisieren.
  
- Wie einfach hat sich der Start mit Ansible gestaltet?
  - Ehrlich gesagt war es gar nicht einfach. Denn allein eine lauffähige Ansible Umgebung bereitzustellen hat sehr viel Zeit in Anspruch genommen.
  
- Was waren die ersten Stolpersteine bei der Umsetzung eines solchen Projekts?
  - Ansible verwendet für die Playbooks die YAML Auszeichnungssprache. Hierbei muss man sich exakt an die Syntax halten, denn auch Leerzeichen spielen eine Rolle.

# Projekt Hürden die genommen werden müssen

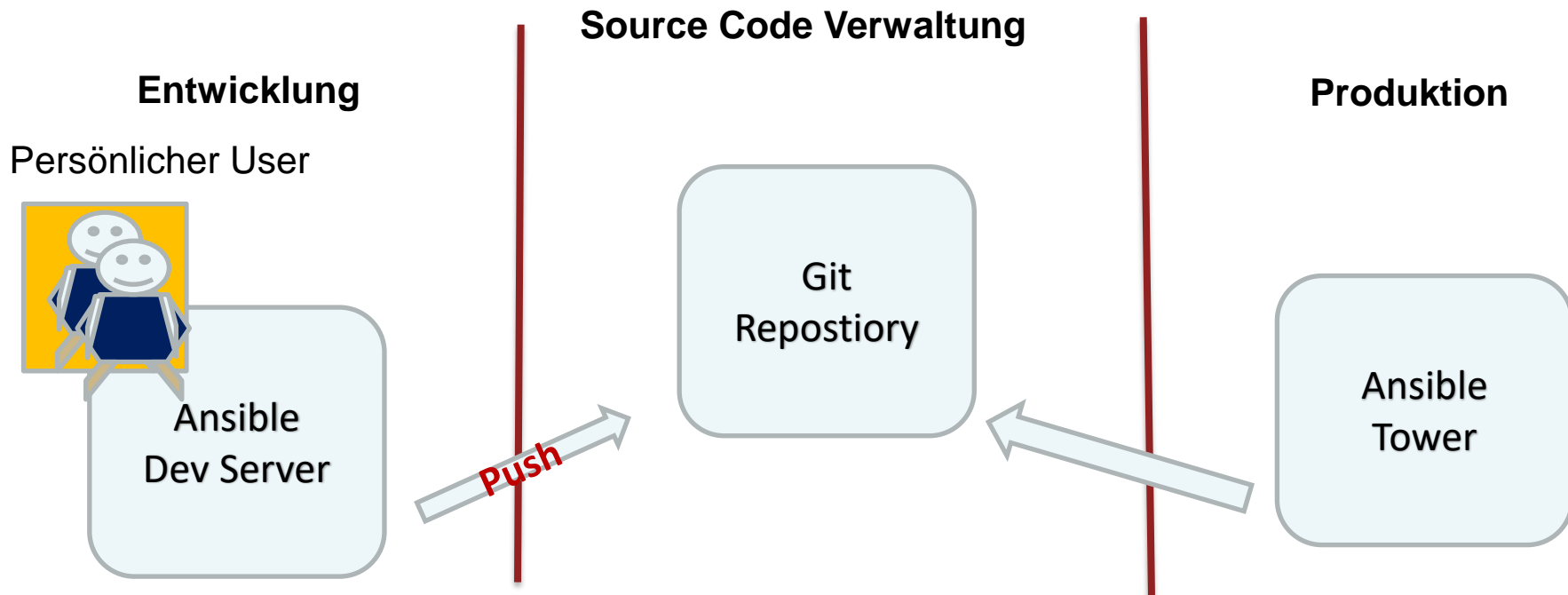
- Wie die Bestandsysteme mit wenig Aufwand in Ansible Tower erfassen?
- Wie einen neuen Server hinzufügen und initial in Betrieb nehmen?
- Wie die Oracle Datenbank Umgebung inventarisieren?

# Wie ein Ansible Projekt aufsetzen - Grundregeln

- Die Git Integration von Anfang an mit einplanen
  - Strukturierung der Projekte
  - Möglichst wenig Shell Skripte “programmieren” , besser „Default“ Funktionalitäten von Ansible Modulen verwenden
    - Ansible Modul Namen voll qualifizierend verwenden!
- Nich so gut => `copy:`                      besser => `ansible.builtin.copy:`
- Zentralen Agenten User auf allen Knoten verwenden / Zugriff über Schlüsselaustausch

# Wie eine Ansible Umgebung strukturieren?

- Git einsetzen!



