Automatisierte Testumgebungen mit Ansible und Vagrant

Tuebix 2018

Mark Pröhl

abstract:

- Software-Entwickler und System-Adminstratoren benötigen häufig virtualisierte Systeme um die Installation, Konfiguration und Funktionalität ihrer Software reproduzierbar testen zu können.
- Ziel: Aufsetzen einer verteilten Testumgebung:
  - Linux-Client (z.B. Ubuntu oder CentOS)
  - Windows-Server
- Vorgestellte Tools:
  - Vagrant (https://www.vagrantup.com)
  - Ansible (https://www.ansible.com)
- Beispiel-Softwareprojekt: `msktutil` (https://github.com/msktutil/msktutil)

about me

- In Tübingen Physik studiert
- Beruflich IT-ler
- Nebenberuflich Author
- Linux seit Anfang 90er
- Themengebiete:
  - Identity Management
  - Security
  - Verzeichnisdienste, Authentisierung und Single Sign-on

about `msktutil`

- Auf GitHub: https://github.com/msktutil/msktutil

  “Msktutil creates user or computer accounts in Active Directory, creates Kerberos keytabs on Unix/Linux systems, adds and removes principals to and from keytabs and changes the user or computer account’s password.”

- einfaches Werkzeug um Kerberos-Keyabs in Active-Directory-Umgebungen zu verwalten

von Hand compilieren und schnell mal Testen

```
$ sudo yum install autoconf gcc-c++ krb5-devel openldap-devel
$ git clone https://github.com/msktutil/msktutil.git
$ cd msktutil/
msktutil $ autoconf
msktutil $ ./configure && make && sudo make install
$ sudo su -
# kinit Administrator@TUEBIX.EXAMPLE.COM
# msktutil create
# kdestroy
```

Erklärung:

- `kinit` holt Kerberos-Tickets
- `msktutil create` macht eine Art “Domain Join”:
  - erzeugt einen Maschinenaccount im AD
  - verwaltet die Datei `/etc/krb5.keytab`
  - nutzt dabei Kerberos für AD-Authentisierung
- `kdestroy` löscht die Tickets wieder

about Vagrant

- https://www.vagrantup.com
- vereinfachtes Deployment von virtuellen Maschinen
- unterstützt verschiedene Virtualisierer:
  - primär VirtualBox
  - Für VMware stehen Plugins zur Verfügung – allerdings kostenpflichtig ;-(
- Anbindung an Konfigurationsmanagement (z.B. Puppet oder Ansible)
- Vagrant Cloud mit vielen vorkonfigurierten VMs:
  https://app.vagrantup.com/boxes/search
- Installation unter CentOS-7:
  - Donwload RPM here: https://www.vagrantup.com/downloads.html
  - `rpm -ihv vagrant_2.1.1_x86_64.rpm`

ein Vagrantfile für Linux

- Initiales `Vagrantfile` erzeugt man mit `vagrant init`
- Beispiel für einen Centos-7 Client:

```
Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  config.vm.box_check_update = false
  config.vm.network "private_network", ip: "192.168.33.101"
  config.vm.hostname = "centos7"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
  end
end
```

ein Vagrantfile für Windows

- Beispiel für einen Windows Server 2016

```
Vagrant.configure("2") do |config|
  config.vm.box = "mwrock/Windows2016"
  config.vm.hostname = "w2k16"
  config.vm.network "private_network", ip: "192.168.33.100"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.gui = true
  end
end
```

ein Vagrantfile für beide Maschinen

```
Vagrant.configure("2") do |config|
  config.vm.define "centos7" do |centos7|
    centos7.vm.box = "centos/7"
    centos7.vm.box_check_update = false
    centos7.vm.network "private_network", ip: "192.168.33.101"
    centos7.vm.hostname = "centos7.tuebix.example.com"
    centos7.vm.provider "virtualbox" do |vb|
       vb.memory = "1024"
    end
    centos7.vm.provision "shell", inline: <<-SHELL
      yum -y install https://dl.fedoraproject.org/pub/epel/epel-r
#      yum -y upgrade
      yum -y install ansible python-winrm
      yum -y install git krb5-devel openldap-devel autoconf gcc-c
    SHELL
  end
...
```

