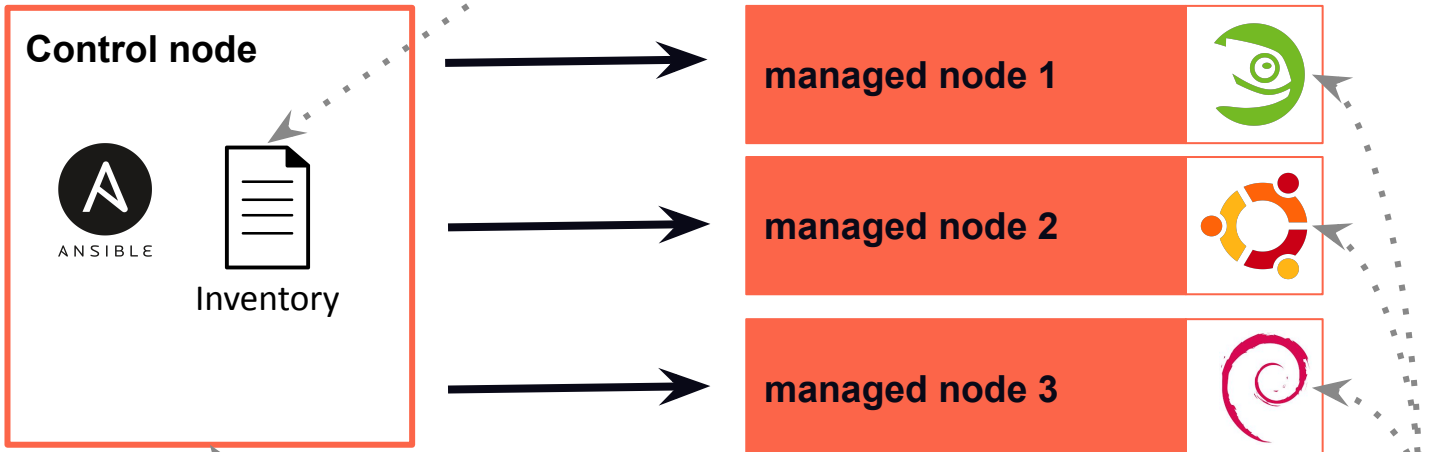
Ansible

TECH
STARTER

Ansible

Grobstruktur

Inventory: Datei in einem INI-Format, die Hosts und Gruppen in Ansible beschreibt. Kann auch über ein Inventarskript bereitgestellt werden.



Control node: Ein System, auf dem Ansible installiert ist.

managed node: Ein Remote-System oder Host, das von Ansible gesteuert wird.

https://docs.ansible.com/ansible/latest/getting_started/index.htm

Was ist ansible?

- Ansible ist ein Open-Source-Tool zur Konfigurationsverwaltung und Bereitstellung

- Es verwendet SSH, um sich mit Servern zu verbinden und die konfigurierten Aufgaben auszuführen. Ansible ermöglicht es, Knoten von einer einzelnen Maschine aus zu steuern und zu konfigurieren.
- Was es von anderer Verwaltungssoftware unterscheidet, ist, dass Ansible die SSH-Infrastruktur nutzt. Das Projekt wurde 2013 gegründet und von Red Hat in 2015 aufgekauft.

Warum Ansible?

- Kein Agent - Solange auf die Box per SSH zugegriffen werden kann und Python installiert ist, kann sie mit Ansible konfiguriert werden.

- Idempotent - Die gesamte Architektur von Ansible ist um das Konzept der Idempotenz strukturiert. Der Kernpunkt hierbei ist, dass Dinge nur dann ausgeführt werden, wenn sie notwendig sind, und dass sie wiederholbar sind, ohne Nebenwirkungen zu verursachen.

Warum Ansible?

