Puppet, Salt oder Ansible

Welches Tool ist das Richtige?

Andy Wirtz, Dr. Jonas Trüstedt

27. November 2018

Tätigkeiten im Rechenzentrum

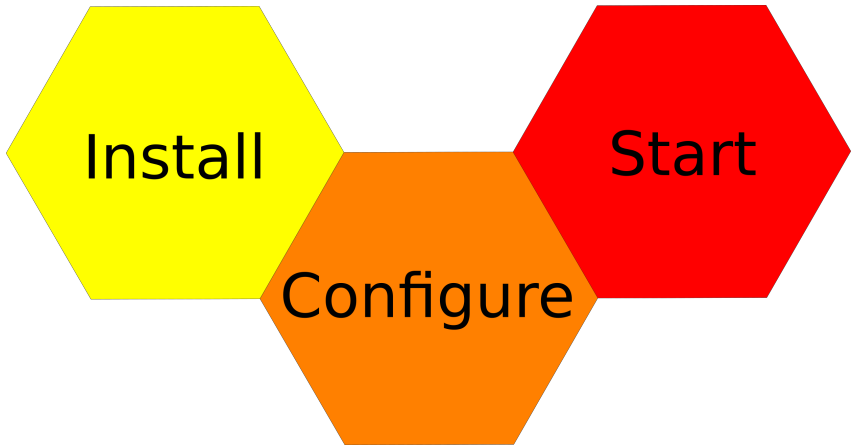
Typische Aufgaben des Systemadministrators:

- einen Host erstellen
- Host mit Applikationen provisionieren
- Applikationen updaten

Konfigurationsmanagement

Eine Applikation muss:

- installiert werden
- konfiguriert werden
- gestartet werden



Beispiel ntp

Verwendung imperativer Shell-Befehle für unterschiedliche Betriebssysteme:

```
yum|apt|zypper install -y ntp

vi /etc/ntp.conf

systemctl start ntpd|ntp|ntpd
systemctl enable ntpd|ntp|ntpd
```

Manuelle Konfiguration

Imperative einzelne Konfiguration:

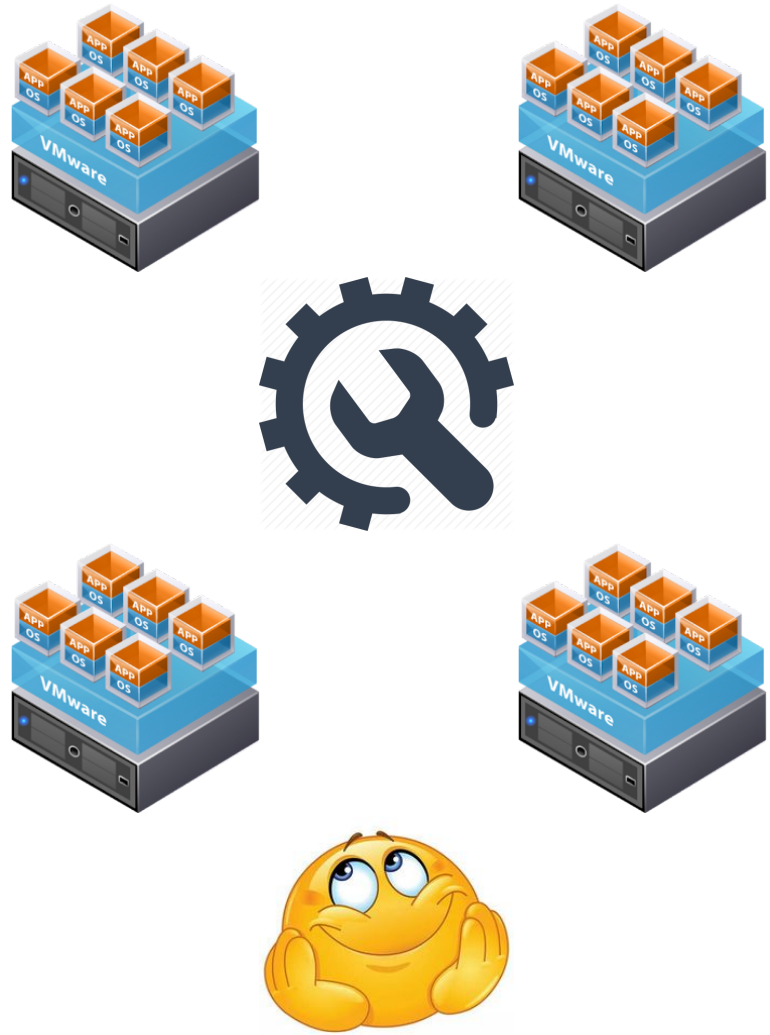
- zu aufwändig
- zu zeitintensiv
- zu fehleranfällig

