Unreachable
Muss es aber nicht

Datum: 17.05.2021
Ort: Berlin
Verfasser: Tommy Buchheim

bundesdruckerei.

Tommy Buchheim (System Manager)

- Geboren 1990
- Urberliner
- Besitzer eines Ausbilderscheins
- 3 Jahren Erfahrung mit Ansible, Tower, Gitlab
- Hundebesitzer
- DnD Spieler

bundesdruckerei.

0. **Vorstellung**
1. **Problemstellung**
   - 1.1 Was ist das Problem
   - 1.2 Wie sieht der Prozess aus
   - 1.3 Wie sieht der Code aus
   - 1.4 Gedanken / Herangehensweise
2. **Methoden und Limits**
   - 2.1 loop
   - 2.2 until & retries
   - 2.3 block & rescue
   - 2.4 wait_for (action plugin)
   - 2.5 wait_for_connection (action plugin)
   - 2.6 pause
3. **Lösung**
   - 3.1 Zielbild
   - 3.2 Ansible Rolle
   - 3.3 Ansible Playbook
4. **Fazit**

Problemstellung.

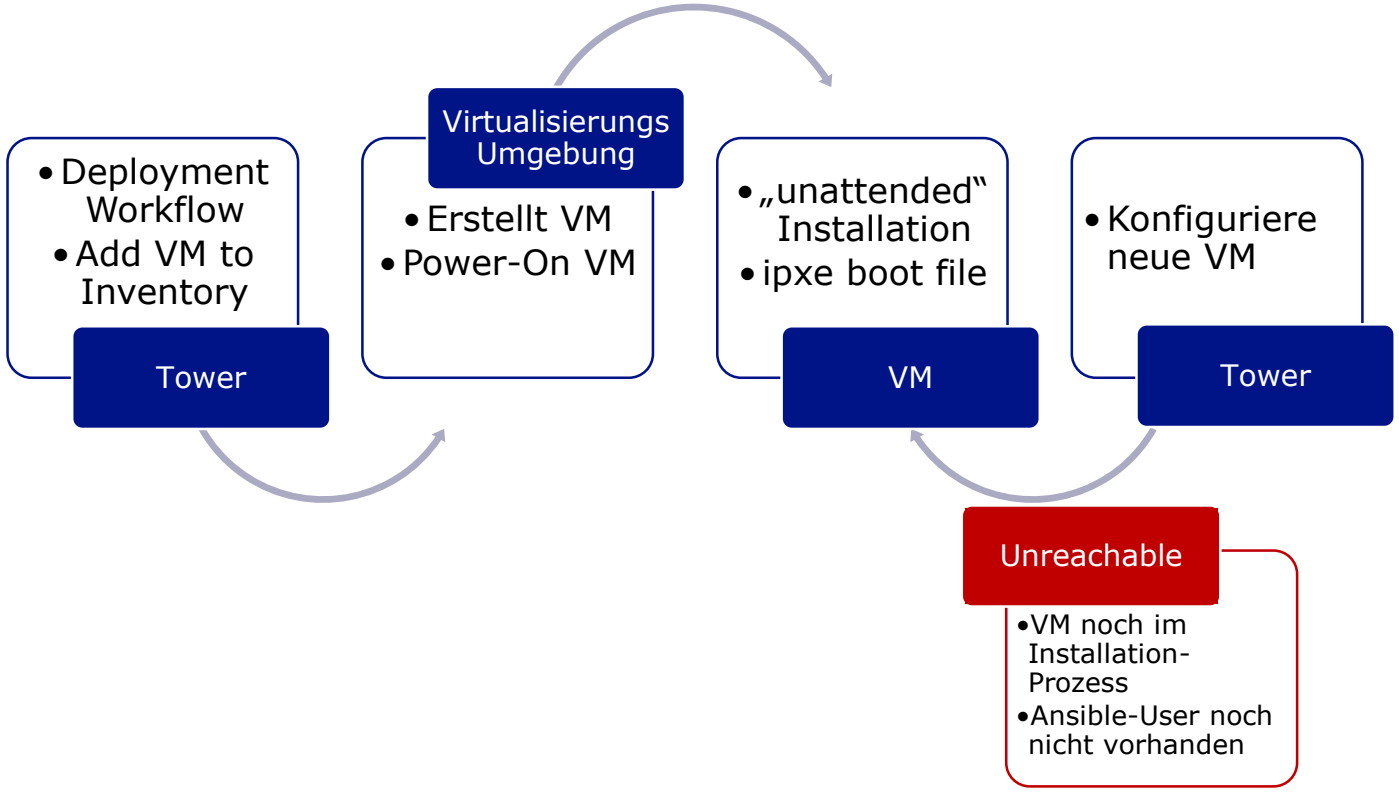
bundesdruckerei.