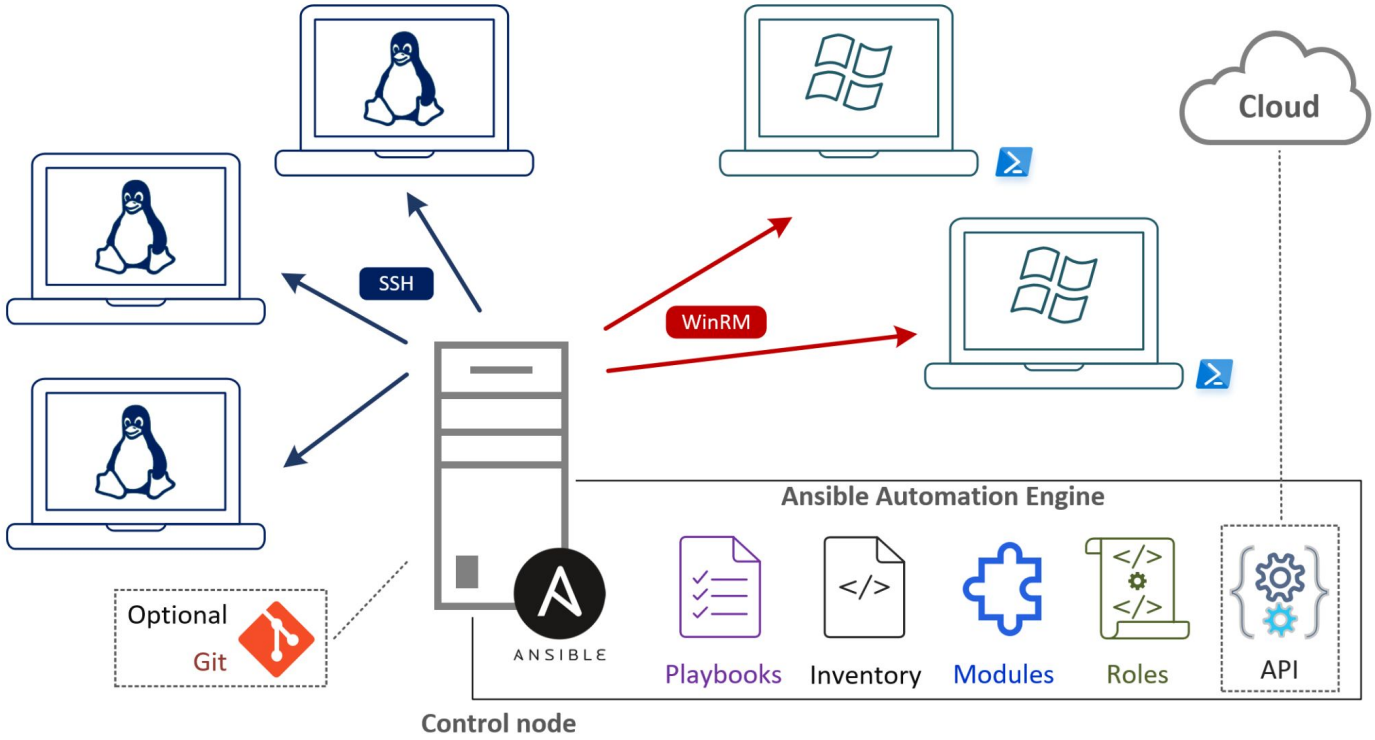
- Deklarativ, nicht prozedural - Andere Konfigurationstools neigen dazu, prozedural zu arbeiten: "mache dies und dann das" und so weiter. Bei Ansible schreibt man eine Beschreibung des gewünschten Zustands der Maschine, und es werden dann die notwendigen Schritte unternommen, um diesen Zustand zu erreichen.

- Geringe Lernkurve - Ansible ist ziemlich einfach zu erlernen.

