Microsoft

Jose Moreno

(jose.moreno@microsoft.com)

12/12/2016

Using Ansible to manage Azure workloads

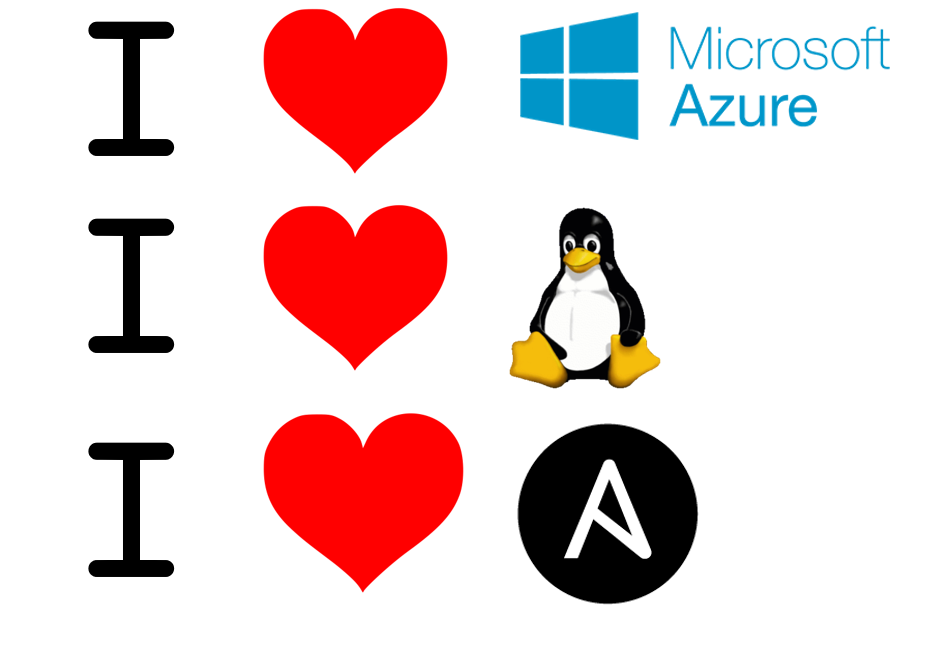


Table of Contents

[Objectives and initial setup 2](#_Toc484560598)

[Introduction to Ansible 4](#_Toc484560599)

[Lab 1: Create Master VM using Azure CLI 5](#_Toc484560600)

[Lab 2: Create Service Principal 8](#_Toc484560601)

[Lab 3: Install Ansible in the Master VM 10](#_Toc484560602)

[Lab 4: Ansible dynamic inventory for Azure 13](#_Toc484560603)

[Lab 5: Creating a VM using an Ansible Playbook 16](#_Toc484560604)

[Lab 6: Running an Ansible playbook on the new VM 20](#_Toc484560605)

[Lab 7: Deleting a VM using Ansible (Optional) 22](#_Toc484560606)

[Conclusion 23](#_Toc484560607)

[End the lab 24](#_Toc484560608)

[References 25](#_Toc484560609)

# Objectives and initial setup

This document contains a lab guide that helps to deploy a basic environment in Azure that allows to test some of the functionality of the integration between Azure and Ansible.

Before starting with this account, make sure to fulfill all the requisites:

* If you are running this lab from the Learn on Demand Systems environment:
  + You will get access to a CentOS VM with a graphical environment, and the Azure CLI installed. Credentials are lab-user / Microsoft123!
  + You will get access to an Azure subscription
* If you are running this lab from your own machine:
  + A valid Azure subscription account. If you don’t have one, you can create your [free azure account](https://azure.microsoft.com/en-us/free/) (<https://azure.microsoft.com/en-us/free/>) today.
  + Install the Azure CLI 2.0 following these instructions: <https://docs.microsoft.com/de-de/cli/azure/install-azure-cli>
* In any case, you should have a basic understanding of vi (or your Linux text editor of choice) in order to make some basic file editing

This lab will cover:

* Introduction to Ansible installation and first steps
* Example of playbooks to interact with Azure in order to create and delete VMs
* Example of playbooks to interact with Azure Linux VMs in order to modify them installing additional software packages or downloading files from external repositories
* Using Ansible’s dynamic inventory information so that VM names to be controlled by Ansible do not need to be statically defined, but are dynamically retrieved from Azure

Along this lab some variables will be used, that might (and probably should) look different in your environment. This is the variables you need to decide on before starting with the lab. Notice that the VM names are prefixed by a (not so) random number, since these names will be used to create DNS entries as well, and DNS names need to be unique.

