

# Konfigurationsmanagement über verschiedene Netze mit Ansible und AWX



Dr. Ottavia Balducci, ATIX AG

Chemnitzer Linux-Tage, 11. März 2023

## 1 Einführung

## 2 Ansible

## 3 AWX

- Einführung
- Grundkomponenten
- Execution Nodes
- Workflow templates
- Zusammenfassung

# Wer bin ich?



- ▶ Ottavia Balducci
- ▶ IT-Beraterin bei ATIX AG

## Mein Fokus:

- ▶ Ansible
- ▶ AWX/AAP
- ▶ orcharhino/Foreman



- ▶ Email: [balducci@atix.de](mailto:balducci@atix.de)

# THE Linux & Open Source Company!



Consulting



Engineering



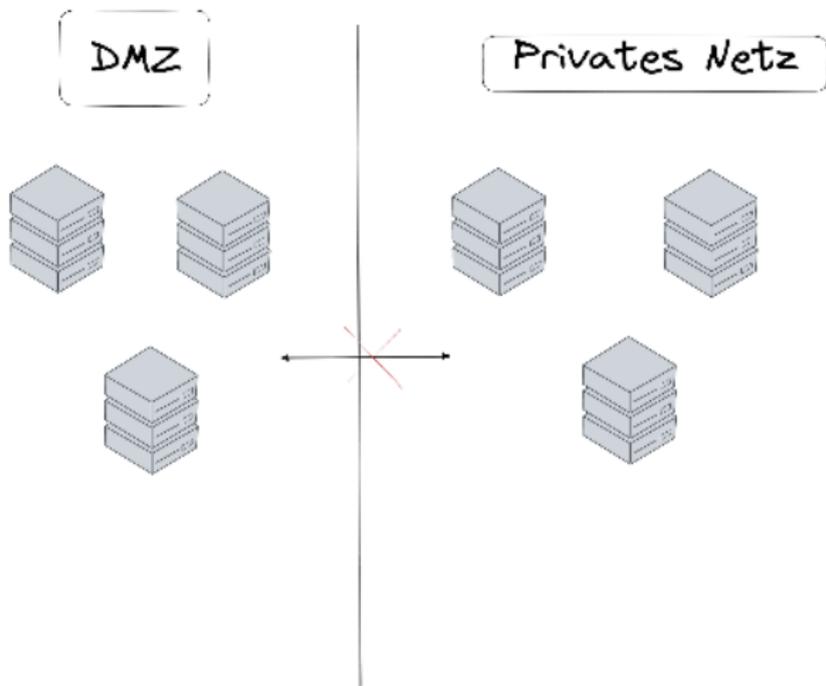
Support



Training

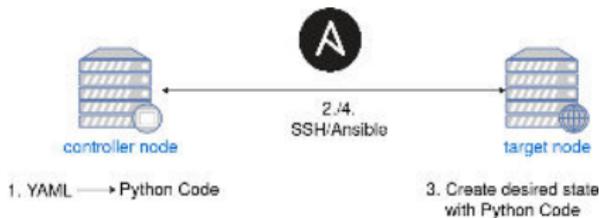


1. Wir möchten Hosts mit Ansible konfigurieren
2. Wir möchten die Gelegenheit haben, diese Hosts aus einer GUI zu konfigurieren  $\Rightarrow$  AWX
3. Wir möchten das auch über verschiedene Netze machen können

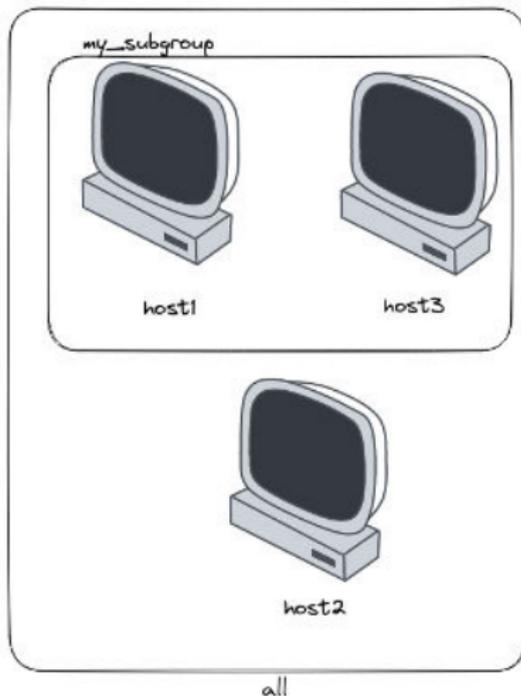


## Ansible

- ▶ Konfigurationsmanagement- und Automatisierungstool
- ▶ Ohne Clients und Daemons
- ▶ Idempotent: Nur Wunschzustand angeben
- ▶ Alle Hosts parallel verwalten
- ▶ Manche Tasks auf Hosts-Untergruppen



- ▶ Definiere Hosts und Hostgruppen
- ▶ YAML- oder INI-Datei
- ▶ Kann dynamisch erstellt werden, z.B.:
  - ▶ VMware
  - ▶ orcharhino/foreman
  - ▶ Cloud Providers
  - ▶ Docker
  - ▶ ...



