# Практика: DOB Модул 4 - Ansible

## Инсталация на Ansible и подготовка на средата

### Ansible

#### SUSE/openSUSE

sudo zypper install ansible

#### RedHat/CentOS

[sudo yum install epel-release]

sudo yum install ansible

#### Debian/Ubuntu

sudo apt-add-repository ppa:ansible/ansible

sudo apt-get update

sudo apt-get install ansible

### Работна среда

Извършваме последователно следните действия

1. Създаваме папка **DOB-Ansible/p1**
2. Създаваме файл **Vagrantfile**:

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

Vagrant.configure(2) do |config|

config.ssh.insert\_key = false

config.vm.define "web" do |web|

web.vm.box="shekeriev/centos-7-64-minimal"

web.vm.hostname = "web.sulab.local"

web.vm.network "private\_network", ip: "192.168.82.100"

web.vm.network "forwarded\_port", guest: 80, host: 8080

end

config.vm.define "db" do |db|

db.vm.box = "shekeriev/centos-7-64-minimal"

db.vm.hostname = "db.sulab.local"

db.vm.network "private\_network", ip: "192.168.82.101"

end

config.vm.define "clnt" do |clnt|

clnt.vm.box = "shekeriev/centos-7-64-minimal"

clnt.vm.hostname = "clnt.sulab.local"

clnt.vm.network "private\_network", ip: "192.168.82.102"

end

end

1. Изпълняваме **vagrant up**

## Първа част

Добавяме хостовете в глобалния инвентарен файл:

1. su -c 'echo '[test-srv]' >> /etc/ansible/hosts'
2. su -c 'echo '192.168.82.100' >> /etc/ansible/hosts'
3. su -c 'echo '192.168.82.101' >> /etc/ansible/hosts'
4. su -c 'echo '[test-clnt]' >> /etc/ansible/hosts'
5. su -c 'echo '192.168.82.102' >> /etc/ansible/hosts'

Или го отваряме за редакция като **root**.

Изпълняваме последователно следните команди:

1. ansible 192.168.82.100 -a "hostname"
2. ansible 192.168.82.100 -a "hostname" -u vagrant
3. ansible 192.168.82.100 -a "hostname" -u vagrant -k

Паролата за потребителя **vagrant** е **vagrant**

Изпълняваме последователно следните команди:

1. ansible test-srv -a "hostname" -u vagrant -k
2. ansible test-srv -a "hostname" -u vagrant -k
3. ansible all -a "hostname" -u vagrant -k

Ако не можем да осъществим връзка, бихме могли да добавим съответните записи в **~/.ssh/known\_hosts** като влезем последователно на всеки хост или изпълним **ssh-keyscan**:

1. ssh [vagrant@192.168.82.101](mailto:vagrant@192.168.81.101)
2. ssh [vagrant@192.168.82.102](mailto:vagrant@192.168.82.102)

Сега можем да изпълним:

1. ansible all -a "hostname" -u vagrant -k
2. ansible all -a "hostname" -u vagrant -k -f 1

Последната команда ограничава броя на едновременно изпълняваните процеси.

Изпълняваме последователно следните команди:

1. ansible all -m command -a "df -h" -u vagrant -k
2. ansible all -m shell -a "df -h" -u vagrant -k
3. ansible all -m shell -a "echo $HOSTNAME" -u vagrant -k
4. ansible all -m shell -a 'echo $HOSTNAME' -u vagrant -k

Обърнете разликата в резултата от изпълнението на последните две команди.

Командите, изпълнени през **command** се изпълняват „директно“, за разлика от тези, изпълнени през **shell**.

Модулът **command** може да бъде пропуснат:

1. ansible all -a "df -h" -u vagrant -k
2. ansible all -a "free -m" -u vagrant -k

Изпълняваме последователно следните действия:

1. Създаваме файл **local\_script.sh** със следното съдържание:

#!/bin/bash

echo 'My hostname is '$HOSTNAME

1. ansible test-srv -m script -a "local\_script.sh" -u vagrant -k

## Втора част

### Инвентарни файлове (1)

Създаване на инвентарен файл

1. Създаваме празен файл **inventory**
2. Добавяме:

web ansible\_host=192.168.82.100 ansible\_user=vagrant ansible\_ssh\_pass=vagrant