1.1 Das Problem in Worte gefasst

Wir haben verschiedene Zonen in denen wir deployen müssen, welche unterschiedliche Laufzeiten in der Installation aufweisen. Es gibt keine zuverlässige Abfrage, die ein fertig installiertes Windows deklariert.

- VM-Deployment
  - Installation „from scrach“
  - Windows Installation mit AD-Integration
- Ansible-Automation-User
  - kommt aus dem AD
- Zonen
  - Inventory
  - Credentials/Ansible-Automation-User
  - Umgebungsvariablen

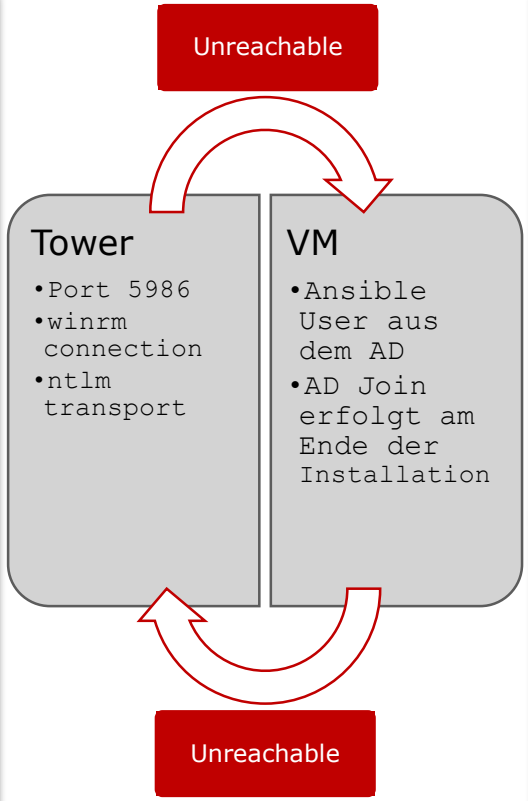
1.2 Deployment Prozess



1.3 Codeansicht

```
---
- name: Deployment
  hosts: Tower
  tasks:
  - name: "Power on and start unattended VM installation"
    vmware_guest_powerstate:
      hostname: "{{ VCENTER }}"
      username: "{{ VCENTER[VCENTER].LOGIN.USER }}"
      password: "{{ VCENTER[VCENTER].LOGIN.PASSWORD }}"
      force: yes
      folder: "{{ VM_FOLDER }}"
      name: "{{ VM_NAME }}"
      state: powered-on

  - name: "Wait until OS is installed"
    vmware_guest_tools_wait:
      hostname: "{{ VCENTER }}"
      username: "{{ VCENTER[VCENTER].LOGIN.USER }}"
      password: "{{ VCENTER[VCENTER].LOGIN.PASSWORD }}"
      validate_certs: no
      name: "{{ VM_NAME }}"
      folder: "{{ VM_FOLDER }}"
    register: WAITOSINSTALLATION
    until: WAITOSINSTALLATION.changed
    retries: 3
    delay: 5
```




```
---
- name: Konfiguriere VM
  hosts: "{{ VM_NAME }}"
  tasks:
  - name: "add Repo entry for /etc/hosts"
    blockinfile:
      path: /etc/hosts
      marker: "# {mark} ANSIBLE MANAGED BLOCK - Repo"
      state: present
      block: |
        {{ REPO_IP}} {{{ REPO_NAME}} {{ REPO_FQDN }}
    when: VM_FAMILY == "linux"

  - name: "Disable protocols for all NICs"
    win_shell: >-
      Get-NetAdapterBinding -ComponentID ms_tcpip6,
ms_lltdio, ms_lldp | Disable-NetAdapterBinding
    when: VM_FAMILY == „windows"
```

bundesdruckerei.

1.4 Gedanken/Herangehensweise

Wann ist etwas „unreachable“?

