Konfigurationsmanagement und Deployment mit Ansible

DI (FH) René Koch
Freelancer
Grazer Linuxtage, 29.04.2017

Inhalt

- Was ist Ansible?
- Inventory
- Ad-Hoc Commands
- Playbooks
- Deployment
- Fragen und Antworten

Was ist Ansible?

Was ist Ansible?

- Problem:
  - Verwalten von vielen Systemen ist zeitaufwändig
  - Manuelle Konfiguration von Diensten ist nicht nachvollziehbar und dadurch fehleranfällig
  - Dokumentation fehlt vielfach
  - Effiziente Möglichkeit für das Ausrollen von Anwendungen benötigt

Was ist Ansible?

- Automatisierung der Provisionierung von Systemen, Applikations-Deployment und Konfigurations-Management
- Kein Agent wird am Zielsystem benötigt
- Verwendung von SSH, PowerShell, APIs
- Standardmäßig parallele Abarbeitung der Task auf mehreren Maschinen
- Einfach lesbare Automatisierungssprache (YAML-Format)

Quelle: https://www.openssh.com/

Was ist Ansible?

- Alternative zu:

CFEngine

puppet

CHEF

SALTSTACK

Was ist Ansible?

- Anforderungen (Kontroll-Maschine)
  - Python 2.6 (oder neuer)
  - Python 3 seit Ansible 2.2 experimentell
  - Jede Linux-Distribution, aber kein Windows
- Zu verwaltende Maschinen
  - Python 2.4 (oder neuer)

Quelle: https://www.python.org/

Was ist Ansible?

- Installation
  - RPM (RHEL, CentOS)

```
yum install epel-release
yum install ansible
```

  - RPM (Fedora)

```
dnf install ansible
```

  - DEB

```
echo „deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty
main“ >> /etc/apt/sources.list
apt-get update
apt-get install ansible
```

  - PIP

```
pip install ansible
```

  - Zusätzliche Pakete können auf der Kontroll- oder Ziel-Maschine benötigt werden (z.B. Python-MySQL-Module)

Inventory

Inventory

- Mit Ansible verwaltete Hosts werden in einem Inventory gepflegt
- Legt fest wie Ansible mit den Hosts kommunizieren soll
- Definition von Gruppen
- ini-Format
- 2 Arten:
  - Statische Inventories
  - Dynamische Inventories

Quelle: http://www.wendia.com/

Inventory

```
myhost.example.com
app.example.com
```

Hostnamen für die Verbindung

```
[webserver]
web01.example.com
web02.example.com
web03.example.com
```

Gruppe

```
[dbserver]
wb01.example.com
wb02.example.com
```

Inventory

```
[development]
dev01.example.com
dev02.example.com
dev03.example.com ansible_ssh_host=192.168.122.3
```

Alternative Verbindungsparameter

```
[gce:vars]
ansible_user=gce-user
```

Variablen

```
[gce:children]
webserver
dbserver
```

Gruppe von Gruppen

```
[webserver]
web[01:10].example.com
```

Regex

```
[dbserver]
db[a:c].example.com
```

Inventory

```
$ inventory/gce.py --list --pretty
{
  "_meta": {
    "hostvars": {
      "instance-1": {
        "ansible_ssh_host": "35.187.121.39",
        "gce_description": "",
        "gce_id": "9041038088844006101",
        "gce_image": "centos-7-v20170327",
        "gce_machine_type": "f1-micro",
        "gce_metadata": {},
        "gce_name": "instance-1",
        "gce_network": "default",
        "gce_private_ip": "10.132.0.2",
        "gce_public_ip": "35.187.121.39",
        "gce_status": "RUNNING",
        "gce_tags": [
          "http-server",
          "https-server"
        ],
        "gce_uuid":
"7f15be332330ff61f3f236d69be61928d1afbb4e",
        "gce_zone": "europe-west1-d"
```


Ad-Hoc Commands

Ad-Hoc Commands

- Kommando (korrekterweise ein Modul) auf einem oder mehreren Hosts ausführen
- Vorteil:
  - Einfach und schnell
  - Benötigt keine Playbooks
  - „Kannst du mal schnell … auf … Systemen?“
- Nachteil:
  - Nicht wiederverwendbar (bash-History ersetzt kein Playbook!)
  - Nur einfache Tasks möglich