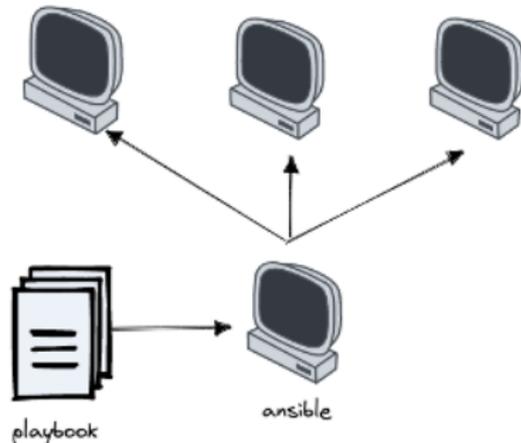
# Grundkomponenten – Inventory



```
---
all:
  hosts:
    192.168.121.1:
    192.168.121.2:
    192.168.121.3:
  vars:
    ansible_user: my_user
    ansible_password: my_password
  children:
    my_subgroup:
      hosts:
        192.168.121.1:
        192.168.121.3:
...

```

- ▶ YAML-Datei
- ▶ Reihenfolge von Befehlen (Tasks)
- ▶ Beschreibt die Schritte für eine größere Aufgabe



# Grundkomponenten – Playbooks



```
---
- hosts: all
  tasks:
    - name: Install Apache packages and deps
      become: true
      ansible.builtin.yum:
        name: "{{ packages }}"
        state: present
      vars:
        packages:
          - httpd
          - httpd-devel
    - name: Copy configuration files.
      ansible.builtin.copy:
        src: "{{ item.src }}"
        dest: "{{ item.dest }}"
        mode: 0644
      loop:
        - src: "/path/to/config/httpd.conf"
          dest: "/var/httpd/conf/httpd.conf"
        - src: "/path/to/config/httpd-vhosts.conf"
          dest: "/var/httpd/conf/httpd-vhosts.conf"
    - name: Make sure Apache is started on boot.
      become: true
      ansible.builtin.service:
        name: httpd
        state: started
        enabled: true
...

