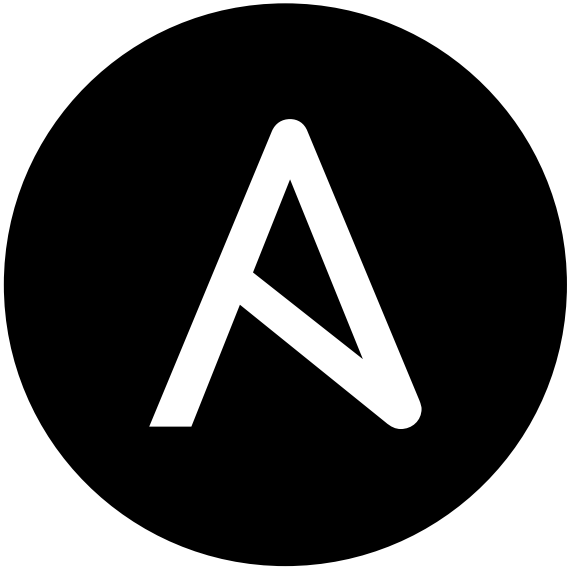


ANSIBLE

vorgetragen von Leon Hutans

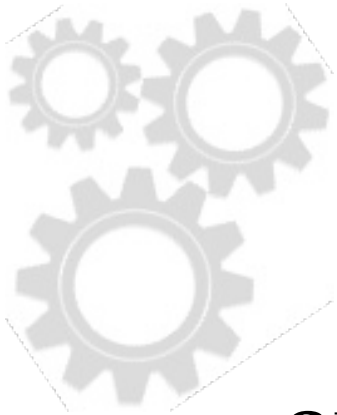
Ausgangspunkt

- **Aufgaben:**
  - Umgebung bereitstellen (Provisioning)
  - Anwendungen ausliefern (Deployment)
  - Instandhaltung der Infrastruktur (Maintenance)
- ursprünglich manuell (von Hand) durchgeführt
- sorgfältige Bearbeitung sehr wichtig

Problemstellung

Wie organisiert man
die Verwaltung mehrerer Server
gut, schnell, einfach und zuverlässig?

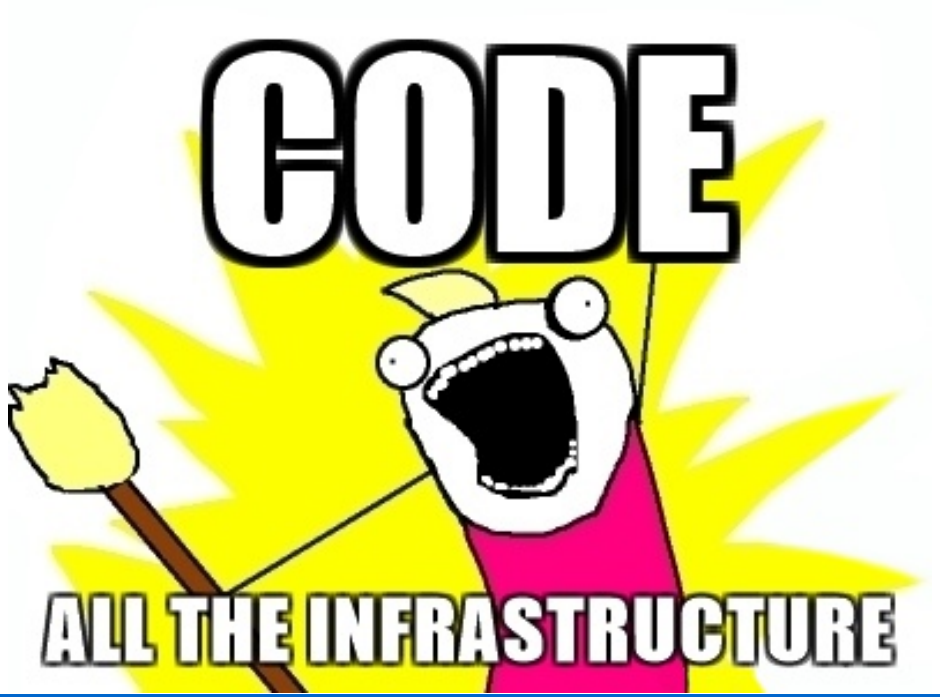
Problemstellung



Können wir diese Arbeit
auf **Maschinen** übertragen
um **Zeit** zu sparen und
menschliche **Fehler** zu reduzieren?

→ **Automatisierung** ←

Wir „programmieren“ die Infrastruktur!



Problemlösung

- **Infrastruktur wie Software behandeln**

  → Nutzung Tools und Prozesse der Software-Entwicklung

  - Version Control
  - Continuous Integration
  - Code Review
  - Automated Testing

Problemlösung

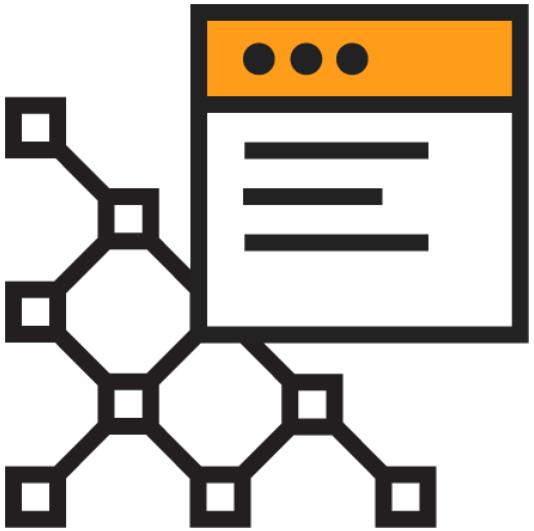
- **Infrastruktur wie Software behandeln**