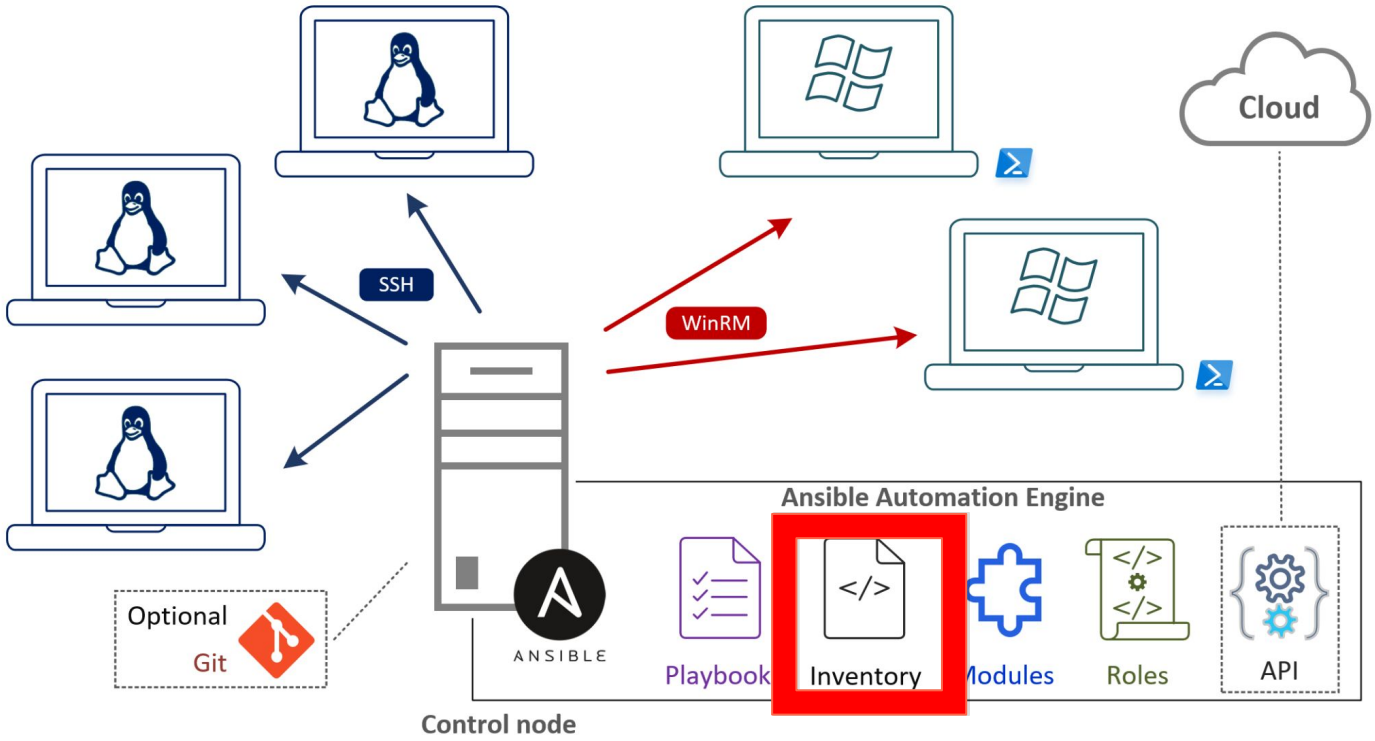
Anwendungsfälle

- (Bereitstellung)
- Konfigurationsmanagement
- App-Bereitstellung
- Kontinuierliche Auslieferung
- Sicherheit & Compliance
- Orchestrierung