Playbook entwickeln  
und Testen über Kommandozeile  
Parameter über Konfigurationsdateien

Beim Ausführen immer automatisch  
Die aktuellste Variante aus dem Repository laden

Nur mit "Git" können Collections (module) automatisch  
in Tower/AWX auch nachgeladen werden!

# Ansible Collection – Grund Idee (1)

- Nachladen von Ansible Module über das “Galaxy” Universum
  - Modulen sind fertige Lösungen für komplexere Konfiguration
  - Wie die sysctl.conf bearbeiten mit ansible.posix.sysctl
  - Manuel nachladen

```
- name: edit sysctl.conf step 2
  ansible.posix.sysctl:
    name: kernel.panic_on_oops
    value: 1
    state: present
```

```
ansible-galaxy collection install ansible.posix
```

# Ansible Collection – Grund Idee (2)

Demo

- Wie aber in Tower konfigurieren?
  - RedHat Tower – Credentials für RedHat hinterlegen (Token)

The screenshot shows the 'EDIT CREDENTIAL' page in Red Hat Tower. The page title is 'CREDENTIALS / EDIT CREDENTIAL'. The main heading is 'Ansible Automation Hub (https://cloud.redhat.com/api/automation-hub/)'. There are two tabs: 'DETAILS' (selected) and 'PERMISSIONS'. The 'DETAILS' tab contains the following fields:

- \* NAME: Ansible Automation Hub (https://cloud)
- DESCRIPTION: (empty)
- ORGANIZATION: Default
- \* CREDENTIAL TYPE: Ansible Galaxy/Automation Hub
- TYPE DETAILS:
  - \* GALAXY SERVER URL: https://cloud.redhat.com/api/aut
  - AUTH SERVER URL: https://sso.redhat.com/auth/real
  - API TOKEN: ENCRYPTED
- \* RUN PROJECT UPDATES WITH HIGHER VERBOSITY: (disabled toggle)
- IGNORE ANSIBLE GALAXY SSL CERTIFICATE VERIFICATION: (enabled toggle)
- ENABLE COLLECTION(S) DOWNLOAD: (enabled toggle)
- FOLLOW SYMLINKS: (disabled toggle)

- Job Settings anpassen
- Nachladen über Collection Definition in Datei  
collections/requirements.yml

```
---  
collections:  
  - ansible.posix
```

# Wie eine Entwicklungs-Umgebung einrichten(1)?

- Wie die Tower Umgebung in der Entwicklung simulieren?
  - Wie „Inventory – Host Vars – Credentials“ aus der Tower Umgebung in der Entwicklung simulieren?
- AWX Python Umgebung einstellen mit:

```
source /var/lib/awx/venv/ansible/bin/activate
```



# Wie eine Entwicklungs-Umgebung einrichten(2)?

- Wie Credentials aus der Tower Umgebung simulieren?
  - **Beispiel zentralen Ansible User hinterlegen**
    - Datei `group_vars/all` anlegen mit

```
ansible_connection: ssh
ansible_ssh_user: ansiblevc
ansible_ssh_pass: <<my_mostly_used_root_pwd>>
```

- In `.gitignore` aufnehmen mit “vi .gitignore”
- Verschlüsseln mit

```
ansible-vault encrypt ./group_vars/all
```

- Playbook mit Password Eingabe aufrufen

```
ansible-playbook --ask-vault-pass execute_command.yml
```