|  |  |
| --- | --- |
| **Description** | **Value used in this lab guide** |
| Azure resource group | ansiblelab |
| Name for provisioning VM | 19761013myvm |
| Username for provisioning VM | lab-user |
| Password for provisioning VM | Microsoft123! |
| Name for created VM (note that this name should be unique, since a DNS entry will be created). | web0119761013 |

# Introduction to Ansible

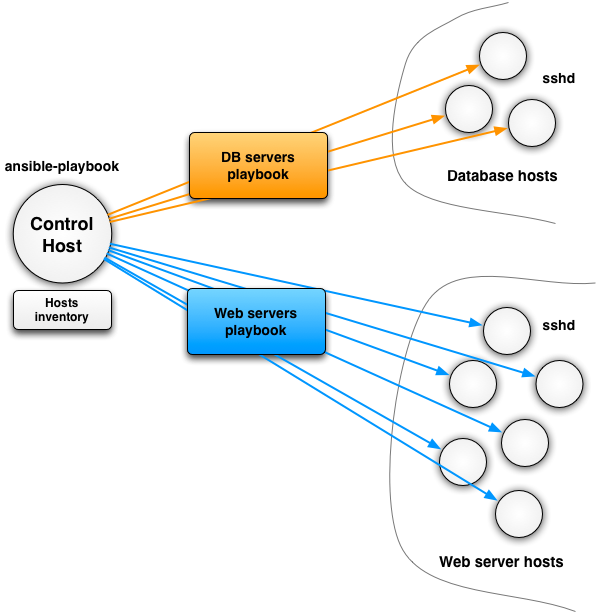
Ansible is a software that falls into the category of “Configuration Management Tools”. These tools are mainly used in order to describe in a declarative language the configuration that should possess a certain machine (or a group of them) in so called playbooks, and then make sure that those machines are configured accordingly.

Playbooks are structured using YAML (Yet Another Markup Language) and support the use of variables, as we will see along the labs.

As opposed to other Configuration Management Tools like Puppet or Chef, Ansible is agent-less, which means that it does not require the installation of any software in the managed machines. Ansible uses SSH to manage Linux machines, and remote Powershell to manage Windows systems.

In order to interact with machines other than Linux servers (for example, with the Azure portal in order to create VMs), Ansible supports extensions called “modules”. Ansible is completely written in Python, and these modules are equally Python libraries. In order to support Azure, Ansible needs the Azure Python SDK.

Additionally, Ansible requires that the managed hosts are documented in a inventory. Alternatively, Ansible supports dynamic inventories for some systems, including Azure, so that the host inventory is dynamically generated at runtime.



**Figure 1**. Overall architecture example from Ansible to configure Database and Web servers

# Lab 1: Create Master VM using Azure CLI

In this lab we will create the "Master VM", a Linux VM in Azure where we will install Ansible, and from which we will further deploy additional VMs using Ansible.