Die Anzahl Instanzen steigt mit der Zeit, die Anzahl Mitarbeiter nicht

Automatisches Konfigurationsmanagement

Verwendung eines Konfigurationsmanagement-Tools:

- Deklarative Formulierung der Infrastruktur
- Abstraktion der einzelnen Ressourcen
- Infrastruktur einfach skalierbar und reproduzierbar



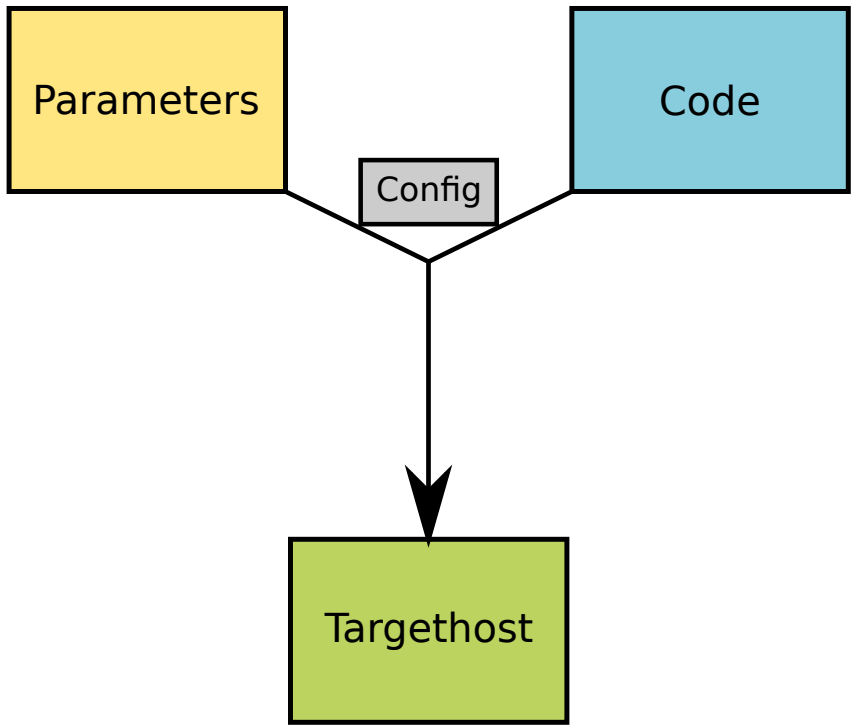
Kernpunkte

- Idempotenz:
  → Mehrmaliges Ausführen möglich
  → Nur bei Änderungen wird angepasst
- Code als Dokumentation:
  → Code lesbar
  → Dokumentation als Code/Parameterset
- Trennung von Parametern und Code:
  → Code mehrfach verwendbar
  → Parameter als spezifische Konfiguration

Vorteile von Konfigurationsmanagement

"Infrastructure as Code":

- Standardisierung
- Automatisierung
- Änderungen an vielen/allen Hosts einfacher
- zentrale Versionskontrolle
- Effizientere Verwaltung der Infrastruktur