Ansible Architektur



Das Inventory



Das Inventory

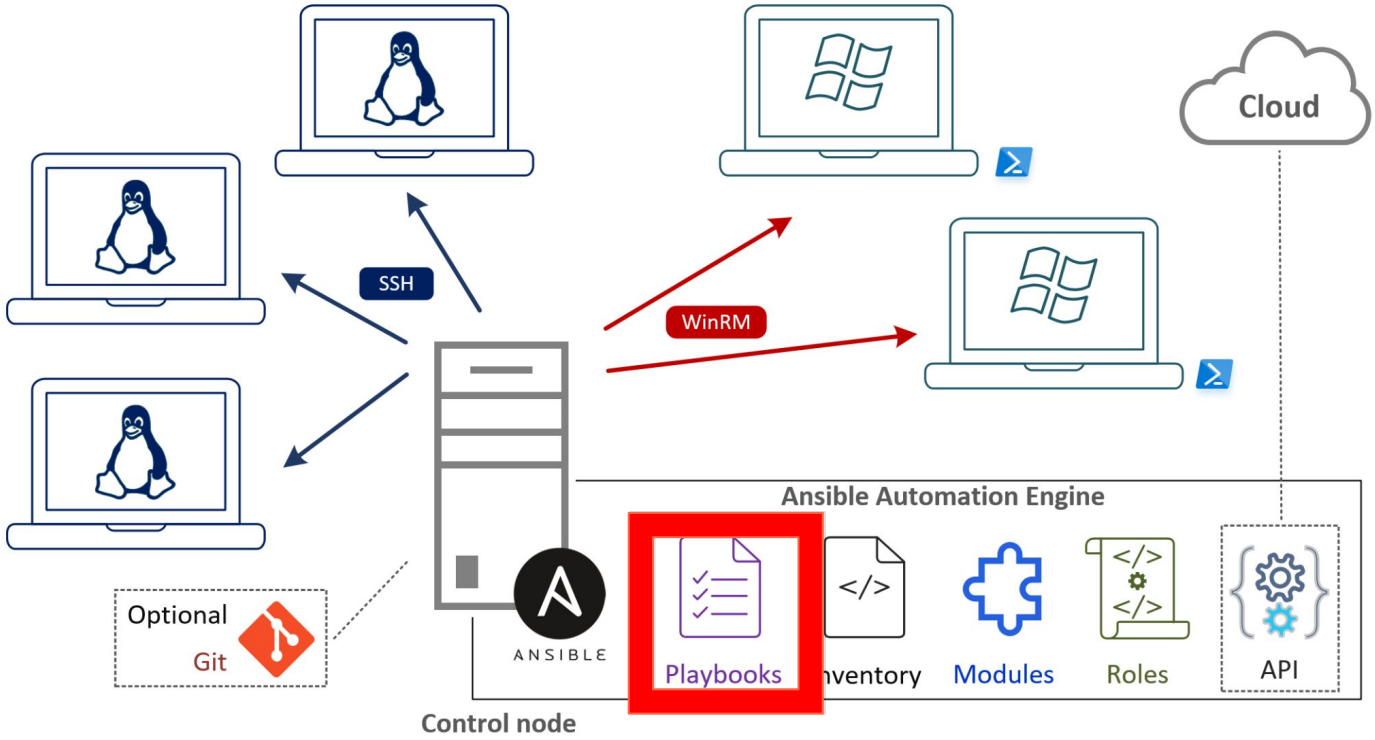
Das Inventory ist eine Beschreibung der Knoten, auf die Ansible zugreifen kann. Standardmäßig wird das Inventory durch eine Konfigurationsdatei beschrieben. Die Konfigurationsdatei listet entweder die IP-Adresse oder den Hostnamen jedes Knotens auf, auf den Ansible zugreifen kann. Jeder Host wird einer Gruppe zugewiesen, wie z.B. Webserver, Datenbankserver usw. Das Inventory kann in verschiedenen Formaten vorliegen, wie YAML, INI usw.

Beispiel Inventory

```
[webservers]
web1 ansible_host=<managed-node-ip1> ansible_user=ubuntu
web2 ansible_host=<managed-node-ip2> ansible_user=ubuntu

[dbservers]
db1 ansible_host=<managed-node-ip3> ansible_user=ubuntu
```