  → Nutzung Tools und Prozesse der Software-Entwicklung

  - Version Control
  - Continuous Integration
  - Code Review
  - Automated Testing

↓

Infrastructure as Code



Infrastructure as Code (IaC)

- auch bekannt als „programmierbare Infrastruktur“
- ähnelt Skript-Programmierung
- Nutzung High-Level- oder deskriptiver Sprachen
  → komplexe und **adaptive** Prozesse möglich
- **Beschreibung / Definition der Infrastruktur**
  - Software der Server
  - Konfiguration von Servern und Software
  - Kommunikationspartner (andere Server, Services)
  - …

  → bietet die Basis für automatisierte Erstellung

Infrastructure as Code (IaC)

- **Resultat:**
  - Änderungen einfacher, schneller, sicherer, zuverlässiger
  - Wiederverwendbarkeit / Reproduzierbarkeit
    - ermöglicht Qualitäts- und Effizienzsteigerung
    - einfaches Testing auf Basis der „finalen“ Infrastruktur
  - höhere Flexibilität für Entwicklungsteams
  - schnellere Deployments

- Umsetzung von IaC mit **CM-Tools** möglich

Configuration Management (CM)

- ermöglichen die **Umsetzung von IaC**
- großes Spektrum solcher Tools
  - verschiedene Stärken und Schwächen

Configuration Management (CM)

- ermöglichen die **Umsetzung von IaC**
- großes Spektrum solcher Tools
  - verschiedene Stärken und Schwächen
- populärste Vertreter:
  - Puppet
  - Chef
  - CFEngine
  - SaltStack
  - **Ansible**

Warum Ansible?

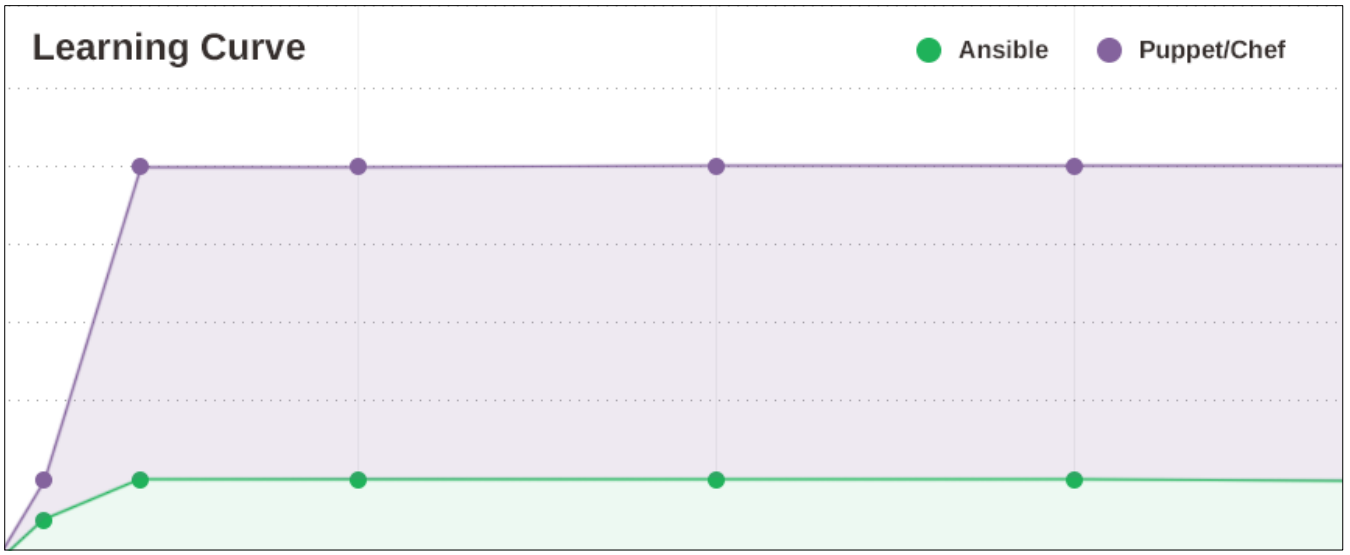
Warum Ansible?

„Directly compared to Chef, Puppet and Saltstack, Ansible is much easier to grasp. You can pick it up and be productive after a couple of hours experimenting and reading through the documentation. This makes Ansible an easy recommendation for small/medium companies without dedicated sys admins or devops.

Martin Rusev
(„What I learned from a year with Ansible“)
(DevOps Consultant aus Berlin)

Warum Ansible?

- Open Source mit guter Dokumentation
- Dokumentation immer auf aktuellen Stand
- schneller und einfacher zu erlernen



https://www.amon.cx/blog/one-year-with-ansible/

Der Unterschied

- **Ansible ist „agentless“**
  - kein zusätzlicher Hintergrundprozess auf Hosts
  - Clients benötigen nur **SSH** und **Python**

Der Unterschied

- **Ansible ist „agentless“**
  - kein zusätzlicher Hintergrundprozess auf Hosts
  - Clients benötigen nur **SSH** und **Python**
- **Ansible ist standardmäßig „push based“**
  - Server „pusht“ Konfigurationsinformation auf Clients
  - Ansible-Nutzer steuert, wann Änderungen geschehen

Der Unterschied

- **Ansible ist „agentless“**
  - kein zusätzlicher Hintergrundprozess auf Hosts
  - Clients benötigen nur **SSH** und **Python**
- **Ansible ist standardmäßig „push based“**
  - Server „pusht“ Konfigurationsinformation auf Clients
  - Ansible-Nutzer steuert, wann Änderungen geschehen
  - **Gegenstück „pull based“** (→ Puppet, Chef, ...)
    - Agent holt sich periodisch Konfiguration vom Server, ermittelt Unterschied, führt Änderungen durch