Quelle: http://www.programmingbasics.org

Ad-Hoc Commands

```
ansible [pattern] -m [module] -a
„[options]“ [flags]
```

- Pattern: Host-Filter
- Module: Ansible Modul (siehe https://docs.ansible.com/ansible/list_of_all_modules.html)
- Options: Optionen für das Ansible Modul
- Flags: zusätzliche Verbindungs-Optionen
  - -i [inventory]: Inventory-Datei
  - -b: sudo verwenden
  - -u [user]: Benutzername für die Verbindung
  - Siehe `man ansible` für weitere Optionen

Ad-Hoc Commands

- Uptime

```
$ ansible testserver -m command -a uptime
testserver | SUCCESS | rc=0 >>
 08:58:17 up 44 min,  1 user,  load average: 0.00, 0.02, 0.0
```

- Logs betrachten

```
$ ansible testserver -a "tail -1 /var/log/messages" -b
testserver | SUCCESS | rc=0 >>
Jan 27 09:01:01 localhost systemd: Starting user-0.slice.
```

- Paket installieren

```
$ ansible testserver -m yum -a "name=httpd state=installed" -b
```

- System-Informationen auslesen

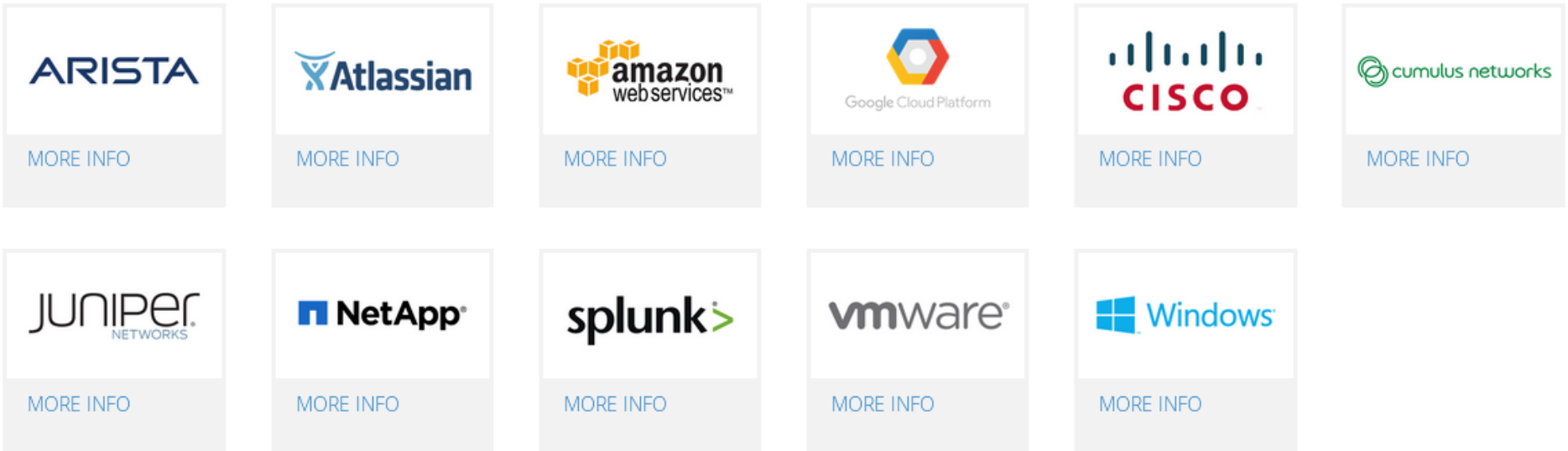
```
$ ansible testserver -m setup
```

Module

https://docs.ansible.com/ansible/list_of_all_modules.html

- yum/apt/zypper/...
- service
- template
- user
- file
- copy

1.034 Module in Ansible 2.3 inkludiert

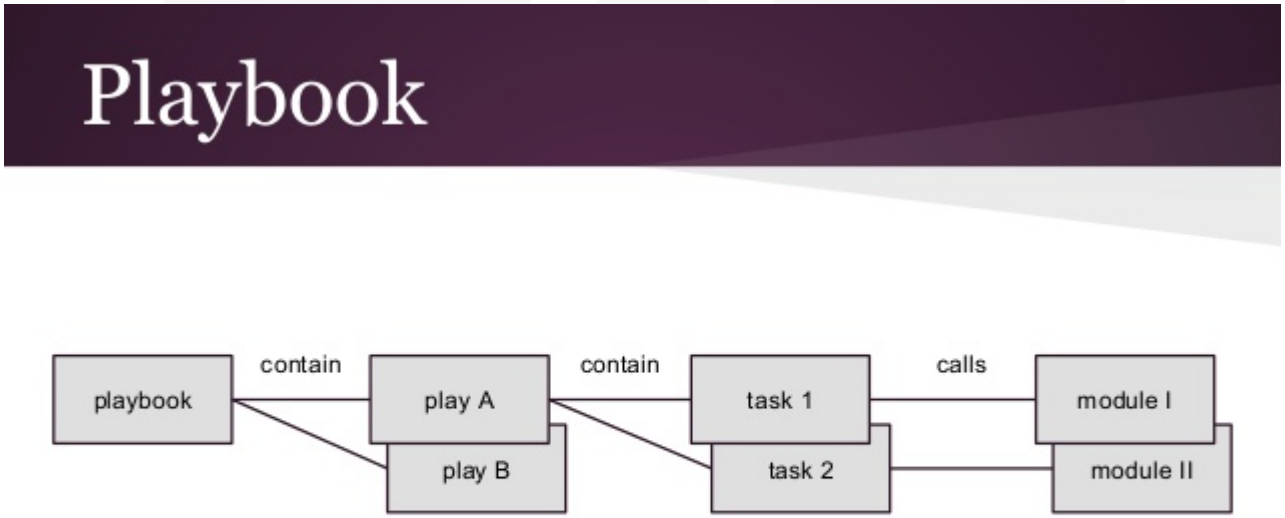


Quelle: https://www.ansible.com/how-ansible-works

Playbooks

Playbooks

- Wiederverwendbare Sammlung von Plays
- Plays beinhalten Tasks, welche Module aufrufen



Quelle: https://image.slidesharecdn.com/ansible-150925121447-lva1-app6891/95/ansible-101-16-638.jpg?cb=1443183393

Playbooks

```
---

- name: Configure nginx webserver
  hosts: webservers
  become: true
  tasks:
    - name: Enable epel repository
      yum:
        name: epel-release
        state: present

    - name: Install nginx
      yum:
        name: nginx
        state: present

    - name: Copy static.html
      copy:
        src: files/static.html
        dest: /usr/share/nginx/html/static.html