Das Playbook



Das Playbook

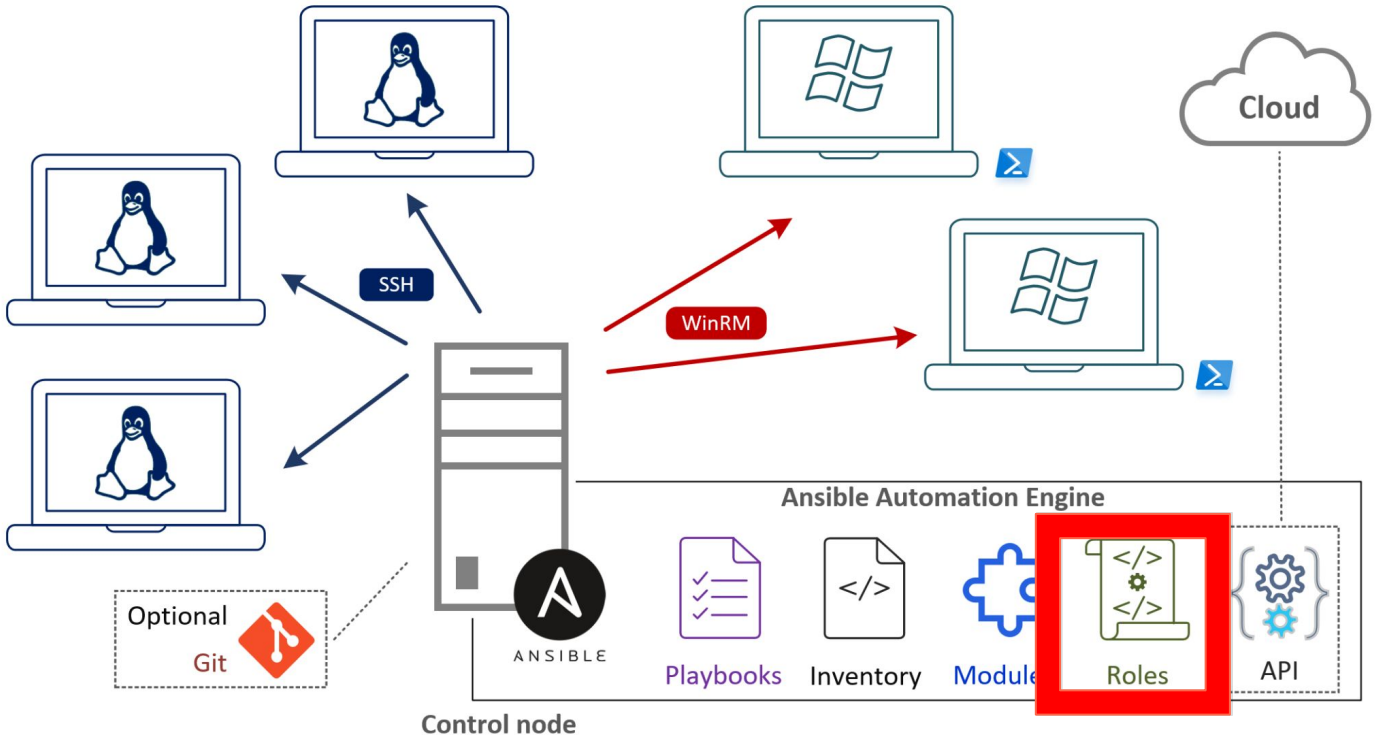
Playbooks sind einfache YAML-Dateien. Diese Dateien beschreiben den gewünschten Zustand Ihrer Systeme. Ansible übernimmt dann die Aufgabe, Ihre Systeme in diesen Zustand zu versetzen, unabhängig davon, in welchem Zustand sie sich derzeit befinden. Playbooks machen Ihre Installationen, Upgrades und die tägliche Verwaltung wiederholbar und zuverlässig.

Playbooks sind einfach zu schreiben und zu pflegen. Sie sind in einer natürlichen Sprache geschrieben, sodass sie leicht weiterentwickelt und bearbeitet werden können. Ein Playbook enthält *Plays*. *Plays* enthalten *Tasks*. *Tasks* rufen *Module* auf.