Tools

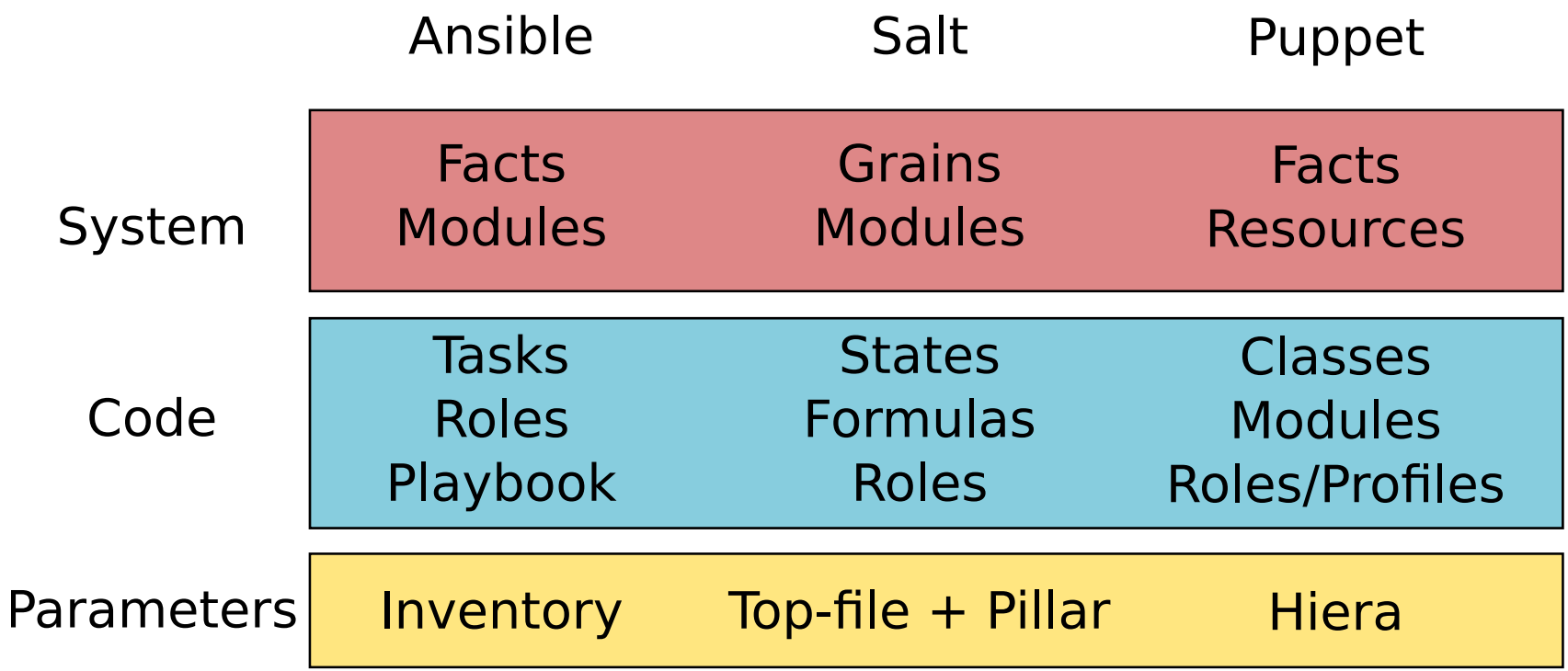
- 

von Red Hat seit 2015, IBM seit 2018:
  - Release: 2012
  - Basis: Python
- 

von puppetlabs:
  - Release: 2005
  - Basis: Ruby
- 

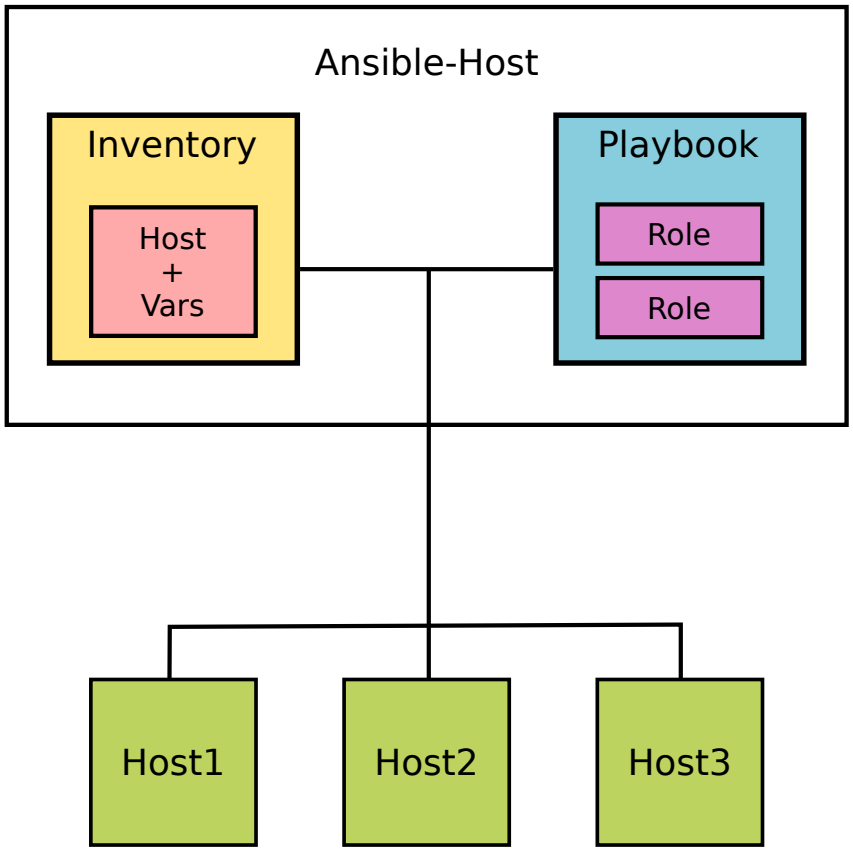
von Saltstack:
  - Release: 2011
  - Basis: Python

