How to achieve easy automatic deployment of Centos7 as VirtualBox VM in a dynamic DNS LAN environment with an Ansible playbook

PLEASE NOTE: This Page describes my solution to some “made up” or very specialized problem/environment! I dont know how usefull this is “as is” in the real world. However, to me it was primarily a training/learning project to get wet with ansible. But hopefuly this is still of use for someone else.

Scenario / Environment:

- Have an internet connected LAN with a local DHCP and “dynamic DNS” Server (bind).
- DHCP Clients provide their FQDN to the DHCP Server and receive IPv4.
- DHCP Server updates local DNS Server (bind) so that the client FQDN becomes resolvable network wide. (Domainname is .lan)
- My VirtualBox virtualisation host is my Ansible host. It itself is a DHCP Client.
- VirtualBox Server contains one predefined/preinstalled template VM
  - Guest OS CentOS7
  - Guest VM template name “centos7-template”
  - Guest template FQDN “centos7-template.lan”
  - Guest Network: enabled, dhcp, bridged, sshd enabled
  - Guest contains “root” user with generic start-password
  - Guests “root” account contains a “authorized_key” (for ssh public key authentication) for use with Ansible or generic administrative SSH access

Problem:

To deploy another fresh Centos7 VM all i need to do is to “clone” the template VM, boot it and change its hostname, reboot it again.

Easy task, right ? But how to automate this with Ansible, so all i'll have to do is fire up just one command line and be flexible with the guests VM name and hostnames?

My Solution:

1. Have Ansible installed on the VirtualBox Host (because, why not?)
2. Have this Ansible Playbook prepared:

deploy-centos7-from-template-on-virtualbox.yml

```
---
- hosts: localhost
  connection: local
  gather_facts: no
  tasks:
  - name: clone template vm
    command: VBoxManage clonevm centos7-template --mode all --name
{{ newVmName }} --register
  - name: temp.start clone vm
    command: VBoxManage startvm {{ newVmName }} --type headless
  - name: add temp host to inventory (ram)
    add_host:
      hostname: "{{ newVmName }}.lan"
  - name: flush local dns cache
    command: systemd-resolve --flush-caches
  - name: wait for new vm to register/update dns hostname and come
available
    shell: /ssd/VirtualBox\ VMs/wait-for-hostname-resolvable.sh
centos7-template.lan 60
  - name: wait for ssh@vm to be accessable
    wait_for:
      timeout: 90
      sleep: 2
      host: "centos7-template.lan"
      port: 22

- hosts: centos7-template.lan
  gather_facts: no
  remote_user: root
  tasks:
  - name: check connection again
    wait_for_connection:
      timeout: 60
  - name: change vms hostname
    command: hostnamectl set-hostname "{{ newVmName }}.lan"
  - name: reboot vm
    reboot:

- hosts: localhost
  gather_facts: no
  connection: local
  tasks:
  - name: flush local dns cache
    command: systemd-resolve --flush-caches
  - name: wait for new vm to register/update dns hostname and come
available
    shell: /ssd/VirtualBox\ VMs/wait-for-hostname-resolvable.sh
"{{ newVmName }}.lan" 60
  - name: wait for ssh@vm to be accessable
```

```
    wait_for:
      timeout: 90
      sleep: 2
      host: "{{ newVmName }}.lan"
      port: 22

- hosts: "{{ newVmName }}.lan"
  gather_facts: no
  remote_user: root
  tasks:
  - name: show fqdn of new vm
    command: hostname -f
```

3. Also have this little Linux shell script with it, so that Ansible has some way to check when a fqdn gets available/resolvable in my dynamic DNS environment, but still have some way to check for a timeout:

   wait-for-hostname-resolvable.sh

```
#!/bin/bash

targetHostname="$1"
timeOut="$2"

if test -z "$targetHostname" ; then
    echo "ERROR: No hostname given."
    echo "Usage: $0 fqdn timeOutSeconds"
    exit -1
fi

if test -z "$timeOut" ; then
    timeOut=60
fi

for seconds in $(seq $timeOut -1 1) ; do
    systemd-resolve --flush-caches
    if host "$targetHostname" 1>/dev/null 2>&1 ; then
        if ping -c1 -w1 "$targetHostname" 1>/dev/null 2>&1 ; then
            break
        else
            #echo "${seconds}s left"
            continue
        fi
    else
        #echo "${seconds}s left"
        sleep 1
    fi
done
```

```
if test $seconds -gt 1 ; then
    true
else
    false
fi
```

- **NOTE:** Ansible comes already with modules to check for hosts and connections (like **wait_for** and **wait_for_connection**). However, i was not successfull with any of those, because in my special network environment hostnames are not always resolvable PLUS i had massive problems with "hostname caching" of the systemd service. So i needed something that takes care of all this, which Ansibles modules seem not to do at the moment. The modules that come with Ansible seem to always asume the hostname is resolvable. If not resolvable the modules fail right on the spot.

4. Have some Ansible "inventory" (hosts) file handy:

   ansibleHosts

```
[templatevms]
centos7-template.lan

#[admin]
#centos7admin.lan

#[cluster]
#centos7node1.lan
#centos7node2.lan
#centos7node3.lan
```

5. Run the Ansible playbook without SSH key checking to prevent answering security questions and provide the VMs new name on the command line:

```
ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -v -i ansibleHosts -e
newVmName=newVMsHostname deploy-centos7-from-template-on-virtualbox.yml
```

6. This will clone the template vm (resulting in new MAC and VBox media UUIDs), start the new VM, ssh into the new VM to change its hostname and reboots it. After new VM is back up it checks connection and hostname by a "hostname -f".

About being "idempotent": Well.... this playbook isnt! Can be run ONLY ONCE for a specific VM name. After that choose a new VM name or delete the existing VM first.

linux, centos, 7, centos7, dynamic, dns, ddny, dyndns, bind, dhcpd, ansible, playbook, deploy, vm, virtualbox, network, local, lan