Beispiel Playbook: nginx installieren

```yaml
- name: Install Nginx on web servers
  hosts: webservers
  become: yes
  tasks:
    - name: Install Nginx
      apt:
        name: nginx
        state: present
```

Rollen


Cloud
SSH
WinRM
Ansible Automation Engine
Optional
Git
ANSIBLE
Playbooks
Inventory
Module
Roles
API
Control node

Rollen

Rollen (*Roles*) sind eine Möglichkeit, Aufgaben in einem Container zusammenzufassen. Wir könnten beispielsweise eine Rolle für die Einrichtung von MySQL haben, eine andere für die Konfiguration von iptables usw. Rollen erleichtern die Konfiguration von Hosts. Jede Rolle kann auf jedem Host oder einer Gruppe von Hosts ausgeführt werden, zum Beispiel:

```
- hosts: all
  roles:
  - role_1
  - role_2
```

Das Playbook im Detail

```yaml
---
# Ein Playbook besteht aus einer oder mehreren Plays.
- name: Name des Plays
  hosts: Zielgruppen-Hosts # Kann 'all' oder eine spezifische Gruppe/Host sein
  become: true # Optional: Wird verwendet, um Aktionen mit erhöhten Rechten auszuführen

  vars: # Optional: Variablen, die innerhalb des Plays verwendet werden
    var_name: Wert

  tasks: # Die Liste der Aufgaben, die ausgeführt werden sollen
    - name: Installation eines Pakets
      ansible.builtin.package:
        name: paketname # Name des zu installierenden Pakets
        state: present # Sorgt dafür, dass das Paket installiert ist (kann auch 'absent'

    - name: Starten und Aktivieren eines Dienstes
      ansible.builtin.service:
        name: dienstname # Name des Dienstes
        state: started # Startet den Dienst (kann auch 'stopped' sein, um ihn zu stoppen
        enabled: yes # Aktiviert den Dienst beim Systemstart (kann auch 'no' sein, um ih

    - name: Eine weitere Aufgabe
      ansible.builtin.another_module:
        key: value
        another_key: another_value
```

Das Playbook im Detail

- `---`: Markiert den Beginn des YAML-Dokuments.
- `- name`: Der Name des Plays, das ausgeführt wird.
- `hosts`: Gibt die Hosts oder Gruppen an, auf denen das Play ausgeführt wird. Diese Hosts werden in der Inventory-Datei definiert.
- `become`: (Optional) Gibt an, ob die Aufgaben als root oder ein anderer Benutzer mit erhöhten Rechten ausgeführt werden sollen.

Das Playbook im Detail

- `vars`: (Optional) Variablen, die innerhalb des Plays verwendet werden können.
- `tasks`: Eine Liste von Aufgaben, die in diesem Play ausgeführt werden sollen. Jede Aufgabe hat:
- `name`: Eine Beschreibung der Aufgabe.
- `ansible.builtin.module_name`: Der Name des Ansible-Moduls, das verwendet wird (z.B. `ansible.builtin.yum`, `ansible.builtin.file`, etc.).
- Parameter: Parameter, die für das Modul spezifisch sind.

Schritte für ein Ansible Deployment

1. Schritt EC2 Instanz(en) erstellen

- Idealerweise mit Terraform

- Geteilten Public Key auf Instanzen bringen
  → ssh-keygen -t rsa -b 2048 : Schlüsselpaar erstellen
  → public key unter .ssh/authorized_keys einfügen

```
ubuntu@ip-10-0-1-132:~$ sudo nano .ssh/authorized_keys
ubuntu@ip-10-0-1-132:~$
```

2. Schritt inventory.ini updaten

- Mit [<hostname-collection>] angeben
- ansible_host: <public-ip>
- Ansible_user: ubuntu username
- ansible_ssh_private_key_file: Pfad zu private key, um auf Instanz zu verbinden
- ansible -i inventory.ini all -m ping: Testet Verbindung

```
[webservers]
web1 ansible_host=18.199.97.21 ansible_user=ubuntu ansible_ssh_private_key_file=../keys/
```

3. Schritt playbook erstellen

```yaml
- name: Install Nginx on web servers
  hosts: webservers
  become: yes
  tasks:
    - name: Install Nginx
      apt:
        name: nginx
        state: present
```

4. Playbook anwenden

```
ansible-playbook -i inventory.ini <playbook>.yml
```

5. Gewünschtes Ergebnis

```
TASK [Install Nginx on webservers] *********************************************
changed: [web1]

PLAY RECAP *********************************************************************
web1                       : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
  ignored=0
```

Zeit für eine Demo

Danke für eure Aufmerksamkeit!