→

Wie loope ich über einen Host der unreachable ist?

→

Finde ich eine bessere Lösung als einen simplen „timeout“?

Methoden und Limits

2.1 loop

- Können umfänglich genutzt werden
  - Bsp. tasks, includes, …
- Benötigt eine Liste als keyword

```
---
- name: Add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  loop:
     - testuser1
     - testuser2
```

Limit:

- Ein loop mit blocks ist nicht möglich

bundesdruckerei.

2.2 until & retries

- Art von loop
- Benötigt 3 Parameter
  - `until`
    - Bedingung die erfülllt sein muss
  - `retries`
    - Anzahl wie oft etwas maximal versucht werden soll
  - `delay`
    - Pause zwischen den Versuchen in Sekunden

```
---
- name: "wait for URLs to be available"
  uri:
    url: "{{ item.URL }}"
    status_code: "{{ item.STATUS_CODE }}"
  register: result
  until: result.status == item.STATUS_CODE
  retries: "{{ item.RETRIES | default(10) }}"
  delay: "{{ item.DELAY | default(5) }}"
  become: false
  when: URL_LIST is defined
  loop: "{{ URL_LIST }}"
```

Limit:

- Funktioniert nicht bei `include_tasks`

bundesdruckerei.

2.3 block & rescue

- Fasst mehrere Tasks zusammen
- `rescue` block wird dann genutzt, wenn ein Task scheitert
- Parameter auf block Ebene gelten für alle Tasks innerhalb des blocks

```
---
- name: Block CentOS
  block:
    - name: task 1
      file:
        path: /tmp/test
        state: present

    - name: task 2
      yum:
        name: [ `httpd`, `memcached` ]
         state: present
  rescue:
    - name: task A
      debug:
        msg: "Error! Something went wrong."
  when: ansible_distribution == `CentOS`
```

Limit:

- Ein loop mit blocks ist nicht möglich

bundesdruckerei.

2.4 wait_for (action plugin)

- Wird auf dem Ansible Host ausgeführt
- Mit `timeout` kann man pausen pro host definieren
- Szenarios auf die man warten kann
  - `port` in Abhängigkeit zum `state`
  - regex match eines Strings in einer Datei
  - `state` einer Datei

```
---
- name: waiting for Port to come online
  local_action:
    module: wait_for
    host: "{{ inventory_hostname }}"
    port: "{{ item.PORT }}"
    state: started
    delay: "{{ item.DELAY | default(10) }}"
    timeout: "{{ item.TIMEOUT | default(20) }}"
  become: false
  when: PORT_LIST is defined
  loop: "{{ PORT_LIST }}"
```

Limit:

- Keine Prüfung ob host nicht mehr unreachable ist

bundesdruckerei.

2.5 wait_for_connection (action plugin)

- Wird auf dem Ansible Host ausgeführt
- Kann über die Parameter `delay`, `sleep`, `connect_timeout` & `timeout` gesteuert werden
- Funktioniert sowohl für Windows als auch Linux

Limit:

- Kann nicht delegiert werden

```
---
# Build a new VM, wait for it to become ready and
continue playbook
- hosts: all
  gather_facts: no
  tasks:
  - name: Clone new VM, if missing
    vmware_guest:
      hostname: '{{ vcenter_ipaddress }}'
      name: '{{ inventory_hostname_short }}'
      template: Windows 2012R2
      customization:
        hostname: '{{ vm_shortname }}'
        runonce:
        - powershell.exe -ExecutionPolicy Unrestricted -
File C:\Windows\Temp\ConfigureRemotingForAnsible.ps1 -
ForceNewSSLCert -EnableCredSSP
    delegate_to: localhost

  - name: Wait for system to become reachable over WinRM
    wait_for_connection:
      timeout: 900
```

bundesdruckerei.