Weitere Features von Ansible

- **Nutzung von YAML**
  - ursprünglich: „Yet Another Markup Language“
  - steht heute für: „YAML Ain't Markup Language“
  - vereinfachte Auszeichnungssprache
  - „Obermenge“ von JSON

YAML – Grundlegende Syntax

- Zeilenumbrüche und Einrückungen sind wichtig!
- Tabulator-Zeichen sind nicht zulässig
- JSON-Syntax ist ebenso zulässig

  z.B.:

```
map: {“key1“: “value1“, “key2“: “value2“, … }
array: [1, 2, 3, “text“, 5, “and so on“]
```

YAML – Grundlegende Syntax

YAML – Grundlegende Syntax

```
---
# Comment
Key: Value
List:
- Item1
- Item2
Map:
   Key: Value
   Map2:
      Key: Value
...
```

start of a document

end of a document

YAML – Beispiel

```
---
# Ein YAML Beispiel
Kasse: 3
Kassierer: Bob
Datum: 2017-12-08
Zahlungsart: Barzahlung
Produkte:
- Name: Schneeschieber
  Menge: 1
  Preis: 14.99
...
```

Unterstützte Betriebssysteme

- Ansible im Vergleich zu anderen Open Source Tools

	AIX	BSD	HP-UX	Linux	OS X	Solaris	Windows	Others
Ansible	Yes	Yes	Yes	Yes	Yes	Yes	Yes (Need linux control machine)	Yes
Chef	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CFEngine	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Puppet	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Salt	Yes	Yes	Partial	Yes	Yes	Yes	Yes	Partial

Unterstützte Betriebssysteme

- Ansible im Vergleich zu anderen Open Source Tools

	AIX	BSD	HP-UX	Linux	OS X	Solaris	Windows	Others
Ansible	Yes	Yes	Yes	Yes	Yes	Yes	Yes (Need linux control machine)	Yes
Chef	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CFEngine	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Puppet	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Salt	Yes	Yes	Partial	Yes	Yes	Yes	Yes	Partial

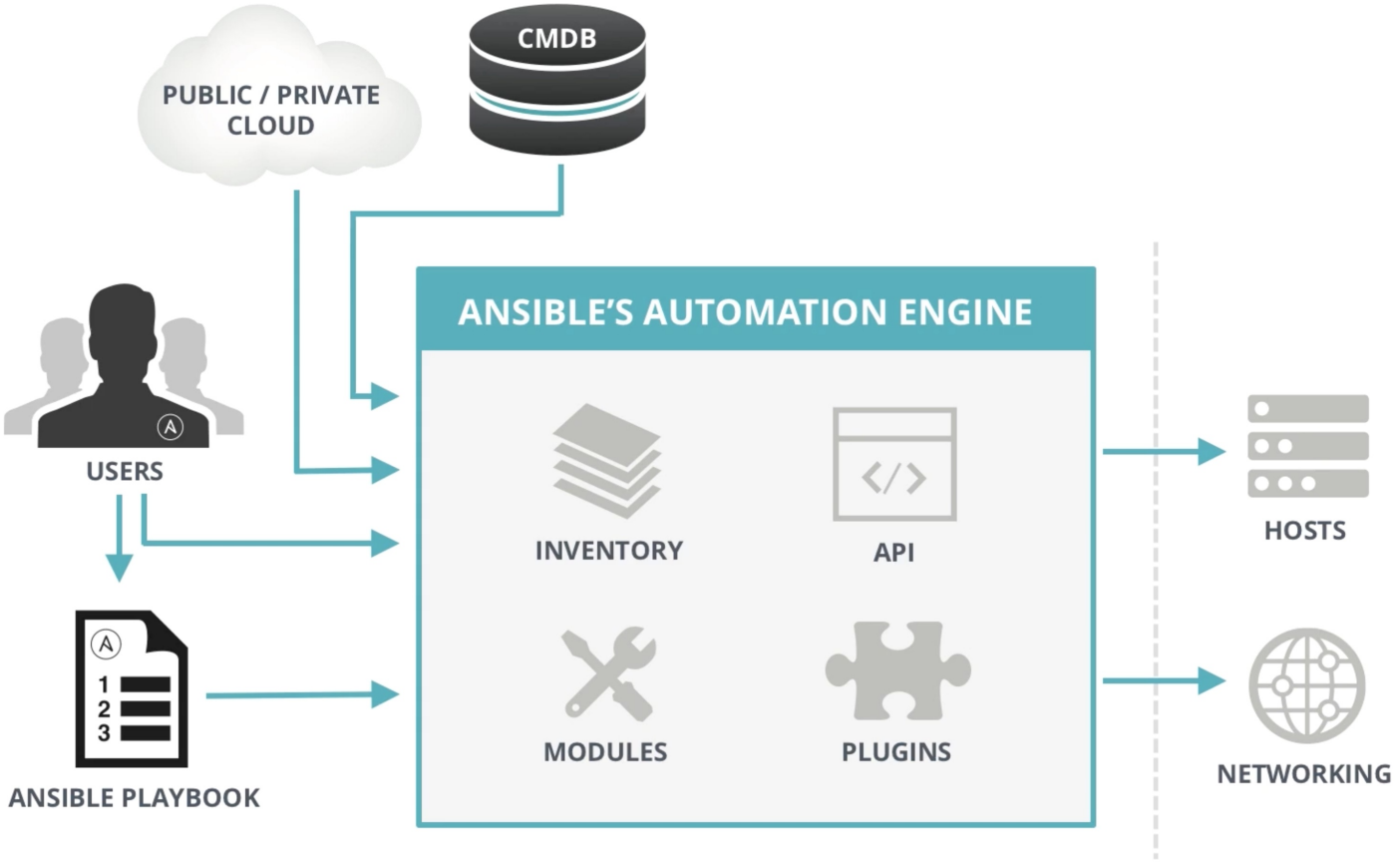
- **Wichtig:**
  - Hosts können mit Windows laufen
  - „Control-Machine“ mit Windows ist nicht unterstützt

    → benötigt wird z.B. Red Hat, Debian, CentOS, OS X, ...