```

Ausgeführt mit

```
ansible-playbook -i inventory.yaml playbook.yaml
```

## Outputbeispiel:

```
PLAY [rocky] *****
TASK [Gathering Facts] *****
ok: [node1]

TASK [install chrony on rocky hosts] *****
ok: [node1]

TASK [set timezone] *****
changed: [node1]

TASK [copy chrony config file] *****
changed: [node1]

TASK [start chrony] *****
changed: [node1]

PLAY RECAP *****
node1: ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

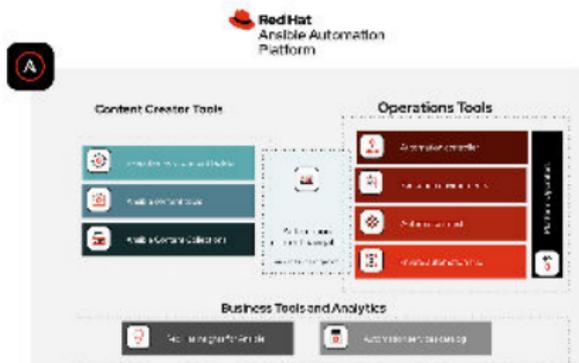
- ▶ AWX/ Ansible Automation Platform
- ▶ Auf Ansible aufgebaut
- ▶ Zum Planen und Verwalten von Ansible Jobs
- ▶ Angenehme Benutzeroberfläche



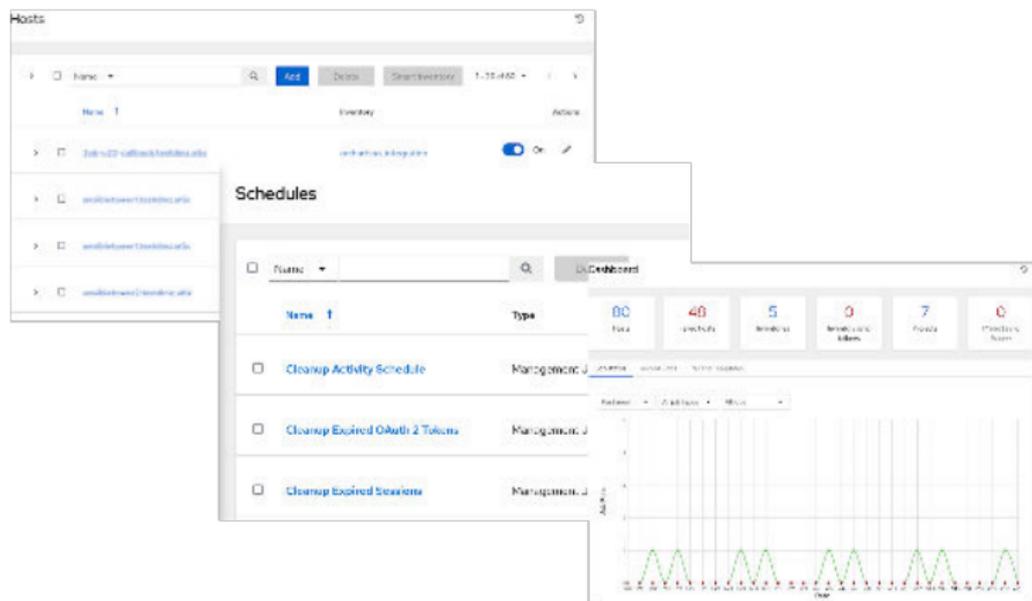
- ▶ Open Source
- ▶ vollkommen anpassbar
- ▶ Web-UI
- ▶ REST API
- ▶ RBAC
- ▶ CI/CD



- ▶ Enterprise-Produkt
- ▶ stabil und zuverlässig
- ▶ Private Automation Hub
- ▶ Enterprise-Inhalte

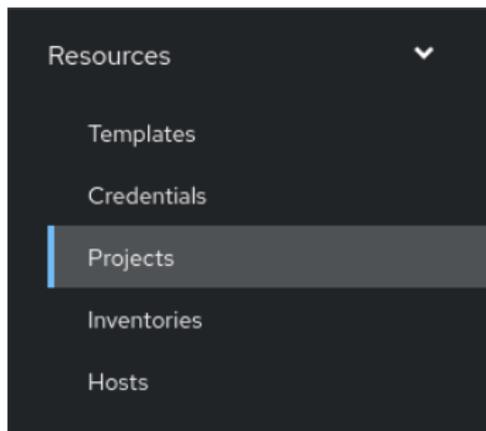


# Benutzeroberfläche



## Projekt:

- ▶ Logische Sammlung von Ansible Playbooks
- ▶ Wird mit einem Versionsverwaltungstool synchronisiert
- ▶ Daraus werden Job Templates abgeleitet

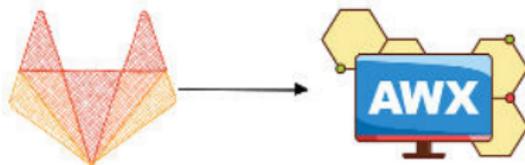


# AWX-Grundkomponenten: Projekte



Mögliche Quellen:

- ▶ Manual
- ▶ Git/Subversion
- ▶ RedHat Insights
- ▶ Remote Archive



>  my\_project

✔ Successful

Git

89b2a4e



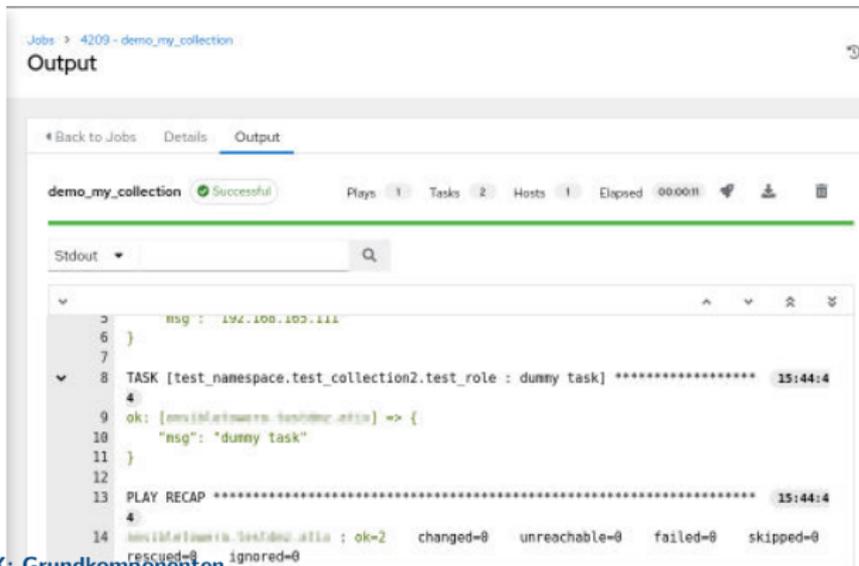
# AWX-Grundkomponenten: Job Templates

## Job Template:

- ▶ Wird aus einem Projekt erstellt
- ▶ Muss ein Playbook aus dem Projekt auswählen
- ▶ Es können zusätzliche Variablen gesetzt werden
- ▶ Man kann ein Inventory zuweisen
- ▶ und mehr ...

# AWX-Grundkomponenten: Job Templates

- ▶ Spezifisches Output für jeden Job unter dem Jobs-Tab
- ▶ Man kann nach fehlgeschlagenen Hosts, übersprungenen Tasks, usw. filtern
- ▶ Auch verfügbar: JSON-Output für jeden Task
- ▶ Ausführlichkeit kann ausgewählt werden