If you are running this lab from a Linux system you could install

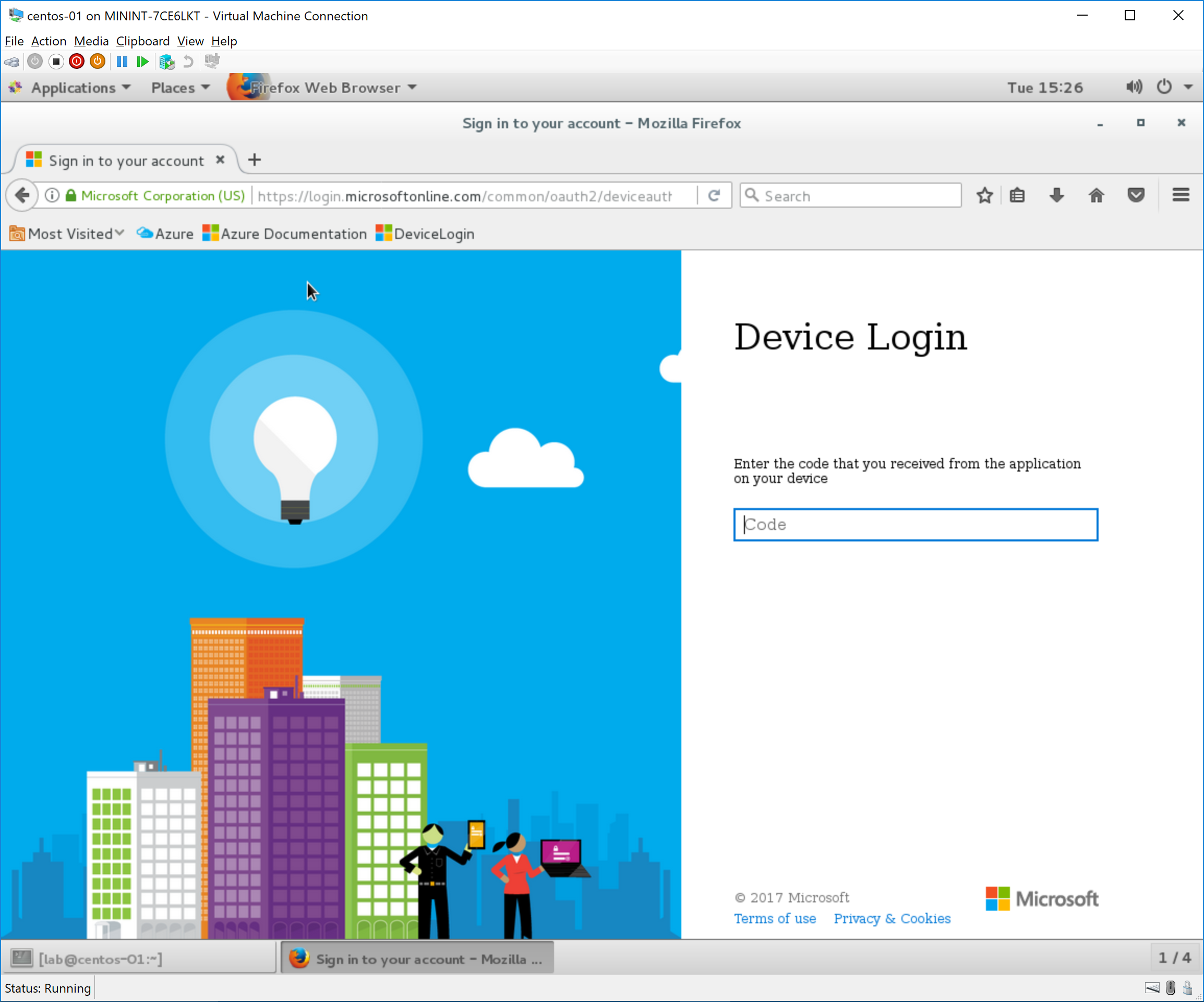
***Note****: if using the CentOS VM provided by Learn On Demand Systems, you can use the key combinations Ctrl+Shift+C and Ctrl+Shift+V to copy and paste text to the terminal window*

1. Log into your system. If you are using the Learn On Demand lab environment, the user for the Centos VM is lab-user, with the password Microsoft123!
2. If you don’t have a valid Azure subscription, but have received a voucher code for Azure, go to <https://www.microsoftazurepass.com/Home/HowTo> for instructions about how to redeem it.
3. Open a terminal window. In Windows, for example by hitting the Windows key in your keyboard, typing "cmd" (without the quotes) and hitting the Enter key. You might want to maximize the command Window so that it fills your desktop.
4. In your terminal window login to Azure.

**az login**

To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code XXXXXXXXX to authenticate.

The “az login” command will provide you a code, that you need to introduce (over copy and paste) in the web page <http://aka.ms/devicelogin>. Open an Internet browser (Firefox is preinstalled int the VM provided by Learn on Demand Systems), go to this URL, and after introducing the code, you will need to authenticate with credentials that are associated to a valid Azure subscription. After a successful login, you can enter the following two commands back in the terminal window in order to create a new resource group, and to set the default resource group accordingly.



**Figure 2**. CLI authentication web site

1. Create a resource group, define it as the default group for further commands, create a Vnet and a subnet, and a Linux machine in that subnet with a public IP address. Here the commands you need for these tasks:

**az group create --name ansiblelab --location westeurope**

**az configure --defaults group=ansiblelab**

**az network vnet create -n ansibleVnet --address-prefixes 192.168.0.0/16 --subnet-name ansibleSubnet --subnet-prefix 192.168.1.0/24**

**az network public-ip create --name masterPip**

**az vm create -n ansibleMaster --image OpenLogic:CentOS:7.3:latest --vnet-name ansibleVnet --subnet ansibleSubnet --public-ip-address masterPip --admin-username lab-user --admin-password Microsoft123!**

{

"fqdns": "",

"id": "/subscriptions/3e78e84b-6750-44b9-9d57-d9bba935237a/resourceGroups/ansiblelab/providers/Microsoft.Compute/virtualMachines/ansibleMaster",

"location": "westeurope",

"macAddress": "00-0D-3A-24-E2-C0",

"powerState": "VM running",

"privateIpAddress": "192.168.1.4",

"publicIpAddress": "**1.2.3.4**",

"resourceGroup": "ansiblelab"

}

***Note****: while this command is running (might take between 10 and 15 minutes), you might wait until it finishes, or in the meantime you can temporarily jump to Lab 2 (Create Service Principal) in a new terminal window. When you are finished with Lab 2 you can come back to this point to finish Lab 1.*

1. The previous command might take 10-15 minutes to run. After you get again the command prompt, connect over SSH to the new VM, using the public IP address displayed in the output of the previous command, and username and password provided in the previous command (lab-user / Microsoft123!). **Please replace 1.2.3.4 with the actual public IP address retrieved out of the last command in Step 2**.