2.6 pause

- Wenn man einen Timeout braucht ganz praktisch
- Wird über `minutes` und `seconds` parametrisiert

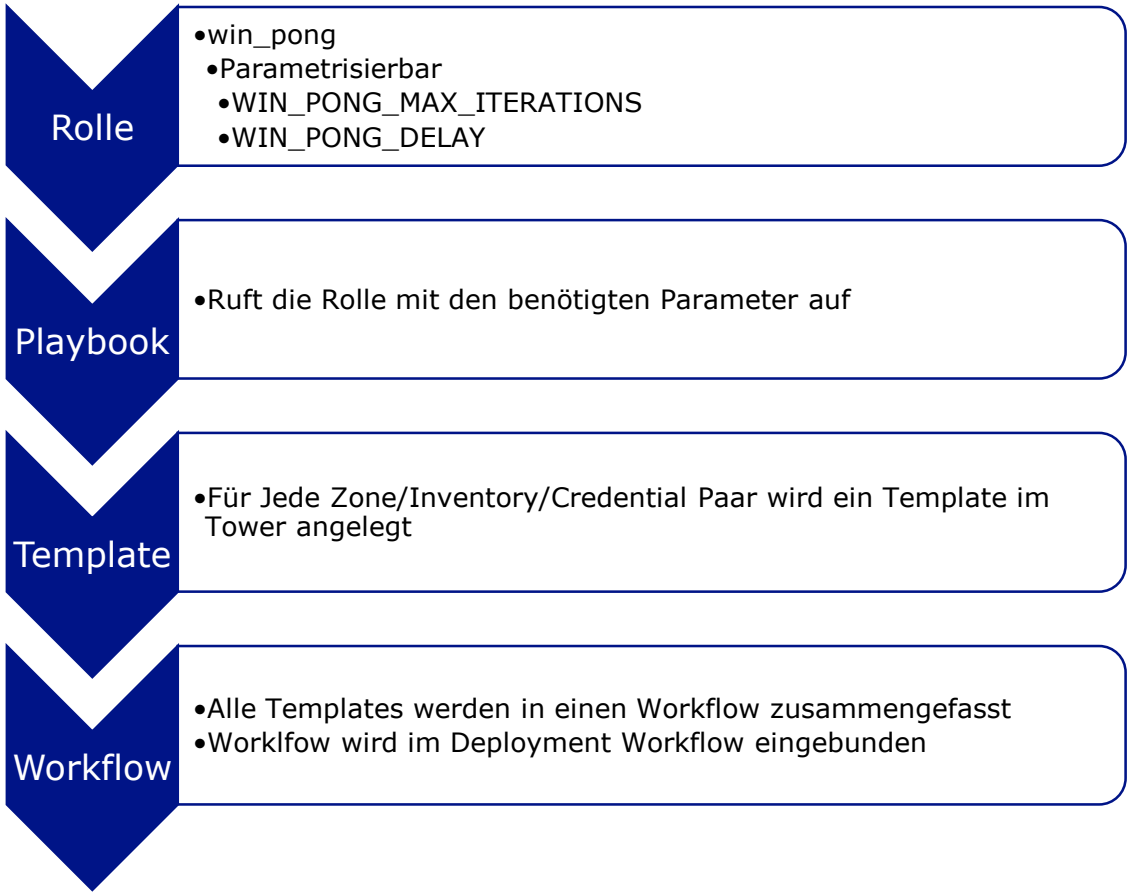
```
---
- name: Pause for 5 minutes
  pause:
    minutes: 5
```

Limit:

- Timeout Wert des Ansible Jobs

Lösung.

3.1 Zielbild


Rolle
•win_pong
•Parametrisierbar
•WIN_PONG_MAX_ITERATIONS
•WIN_PONG_DELAY
Playbook
•Ruft die Rolle mit den benötigten Parameter auf
Template
•Für Jede Zone/Inventory/Credential Paar wird ein Template im Tower angelegt
Workflow
•Alle Templates werden in einen Workflow zusammengefasst
•Worklfow wird im Deployment Workflow eingebunden

3.2 Ansible Rolle Prozessansicht

- `include_tasks` wird noch bevor die 1. Verbindung zum Host aufgebaut wird ausgeführt
- `loop` über `include_tasks`
- `meta: end_host` ist das „`until`“ aus dem `loop`

bundesdruckerei.