ein Vagrantfile für beide Maschinen (cont.)

```
...
  config.vm.define "w2k16" do |w2k16|
    w2k16.vm.box = "jacqinthebox/windowsserver2016"
    w2k16.vm.hostname = "w2k16"
    w2k16.vm.network "private_network", ip: "192.168.33.100"
    w2k16.vm.provider "virtualbox" do |vb|
       vb.memory = "1024"
       vb.gui = true
    end
  end
end
```

- Starten mit `vagrant up`

about Ansible

- https://www.ansible.com
- Werkzeug für Konfigurationsmanagement, Automatisierungs und Orchestrierung
- Nutzt native Protokolle: SSH / Windows Remote Management (WinRM)
- Beschreibung erfolgt in YAML
- Unterstützt Ad-hoc-Kommandos und “Playbooks”
- Mitgelieferte Module: `ansible-doc -l`
- Ansible Galaxy:
  - https://galaxy.ansible.com
  - `ansible-galaxy search`
  - zahlreiche zusätzliche “Roles”
- Installation:
  - `yum install ansible`
  - `yum install python-winrm"` für Windows Remote Management

Ansible testen

- Auf `centos7` anmelden:

```
$ vagrant ssh centos7
[vagrant@centos7 ~]$
```

- `root` werden und nach `/vagrant` wechseln:

```
[vagrant@centos7 ~]$ sudo su -
[root@centos7 ~]# cd /vagrant/
[root@centos7 vagrant]#
```

- Die Datei `/etc/ansible/hosts` sollte so aussehen:

“‘ w2k16 ansible_host=192.168.33.100 ansible_port=5985 ansible_user=vagrant ansible_password=vagrant ansible_connection=winrm ansible_winrm_transport=ntlm

centos7 ansible_connection=local “‘

Ansible testen (cont.)

- Test mit den Modulen `ping`, bzw. `win_ping`:

```
[root@centos7 vagrant]# ansible centos7 -m ping
centos7 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[root@centos7 vagrant]# ansible w2k16 -m win_ping
w2k16 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Ansible testen (cont.)

- Test mit Ad-hoc-Kommando:

```
[root@centos7 vagrant]# ansible centos7 -m shell \
                                    -a 'hostname;whoami'
centos7 | SUCCESS | rc=0 >>
centos7
root
```

Ein Playbook für den DC

```
---
- name: promote windows server
  hosts:
    - w2k16
  tasks:
    - win_feature:
        name: AD-Domain-Services
        state: present
        include_management_tools: yes
        include_sub_features: yes
    - win_domain:
        dns_domain_name: tuebix.example.com
        safe_mode_password: P@ssw0rd
      register: dcpromo_out
    - win_reboot:
      when: dcpromo_out.reboot_required
```

- Bug: der Reboot funktioniert nicht und muss von Hand ausgeführt werden

Ein Playbook für den CentOS Client. . .

```
---
- name: configure centos client
  hosts:
    - centos7
  tasks:
    - name: confiure dns resolver
      copy:
        content: |
                 search tuebix.example.com
                 nameserver 192.168.33.100
        dest: /etc/resolv.conf
    - name: confiure kerberos client libraries
      copy:
        content: |
                 [libdefaults]
                 default_realm = TUEBIX.EXAMPLE.COM
        dest: /etc/krb5.conf
    - name: checkout msktutil
      git:
```

Fragen / Feedback?

- Kontakt: mark@mproehl.net
- Links:
  - https://www.vagrantup.com
  - https://www.ansible.com
  - https://github.com/msktutil/msktutil.git
  - https://www.kerberos-buch.de/tuebix-2018