**ssh** [**lab-user@1.2.3.4**](mailto:lab-user@1.2.3.4)

The authenticity of host '1.2.3.4 (1.2.3.4)' can't be established.

ECDSA key fingerprint is 09:7f:7e:fc:34:d9:9f:ff:a6:5c:de:50:5a:5a:4f:14.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '1.2.3.4' (ECDSA) to the list of known hosts.

Password:

[lab-user@ansibleMaster ~]$

1. Install Azure CLI 2.0 in the provisioning machine ansibleMaster:

**sudo yum update -y**

**sudo yum install -y gcc libffi-devel python-devel openssl-devel**

**curl -L https://aka.ms/InstallAzureCli | bash**

***Note****: you can just press Enter to accept the default answer when the installation program asks you a question. In the last question, make sure to tell the script to update the PATH variable (answer 'Y').*

**What we have learnt**

You can use the Azure CLI from any platform (including Linux) to manage Azure. In this section you used basic Azure CLI commands to create virtual networks and a CentOS virtual machine.

# Lab 2: Create Service Principal

This step is required so that Ansible can log in to Azure with non-interactive authentication. We will define a service principal and an application ID, and we will give permissions to the service principal to operate on the resource group that we created in Lab 1.

As best practice, you should give the minimum permissions required to your service principals. See for more information: <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-authenticate-service-principal-cli>).

1. Create Active Directory application:

**az ad app create --password ThisIsTheAppPassword --display-name ansibleApp --homepage ansible.mydomain.com --identifier-uris ansible.mydomain.com**

{

"appId": "**11111111-1111-1111-1111-111111111111**",

"appPermissions": null,

"availableToOtherTenants": false,

"displayName": "ansibleApp",

"homepage": "ansible.mydomain.com",

"identifierUris": [

"ansible.mydomain.com"

],

"objectId": "55555555-5555-5555-5555-555555555555",

"objectType": "Application",

"replyUrls": []

}

1. Create Service Principal:

**az ad sp create --id 11111111-1111-1111-1111-111111111111**

{

"appId": "**11111111-1111-1111-1111-111111111111**",

"displayName": "ansibleApp",

"objectId": "**44444444-4444-4444-4444-444444444444**",

"objectType": "ServicePrincipal",

"servicePrincipalNames": [

"**11111111-1111-1111-1111-111111111111**",

"ansible.mydomain.com"

]

}

1. Find out your subscription and tenant IDs:

**az account show**

{

"environmentName": "AzureCloud",

"id": "**22222222-2222-2222-2222-222222222222**",

"isDefault": true,

"name": "Your Subscription Name",

"state": "Enabled",

"tenantId": "**33333333-3333-3333-3333-333333333333**",

"user": {

"name": "your.name@microsoft.com",

"type": "user"

}

}

1. Assign the Contributor role to the principal for our resource group (remember we have specified the default resource group in Lab 1, so we do not need to specify it again), using the object ID for the service principal:

**az role assignment create --assignee 44444444-4444-4444-4444-444444444444 --role contributor**

{

"id": "/subscriptions/**22222222-2222-2222-2222-222222222222**/resourceGroups/ansiblelab/providers/Microsoft.Authorization/roleAssignments/66666666-6666-6666-6666-666666666666",

"name": "66666666-6666-6666-6666-666666666666",

"properties": {

"principalId": "**44444444-4444-4444-4444-444444444444**",

"roleDefinitionId": "/subscriptions/**22222222-2222-2222-2222-222222222222**/providers/Microsoft.Authorization/roleDefinitions/77777777-7777-7777-7777-777777777777",

"scope": "/subscriptions/**22222222-2222-2222-2222-222222222222**/resourceGroups/ansiblelab"

},

"resourceGroup": "ansiblelab",

"type": "Microsoft.Authorization/roleAssignments"

}

Note the following values of your output, since we will use them later. In this guide they are marked in different colors and dummy values for easier identification:

1. Subscription ID: **22222222-2222-2222-2222-222222222222**
2. Tenant ID: **33333333-3333-3333-3333-333333333333**
3. Application ID (also known as Client ID): **11111111-1111-1111-1111-111111111111**
4. Password: **ThisIsTheAppPassword**

**What we have learnt**

Username and password is not a good authentication method for automation solutions, since it is interactive. The solution for non-interactive authentication in Azure is called "Service Principal" where an application can authenticate with a pre-defined password, and it gets specific permissions within a certain scope.

As alternative to password authentication for the application, digital certificates can be used, but that is out of the scope of this lab.