```
Jobs > 4209 - demo_my_collection
Output

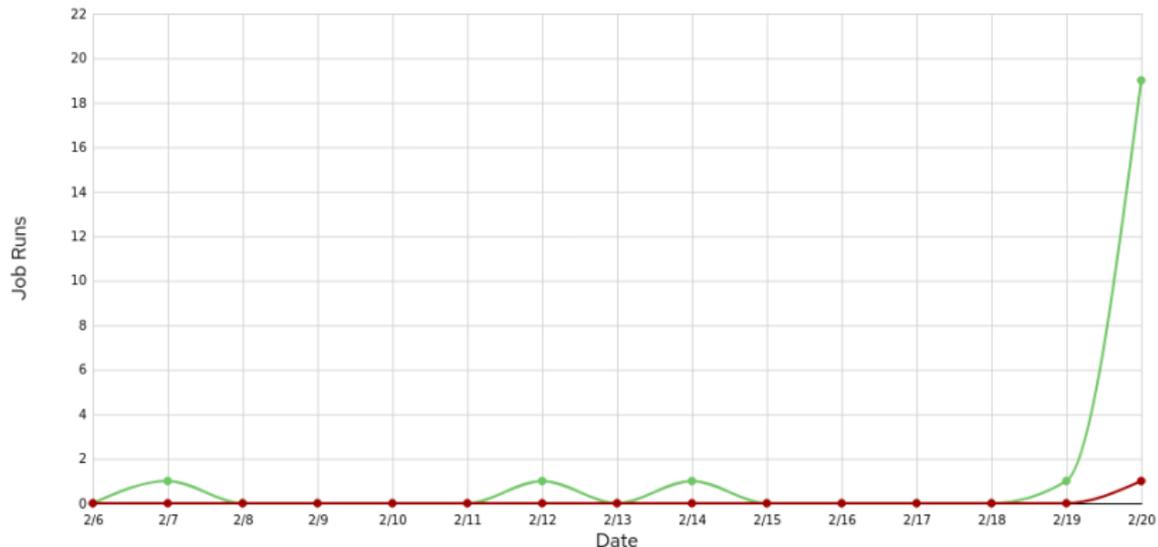
Back to Jobs Details Output

demo_my_collection Successful Plays: 1 Tasks: 2 Hosts: 1 Elapsed: 00:00:11

Stdout
msg: 192.108.100.111
}
TASK [test_namespace.test_collection2.test_role : dummy task] ***** 15:44:4
ok: [localhost:localhost ansible_host=192.108.100.111] => {
  "nsg": "dummy task"
}
PLAY RECAP ***** 15:44:4
ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

# AWX-Grundkomponenten: Job Templates

Dashboard:



- ▶ Sammlung von Target-Hosts (analog zu Inventory-Dateien)
- ▶ Eigene Resource in der Weboberfläche
- ▶ Gruppen und Untergruppen möglich
- ▶ Zusatzvariablen möglich
- ▶ Werden den Jobs zugewiesen

## Standard-Inventories:

- ▶ Hosts per Hand ausgewählt

[Inventories](#) > [Demo Inventory](#)

### Hosts

[◀ Back to Inventories](#) [Details](#) [Access](#) [Groups](#) [Hosts](#) [Sources](#) [Jobs](#)

Name ↑	Description ↓
<input type="checkbox"/> localhost	

# AWX-Grundkomponenten: Hosts



- ▶ eigene Resource
- ▶ eigene Variablen
- ▶ kann deaktiviert werden

Hosts

Name	Inventory	Actions
<input type="checkbox"/> <a href="#">2019-03-01-callback-awx@ms.atix</a>	orchestration-integration	<input checked="" type="checkbox"/> On
<input type="checkbox"/> <a href="#">awx@kubernetes@ms.atix</a>	Test	<input checked="" type="checkbox"/> On
<input type="checkbox"/> <a href="#">awx@kubernetes@ms.atix</a>	orchestration-integration	<input checked="" type="checkbox"/> On
<input type="checkbox"/> <a href="#">awx@kubernetes@ms.atix</a>	Test	<input checked="" type="checkbox"/> On

# AWX-Grundkomponenten: Dynamische Inventories



- ▶ Hosts werden aus einer Quelle gezogen
- ▶ Verschiedene Quellen möglich
- ▶ Automatische Gruppen hängen von der Quelle ab

The screenshot displays the AWX web interface for managing dynamic inventories. At the top, there is a search bar with a dropdown menu set to 'Name', a search icon, and three buttons: 'Add' (blue), 'Delete' (grey), and 'Sync all' (blue). To the right of these buttons, it shows '1 - 1 of 1' with navigation arrows. Below this is a table with the following columns: 'Name', 'Status', 'Type', and 'Actions'. The table contains one entry: 'orcharhino' with a 'Successful' status (indicated by a green checkmark), and 'Type' 'Red Hat Satellite 6'. The 'Actions' column for this entry contains a refresh icon and an edit icon. At the bottom of the table, there is a pagination bar showing '1 - 1 of 1 items', navigation arrows, and '1 of 1 page'.