Terminologie



Ansible - allgemein

- SSH Verbindung zu Hosts (sshkeys)
- Kein Client nötig
- Abbildung der Infrastruktur in einem Inventar
- Community-roles in Ansible Galaxy



Ansible - Beispiel

ntp.yaml:

```
tasks:
- name: install ntp
  package:
    name: ntp
    state: present

- name: Generate ntp.conf
  template:
    src: ntp.conf.j2
    dest: /etc/ntp.conf

- name: Ensure NTP is
    running
  service:
    name: ntpd
    state: started
    enabled: yes
```

Ausschnitt aus ntp.conf.j2:

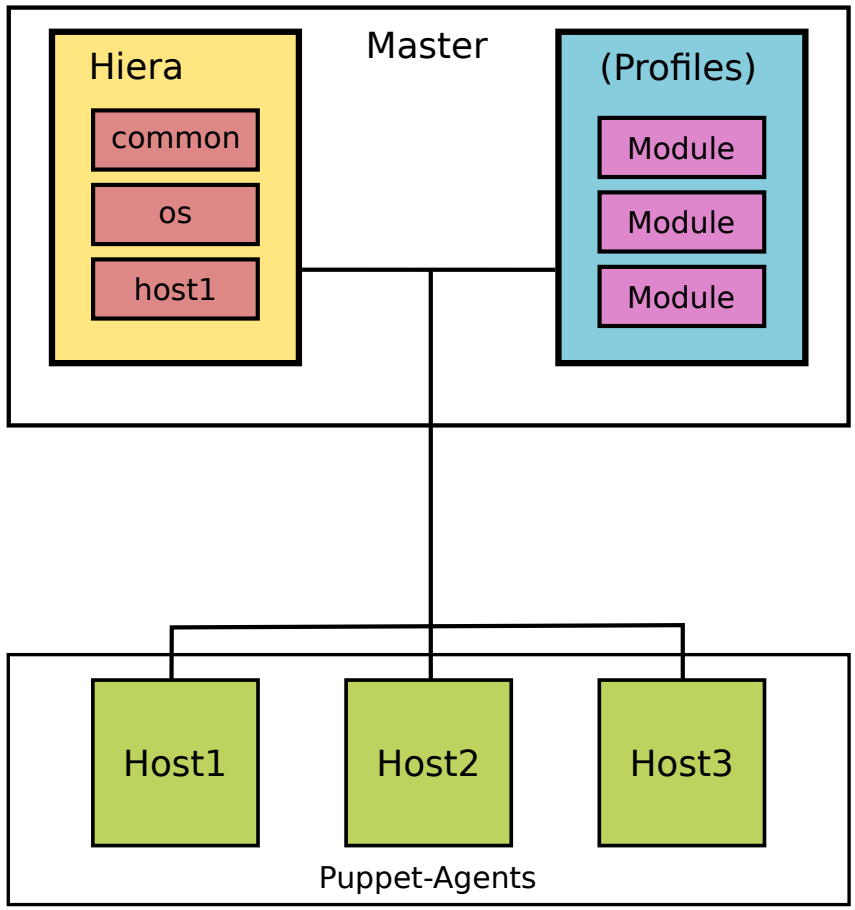
```
[...]
{% for item in ntp_servers
    %}
server {{ item }}
{% endfor %}
[...]
```

Inventory.yaml (ini-file or yaml):

```
base:
  hosts:
    host1.example.com
    host2.example.com
  vars:
    ntp_servers:
      - ntp1.server.org
      - ntp2.server.org
```

Puppet - allgemein

- Clientbasiert
- Periodische Durchläufe (Standard: alle 30 min)
- Community-Module auf Puppet Forge
- Hiera zur Verwaltung von Parametern



Puppet - Beispiel

ntp: init.pp:

```
package{ ntp:
  ensure => latest,
}

file { /etc/ntp.conf:
    ensure  => present,
    content => epp('ntp/ntp.
        conf.epp'),
  }

service{ 'ntpd':
  ensure => running,
  enable => true
}
```