# Lab 3: Install Ansible in the Master VM

At this point we have our master VM running in Azure, and we have configured a service principal for automation. This section will install Ansible and the Azure Python SDK on the master VM that was created in the previous steps.

1. Install required software packages

**sudo yum install -y python-devel openssl-devel git gcc epel-release**

**sudo yum install -y ansible python-pip jq**

**sudo pip install --upgrade pip**

1. Install Azure Python SDK. At the time of this writing, the latest supported version is 2.0.0rc5. With this version, the package msrestazure needs to be installed independently. Additionally, we will install the package DNS Python so that we can do DNS checks in Ansible playbooks (to make sure that DNS names are not taken)

**sudo pip install azure==2.0.0rc5**

**sudo pip install msrestazure dnspython packaging**

1. We will clone some Github repositories, such as the ansible source code (which includes the dynamic inventory files such as azure\_rm.py), and the repository for this lab.

**git clone git://github.com/ansible/ansible.git --recursive**

**git clone git://github.com/erjosito/ansible-azure-lab**

1. Lastly, you need to create a new file in the directory ~/.azure using the credentials for the service principal generated in the previous sections. The filename is ~/.azure/credentials

**touch ~/.azure/credentials**

**cat <<EOF > ~/.azure/credentials**

**[default]**

**subscription\_id=22222222-2222-2222-2222-222222222222**

**client\_id=11111111-1111-1111-1111-111111111111**

**secret=ThisIsTheAppPassword**

**tenant=33333333-3333-3333-3333-333333333333**

**EOF**

***Note****: don’t forget to replace the numbers with the actual information you retrieved when you created the service principal*

1. And lastly, we will create a pair of private/public keys, and install the public key in the local machine, to test the correct operation of Ansible.

**ssh-keygen -t rsa**

Generating public/private rsa key pair.

Enter file in which to save the key (/home/lab-user/.ssh/id\_rsa):

Created directory '/home/lab-user/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/lab-user/.ssh/id\_rsa.

Your public key has been saved in /home/lab-user/.ssh/id\_rsa.pub.

The key fingerprint is:

81:86:f7:9c:6b:34:3a:5a:b2:d9:49:c4:8b:36:19:3b lab-user@ansibleMaster

The key's randomart image is:

+--[ RSA 2048]----+

| |

| . . |

| . + . |

| + o o |

| . o S |

| \* + o |

| E \* o |

| . @ + |

| + o |

+-----------------+

**chmod 755 ~/.ssh**

**touch ~/.ssh/authorized\_keys; chmod 644 ~/.ssh/authorized\_keys**

**ssh-copy-id lab-user@127.0.0.1**

You can verify that when trying to ssh to the local machine, no password will be requested:

[lab-user@ansibleMaster ~]$ **ssh 127.0.0.1**

Last login: Tue Jun 6 20:39:03 2017 from mymachine.mydomain.com

[lab-user@ansibleMaster ~]$

**What we have learnt**

Ansible can be installed on an Azure VM exactly the same as in other Linux systems

# Lab 4: Ansible dynamic inventory for Azure

Ansible allows to execute operations in machines that can be defined in a static inventory in the machine where Ansible runs. But what if you would like to run Ansible in all the machines in a resource group, but you don’t know whether it is one or one hundred? This is where dynamic inventories come into place, they discover the machines that fulfill certain requirements (such as existing in Azure, or belonging to a certain resource group), and makes Ansible execute operations on them.

1. In this first step we will test that the dynamic inventory script is running, executing it with the parameter “--list”. This should show JSON text containing information about all the VMs in your resource group.

**python ./ansible/contrib/inventory/azure\_rm.py --list | jq**

{

"azure": [

"ansibleMaster"

],

"westeurope": [

"ansibleMaster"

],

"ansibleMasterNSG": [

"ansibleMaster"

],

"ansiblelab": [

"ansibleMaster"

],

"\_meta": {

"hostvars": {

"ansibleMaster": {

"powerstate": "running",

"resource\_group": "ansiblelab",

"tags": {},

"image": {

"sku": "7.3",

"publisher": "OpenLogic",

"version": "latest",

"offer": "CentOS"

},

"public\_ip\_alloc\_method": "Dynamic",

"os\_disk": {

"operating\_system\_type": "Linux",

"name": "osdisk\_vD2UtEJhpV"

},

"provisioning\_state": "Succeeded",

"public\_ip": "52.174.19.210",

"public\_ip\_name": "masterPip",

"private\_ip": "192.168.1.4",

"computer\_name": "ansibleMaster",

...

}

}

}

}

***Note****: "jq" is a command-line JSON interpreter, that you can use here to make the JSON output readable. Try to use the previous command without the " | jq" part and see the effect.*

1. Now we can test Ansible functionality. But we will not change anything on the target machines, just test reachability with the Ansible function “ping”.