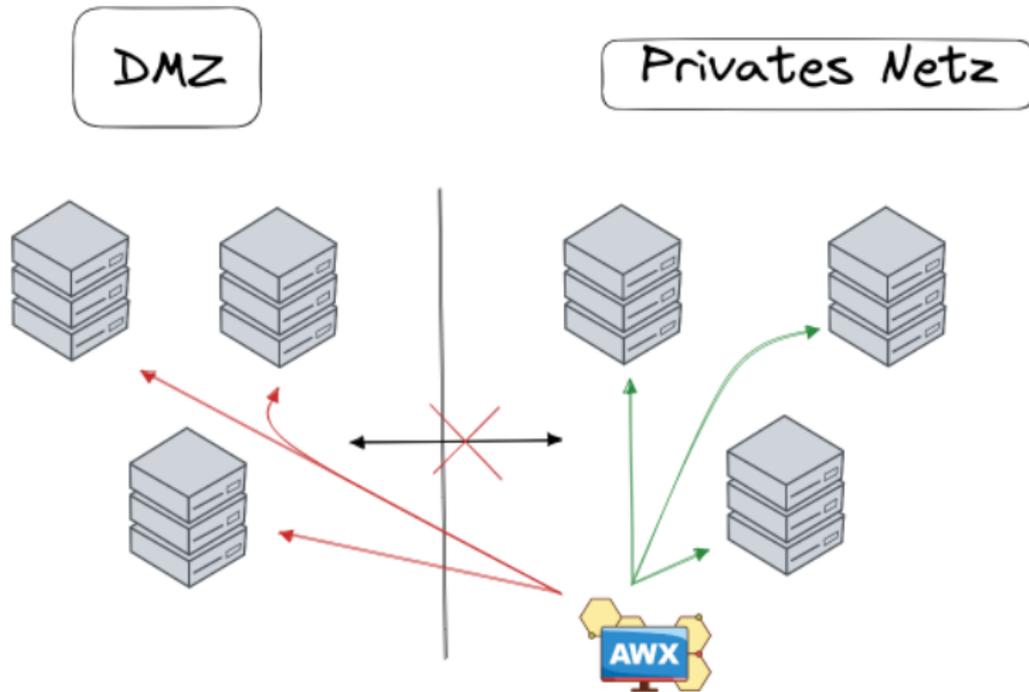
Name	Status	Type	Actions
orcharhino	Successful	Red Hat Satellite 6	

- ▶ Projekte
- ▶ Job Templates
- ▶ Inventories

Damit kann man schon alle Hosts bedienen, die von AWX erreichbar sind!

# Unser Problem - 1

Die Hosts liegen in verschiedenen Netzen und können nicht alle gleichzeitig von AWX erreicht werden!



## Unser Problem - 2

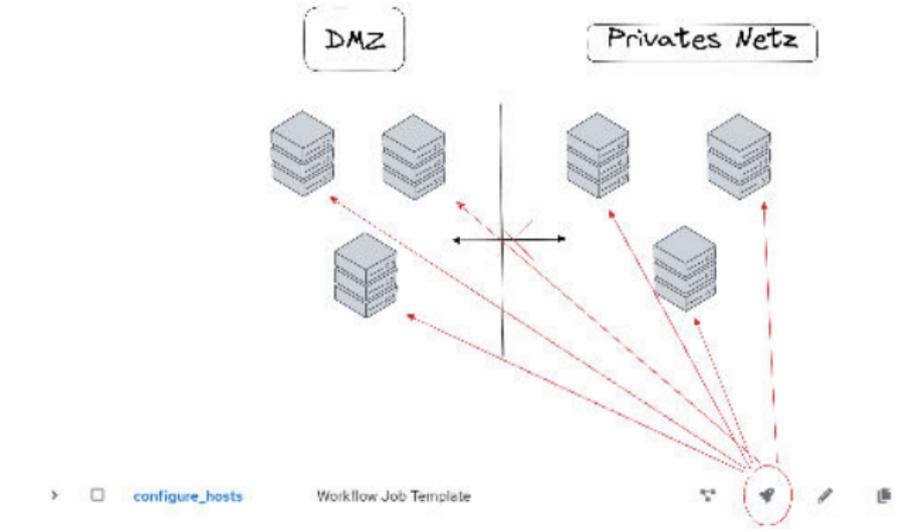


Außerdem haben wir nur ein einziges flaches Inventory:

- ▶ dynamisch erstellt
- ▶ keine geeigneten Gruppen

```
---  
all:  
  hosts:  
    host-1-dmz:  
    host-2-dmz:  
    host-3-privat:  
    host-4-dmz:  
    host-5-privat:  
    host-6-privat:  
    # ...  
...
```

⇒ Wie können wir trotzdem alle Hosts auf einmal mit einem Knopfdruck konfigurieren?



# Spoiler Alert!

Lösung:



AWX + Execution Nodes + Workflow Template

## Execution Nodes:

- ▶ Zusätzliche Nodes, die Ansible-Jobs ausführen
- ▶ Können keine Management-Jobs ausführen
- ▶ Können keine Web-Anfragen beantworten
- ▶ Kommunizieren über Receptor auf Port 27199

# Topology

## Topology:

- ▶ Netz aus Controller- und Execution-Nodes

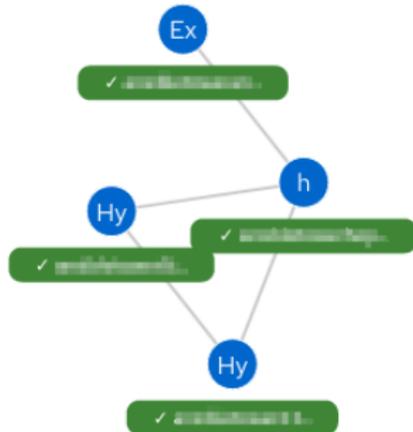
### Legend

#### Node types

- C** Control node
- Ex** Execution node
- Hy** Hybrid node
- h** Hop node

#### Status types

- Healthy
- Error
- Disabled



# Execution Nodes hinzufügen



Execution Nodes können in der AWX-GUI hinzugefügt werden!