Die Architektur

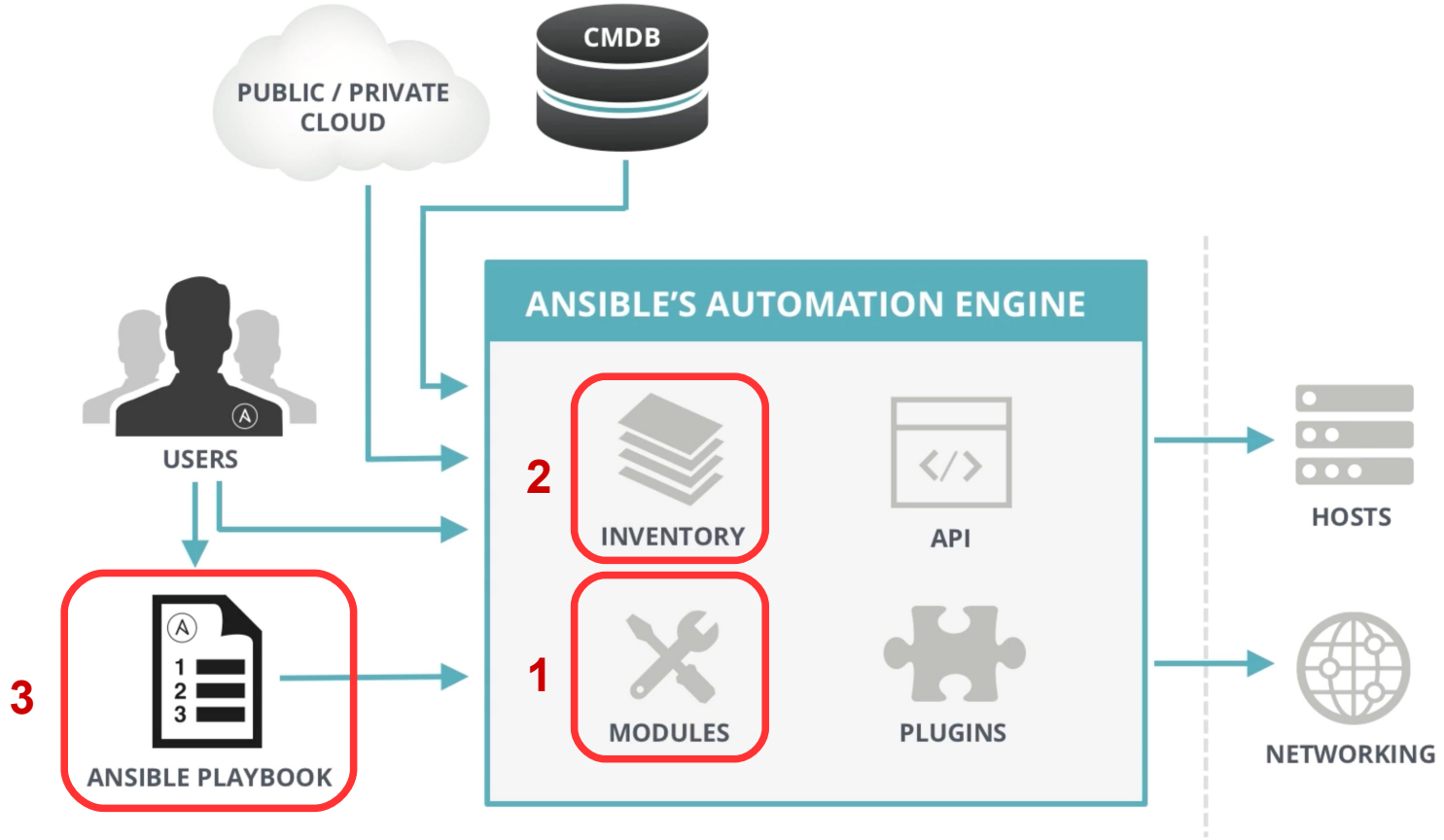
Architektur - Übersicht

- Design-Ziele:
  - minimalistisch, sicher, zuverlässig, leicht erlernbar



Architektur - Übersicht

- Design-Ziele:
  - minimalistisch, sicher, zuverlässig, leicht erlernbar



„Ansible Modules“

- Ansible besitzt „module library“
- kleine **Programme**
- vom Server (Control-Machine) zu Clients (Hosts) „gepusht“
- Ausführung auf Host-Maschine mit anschließender Entfernung
- Beispiele:
  apt , yum , service , template , pip , ping , ...