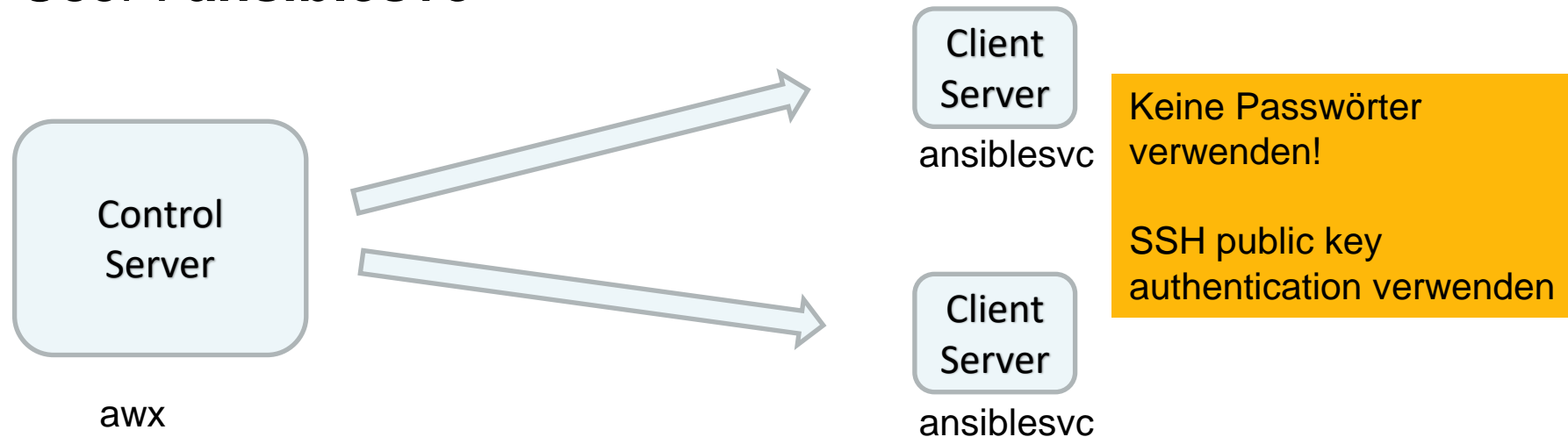
# Wie eine Entwicklungs-Umgebung einrichten(3)?

- Parameter in eigener Param-Datei hinterlegen und mit der **-e** Option aufrufen
  - In Tower werden diese dann in den “EXTRA VARIABLES” auf dem Template hinterlegt
  - Verzeichnis „**debug\_params/**“ im Projekt Ordner anlegen
  - Parameter Daten mit „**<play\_book\_name>\_param.yml**“ mit den notwendigen Parameter anlegen
  - In „.gitignore“ diese Verzeichnis aufnehmen um zu verhindern das sensible Daten über git verteilt werden!
  - Aufrufen mit der **-e** Option (optional mit **--step** für die schrittweise Ausführung), auf das **@** (für Datei) achten!

```
ansible-playbook install_ahf.yml
--ask-vault-pass --limit apex01.pipperr.local
-e @./debug_params/install_ahf_param.yml --step
```

# Regel 1 - Nicht mit Root arbeiten

- Control Server (**Oracle Linux Automation Manager**)
  - Tower User (ruft Plays auf) : **awx**
  - Software Owner  
(hier werden die Skripte entwickelt und getestet) : **ansible**
- Client Server (**Oracle Linux Automation Engine**)
  - User : **ansiblevc**



# Regel 2 - Möglichst mit Ansible Methoden arbeiten

- Beispiel: Plattenplatz ermitteln aus den ansible Facts
  - Welche Facts stehen im Default zur Verfügung

```
ansible <hostname> -m ansible.builtin.setup --ask-vault-pass
```

- In Playbook in einer Variable verwenden

```
---
- hosts : bvmmdb32
  vars:
    - size_total: "{{ ansible_mounts|json_query('[?mount == `/opt`].size_total') }}"
    - size_free : "{{ ansible_mounts|json_query('[?mount == `/opt`].size_available') }}"

  tasks:
    - name: check Diskspace
      ansible.builtin.debug:
        msg: "The filesystem has {{ ( size_total[0]/1024/1024 )|round(1,'common') }}
              MB space, free is {{ (size_free[0]/1024/1024)|round(2) }} MB"
```

Jinja2 templating system

[https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible\\_templating\\_jinja2](https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible_templating_jinja2)

# Wie Ansible Tower initial betanken?

- Wie alle Hosts initial auf einmal erfassen?
  - Z.B. aus vorhandener Keepass Datei mit Python auslesen

[https://www.pipperr.de/dokuwiki/doku.php?id=python:python\\_read\\_keepass\\_file](https://www.pipperr.de/dokuwiki/doku.php?id=python:python_read_keepass_file)
- Wie nachträglich einen Host erfassen?
  - Z.B. mit dem AWX Cli