[Instances](#)

Create new Instance ↻

Host Name *	Description	Instance State ⓘ
<input type="text"/>	<input type="text"/>	installed
Listener Port * ⓘ	Instance Type ⓘ	Options
<input type="text" value="27199"/>	execution	<input checked="" type="checkbox"/> Enable Instance ⓘ
<input type="button" value="Save"/>	<input type="button" value="Cancel"/>	

Neu seit Version 21.7.0.

# Execution Nodes hinzufügen



Ansible Automation Platform:  
Execution Nodes können im Inventory bei der Installation deklariert werden

```
[automationcontroller]
mycontroller1.example
mycontroller2.example

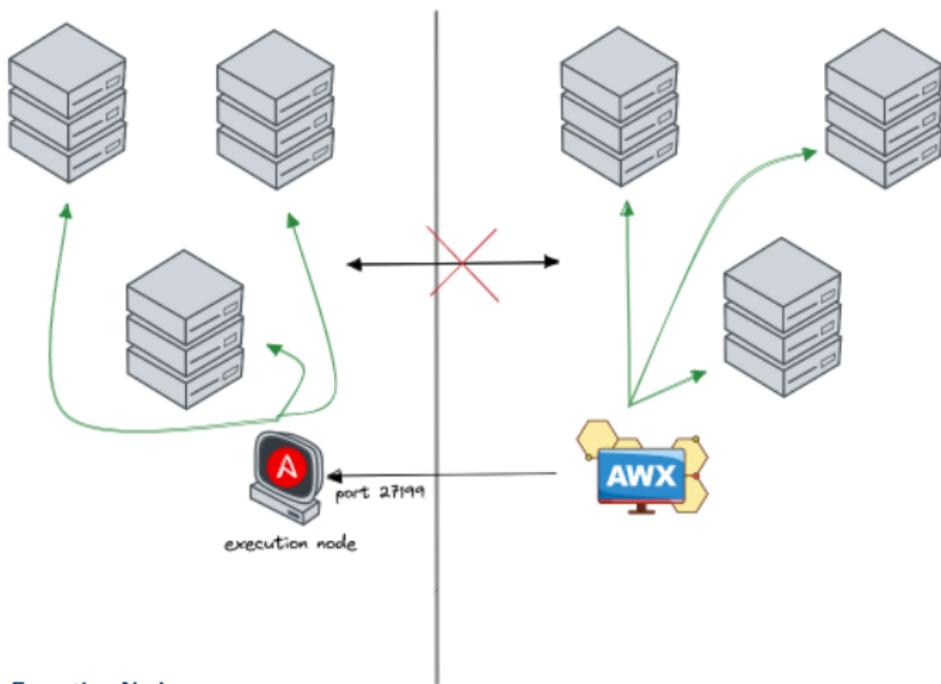
[automationcontroller:vars]
peers=hop.example

[execution_nodes]
hop.example          node_type=hop
execution.example    node_type=execution
# this defaults to execution
execution2.example
```