**ansible -i ./ansible/contrib/inventory/azure\_rm.py all -m ping**

The authenticity of host '1.2.3.4 (1.2.3.4)' can't be established.

ECDSA key fingerprint is 09:7f:7e:fc:34:d9:9f:ff:a6:5c:de:50:5a:5a:4f:14.

Are you sure you want to continue connecting (yes/no)? yes

ansibleMaster | SUCCESS => {

"changed": false,

"ping": "pong"

}

**ansible -i ./ansible/contrib/inventory/azure\_rm.py all -m ping**

ansibleMaster | SUCCESS => {

"changed": false,

"ping": "pong"

}

***Note****: The first time you run the command you will have to acknowledge the host's authenticity, after that it should run automatically*

1. If you already had VMs in your Azure subscription, they probably didn't pop up in the previous steps in this lab. The reason is because when we created the service principal, the scope was set to the resource group.  
   Still, you could refine the inventory script in order to return only the VMs in a certain resource group, in case your service principal had a broader scope. To that purpose, we will modify the .ini file that controls some aspects of azure\_rm.py. This .ini file is to be located in the same directory as the Python script: **~/ansible/contrib/inventory/azure\_rm.ini**. You need to find the line that specifies which resource groups are to be inspected, uncomment it and change it to something like this:

**resource\_groups=ansiblelab**

***Note****: edit the file* ***~/ansible/contrib/inventory/azure\_rm.ini*** *with a text editor such as vi. Note that you can filter not only per resource group, but per location, and more importantly, per tag.*

1. You can actually do much more with ansible, such as running any command on all the VMs returned by the dynamic inventory script, in this case “/bin/uname -a”

**ansible -i ~/ansible/contrib/inventory/azure\_rm.py all -m shell -a "/bin/uname -a"**

***Note****: the command "uname -a" returns some information about the machine where it is executed, such as the Kernel version, the date and time and the CPU architecture*

**What we have learnt**

Dynamic Inventory is an Ansible feature that allows for a certain operation to be executed on a list of VMs which is not defined statically, but is evaluated at execution time. For example, on all VMs in Azure in a certain resource group or a certain location.

# Lab 5: Creating a VM using an Ansible Playbook

Now that we have Ansible up and running, we can deploy our first playbook in order to create a VM. This playbook will not be executed using the dynamic inventory function, but on the localhost. This will trigger the necessary calls to Azure so that all required objects are created. We will be using the playbook example that was cloned from the Github repository for this lab in previous sections, which you should have stored in **~/ansible-azure-lab/new\_vm\_web.yml**.

1. You need to change the public SSH key that you will find inside of **~/ansible-azure-lab/new\_vm\_web.yml** with your own key, which you can find using this command:

**cat ~/.ssh/id\_rsa.pub**

***Note****: edit the file* **~/ansible-azure-lab/new\_vm\_web.yml** *with a text editor such as vi from the command line. You can find the SSH string towards the end. It is important that the username matches the user name in your ansibleMaster VM ("lab-user", if you followed the lab guide correctly)*

1. You can use the following commands to double check the vnet and subnet that were used to create the master VM. that information (note that the outputs have been truncated so that they fit to the width of this document).

**az network vnet list -o table**

Location Name ProvisioningState ResourceGroup ResourceGuid

---------- ----------- ------------------- --------------- -------------

westeurope **ansibleVnet** Succeeded ansiblelab ...

**az network vnet subnet list --vnet-name ansibleVnet -o table**

AddressPrefix Name ProvisioningState ResourceGroup

--------------- ------------- ------------------- ---------------

192.168.1.0/24 **ansibleSubnet** Succeeded ansiblelab

1. Now we have all the information we need, and we can run all playbook with all required variables. Note that variables can be defined inside of playbooks, or can be entered at runtime along the ansible-playbook command with the “**--extra-vars**” option. As VM name please use **only lower case letters and numbers** (no hyphens, underscore signs or upper case letters), and a unique name, for example, using your birthday as suffix).

**ansible-playbook ~/ansible-azure-lab/new\_vm\_web.yml --extra-vars "vmname=web19761013 resgrp=ansiblelab vnet=ansibleVnet subnet=ansibleSubnet"**

[WARNING]: provided hosts list is empty, only localhost is available

PLAY [CREATE VM PLAYBOOK] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [debug] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost] => {

"changed": false,

"msg": "Public DNS name web2017060673.westeurope.cloudapp.azure.com resolved to IP NXDOMAIN. "

}

TASK [Check if DNS is taken] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

skipping: [localhost]

TASK [Create storage account] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost]

TASK [Create security group that allows SSH and HTTP] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost]

TASK [Create public IP address] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost]

TASK [Create NIC] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost]