„Ansible Inventory“

- **Liste von Host-Maschinen** (IP-Adressen...)
- Gruppierung von Hosts möglich
- Format der Inventory-Datei standardmäßig „**INI**“
- mehrere Dateien gleichzeitig nutzbar

„Ansible Inventory“

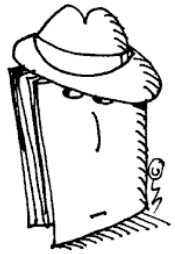
- Beispiel:

```
mail.example.de

[webservers]
webserver1.example.de
webserver2.example.de

[dbservers]
db1.example.org
db2.example.org
```

„Ansible Inventory“

- Weitere Informationen:
  - Standard SSH Ports angenommen
    → bei Abweichung explizit (mit „:“) angeben
  - **Alias**se:

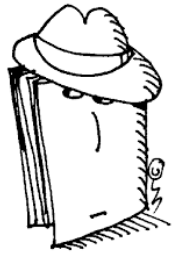
```
jumper  ansible_port=22  ansible_host=192.0.2.50
```



„Ansible Inventory“

- Weitere Informationen:
  - Standard SSH Ports angenommen
    → bei Abweichung explizit (mit „:“) angeben
  - **Aliasse:**

```
jumper  ansible_port=22  ansible_host=192.0.2.50
```



  - **Benutzer und Verbindungstyp:**

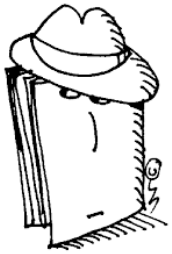
```
localhost          ansible_connection=local
s1.example.com     ansible_connection=ssh    ansible_user=admin
```

    → viele weitere Verbindungstypen (z.B. auch „docker“)

„Ansible Inventory“

- Weitere Informationen:
  - Standard SSH Ports angenommen
    → bei Abweichung explizit (mit „:“) angeben
  - **Aliasse:**

```
jumper  ansible_port=22  ansible_host=192.0.2.50
```



  - **Benutzer und Verbindungstyp:**

```
localhost        ansible_connection=local
s1.example.com   ansible_connection=ssh    ansible_user=admin
```

    → viele weitere Verbindungstypen (z.B. auch „docker“)
  - **Variablen** für Hosts und Host-Gruppen:

```
host1  http_port=80  maxRequestsPerChild=100
```

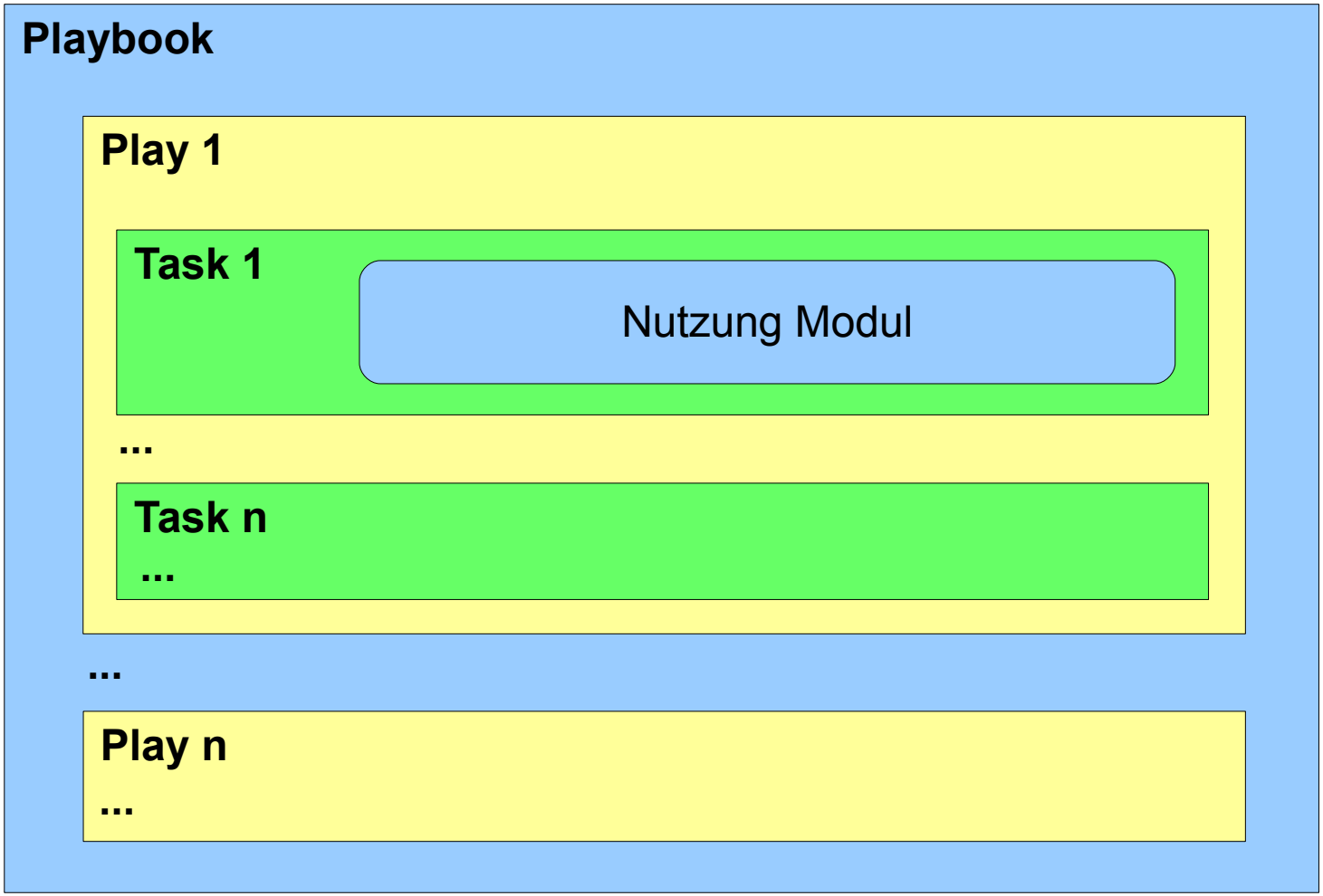
„Ansible Playbooks“

- Ansible's Sprache zur Beschreibung von
  - Konfiguration
  - Deployment
  - Orchestrierung
- Playbook-Format ist **YAML**