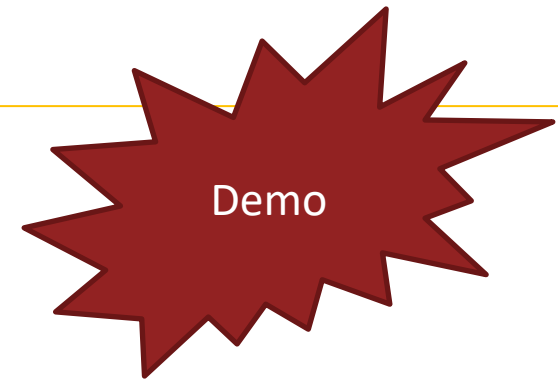
[https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible\\_tower\\_cli\\_awx](https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible_tower_cli_awx)
- Playbook um den Host initial zu konfigurieren
  - Ansible Service User anlegen
  - Key Austausch
  - Weitere Default Einstellungen wie ssh Zugriff einschränken etc.

# Das Ansible Henne-Ei Problem

---

- Wie den Service User auf den Clients anlegen?
  - A) Manuell
  - B) Skript
    - Ablauf in unsere Umgebung per Skript:
      - Host mit root Password als “Host Var” in speziellen Install Inventory in Tower anlegen
      - Playbook für das Anlegen des Users / Key Austausch / Härten des Systems starten
      - Host im „Setup Inventory“ löschen (Passwort in Tower wieder damit entfernt)
      - Host in produktiven Inventory anlegen (ohne PWD, da ja jetzt Key Austausch definiert)
    - Skript fragt zu Beginn die notwendigen Parameter ab und führt alle Schritte in Richtung Tower per REST API durch

# Ansible AWX / Tower “fernsteuern”



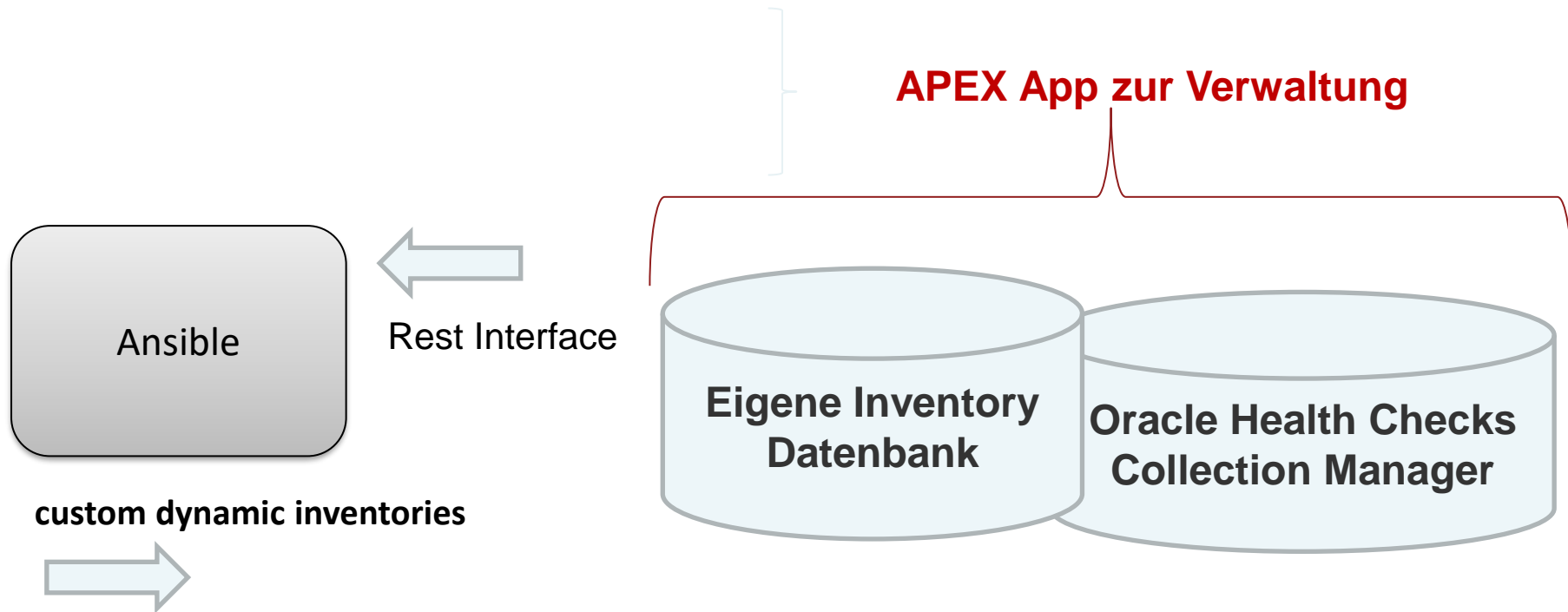
- Per Rest API steuern
- Vorteil:
  - Schnelle Pflege, ohne viel durch Oberflächen navigieren zu müssen

```
awx host create --conf.insecure
--name apex01
--inventory 1
--description "APEX Dev Host"
--variables '{ "ansible_connection":"ssh",
               , "ansible_ssh_user":"root",
               , "ansible_ssh_pass":"mysecret1"} '
```

