Microsoft

Jose Moreno

PDU CSA-P

12/12/2016

Using Ansible to manage Azure workloads

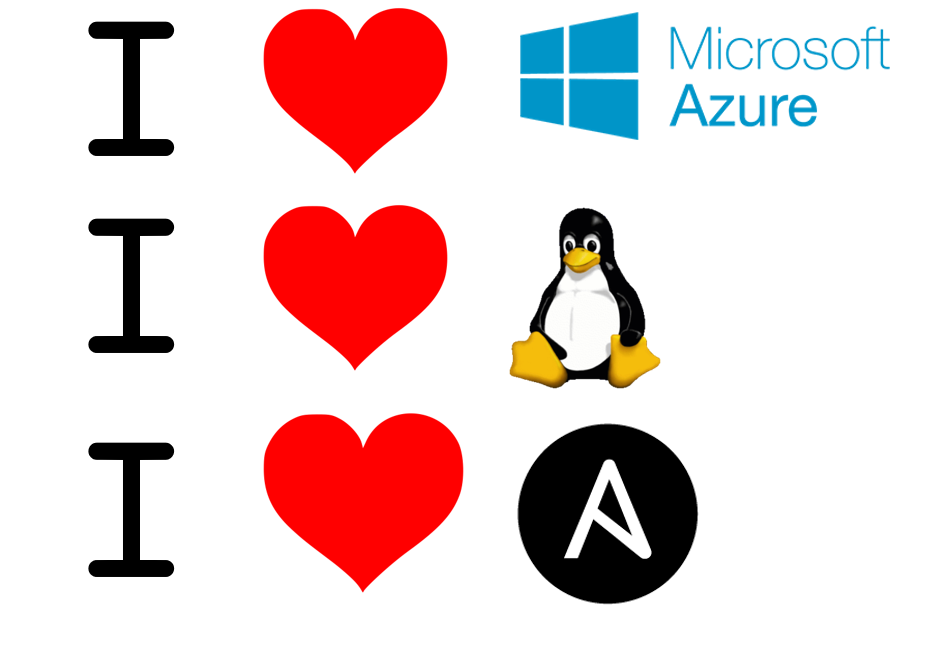


Table of Contents

[Objectives and initial setup 2](#_Toc469611494)

[Introduction to Ansible 3](#_Toc469611495)

[Lab 1: Create Control VM using Azure CLI 4](#_Toc469611496)

[Lab 2: Create Service Principal 5](#_Toc469611497)

[Lab 3: Install Ansible in the provisioning VM 7](#_Toc469611498)

[Lab 4: Ansible dynamic inventory for Azure 9](#_Toc469611499)

[Lab 5: Creating a VM using an Ansible Playbook 11](#_Toc469611500)

[Lab 6: Running an Ansible playbook on the new VM 15](#_Toc469611501)

[Lab 7: Deleting a VM using Ansible (Optional) 17](#_Toc469611502)

[End the lab 18](#_Toc469611503)

[**References** 19](#_Toc469611504)

# 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:

* 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.
* If you are using Windows 10, you can [install Bash shell on Ubuntu on Windows](http://www.windowscentral.com/how-install-bash-shell-command-line-windows-10) (<http://www.windowscentral.com/how-install-bash-shell-command-line-windows-10>). To install Azure CLI, download and [install the latest Node.js and npm](https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions) for Ubuntu (<https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions>). Then, follow the [instructions](https://azure.microsoft.com/en-us/documentation/articles/xplat-cli-install/) (**Option-1**): <https://azure.microsoft.com/en-us/documentation/articles/xplat-cli-install/>
* If you are using MAC or another windows version, install Azure CLI, following **Option-2**: <https://azure.microsoft.com/en-us/documentation/articles/xplat-cli-install/>

The labs 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 | 19761013web01 |

# 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.

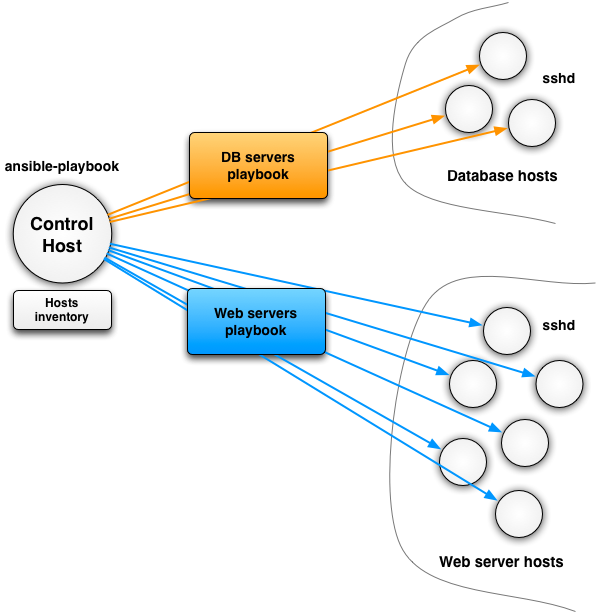


Figure 1. Overall architecture example from Ansible to configure Database and Web servers