- „Ansible Module sind die Werkzeuge,
  Playbooks sind die Bedienungsanleitung,
  Host-Inventory ist das rohe Material“
- bei erneuter Ausführung selbes Ergebnis wie bei erster
  → **Idempotenz**

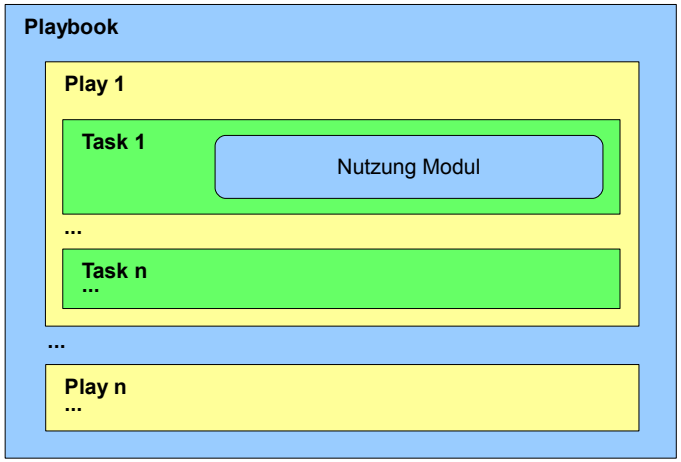
„Ansible Playbooks“

- **Vereinfachte Struktur:**



„Ansible Playbooks“

- **Komponenten:**
  - **Playbook:**
    - besteht aus einem oder mehreren Plays
  - **Play:**
    - soll Gruppe von Hosts eine Reihe von Rollen („**Roles**“) zuweisen → repräsentiert durch „Tasks“
  - **Task:**
    - Aufruf eines Moduls



„Ansible Playbooks“

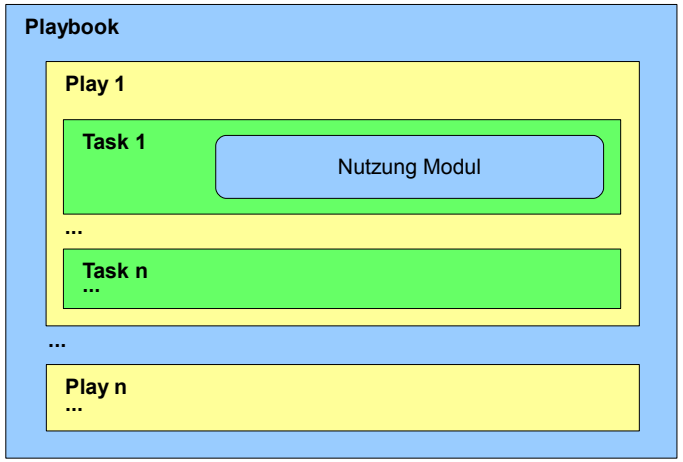
- Beispiel mit „Roles“:

```
---
- hosts: webservers
  roles:
    - common
    - webapp

- hosts: dbservers
  roles:
    - common
    - database
```

„Ansible Playbooks“

- Beispiel mit „Roles“:

```
---
- hosts: webservers
  roles:
    - common
    - webapp

- hosts: dbservers
  roles:
    - common
    - database
```



→ Wo sind die Tasks?

„Ansible Playbooks“

- **Roles:**
  - Variante, um automatisch verschiedene Tasks usw. aus einer bekannten Ordner-Struktur zu laden
  - Beispiel-Struktur:

```
roles/
    common/
        tasks/
        templates/
        vars/
        defaults/
        meta/
    webapp/
        tasks/
        handlers/
        files/
        defaults/
        meta/
    ...
```

Wo sind die Tasks?

„Ansible Playbooks“

- **Roles:**
  - Variante, um automatisch verschiedene Tasks usw. aus einer bekannten Ordner-Struktur zu laden
  - Beispiel-Struktur:

```
roles/
    common/
        tasks/
        templates/
        vars/
        defaults/
        meta/
    webapp/
        tasks/
        handlers/
        files/
        defaults/
        meta/
    ...
```

Konkretes Beispiel: Playbook mit zwei Plays

```yaml
---
- hosts: webservers
  order: inventory

  vars:
    http_port: 80
    config_destination: /etc/httpd.conf

  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum:
      name=httpd
      state=latest

  - name: write the apache config file
    template: src=/srv/httpd.j2 dest={{ config_destination }}
    notify:
    - restart apache

  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes

  handlers:
    - name: restart apache
      service: name=httpd state=restarted


- hosts: databases

  tasks:
  - name: test connection
    ping:
      remote_user: yourname
      become: yes
      become_user: root

  - name: ensure postgresql is at the latest version
    yum: name=postgresql state=latest
    when: ansible_os_family == "CentOS"

  - name: ensure that postgresql is started
    yum: name=postgresql state=started
    when: ansible_os_family == "CentOS"
```

Play 1

```yaml
---
- hosts: webservers
  order: inventory                                          => seit 2.4 – Host-Reihenfolge (z.B. sorted...)

  vars:
    http_port: 80
    config_destination: /etc/httpd.conf

  remote_user: root                                         => Benutzer - seit Ansible 1.4 so (vorher nur "user")

  tasks:
  - name: ensure apache is at the latest version            => Task-Name (gleichzeitig Beschreibung)
    yum:                                                    => zu verwendendes Modul
      name=httpd                                            => Modul-Argumente (auch einzeilig möglich)
      state=latest                                             (hier Verwendung der sog. „Complex Args“-Syntax)

  - name: write the apache config file
    template: src=/srv/httpd.j2 dest={{ config_destination }}   => Jinja2 Template-Datei verarbeiten und auf Host speichern
    notify:                                                 => Aufruf des "handlers"
    - restart apache

  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes           => Nutzung des Moduls „service“

  handlers:                                                 => nur einmal ausgeführte Aktionen durch "notify" getriggert
    - name: restart apache
      service: name=httpd state=restarted
```