[https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible\\_tower\\_cli\\_awx](https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible_tower_cli_awx)

# Wie die Datenbank Umgebung katalogisieren?

- Wie und wo werden die Master Daten unserer Umgebung hinterlegt?
  - Z.b. als „**host\_vars**“ auf dem Host
  - Wie das aber automatisch erfassen und verwalten?



Hier arbeiten wir noch an einer einfachen und übersichtlichen Lösung

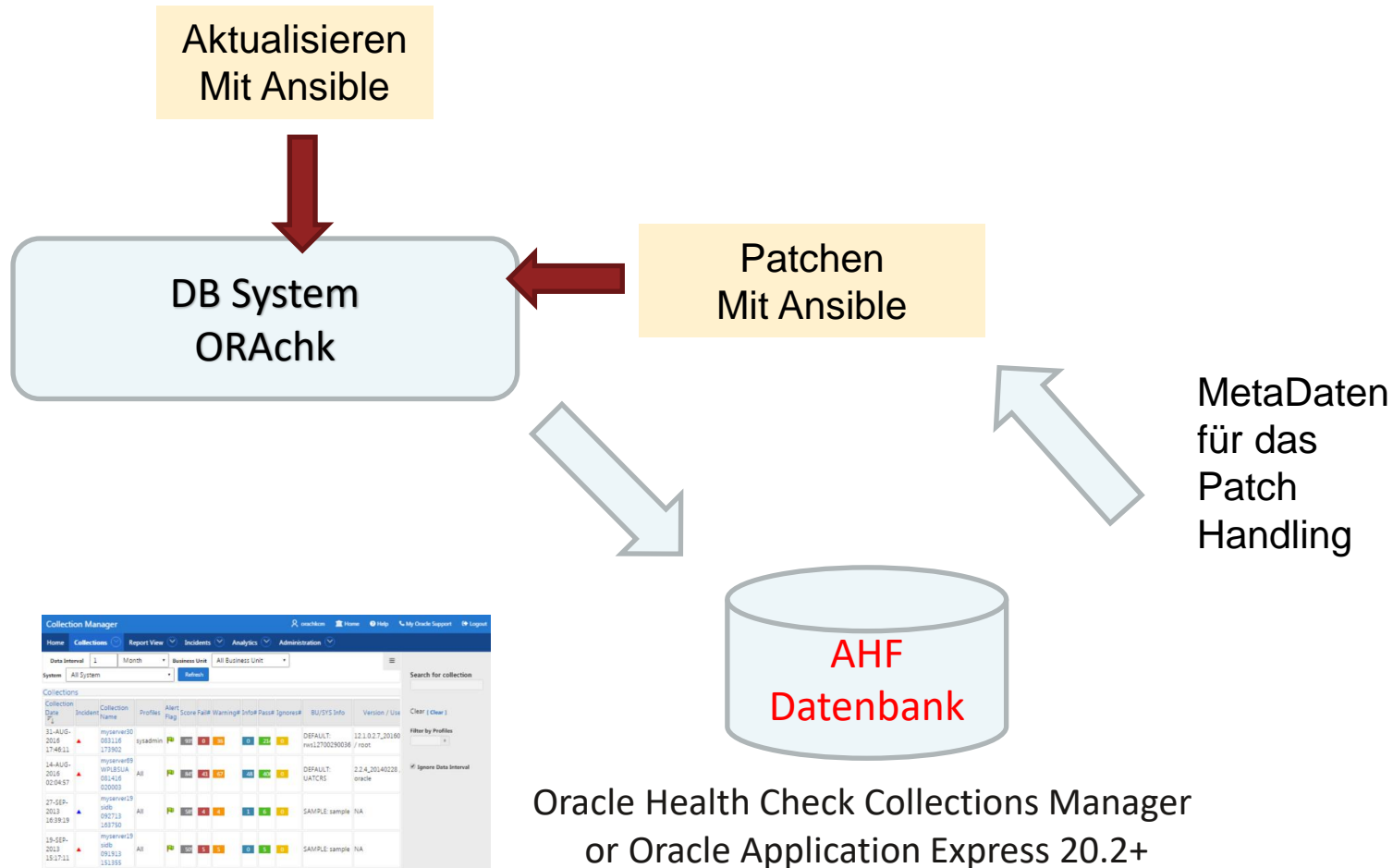


# Projekt Idee – AHF <—>Ansible Integration

---

- Oracle AHF Integration in die Ansible Welt für das Inventarisieren und Protokollieren der DB Umgebung
  - Verteilen und regelmäßig Updaten AHF mit Ansible
  - Ergebnis orachk Reports in Datenbank schreiben
  - Datenbank für die Inventarisierung verwenden
  - Datenbank Information ist die Basis für das Patchen mit Ansible
  - Patch Deployment über Ansible

# Projekt Idee – AHF <=> Ansible Integration



<https://docs.oracle.com/en/engineered-systems/health-diagnostics/exachk/oexug/apex5-managing-oracle-health-check-collection-manager.html>

# Wie am einfachsten den Patch von MOS laden?

- Ein gutes Tool dazu:
  - <https://github.com/MarisElsins/getMOSPatch>

```
java -jar getMOSPatch.jar MOSUser=Gunther.Pipperr@nomail.de patch=30166242 download=all
Enter your MOS password:
-
We're going to download patches for the following Platforms/Languages:
 4L - German (D)
226P - Linux x86-64

Processing patch 30166242 for German (D) and applying regexp .* to the filenames:
No files available

Processing patch 30166242 for Linux x86-64 and applying regexp .* to the filenames:
1 - AHF-LINUX_v21.3.0.zip
Enter Comma separated files to download: all
All files will be downloaded because download=all was specified.

Downloading all selected files:
Downloading AHF-LINUX_v21.3.0.zip: 392MB at average speed of 91396KB/s - DONE!
```

Beispiel: AHF Download

# AHF Install und Update

- Oracle AUTONOMOUS HEALTH FRAMEWORK (AHF) auf allen DB Systemen installieren und regelmäßig updaten
  - AHF von Support Portal herunterladen
  - Auf das Zielsystem kopieren