1. Записваме и излизаме
2. Изпълняваме

ansible web -i inventory -a "hostname"

1. Отваряме файла и създаваме група:

[grp-webservers]

web

1. Изпълняваме

ansible grp-webservers -i inventory -a "hostname"

1. Отваряме файла и добавяме втори хост:

db ansible\_host=192.168.82.101 ansible\_user=vagrant ansible\_ssh\_pass=vagrant

1. И група за него:

[grp-databases]

db

1. Запазваме и изпълняваме

ansible grp-databases -i inventory -a "hostname"

1. Отваряме файла и добавяме трети хост:

clnt ansible\_host=192.168.82.102 ansible\_user=vagrant ansible\_ssh\_pass=vagrant

1. И група за него:

[grp-stations]

clnt

1. След това добавяме обща група:

[grp-servers:children]

grp-webservers

grp-databases

1. И най-накрая добавяме променливи за групата:

[grp-servers:vars]

ansible\_user=vagrant

ansible\_ssh\_pass=vagrant

1. Изпълняваме

ansible grp-servers -i inventory -a "hostname"

### Променливи (2)

Глобално ниво:

1. Създаваме директории **2/group\_vars** и **2/host\_vars**
2. Копираме файла **inventory** от папка **1**
3. Създаваме файл **all** в папка **2/group\_vars**

---

# Global level user

username: user\_all

1. ansible grp-servers -i inventory -m user -a "name={{username}} password=Password1" --become

Групово ниво:

1. Създаваме файл **grp-webservers** в папка **2/group\_vars**

---

# Group level user

username: user\_group

1. ansible grp-servers -i inventory -m user -a "name={{username}} password=Password1" --become

Ниво хост:

1. Създаваме файл file **web** in **2/host\_vars**

---

# Host level user

username: user\_host

1. ansible grp-servers -i inventory -m user -a "name={{username}} password=Password1" --become

### Конфигурации (3)

Работим в папка 3. Копираме инвентарния файл от папка 2.

Използване на локален конфигурационен файл:

1. Премахваме записите за трите хоста от файла **~/.ssh/known\_hosts**
2. Изпълняваме

ansible clnt -i inventory -a "hostname" -u vagrant -k

1. Създаваме файл **ansible.cfg**

[defaults]

host\_key\_checking=false

1. ansible clnt -i inventory -a "hostname" -u vagrant -k
2. cat ~/.ssh/known\_hosts

Променливи на средата:

1. Създаваме променлива

export ANSIBLE\_HOST\_KEY\_CHECKING=true

1. Проверяваме я

echo $ANSIBLE\_HOST\_KEY\_CHECKING

1. Изпълняваме:

ansible db -i inventory -a "hostname" -u vagrant -k

1. Премахваме променливата

unset ANSIBLE\_HOST\_KEY\_CHECKING

1. Изпълняваме отново

ansible db -i inventory -a "hostname" -u vagrant -k

1. Проверяваме

cat ~/.ssh/known\_hosts

Допълнителни настройки:

1. Редактираме фйла **ansible.cfg**

private\_key\_file = /home/devops/.vagrant.d/insecure\_private\_key

ansible\_user = vagrant

remote\_user = vagrant

1. Редактираме файла **inventory** като премахваме всичката информация за потребители и пароли:

web ansible\_host=192.168.82.100

db ansible\_host=192.168.82.101

clnt ansible\_host=192.168.82.102

[grp-webservers]

web

[grp-databases]

db

[grp-stations]

clnt

[grp-servers:children]

grp-webservers

grp-databases

1. Изпълняваме

ansible web -i inventory -a "hostname"

### Modules (4)

Предварителна подготовка - работим в папка 4 и копираме файлове от папка 3:

1. Копираме инвентарния и конфигурационния файл:

cp ../3/ansible.cfg .

cp ../3/inventory .

Преглеждаме списъка от достъпни модули

1. ansible-doc -l
2. ansible-doc yum

Инсталираме Apache HTTP на нашите уеб сървъри (**grp-webservers**):

1. ansible grp-webservers -i inventory -m yum -a "name=httpd state=present" --become
2. ansible grp-webservers -i inventory -m service -a "name=httpd state=started enabled=true" --become
3. Тестваме в браузър на хоста - <http://localhost:8080>

Инсталираме MariaDB на нашите сървъри за бази от данни (**grp-dbservers**):

1. ansible grp-databases -i inventory -m yum -a "name=mariadb,mariadb-server state=present" --become
2. ansible grp-databases -i inventory -m service -a "name=mariadb state=started enabled=true" --become
3. Влизаме в системата и проверяваме дали всичко работи

## Трета част

### Playbooks

Ако все още не сме го направили, можем да спрем всички машини от предходните упражнения.