```

Playbooks

```
ansible-playbook [options]
playbook.yml
```

- Options:
  - -e: extra Variablen
  - -i [inventory]: Name des Inventories
  - -f [forks]: Anzahl der parallel abzuarbeitenden Hosts
  - Siehe `man ansible-playbook` für weitere Optionen
- Playbook: Name des Playbooks

Playbooks

```
$ ansible-playbook web-notls.yml

PLAY [Configure nginx webserver] ***********************************************

TASK [setup] *******************************************************************
ok: [instance-1]

TASK [Enable epel repository] **************************************************
ok: [instance-1]

TASK [Install nginx] ***********************************************************
changed: [instance-1]

TASK [Copy index.html] *********************************************************
changed: [instance-1]

TASK [Start and enable nginx] **************************************************
changed: [instance-1]


PLAY RECAP *********************************************************************
instance-1                 : ok=9    changed=5    unreachable=0    failed=0
```

Playbooks

- Tasks werden in der Reihenfolge wie im Play angegeben ausgeführt
- Tasks können inkludiert werden

```
tasks:
  - include: tasks/{{ ansible_distribution }}.yml
  - include: tasks/common.yml
```

- Schlägt ein Task fehl, laufen für diesen Host keine weiteren mehr
- Module sind idempotent um einen gewissen Status zu gewährleisten (im Gegensatz zu Shell-Skripten)

Playbooks

- Handler sind spezielle Tasks, die einmalig am Ende des Playbook-Laufs aufgerufen werden

```
tasks:
  - name: Copy nginx https vhost config
    template:
       src: templates/https.conf.j2
       dest: /etc/nginx/conf.d/https.conf
       mode: 0644
    notify: Restart nginx

  - name: Some more tasks
    ...

handlers:
  - name: Restart nginx
    service:
      name: nginx
      state: restarted
```