# Lab 1: Create Control VM using Azure CLI

1. Create provisioning machine using Azure CLI from a Linux shell (because we will connect to a new VM using SSH public/private key authentication).

**azure login**

**azure group create ansiblelab westeurope**

**azure vm quick-create -g ansiblelab -n vm-00 -l westeurope -w 19761013myvm -u lab-user -M .ssh/id\_rsa.pub -p Microsoft123! -Q "OpenLogic:CentOS:7.2:latest" -s "Visual Studio Enterprise" -y Linux**

**ping 19761013myvm.westeurope.cloudapp.azure.com**

1. Connect to the newly created machine

**ping 19761013myvm.westeurope.cloudapp.azure.com**

1. Install Azure CLI in the provisioning machine vm00

**sudo yum update -y**

**curl --silent –location** [**https://rpm.nodesource.com/setup\_4.x**](https://rpm.nodesource.com/setup_4.x) **| sudo bash -**

**sudo yum install -y nodejs**

**sudo npm install azure-cli -g**

# Lab 2: Create Service Principal

See for more information: <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-authenticate-service-principal-cli>)

1. Create Service Principal:

**$ azure ad sp create -n ansiblelab -p ThisIsTheAppPassword**

info: Executing command ad sp create

+ Creating application ansiblelab

+ Creating service principal for application **11111111-1111-1111-1111-111111111111**

data: Object Id: 44444444-4444-4444-4444-444444444444

data: Display Name: ansiblelab

data: Service Principal Names:

data: 11111111-1111-1111-1111-111111111111

data: <http://ansiblelab>

info: ad sp create command OK

1. Find out your subscription and tenant IDs:

**$ azure account show**

info: Executing command account show

data: Name : Your Subscription Name

data: ID : **22222222-2222-2222-2222-222222222222**

data: State : Enabled

data: Tenant ID : **33333333-3333-3333-3333-333333333333**

data: Is Default : true

data: Environment : AzureCloud

data: Has Certificate : No

data: Has Access Token : Yes

data: User name : youremail@yourcompany.com

data:

info: account show command OK

1. Assign the Contributor role to the principal for your subscription, using the object ID for the service principal:

**$ azure role assignment create --objectId 44444444-4444-4444-4444-444444444444 -o** Contributor -c /subscriptions/22222222-2222-2222-2222-222222222222/

info: Executing command role assignment create

+ Finding role with specified name

-data: RoleAssignmentId : /subscriptions/22222222-2222-2222-2222-222222222222/providers/Microsoft.Authorization/roleAssignments/86e6bcd4-8061-41d1-8a7f-74219a268d51

data: RoleDefinitionName : Contributor

data: RoleDefinitionId : b24988ac-6180-42a0-ab88-20f7382dd24c

data: Scope : /subscriptions/22222222-2222-2222-2222-222222222222

data: Display Name : ansiblelab

data: SignInName : undefined

data: ObjectId : 44444444-4444-4444-4444-444444444444

data: ObjectType : ServicePrincipal

data:

+

info: role assignment create command OK

Note the following values of your output, since we will use them later. In this guide they are marked in different colors 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**

# Lab 3: Install Ansible in the provisioning VM

This section will install Ansible and the Azure Python SDK on the provisioning 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**

**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**

**sudo pip install dnspython**

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/Azure-Ansible-Examples**

1. Lastly, you need to create a new file in the directory ~/.azure (create it if it does not exist), using the credentials generated in the previous sections. The filename is ~/.azure/credentials

**mkdir ~/.azure**

**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**

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**

**chmod 755 ~/.ssh**

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

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

# 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 subscription.

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

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**

1. If you already had VMs in your Azure subscription, they probably popped up in the previous steps in this lab. We can refine the inventory script in order to return only the VMs in a certain resource group. 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**

1. Now you can do again the reachability test with “ping”, and verify that only the provisioning VM (the only VM in our resource group) is tested.