Ausschnitt aus ntp.conf.epp:

```
[...]
<% $ntp::servers.each |
    $server| {-%>
server <%= $server %>
<% } -%>
[...]
```

Puppet - Beispiel

common.yaml:

```
classes:
- ntp

ntp::servers:
  - ntp1.server.org
  - ntp2.server.org
```

host5.example.com.yaml:

```
ntp::servers:
  - ntp1.test.server.com
  - ntp2.test.server.com
```

Hiera-config hiera.yaml:

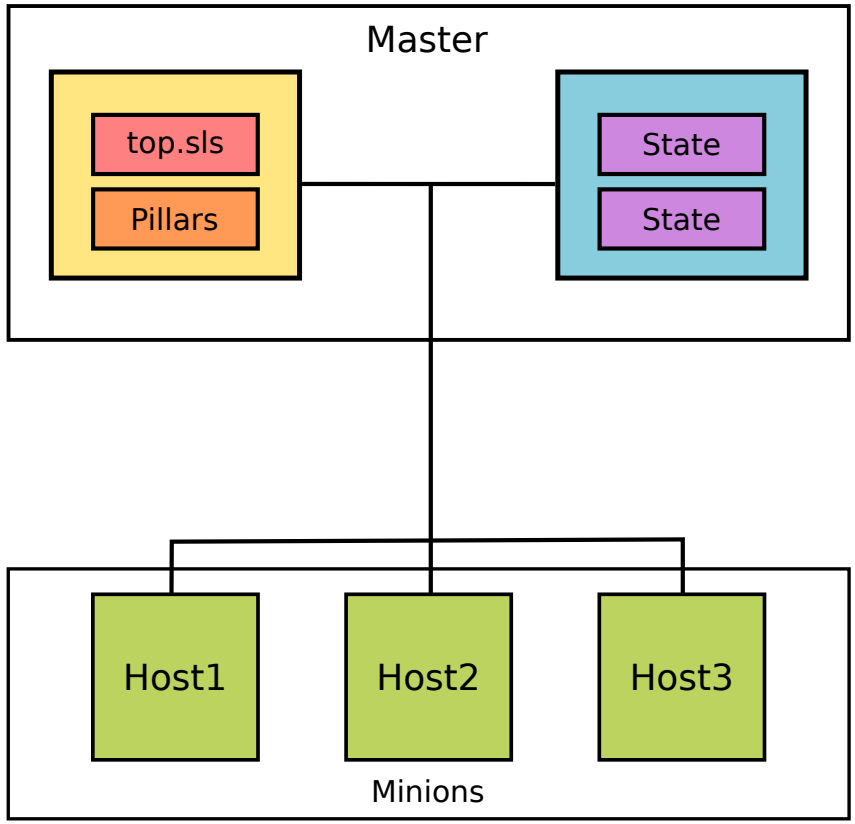
```
hierarchy:
  - name: "Per-node_data"
    path: "nodes/%{trusted.
        certname}.yaml"

  - name: "Per-OS_defaults"
    path: "os/%{facts.os.
        family}.yaml"

  - name: "Common_data"
    path: "common.yaml"
```

Salt - allgemein

- Clientbasiert & SSH-Variante
- Eventbasierte Auslöser (Reactor)
- Periodische Durchläufe möglich
- Infrastruktur aufgeteilt in:
  - Top.sls (wie Inventory)
  - Pillars (ausgelagerte Parameter)
- Wenig Community-Formulas



Salt - Beispiel

ntp.sls:

```
ntp:
  pkg.installed:
    - name: ntp

ntp_conf:
  file.managed:
    - name: /etc/ntp.conf
    - template: jinja
    - source: salt://ntp/ntp
        -client.conf

ntp_running:
  service.running:
    - name: ntpd
    - enable: True
```

Ausschnitt aus ntp-client.conf:

```
[...]
{% set ntpservers = salt['
    pillar.get']('ntp:
    servers') %}
{% for ntpserver in
    ntpservers -%}
server {{ ntpserver }}
{% endfor %}
[...]
```

Salt - Beispiel

top.sls:

```
base:
  '*':
    - ntp
  'webserver':
    - httpd
```

Ausschnitt aus pillar ntp:

```
[...]
ntp:
  servers: ['ntp1.example.
      com','ntp2.example.com
      ','ntp3.example.com']
[...]
```

Salt-SSH mit Roster-File für Targets:

```
Minion1:
  host: host1.example.com
Minion2:
  host: host2.example.com
```