# Lösung für unser Problem

DMZ

Privates Netz



# Instance Groups

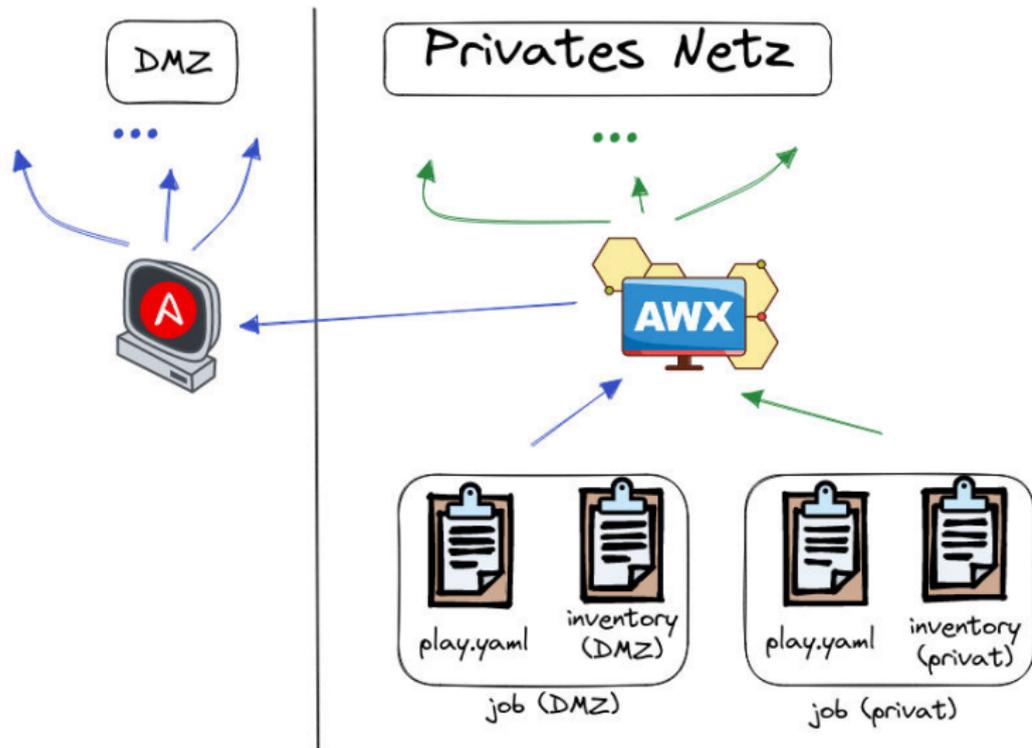


- ▶ Gruppe von Nodes, die einen Job ausführen können
- ▶ Policies für die Gruppe möglich (z.B. Anzahl der Jobs)
- ▶ Kann in jedem Job Template ausgewählt werden

Instance Groups 

A search input field with a magnifying glass icon on the left. The text 'controlplane' is entered, followed by a small 'x' icon to clear the text.

# Lösung für unser Problem



# Unser Problem



- ▶ Wir haben nur ein einziges Inventory
- ▶ Das Inventory ist dynamisch erstellt
- ▶ Es ist flach, d.h. es gibt keine geeigneten Gruppen

```
---  
all:  
  hosts:  
    host-1-dmz:  
    host-2-dmz:  
    host-3-privat:  
    host-4-dmz:  
    host-5-privat:  
    host-6-privat:  
    # ...  
...
```

Erstelle einen neuen Job:

- ▶ Soll als Input das flache Inventory haben
- ▶ Soll als Output zwei Inventories zurückgeben
- ▶ Trennung z.B. mittels regex anhand der Hostnamen
- ▶ Keine Verbindung zu den Hosts nötig, weil keine Facts gebraucht werden
- ▶ Soll auf localhost laufen

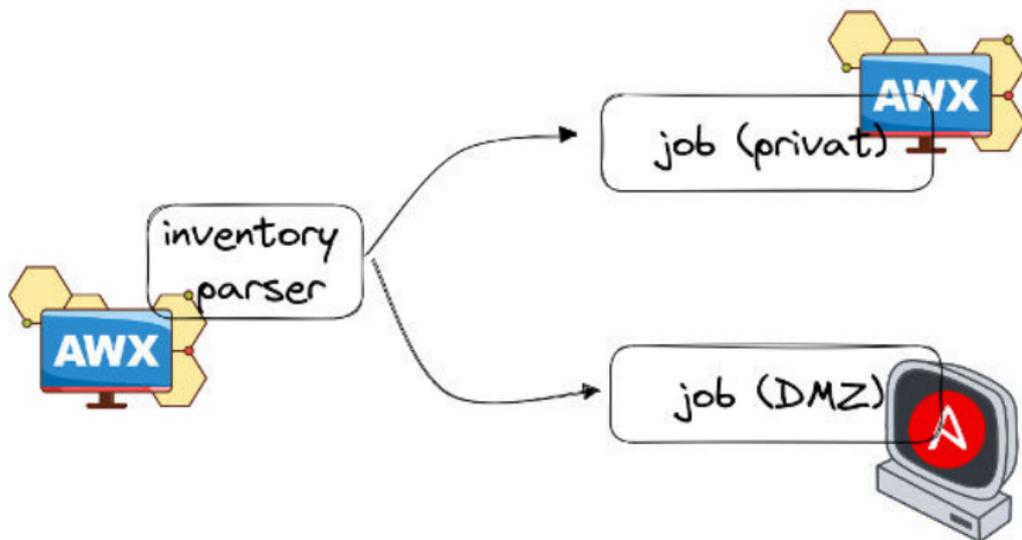
# Inventory parser



```
---
- hosts: "{{{ inithosts }}"
  gather_facts: false
  tasks:
    - name: "Add hosts to dmz"
      ansible.builtin.add_host:
        name: "{{{ item }}"
        groups: dmz
      loop: "{{{ ansible_play_hosts }}"
      when: '"dmz" in item'
      delegate_to: localhost
    - name: "Add hosts to private"
      ansible.builtin.add_host:
        name: "{{{ item }}"
        groups: private
      loop: "{{{ ansible_play_hosts }}"
      when: '"privat" in item'
      delegate_to: localhost
    - name: "Pass the new groups to the next jobs"
      ansible.builtin.set_stats:
        data:
          dmz_hosts: "{{{ groups.dmz | default('localhost') }}"
          private_hosts: "{{{ groups.private | default('localhost') }}"
        aggregate: false
      delegate_to: localhost
```