TASK [Create VM] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : **ok=6 changed=5 unreachable=0 failed=0**

***Note****: some errors you might get at this step, if you enter a "wrong" VM name (see the appendix for more details):*

* *fatal: [localhost]: FAILED! => {"changed": false, "failed": true, "msg": "The storage account named your-vm-name is already taken. - Reason.already\_exists"}  
  Resolution: use another name for your VM, that one seems to be already taken*
* *fatal: [localhost]: FAILED! => {"changed": false, "failed": true, "msg": "Error creating or updating 19761113web01 - Azure Error: InvalidDomainNameLabel\nMessage: The domain name label 19761113web01 is invalid. It must conform to the following regular expression: ^[a-z][a-z0-9-]{1,61}[a-z0-9]$."}  
  Resolution: use another name for your VM following the naming syntax. In this case, the problem was that VM names should not start with a number, but with a lower case letter*

1. While the playbook is running, have a look in another console inside of the file **~/ansible-azure-lab/new\_vm\_web.yml**, and try to identify the different parts it is made out of.
2. You can run the dynamic inventory, to verify that the new VM is now detected by Ansible

**python ./ansible/contrib/inventory/azure\_rm.py --list | jq**

{

"westeurope": [

"ansibleMaster",

"web2017060673"

],

"web2017060673": [

"web2017060673"

],

"\_meta": {

"hostvars": {

"ansibleMaster": {

"powerstate": "running",

"resource\_group": "ansiblelab",

...

"ansible\_host": "52.174.19.210",

"name": "ansibleMaster",

},

"web2017060673": {

"powerstate": "running",

"resource\_group": "ansiblelab",

...

"ansible\_host": "52.174.198.220",

"name": "web2017060673",

"fqdn": "web2017060673.westeurope.cloudapp.azure.com",

}

}

},

"ansibleMasterNSG": [

"ansibleMaster"

],

"azure": [

"ansibleMaster",

"web2017060673"

],

"ansiblelab": [

"ansibleMaster",

"web2017060673"

]

}

1. Using the dynamic inventory, run the ping test again, to verify that the dynamic inventory file can see the new machine. The first time you run the test you will have to verify the SSH host key, but successive attempts should run without any interaction being required:

**$ ansible -i ~/ansible/contrib/inventory/azure\_rm.py all -m ping**

The authenticity of host '52.174.47.97 (52.174.47.97)' can't be established.

ECDSA key fingerprint is 48:89:dc:6d:49:77:2d:85:50:6b:73:90:70:c6:05:5c.

Are you sure you want to continue connecting (yes/no)? ansibleMaster | SUCCESS => {

"changed": false,

"ping": "pong"

}

yes

your-vm-name | SUCCESS => {

"changed": false,

"ping": "pong"

}

**$ ansible -i ~/ansible/contrib/inventory/azure\_rm.py all -m ping**

vm-00 | SUCCESS => {

"changed": false,

"ping": "pong"

}

your-vm-name | SUCCESS => {

"changed": false,

"ping": "pong"

}

***Note****: the first time you connect to the new VM you need to manually accept the SSH fingerprint, further attempts will work without manual intervention*

**What we have learnt**

Ansible playbooks can be used not only to interact with Linux VMs running on Azure, but with Azure itself. In this section we used an Ansible playbook (which is executed against the local host) to create a new Linux VM in Azure.

As with any Ansible deployment, getting password-less SSH authentication right is a critical step. For that purpose, the creation of the VM in Azure needs to make sure that the right users with the right SSH public keys are deployed.

# Lab 6: Running an Ansible playbook on the new VM

In this section we will run another Ansible playbook, this time to configure the newly created machine. As example, we will run a very simple playbook that installs a software package (httpd) and downloads an HTML page from a Github repository. If everything works, after running the playbook you will have a fully functional Web server.

You will probably be thinking that if the purpose of the exercise is creating a Web server, there are other quicker ways in Azure to do that, for example, using Web Apps, or Azure Linux Script Extensions. Please consider that we are using this as example, you could be running an Ansible playbook to do anything that Ansible supports, and that is a lot.