  - Installieren oder Updaten
  - Konfigurieren
  - Orachk Bericht erzeugen und in Repository DB laden



[https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible\\_oracle\\_ahf\\_install](https://www.pipperr.de/dokuwiki/doku.php?id=linux:ansible_oracle_ahf_install)

# Was benötigen wir dazu?

---

- Inventory
  - Liste der zu orchestrierenden Server
- Module
  - Skript Lib's mit fertigen Methoden für die eigentlichen Aufgaben
- Playbooks
  - Skript für die eigentliche Aufgabe; in Tasks organisiert
- Roles
  - Wiederverwendbare Aufgaben, die mit in Playbooks eingebunden werden können
- Templates
  - Vorlagen für Scripts die auf dem Client ausgeführt werden sollen mit dem „Jinja2 templating system“ für die Definition von Variablen

# Ein Oracle Ansible Patch Modul definieren

- PreDowntime
  - Oracle Home und Patch Stand feststellen
  - Opatch aktualisieren
  - Patch Software bereitstellen
  - DB Health Check durchführen
  - Patch mit Opatch prüfen ( prereq CheckConflictAgainstOHWithDetail etc.)
- Downtime
  - Alle Dienste/Programm aus dem Oracle Home stoppen
    - Applikation der DB stoppen
    - Listener stoppen
    - DB sauber herunterfahren
    - Mit Optach Patch überprüfen
    - Mit Opatch Patch(e) einspielen
    - DB Starten
    - Patch in DB einspielen
    - DB weiter Maßnahmen durchführen ( je nach Bedarf wie utlrp )
- PostDowntime
  - Listener starten
  - Applikation der DB neu starten

# Gibt es das bereits?

- <https://github.com/oravirt/ansible-oracle>
- <https://github.com/oravirt/ansible-oracle-modules>
- <https://github.com/iarov/ansible-orapatch>
  - <https://blog.pythian.com/automating-oracle-patching-with-an-ansible-module/>

## Vorteil:

- Viel der benötigten Komplexität ist bereits umgesetzt

## Nachteil:

- Hohe Komplexität der Skripte
- Für einfache Aufgaben sind eigene Skripte verständlicher

**Und wann gibt es das direkt von Oracle? .-)**

# Reine Ansible Lösung?

---

- Wie aber das ganze so umsetzen das möglichst wenig Skript Aufwand notwendig wird?
- Wie möglichst viel Ansible Basis Verhalten verwenden?