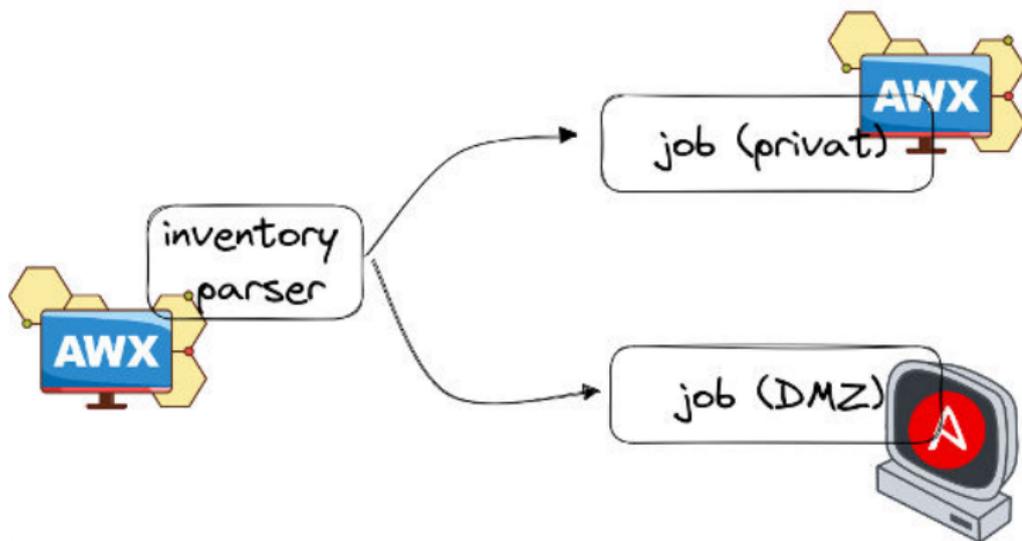
# Wichtig!

- ▶ Die neuen Gruppen müssen an zwei verschiedene Jobs weitergegeben werden
- ▶ Sonst könnten sie nicht auf zwei verschiedenen Nodes ausgeführt werden



# Workflow Templates

- ▶ Kette von Job Templates
- ▶ Zum Beispiel nützlich für:
  - ▶ Fehlerbehandlung
  - ▶ Genehmigung von Jobs
  - ▶ ... und unser Problem!

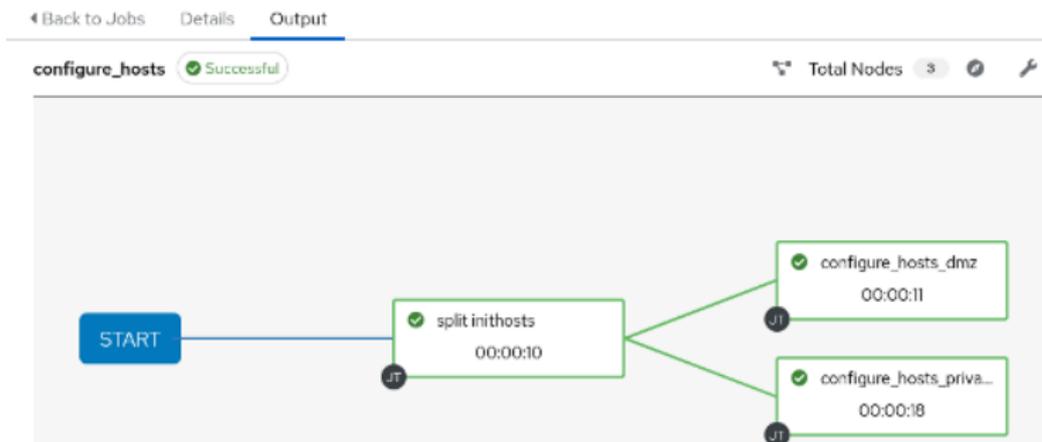


Verschiedene Arten von Nodes:

- ▶ Job Template
- ▶ Workflow Job Template
- ▶ Project Sync
- ▶ Approval
- ▶ Inventory Source Sync
- ▶ Management Job

Ein Workflow Template welches drei Jobs ausführt:

1. Aufteilen der Hosts in die korrekten Inventories
2. Ausführen des Jobs im DMZ durch die Execution Node
3. Ausführen des Jobs im privaten Netz von AWX selbst

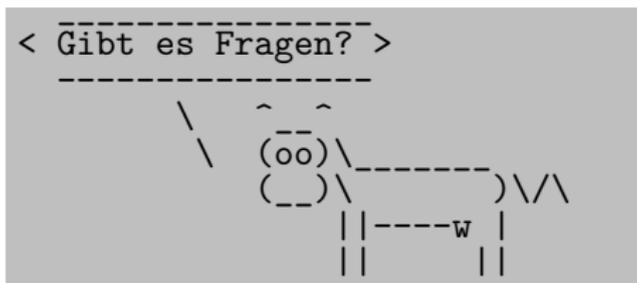




- ▶ AWX als Tool für die Automatisierung und Verwaltung von Ansible-Projekten
- ▶ Komplexe Topologie-Szenarien dank Execution Nodes möglich
- ▶ Workflow Templates zur Abstimmung zwischen Jobs

Kurz: AWX ist cool!

Danke für eure Aufmerksamkeit!



- ▶ WebAssembly auf der Serverseite: Was ist WASI?, Dr. P. Fries
- ▶ Autoscaling in Kubernetes – From Zero to Hero, T. Manske und L. Paluch
- ▶ Das Chaos überblicken – Monitoring & Tracing in Kubernetes, S. Rauch und V. Welker