AdHoc-Befehle/Community

Einzelne Befehle direkt auf Hosts ausführen:

- Puppet: Bolt/Tasks
- Ansible: AdHoc-Commands
- Salt: AdHoc-Commands

Fertige Bausteine aus der Community:

- Puppet Forge: ~5800 Module
- Ansible Galaxy: ~17000 Roles
- Salt Formulas (Github): ~300 Formulas

Integration in anderen Produkten

Beispiele:

Als Installer:

- Red Hat Openshift → Ansible
- Suse CaaSP → Salt
- Foreman-Installer → Puppet

Foreman/orcharhino/Satellite:

- Puppet
- Ansible
- Salt

Einsatzmöglichkeiten

- Basiskonfiguration für Infrastruktur:
  - Spezielle Konfigurationen
  - Admin-User inkl. ssh-keys
  - Anbinden an Logging/Monitoring
- Installer für Software/Umgebungen:
  - Serverspezifische Konfigurationen
  - Spezielle Anwendungen
  - Clusterumgebungen wie Kubernetes/Openshift
- Orchestrierung:
  - Containerhosts
  - Anwendungen mit mehreren Instanzen

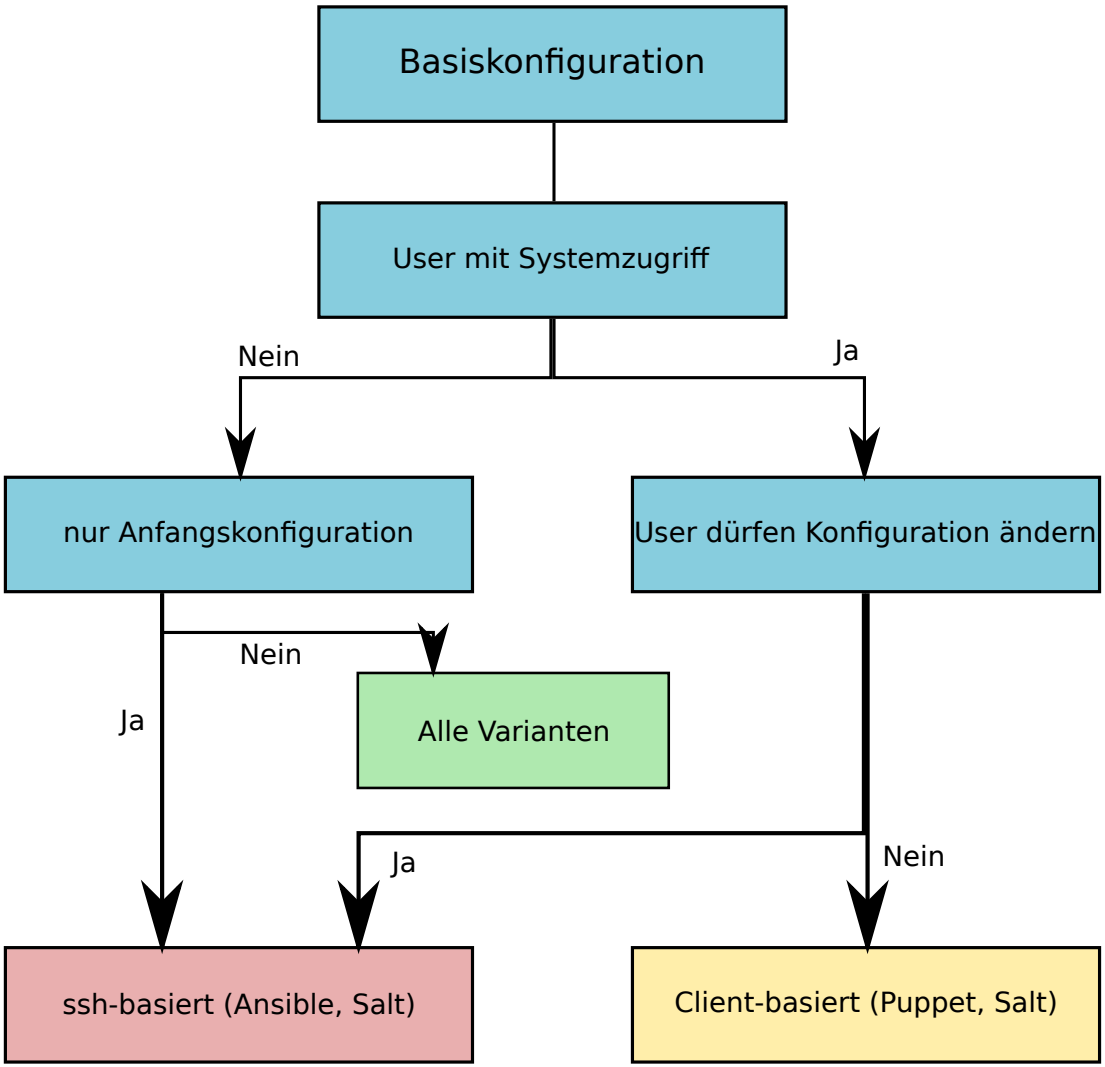
Welches Tool ist das Richtige?