Playbooks

- Templates sind dynamisch generierte Dateien, welche Variablen auflösen

```
- name: Configure nginx webserver
  hosts: webservers
  become: true
  vars:
    nginx_key: localhost.key
    nginx_cert: localhost.c
  tasks:
    - name: Copy nginx https
      template:
         src: templates/htt
         dest: /etc/nginx/c
         mode: 0644
      notify: Restart nginx
```

```
# Settings for a TLS enabled server.
server {
      listen       443 ssl http2 default_server;
      listen       [::]:443 ssl http2 default_server;
      server_name  _;
      root         /usr/share/nginx/html;
      ssl_certificate "/etc/pki/nginx/{{ nginx_cert }}";
      ssl_certificate_key "/etc/pki/nginx/{{ nginx_key }}";
      ssl_session_cache shared:SSL:1m;
      ssl_session_timeout  10m;
      ssl_ciphers HIGH:!aNULL:!MD5;
      ssl_prefer_server_ciphers on;
      location / {
      }
}
```

Playbooks

- Templates werden mittels Jinja2 gerendert:
  http://jinja.pocoo.org/docs/2.9/
- In der Regel nur wenige Jinja2 Features nötig
- z.B. Schleifen und Abfragen möglich

```
{% for server in ntp_servers %}
server {{ server }} iburst
{% endfor %}
```

Playbooks

- Schleifen in Ansible Playbooks (meistens) mittels with_items

```
- name: Common tasks
  hosts: all
  become: true

  vars:
    admin_tools:
      - telnet
      - nmap
      - net-tools
      - bind-utils
      - tcpdump

  tasks:
    - name: Install admin tools
      yum:
        name: "{{ item }}"
        state: installed
      with_items: "{{ admin_tools }}"
```

Playbooks

- When-Abfragen in Playbooks

```
- name: Install "{{ webserver.name }}"
  yum:
    name: "{{ webserver.name }}"
    state: present

- name: Deploy httpd.conf
  template:
    src: templates/httpd.conf.j2
    dest: /etc/httpd/conf/httpd.conf
    owner: root
    group: root
    mode: 0644
  notify: Restart apache
  when: webserver.name == "httpd"
```

Playbooks

- Rollen dienen dazu Playbooks übersichtlicher zu machen
- Rollen sind in anderen Playbooks wiederverwendbar
- Rollen beinhalten Variablen, Tasks, Handler, Files, Templates, Defaults und Meta-Daten
- Fertige Rollen gibt es auf https://galaxy.ansible.com/ zum Downloaden

Playbooks

- Einbinden von Rollen anstelle von Tasks

```
---

- name: Deploy common tasks
  hosts:
    - "{{ hosts }}"
  become: True
  roles:
    - register-satellite6
    - common
    - { role: icinga2, when: icinga2_manage == true }
    - { role: bareos, when: bareos_manage == true }
```

```
roles/common/
├── defaults
│   └── main.yml
├── files
│   ├── aliases
│   └── bashrc
├── handlers
│   └── main.yml
├── meta
├── tasks
│   └── main.yml
├── templates
│   └── ntp.conf.j2
└── vars
```

Deployment

Deployment

- Ansible erlaubt nicht nur Konfigurations-Management sondern auch Deployen von VMs und Anwendungen
  - Amazon EC2
  - Azure
  - Google
  - Openstack
  - VMware
  - und viele mehr

Quelle: https://www.brightwork.com

Deployment

- Beispiel Google Compute Engine

```
---

- name: Create instances
  hosts: localhost
  connection: local
  gather_facts: no

  tasks:
    - name: Create instance
      gce:
        instance_names: "{{ gce_instance_names }}"
        machine_type: "{{ gce_machine_type }}"
        image: "{{ gce_image }}"
        service_account_email: "{{ gce_service_account_email }}"
        credentials_file: "{{ gce_credentials_file }}"
        project_id: "{{ gce_project_id }}"
        zone: "{{ gce_zone_id }}"
```

Deployment

- Beispiel Wordpress

```
- name: Download wordpress
  get_url:
    url: https://wordpress.org/latest.zip
    dest: "{{ wordpress.webroot }}/latest.zip"
    mode: 0644

- name: Download and extract wordpress package
  unarchive:
    src: https://wordpress.org/latest.zip
    dest: "{{ wordpress.webroot }}"
    remote_src: yes

- name: Get uniq keys for wordpress
  command: curl https://api.wordpress.org/secret-key/1.1/salt/
  register: securekeys

- name: Configure wordpress
  template:
    src: templates/wp-config.php.j2
    dest: "{{ wordpress.webroot }}/wordpress/wp-config.php"
    mode: 0644
```

Fragen und Antworten

DI (FH) René Koch
Freelancer
Grazer Linuxtage, 29.04.2017