Use case: Cloudstack + Ansible
About Me

- Sebastian Bretschneider
- Since 2011 System Engineer at BIT.Group GmbH – member of intelligence group
- Cloudstack
- Ansible
- Ceph
- Linux
- Infrastructure

CloudStack Berlin & Dresden, Germany
https://www.meetup.com/german-CloudStack-user-group

Ansible Dresden, Germany
https://www.meetup.com/Ansible-Dresden
Overview BIT.Group GmbH – member of itelligence group

- 350+ employees in Dresden, Bautzen, Hanover and Shanghai
- SAP Consulting, Development and Support
- SAP partner and service provider for SAP SE
BIT.Group GmbH as part of itelligence / NTT DATA Group

- Since June 2016 BIT.Group GmbH officially part of itelligence and NTT DATA Group
- Know-how, flexibility and internationality as part of NTT DATA network

- Together internationally leading full IT service provider with:
  - 3,500+ active SAP customers
  - Locations in 40+ countries
  - $1.5 billion in SAP revenue worldwide
  - Over 9,000 SAP experts worldwide
Content

1. Challenge
2. Environment
3. Ansible
4. Examples
5. Parts
6. Big Picture
7. Results
Challenge

- User friendly interface with all sub-services
- All operations changeable by admins
- Make cloud-consistent configuration
- Use open source
- Scalability
- Modular design
- Simple
Environment

- Hypervisor
  - KVM

- Automation
  - Ansible

- Storage
  - Ceph
  - NFS

- Network
  - VLAN
  - Shared networks
  - Isolated networks

- BIT.Cloud
  - Portal
  - Ansible Daemon
What is Ansible

Simple
- Get productive quickly
- Human readable automation
- No special coding skills required
- Tasks executed in order

Powerful
- Orchestrate the app lifecycle
- App deployment
- Configuration management
- Workflow orchestration

Agentless
- Predictable, reliable and secure
- Agentless architecture
- Use Open SSH & WinRM
- No agents to exploit or update
Ansible Modules

- Cloudstack Modules
  - cs_facts – Gather facts on instances
  - cs_firewall – Manages firewall rules
  - cs_ip_address – Manages public IP address associations
  - cs_volume – Manages volumes
  - ...
  - cs_domain
  - cs_cluster
  - ...

- 36 Cloudstack modules overall (April 2017)
- Development/Improvements by BIT.Group
Get it working

- **Python module**
  - `pip install cs`
  - `apt-get install python-cs`

- **Credentials File**
  - `.cloudstack.ini → home directory`
  - `CLOUDSTACK_CONFIG → environment variable pointing to .ini file`
  - `Cloudstack.ini → working directory`

- **cloudstack.ini**

  [cloudstack] # global or region – passed by arg api_region
  
  endpoint = https://cloudstack-management.server/client/api
  
  key = api key
  
  secret = api secret
  
  Timeout = 60
Workflow

A → cs_instance → CloudStack
Workflow

configure

cloudstack
Examples: CloudStack VM

- **Playbook**
  - hosts: localhost
    - roles:
      - cs-vm

- **Role: tasks/main.yml**
  - block:
    - include: tasks/create_vm.yml
    - include: tasks/add_ansible_host.yml
    - include: tasks/ssh_key_rollout.yml
      when: vm_action == "create"

[ ... ]
Examples: CloudStack VM

- **create_vm.yml**
  
  name: Creating Virtual Machine

  local_action:

  module: cs_instance

  name: "{{ vm_name }}"
  template: "{{ os_template }}"
  hypervisor: "{{ cs_hypervisor }}"
  project: "{{ cs_project }}"
  zone: "{{ cs_zone }}"
  service_offering: "{{ cs_service_offering }}"
  networks: "{{ cs_networks }}"
  domain: "{{ cs_domain }}"
  tags:
    - { key: CostCenter, value: "{{ vm_costcenter }}" }

  state: started

  register: cs_vm  #get information about VM e.g. root password
Examples: Port Forwarding

- **add_portforwarding.yml**
  - name: Forwarding Ports in CloudStack

    ```yaml
    local_action:
      module: cs_portforward
      domain: "{{ cs_domain }}"
      project: "{{ cs_project }}"
      ip_address: "{{ cs_public_ipv4 }}"
      vm: "{{ vm_name }}"
      public_port: "{{ public_port }}"
      public_end_port: "{{ public_end_port }}"
      private_port: "{{ private_port }}"
      private_end_port: "{{ private_end_port }}"
      protocol: "{{ proto }}"
      open_firewall: "{{ open_firewall }}"
      state: present
    ```
Parts

- **Portal**
  - API
  - Communication with Services
  - User/Customer Interface
Parts

- Daemon

  - playbook handling – list, run, run command, read metadata
  - job handling – overview (running jobs), status, notify (Portal)
  - statistics – failed, queue, running, abort
  - job scheduling – reschedule failed, delay, cron

- REST API via HTTPS
- json response
All together

Portal

Cloudstack

VMs

Ansible + Daemon

write

read
Use Case

- **Results**
  - UI with integrated Services
    - CloudStack
    - Automation
    - User / Role Management
    - Tickets, Monitoring, Backup, ...

- Write Operations with Ansible only
- Modular Design

- OpenSource

- API
- Scalable
Questions?

Contact
Sebastian Bretschneider
Cloud Infrastructure Architect
T +49 3591 5253-1214
sebastian.bretschneider@bitgroup.de

BIT.Group GmbH – member of itelligence group

We make the most of SAP® solutions!