Подготвяме работната директория:

1. Бидейки в работната папка **DOB-Ansible** изпълняваме следните команди

mkdir -p p3/{1..2}

cd p3

1. Копираме **Vagrantfile** от **p2** и изпълняваме

vagrant up

1. Изпълняваме

cd 1

1. Копираме **inventory** файла от **../../p2/2**

cp ../../p2/2/inventory .

1. Създаваме конфигурационен файл **ansible.cfg**

[defaults]

host\_key\_checking = False

hostfile=iventory

1. Създаваме файл **playbook.yml** със съдържание:

---

- hosts: grp-webservers

become: true

tasks:

- name: Install Apache HTTP Server

yum: name=httpd state=present

- name: Start Apache HTTP Server and Enable it

service: name=httpd state=started enabled=true

1. Изпълняваме

ansible-playbook playbook.yml

1. Нека отворим отново файла **playbook.yml** и да добавим:

- hosts: grp-databases

become: true

tasks:

- name: Install MariaDB Server

yum: name=mariadb,mariadb-server state=present

- name: Start and enable MariaDB

service: name=mariadb state=started enabled=true

1. Изпълняваме отново:

ansible-playbook playbook.yml

Нека сега да направим невъзможна комуникацията с единия хост:

1. Преди да пристъпим към действие, трябва да променим локалната конфигурация. За целта отваряме файла **ansible.cfg** и добавяме:

retry\_files\_enabled = True

retry\_files\_save\_path = ~/.ansible-retry

1. Отваряме **inventory** файла и извършваме следните промени - премахваме променливите
2. След това добавяме премахнатата информация само срещу единия хост в началото на файла
3. Запазваме и стартираме повторно:

ansible-playbook playbook.yml

За недостъпните хостове се създава специален файл, който след отстраняване на проблема с достъпа, бихме могли да използваме, за да повторим операцията само срещу тях. За целта изпълняваме следните действия:

1. Отваряме **inventory** файла и го възстановяваме към предходното му състояние - добавяме секция с променливи в края

[grp-servers:vars]

ansible\_user=vagrant

ansible\_ssh\_pass=vagrant

1. След това изпълняваме:

ansible-playbook playbook.yml --limit @/home/devops/.ansible-retry/playbook.retry

### Няколко допълнителни техники

#### Register & Debug

Нека да направим още един файл с име **register.yml** и да въведем следния текст:

---

- hosts: clnt

become: false

tasks:

- name: Get system's kernel version

shell: /usr/bin/uname -r

register: kver

- name: Debug info

debug: var=kver

Сега можем да изпълним:

ansible-playbook register.yml

#### Copy

Да пробваме и още една техника - за копиране на файл(-ове). Нека направим нов файл - **copy.yml** и да въведем следния текст:

---

- hosts: grp-webservers

become: true

tasks:

- name: Copy new index.html

copy: src=html/index.html dest=/var/www/html/

Нека направим и нова папка **html**, в която да запишем файл **index.html** със следното съдържание:

<h2>Hello, Ansible!</h2>

Сега можем да изпълним:

ansible-playbook copy.yml

И да проверим резултата в браузъра, като въведем <http://localhost:8080>

#### Conditional

Нека променим **inventory** файла, така че да добавим още един уеб сървър:

web ansible\_host=192.168.82.100

webu ansible\_host=192.168.82.105

…

[grp-webservers]

web

webu

Сега можем да добавим следното във **Vagrant** файла:

config.vm.define "webu" do |webu|

webu.vm.box="ubuntu/trusty64"

webu.vm.hostname = "webu.sulab.local"

webu.vm.network "private\_network", ip: "192.168.82.105"

webu.vm.network "forwarded\_port", guest: 80, host: 8081

end

И да изпълним повторно **vagrant up**

Нека да създадем още един файл **webservers.yml**:

---

- hosts: grp-webservers

become: true

tasks:

- name: Install Apache HTTP Server

yum: name=httpd state=present

when: ansible\_os\_family == "RedHat"

- name: Start Apache HTTP Server and Enable it

service: name=httpd state=started enabled=true

when: ansible\_os\_family == "RedHat"

- name: Install Apache HTTP Server on Ubuntu

apt: name=apache2 state=present

when: ansible\_os\_family == "Debian"

- name: Start Apache HTTP Server and Enable it on Ubuntu

service: name=apache2 state=started enabled=true

when: ansible\_os\_family == "Debian"

И да изпълним:

ansible-playbook webservers.yml

Нека сега да отворим следния адрес <http://localhost:8081> в браузър.