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

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 ./azure\_rm.py all -m shell -a "/bin/uname -a"**

# 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 **~/Azure-Ansible-Examples/azure-playbooks/new\_vm\_web.yml**.

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

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

1. You will need to write the IP address of the provisioning machine, that you can find with this command.

**ip a**

1. Now we need to pieces of information so that we can place the new VM in the same subnet as the provisioning VM: the vnet and the subnet. You can use the following commands to find that information (note that the outputs have been truncated so that they fit to the width of this document):

**$ azure network vnet list -g ansiblelab**

**info: Executing command network vnet list**

**+ Looking up virtual networks**

**data: Name Location Resource group**

**data: ----------------------------- ---------- --------------**

**data: vm-00-weste-hl2w86f529j7-vnet westeurope ansiblelab**

**info: network vnet list command OK**

**> azure network vnet subnet list -e vm-00-weste-hl2w86f529j7-vnet -g ansiblelab**

**info: Executing command network vnet subnet list**

**+ Looking up the virtual network "vm-00-weste-hl2w86f529j7-vnet"**

**+ Getting virtual network subnets**

**data: Name Provisioning state Address prefix**

**data: ----------------------------- ------------------ --------------**

**data: vm-00-weste-hl2w86f529j7-snet Succeeded 10.0.1.0/24**

**info: network vnet subnet list command OK**

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, prefixing it with your birthday).

**ansible-playbook ~/Azure-Ansible-Examples/azure-playbooks/new\_vm\_web.yml --extra-vars "vmname=19761013web01 resgrp=ansiblelab vnet=vm-00-weste-hl2w86f529j7-vnet subnet=vm-00-weste-hl2w86f529j7-snet"**

1. While the playbook is running, have a look in another console inside of the file **~/Azure-Ansible-Examples/azure-playbooks/new\_vm\_web.yml**, and try to identify the different parts it is made out of. When the playbook has been executed successfully, the output should be similar to this one. If it is not, check the appendix for possible error causes:

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

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

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

**ok: [localhost] => {**

**"msg": "Public DNS name web011138.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**

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)? vm-00 | SUCCESS => {**

**"changed": false,**

**"ping": "pong"**

**}**

**yes**

**19761013web01 | SUCCESS => {**

**"changed": false,**

**"ping": "pong"**

**}**

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

**vm-00 | SUCCESS => {**

**"changed": false,**

**"ping": "pong"**

**}**

**19761013web01 | SUCCESS => {**

**"changed": false,**

**"ping": "pong"**

**}**

# 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. 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 **~/Azure-Ansible-Examples/azure-playbooks/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.

**$ ansible-playbook -i ~/ansible/contrib/inventory/azure\_rm.py ~/Azure-Ansible-Examples/azure-playbooks/httpd.yml --extra-vars "vmname=19761013web01"**

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

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

changed: [19761013web01]

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

changed: [19761013web01]

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

changed: [19761013web01]

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

changed: [19761013web01]

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

19761013web01 : 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://19761013web01.westeurope.cloudapp.azure.com**.

# 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://19761013web01.westeurope.cloudapp.azure.com**.

**$ ansible-playbook ~/Azure-Ansible-Examples/azure-playbooks/delete\_vm.yml --extra-**vars "vmname=19761013myweb 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**

# 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:

**$ azure group delete ansiblelab**

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

**$ azure ad sp show -o 44444444-4444-4444-444444444444**

info: Executing command ad sp show

+ Getting Active Directory service principals

data: Object Id: 44444444-4444-4444-444444444444

data: Display Name: ansiblelab

data: Service Principal Names:

data: http://ansiblelab

data: 11111111-1111-1111-1111111111

data:

info: ad sp show command OK

**$ azure ad sp delete -o 44444444-4444-4444-444444444444**

For the application we first need to find out the object ID, out of the application ID

**$ azure ad app show -a 11111111-1111-1111-1111111111**

info: Executing command ad app show

+ Getting Active Directory application(s)

data: AppId: 11111111-1111-1111-1111111111

data: ObjectId: 55555555-5555-5555-555555555555

data: DisplayName: ansiblelab

data: IdentifierUris: 0=http://ansiblelab

data: ReplyUrls:

data: AvailableToOtherTenants: False

data: HomePage: http://ansiblelab

data:

info: ad app show command OK

**$ azure ad app delete -o 55555555-5555-5555-555555555555**

## **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/>