1. We will be using the example playbook that was downloaded from Github **~/ansible-azure-lab/httpd.yml**. Additionally, we will be using the variable “vmname” in order to modify the “hosts” parameter of the playbook, that defines on which host (out of the ones returned by the dynamic inventory script) the playbook will be run. First we will verify that there is no Web server running on the VM:

**curl** [**http://your-vm-name.westeurope.cloudapp.azure.com**](http://19761013web01.westeurope.cloudapp.azure.com)

curl: (7) Failed connect to your-vm-name.westeurope.cloudapp.azure.com:80; Connection refused

And now we will install the HTTP server with our Ansible playbook:

**ansible-playbook -i ~/ansible/contrib/inventory/azure\_rm.py ~/ansible-azure-lab/httpd.yml --extra-vars "vmname=your-vm-name"**

PLAY [Install Apache Web Server] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Ensure apache is at the latest version] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [your-vm-name]

TASK [Change permissions of /var/www/html] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [your-vm-name]

TASK [Download index.html] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [your-vm-name]

TASK [Ensure apache is running (and enable it at boot)] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [your-vm-name]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

your-vm-name : **ok=4 changed=4 unreachable=0 failed=0**

1. Now you can test that there is a Web page on our VM using your Internet browser and trying to access the location **http://your-vm-name.westeurope.cloudapp.azure.com**, or using curl from the master VM

**curl** [**http://your-vm-name.westeurope.cloudapp.azure.com**](http://19761013web01.westeurope.cloudapp.azure.com)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Hello World</title>

</head>

<body>

<h1>Hello World</h1>

<p>

<br>This is a test page

<br>This is a test page

<br>This is a test page

</p>

</body>

</html>

**What we have learnt**

Once the VM is created in Azure, Ansible can be used to configure it via standard Ansible playbooks. Using dynamic inventories is not necessary to have a static list of the existing VMs in Azure.

# Lab 7: Deleting a VM using Ansible (Optional)

Finally, similarly to the process to create a VM we can use Ansible to delete it, making sure that associated objects such storage account, NICs and Network Security Groups are deleted as well. For that we will use the playbook in this lab’s repository delete\_vm.yml:

1. Now you can test that there is a Web page on our VM using your Internet browser and trying to access the location **http://your-vm-name.westeurope.cloudapp.azure.com**.

**ansible-playbook ~/ansible-azure-lab/delete\_vm.yml --extra-vars "vmname=your-vm-name resgrp=ansiblelab"**

[WARNING]: provided hosts list is empty, only localhost is available

PLAY [Remove Virtual Machine and associated objects] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Remove VM and all resources] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost]

TASK [Remove storage account] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=2 changed=0 unreachable=0 failed=0

1. Verify that the VM does not exist any more using Ansible’s dynamic inventory functionality:

**ansible -i ~/ansible/contrib/inventory/azure\_rm.py all -m ping**

**What we have learnt**

Ansible playbooks can be used for the full lifecycle of a VM. Not only creation and management, but deletion as well.

# Conclusion

In this lab we have seen how to use Ansible for two purposes:

On one side, Ansible can be used in order to create VMs, in a similar manner than Azure quickstart templates. If you already know Ansible and prefer using Ansible playbooks instead of native Azure JSON templates, you can certainly do so.

On the other side, and probably more importantly, you can use Ansible in order to manage the configuration of all your virtual machines in Azure. Whether you have one VM or one thousand, Ansible will discover all of them (with its dynamic inventory functionality) and apply any playbooks that you have defined, making server management at scale much easier.

All in all, the purpose of this lab is showing to Ansible admins that they can use the same tools in Azure as in their on-premises systems.

# End the lab

To end the lab, simply delete the resource group that you created in the first place (**ansiblelab** in our example) from the Azure portal or from the Azure CLI:

**az group delete --name ansiblelab**

Optionally, you can delete the service principal and the application that we created at the beginning of this lab:

**az ad sp show --id 44444444-4444-4444-444444444444**

{

"appId": "**11111111-1111-1111-1111-111111111111**",

"displayName": "ansibleApp",

"objectId": "**44444444-4444-4444-444444444444**",

"objectType": "ServicePrincipal",

"servicePrincipalNames": [

"ansible.mydomain.com",

"**11111111-1111-1111-1111-111111111111**"

]

}

**az ad sp delete --id 44444444-4444-4444-444444444444**

For the application we need application ID that we can get out of the "az ad sp show" command above:

**az ad app show --id 11111111-1111-1111-1111111111**

{

"appId": "11111111-1111-1111-1111111111",

"appPermissions": null,

"availableToOtherTenants": false,

"displayName": "ansibleApp",

"homepage": "ansible.mydomain.com",

"identifierUris": [

"ansible.mydomain.com"

],

"objectId": "55555555-5555-5555-5555-555555555555",

"objectType": "Application",

"replyUrls": []

}

**az ad app delete --id 11111111-1111-1111-1111111111**

## **References**

Useful links:

* Ansible web page: <https://www.ansible.com>
* Azure portal: <https://portal.azure.com>
* Using CLI to créate a Service Principal: <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-authenticate-service-principal-cli>
* Ansible documentation – Getting started with Azure: <https://docs.ansible.com/ansible/guide_azure.html>
* Azure CLI installation on Linux and Mac: <https://azure.microsoft.com/en-us/downloads/cli-tools-install/>
* vi cheat sheet: <http://www.lagmonster.org/docs/vi.html>