Allgemeine Antwort:

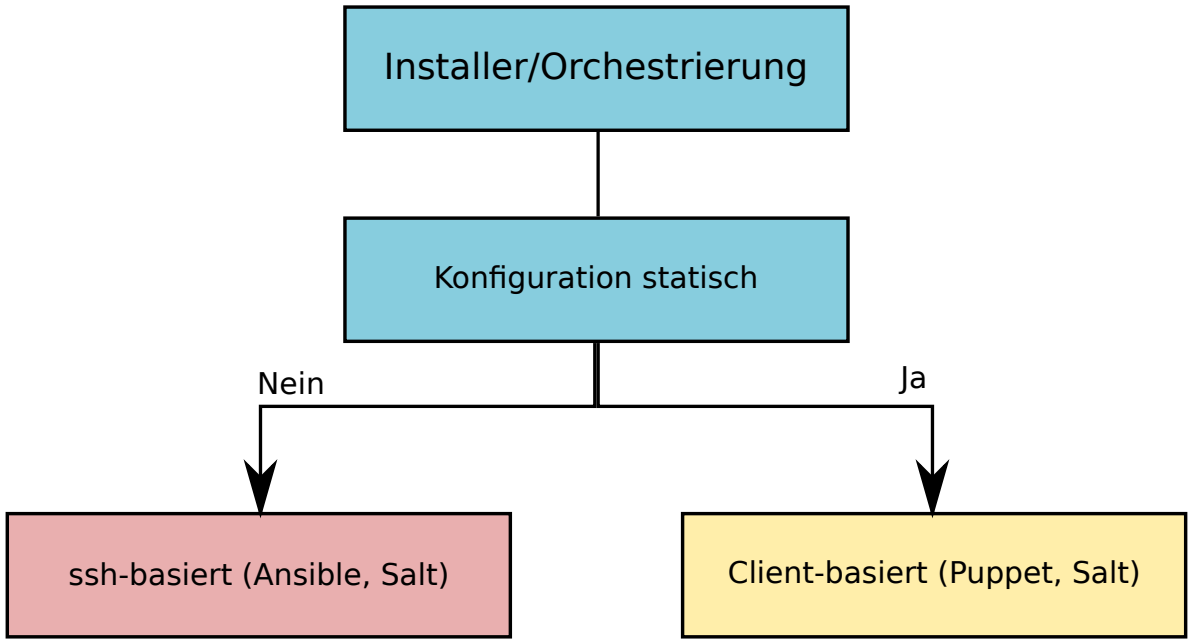
⇒ *"Kommt drauf an"*

⇒ Fast alles kann mit jedem Tool *irgendwie* umgesetzt werden

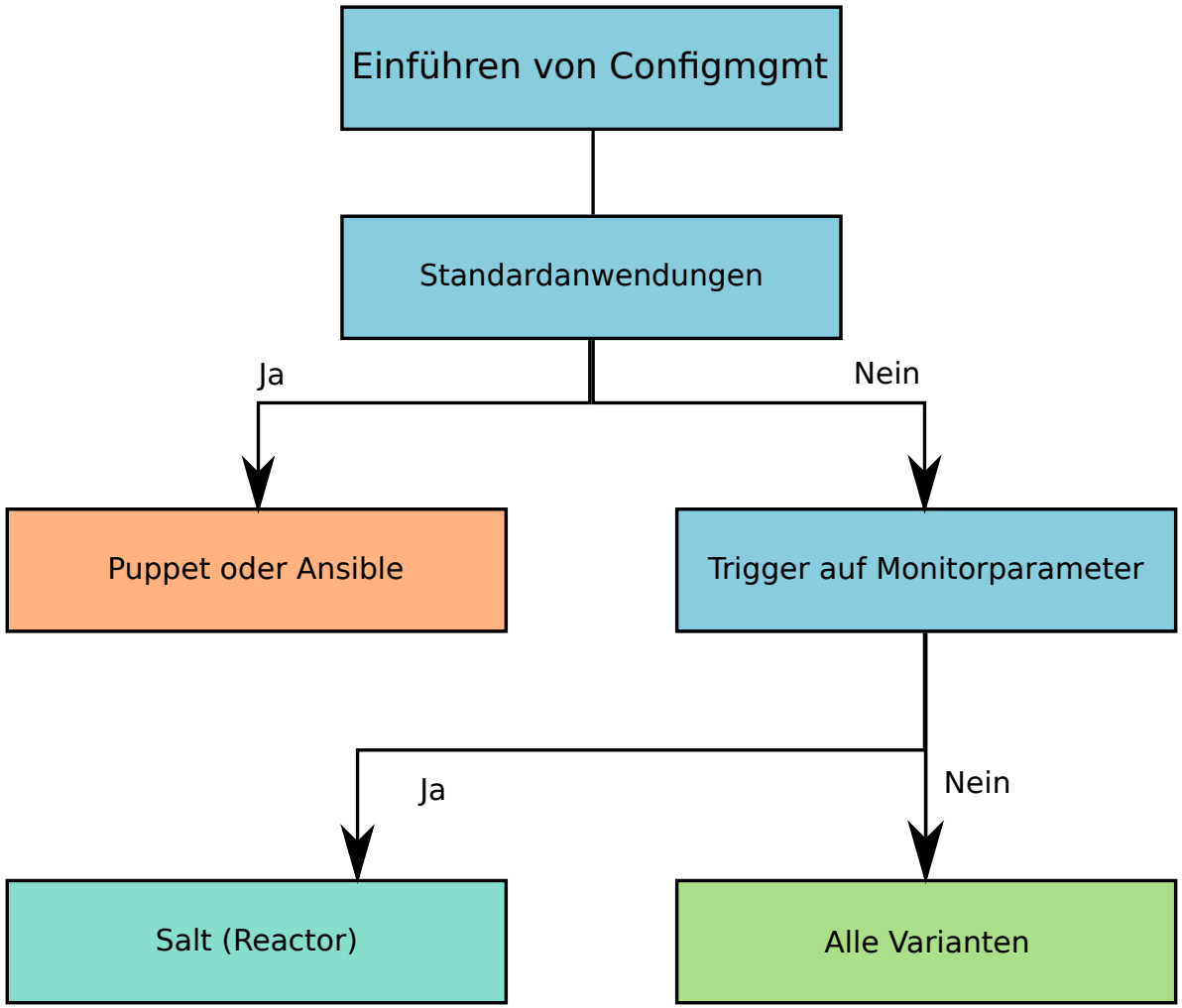
Welches Tool ist das Richtige?



Welches Tool ist das Richtige?



Welches Tool ist das Richtige?



Client-based vs ssh-based

Client-basiert:

- Fokus auf Stabilität/Persistenz
- Infrastructure as Code
- Zentrale Verwaltung
- Höhere Abstraktion
  → Heterogene Systeme

→ Stärken bei persistenten Konfigurationen

ssh-basiert:

- Fokus auf Flexibilität
- Verwaltung je Anwendung/Umgebung
- Verwendung mit Userrechten
- Einfache Remote-Ausführung

→ Stärken bei Orchestrierung/ Deployment

Kombination von Tools

- Kombination unterschiedlicher Tools möglich
- Mehrfache Verwendung von gleichen Client-based Tools
  → Mehrere Puppet-/Saltmaster technisch schwierig
  → Workflows für die zentrale Verwaltung nötig
- Mehrfache Verwendung von gleichen ssh-based Tools
  → In eigenen Inventories möglich

→ Verwaltung pro Abteilung möglich

→ Geeignete Wahl der Tools pro Abteilung/Anwendung möglich

Zusammenfassung

- Arbeitserleichterung durch Configurationmanagement
- Zentrale Verwaltung von Servern/Anwendungen
- Reproduzierbarkeit
- 3 Tools mit vielen Gemeinsamkeiten:
  - Ansible
  - Puppet
  - Saltstack
- Häufig Kombination der Tools
- Wahl des Tools je nach Einsatzzweck

Training/Schulung @ ATIX

puppet

www.atix.de

ANSIBLE

#ATIX #linuxstammtisch #configmgmt #puppet #ansible #saltstack