Play 2

```yaml
- hosts: databases

  tasks:
  - name: test connection
    ping:
      remote_user: yourname                                 => remote_user auch per Task definierbar
      become: yes                                           => root-Rechte einräumen (Privilege Escalation)
      become_user: root                                     => "root" ist Standard (also hier eigentlich überflüssig)

  - name: ensure postgresql is at the latest version
    yum: name=postgresql state=latest
    when: ansible_os_family == "CentOS"                     => führe nur aus, wenn Host CentOS nutzt (Conditionals)

  - name: ensure that postgresql is started
    yum: name=postgresql state=started
    when: ansible_os_family == "CentOS"
```

„Ansible Playbooks“

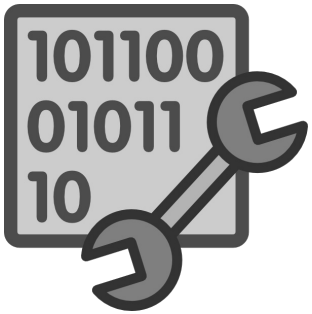
- Playbook ausführen mit:

```
ansible-playbook playbook.yml -f 10
```

(Parallelisierungslevel)

- Parallele Ausführung:
  - **Standard:** für so viele Hosts wie möglich parallel
  - Einschränkung im Playbook mit „serial“ keyword

Ad-Hoc Commands



Ad-Hoc Commands

- ad hoc = „für diesen einen Augenblick/Zweck“
- für schnelle, meist einmalige Aktionen
- **einfacher Befehlsaufbau:**

```
ansible <pattern> -m <module_name> -a <arguments>
```

Ad-Hoc Commands

- ad hoc = „für diesen einen Augenblick/Zweck“
- für schnelle, meist einmalige Aktionen
- **einfacher Befehlsaufbau:**

```
ansible <pattern> -m <module_name> -a <arguments>
```

Ad-Hoc Commands - Patterns

- um die Kommunikationspartner (Hosts) anzugeben
- Adressierung einzelner Hosts oder ganzer Gruppen aus Einträgen der Inventory-Datei
- Beispiel Ad-Hoc Command:

```
ansible webservers -m service -a „name=httpd state=restarted“
```

Ad-Hoc Commands - Patterns

- um die Kommunikationspartner (Hosts) anzugeben
- Adressierung einzelner Hosts oder ganzer Gruppen aus Einträgen der Inventory-Datei
- Beispiel Ad-Hoc Command:

```
ansible webservers -m service -a „name=httpd state=restarted“
```

→ Gruppe = „webservers“

→ Modul = „service“

→ Modul-Argumente: „name=httpd“ , „state=restarted“

Ad-Hoc Commands - Patterns

- Weitere Möglichkeiten:
  - **alle Hosts** adressieren: **all** oder *****
  - **einzelne Hosts**, z.B.: **server1.example.com** oder **192.0.2.81**
  - aus **mehreren Gruppen**: **webservers:dbservers**
    („ : “ = „OR“ → Host ist in Gruppe 1 oder in Gruppe 2)
  - bestimmten **Gruppen ausschließen**: **webservers:!backupservers**
    (Hosts, die in Gruppe 1 sind, aber nicht in Gruppe 2)
  - aus **Schnittmenge** zweier Gruppen: **webservers:&testservers**
    (Hosts, die in Gruppe 1 und ebenso in Gruppe 2 sind)
  - **Kombination der Operationen** ist möglich, z.B.:
    webservers:dbservers:&testservers:!backupservers
    (Hosts aus webservers und dbservers, die auch in testservers sind, jedoch nicht in backupservers)

Ad-Hoc Commands - Patterns

- Weitere Möglichkeiten:
  - **Variablen:** **webservers:!{{excluded}}:&{{required}}**
  - **Wildcards:** **server1*.com:webservers**
  - **Indices:**
    - **webservers[0]** (erster Host der Gruppe)
    - **webservers[-1]** (letzter Host der Gruppe)
    - **webservers[0:2]** (webservers[0], [1] und [2])
    - **webservers[2:]** (webservers[2] und restliche)
  - **Reguläre Ausdrücke:** Anfang mit „ ~ “ symbolisiert

Ad-Hoc Commands - Beispiele

```
ansible weimar -a „/sbin/reboot“
```

→ lasse alle Hosts der Gruppe „weimar“ rebooten

Ad-Hoc Commands - Beispiele

```
ansible weimar -a „/sbin/reboot“
```

→ lasse alle Hosts der Gruppe „weimar“ rebooten

```
ansible weimar -a „/usr/bin/foo“ -u bob
```

→ führe den Befehl als Nutzer „bob“ aus

Ad-Hoc Commands - Beispiele

```
ansible weimar -a „/sbin/reboot“
```
→ lasse alle Hosts der Gruppe „weimar“ rebooten

```
ansible weimar -a „/usr/bin/foo“ -u bob
```
→ führe den Befehl als Nutzer „bob“ aus

```
ansible weimar -m shell -a 'echo Hello World!'
```
→ nutze das „shell“-Modul und übergebe das Argument

Ad-Hoc Commands - Beispiele

```
ansible weimar -a „/sbin/reboot“
```
→ lasse alle Hosts der Gruppe „weimar“ rebooten

```
ansible weimar -a „/usr/bin/foo“ -u bob
```
→ führe den Befehl als Nutzer „bob“ aus

```
ansible weimar -m shell -a 'echo Hello World!'
```
→ nutze das „shell“-Modul und übergebe das Argument