3.2 Ansible Rolle Codeansicht

defaults/main.yml

```
---
WIN_PONG_MAX_ITERATIONS: 5
WIN_PONG_DELAY: 30
```

tasks/main.yml

```
---
- name: include Pong
  include_tasks: pong.yml
  loop: "{{ range(0,(WIN_PONG_MAX_ITERATIONS|int))|list }}"
  loop_control:
    extended: yes
  ignore_errors: yes

- name: "Error"
  debug:
    msg: "{{ result }}"
  failed_when: (result.ping is not defined) or not
(result.ping == "pong")
```

tasks/pong.yml

```
---
- name: Pong Windows Host
  win_ping:
  register: result
  no_log: true

- name: "Status Update"
  debug:
    msg: "STATUS = Current Iteration: {{ ansible_loop.index
}} | {{ ansible_loop.revindex0 }} Iterations left."

- name: "Pong"
  debug:
    msg: "{{ inventory_hostname_short }} send a pong."
  when: (result.ping is defined) and (result.ping == "pong")

- name: "Pong?"
  meta: end_host
  when: (result.ping is defined) and (result.ping == "pong")

- name: "Delay {{ WIN_PONG_DELAY }} seconds"
  pause:
    seconds: "{{ WIN_PONG_DELAY }}"
```

bundesdruckerei.

3.3 Ansible Playbook Codeansicht

- `VM_NAME` ist die Variable die auch im Deployment Prozess genutzt wird
- Parametrisiert mit `WIN_PONG_MAX_ITERATIONS` und `WIN_PONG_DELAY`
- `gather_facts: no` , da diese nur stören würden
- `ignore_unreachable: yes` , wichtigster Parameter, damit die Rolle funktioniert

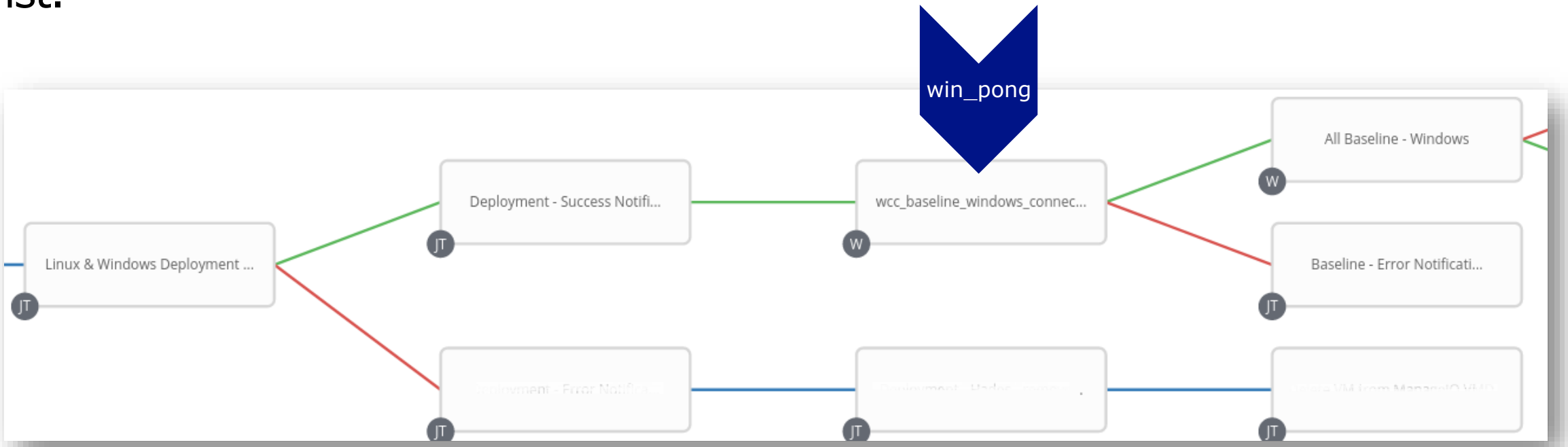
```
---
- name: Ping Pong Windows
  hosts: "{{ VM_NAME }}"
  gather_facts: no
  ignore_unreachable: yes
  vars:
    WIN_PONG_MAX_ITERATIONS: 20
    WIN_PONG_DELAY: 10
  roles:

    - role: 'win_pong'
```

Fazit.

Um die Ecke Denken

Für jedes Problem gibt es auch eine einfache Lösung und dann noch eine Lösung auf die man nicht gleich kommt, die aber am Ende genauso simple ist.



Tommy Buchheim
System Manager IT SI BCS
Email: tommy.buchheim@bdr.de

Vielen Dank.

Frage?

Standet Ihr mal vor einer ähnlichen Herausforderung und welche kreative Lösung habt ihr dafür gefunden?