# Erfahrungsbericht – Projektfazit (1)

- Würdest du nochmal mit Ansible diesen Ansatz verfolgen?
  - Ja, denn umso mehr wir mit Ansible arbeiten umso mehr stellen wir auch fest wie mächtig Ansible ist.
  - Mittlerweile gibt es Git-Repositories mit viele Playbook Beispielen zu diversen Themen.
  - Aber auch Oracle selbst arbeitet wohl in der Oracle Cloud mit Ansible und unterstützt die Installation auf Oracle VMs da es jetzt ja eine Ansible Variante von Oracle gibt.
  - Ich hoffe, dass Oracle in Zukunft auch Playbooks zur automatisieren von diversen Tätigkeiten anbieten wird

# Erfahrungsbericht – Projektfazit (2)

---

- Was sollte in einem Ansible Projekt auf jeden Fall früh im Fokus stehen?
  - Wie gesagt, man muss konzeptionell vorgehen. Denn sonst kann man sich im Detail verlaufen.
  - Was wichtig ist, man sollte soweit es geht mit Ansible Bordmitteln arbeiten.
  - Vielen weichen schnell auf Python Skripte um, was dadurch noch komplexer wird denn man verliert irgendwann den Überblick.

# Erfahrungen aus der Einführung

---

- “Klassisch” eingeführt
  - Tower installiert, einfache „Redhat“ Tests mit lokalen Workbooks – erste Einschränkungen – wenig genutzt
  - Weitere Skripte erstmal unter root Lokal entwickelt
  - Erste Erfolge mit dem Einbinden aller Server
  - Viele kleine Schwierigkeiten mit den “kleinen” Dingen
  - Dann doch strukturiert angepasst und auf GIT Funktionalität umgestiegen
  - Suche nach fertigen Lösungen, da eine eigene Entwicklung immer aufwendiger wird
  - Der Nutzen stellt sich langsam ein

# Was beim nächsten Mal gleich zu Begin einführen?

- Benutzer-Konzept erstellen
- Namens-Konzept für Projekte und Templates
- Lokaler Git Hub Server mit zentralen Projekten
- Aufsetzen des eigentlichen „Tower/AWX“ Servers
- Lokale Entwicklung mit Hilfs-Skripten, um die Tower Umgebung zu simulieren

# Fazit

---

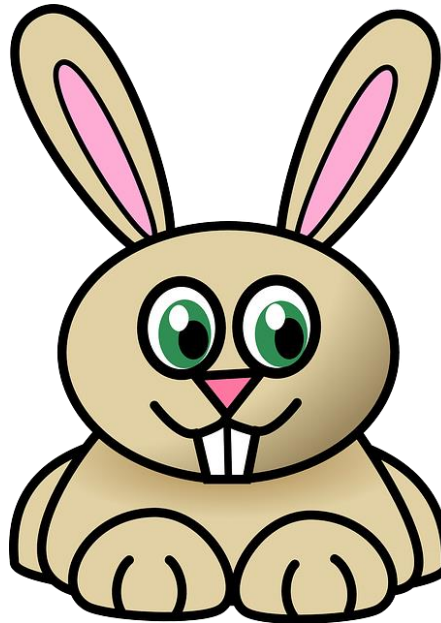
- Eine Ansible Einführung als Projekt sehr ernst nehmen
  - Aufwand steckt in der Einführung / Dokumentation / Strukturierung / Pflege weniger in der Entwicklung von einzelnen Skripts!

# Mehr

---

- Blog Gunther Pippèrr  
<https://www.pipperr.de/dokuwiki/doku.php>
- Source Code vom Vortrag:  
[https://github.com/gpipperr/DOAG\\_2021\\_ansible\\_oracle\\_patch\\_management](https://github.com/gpipperr/DOAG_2021_ansible_oracle_patch_management)
- Wieder mal eine andere Skript Library
  - <https://github.com/gpipperr/OraPowerShell>
- Bildmaterial : <https://pixabay.com>

Fragen ?



# FAQ



# F&A

Fragen

**Ansible im Einsatz**