Pakete auf Host-Maschinen verwalten:

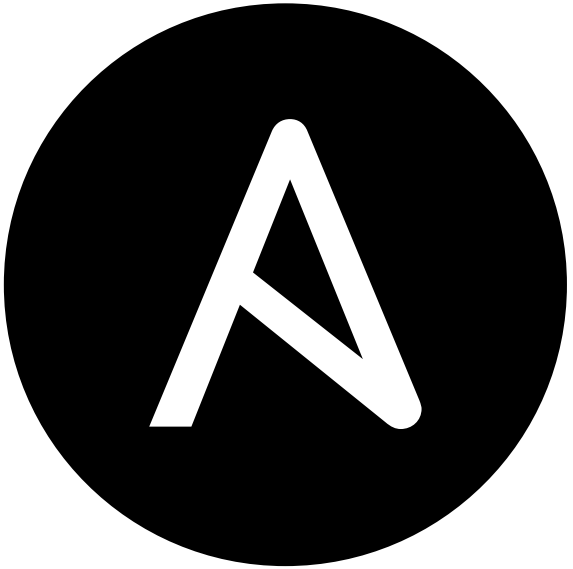
```
ansible webservers -m apt -a “name=foo state=present“
```
→ stellt sicher, dass „foo“ auf den Hosts vorhanden / installiert ist

```
ansible webservers -m apt -a “name=foo state=absent“
```
→ stellt sicher, dass „foo“ nicht auf den Hosts vorhanden ist / deinstalliert wird

Noch nicht genug?

Ansible Galaxy

- offizieller Community-Treffpunkt
- zum **Austausch von „Ansible Roles“**
- Download der verfügbaren „Roles“ mit:
  ```
  ansible-galaxy install --roles-path <path> username.rolename
  ```
- Weitere Informationen unter:
  https://galaxy.ansible.com/intro



ANSIBLE

vorgetragen von Leon Hutans

Quellen und Weiterführende Links

- https://www.upguard.com/articles/the-7-configuration-management-tools-you-need-to-know
- https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software
- https://de.wikipedia.org/wiki/Ansible
- https://puppet.com/solutions/infrastructure-as-code
- http://www.silicon.de/41631054/infrastructure-as-code-programmierung-fuer-administratoren/
- http://www.searchenterprisesoftware.de/definition/Infrastructure-as-Code-IAC
- https://opensource.com/business/16/9/what-are-configuration-management-tools
- https://www.amon.cx/blog/one-year-with-ansible/
- https://www.ansible.com/how-ansible-works
- http://whatis.techtarget.com/definition/agentless
- https://learnxinyminutes.com/docs/de-de/yaml-de/
- http://yaml.org/spec/1.2/spec.html#id2759572
- http://docs.ansible.com/ansible/latest/YAMLSyntax.html
- https://github.com/ansible-semaphore/semaphore
- http://docs.ansible.com/ansible/latest/dev_guide/overview_architecture.html
- http://docs.ansible.com/ansible/latest/modules.html
- http://docs.ansible.com/ansible/latest/modules_by_category.html
- http://docs.ansible.com/ansible/latest/dev_guide/developing_modules.html
- https://docs.ansible.com/ansible/devel/plugins.html
- http://www.paramiko.org/
- http://docs.ansible.com/ansible/latest/intro_inventory.html
- http://docs.ansible.com/ansible/latest/playbooks.html
- http://docs.ansible.com/ansible/latest/playbooks_reuse_roles.html
- http://docs.ansible.com/ansible/latest/become.html
- http://docs.ansible.com/ansible/latest/intro_adhoc.html
- https://galaxy.ansible.com/intro
- http://docs.ansible.com/ansible/latest/glossary.html

Bildquellen

- Ansible Logo
  https://www.ansible.com
- „Handarbeit“
  https://www.1001freedownloads.com/free-cliparts/?tag=hand&page=8
- „Code all the infrastructure“
  https://memegenerator.net/img/instances/500x/44959751/code-all-the-infrastructure.jpg
- Infrastructure as Code - Grafik
  https://puppet.com/sites/default/files/2017-05/WindowInfra.png
- „Reusable“
  http://weclipart.com/gimg/2BC6322336F5D284/recycle-clip-art-recycle-clip-art-6.png
- „Questionmark Man“
  http://cliparting.com/wp-content/uploads/2016/06/Person-thinking-with-question-mark-free-clipart.jpg
- SaltStack Logo
  http://saltstack.com/wp-content/uploads/2014/12/saltStack_horizontal_dark_800x251.png
- CFEngine Logo
  https://cfengine.com/wp-content/uploads/2014/06/logo_with_box.png
- Puppet Logo
  https://puppet.com/themes/hoverboard/images/puppet-logo/puppet-logo-amber-white-lg.png
- Learning Curve Ansible vs Puppet/Chef
  https://www.amon.cx/blog/one-year-with-ansible/
- „Architektur Männchen“
  http://www.elisabethhubert.com/wp-content/uploads/2012/02/BlueArchitect.jpg
- Architektur Ansible
  https://moore3071.github.io/ansible-tutorial/architecture.png
- „Configure data“
  http://www.clker.com/cliparts/8/a/4/a/1194996613360877581kconfigure.svg.hi.png
- „Search“
  http://clipart-library.com/clipart/714589.htm
- „Questioning man“
  https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRuSGHMFNESYPRXt0zQmRc9jewcD0PgtdDirbIhq7Sz1ALUgcJ6bw
- „Package Manager“
  https://cdn.xl.thumbs.canstockphoto.de/3d-k%C3%A4sten-plazierung-postpaket-mann-stock-illustrationen_csp12282056.jpg