### Шаблони

Нека изпълним следната последователност от действия, за да видим макар и в много базов пример работата с шаблони:

1. Отиваме в папка **p3/2**

cd ../2

1. Копираме следните файлове от папка **1**

cp ../1/ansible.cfg .

cp ../1/inventory .

cp ../1/webservers.yml

1. Създаваме папка **templates**

mkdir templates

1. Създаваме файл **index.j2** в папка **templates** със следното съдържание:

<html>

<head>

<title>Hello!</title>

</head>

<body>

<h2>Hello from Ansible on {{ v\_host\_type }}!</h2>

</body>

</html>

1. Променяме съдържанието на **webservers.yml** както следва:

---

- hosts: grp-webservers

become: true

tasks:

- name: Install Apache HTTP Server

yum: name=httpd state=present

when: ansible\_os\_family == "RedHat"

- name: Start Apache HTTP Server and Enable it

service: name=httpd state=started enabled=true

when: ansible\_os\_family == "RedHat"

- name: Deploy index.j2 on RedHat

vars:

v\_host\_type: RedHat

template: src=templates/index.j2 dest=/var/www/html/index.html

when: ansible\_os\_family == "RedHat"

- name: Install Apache HTTP Server on Ubuntu

apt: name=apache2 state=present

when: ansible\_os\_family == "Debian"

- name: Start Apache HTTP Server and Enable it on Ubuntu

service: name=apache2 state=started enabled=true

when: ansible\_os\_family == "Debian"

- name: Deploy index.j2 on Debian

vars:

v\_host\_type: Debian

template: src=templates/index.j2 dest=/var/www/html/index.html

when: ansible\_os\_family == "Debian"

1. Изпълняваме командата:

ansible-playbook webservers.yml

1. Отваряме в браузър адреса: <http://localhost:8080>
2. Отваряме във втори прозорец адреса: <http://localhost:8081>

## Работа в хетерогенна среда

Нека да добавим една **Windows** машина към нашата виртуална инфраструктура. Това може да стане като импортираме вече готова или инсталираме нова.

При всички положения трябва да има мрежова свързаност и да е видима от нашия **Ansible** хост. След като се уверим, че в мрежово направление всичко е наред, можем да пристъпим към настройката на защитната стена така, че да позволява отдалечено управление.

Конфигурирането на защитната стена може да стане по няколко начина. Тъй като нашият случай имаме **Windows Server 2016 Core**, то можем да използваме командата **sconfig**.

За цялостно и автоматизирано настройване на **WinRM** можем да използваме следния скрипт:

<https://github.com/ansible/ansible/blob/devel/examples/scripts/ConfigureRemotingForAnsible.ps1>

Един от вариантите да доставим този скрипт на нашия Windows Core хост е да направим споделена папка. За целта стартираме PowerShell и изпълняваме следните команди:

mkdir C:\Temp

New-SmbShare -Name Temp -Path 'C:\Temp' -FullAccess Everyone

Set-NetFirewallRule -Name 'FPS-SMB-In-TCP' -Enabled True

След това стартираме копирания конфигурационен скрипт.

Сега на нашия **Ansible** хост трябва да добавим информация в инвентарния файл:

[win]

192.168.82.110

[win:vars]

ansible\_user: Administrator

ansible\_password: Password1

ansible\_port: 5986

ansible\_connection: winrm

ansible\_winrm\_server\_cert\_validation: ignore

Като пред-последна стъпка инсталираме **pip**, ако вече не е наличен:

sudo yum install python-pip

Най-накрая инсталираме съответния **Python** модул на нашата работна станция:

pip install pywinrm

Можем да изпълним следната команда, за да съберем информация за отдалечения **Windows** хост:

ansible win -i inventory -m setup