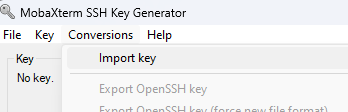
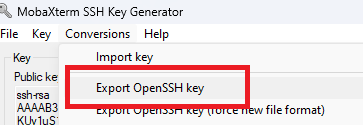
**ANSIBLE AND TERRAFORM LAB**  
**PREPARATOTY STEPS**

1. Generate a SSH key (public and private) that will be used during your VM deployments with Terraform.
2. Import the private key to your preferred ssh key generator as shown in the screenshot below.  
   
3. Convert the ssh private key from .ppk to openssh by clicking the “**Export OpenSSH Key**”. See screenshot below  
   

This essence for these steps is to enable the ansible master VM connect to the slave VMs to perform custom configurations. The file extension .ppk indicates that this is a PuTTY Private Key file, which is not directly compatible with OpenSSH which is the SSH client used in Linux. To use a .ppk file with OpenSSH, you typically need to convert it to the OpenSSH format.

1. After converting the private key to OpenSSH, save the new private and public key which will be used for our VM deployment with Terraform.
2. Move the private and public key to the directory hosting the configuration files for your VM(s) deployment. This makes terraform easily pickup the public key to be used during the VM creation.

**TERRAFORM STEPS**

To select the public key, we created to be applied to the VM, the terraform code block for this action will look like this.

***os\_profile\_linux\_config {***

***disable\_password\_authentication = false***

***ssh\_keys {***

***#Path to the directory where the SSH Key will be stored for the user created during VM Creation***

***path = "/home/${var.admin\_username}/.ssh/authorized\_keys"***

***#Location of Public Key in the local system when the SSH Key was generated***

***key\_data = file("${path.module}/VMSSHpublickey.pub")***

***}***

***}***

**var.admin\_username** is the name of the default user required when creating a VM on Azure. The name is specified in your .tfvars file.

Next, we need terraform to write the public IP address of the VM(s) that would be deployed with terraform so we can easily create an ansible inventory file. This step is very useful if you are creating multiple VMs.

The output code block will look like this.

***#IP address collection for Ansible Inventory***

***output "public\_ip\_addresses" {***

***value       = [for vm in azurerm\_public\_ip.example : vm.ip\_address]***

***}***

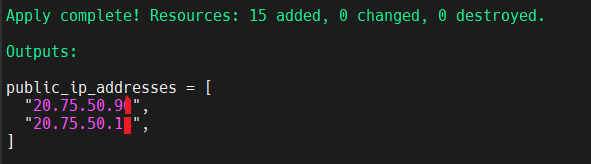
**N/B:** This assumes I added the count index in my terraform code to enable me deploy multiple VMs.

**IMPORTANT INFORMATION**

* For this lab, I have an existing VM where I have installed Ansible and this VM will be used as the Master VM.
* Secondly, I am running terraform on a Windows Machine but using Mobaxterm to give me a Linux environment for executing Linux commands on my PC.
* Upload the private key we created to the directory of your choice using Mobaxterm or WinSCP. This will be important as we need to specify the path of our private key in our Master VM that will be used to create our inventory file. (For this lab, the private key is in my **/home/Nezie/Ansible** directory).

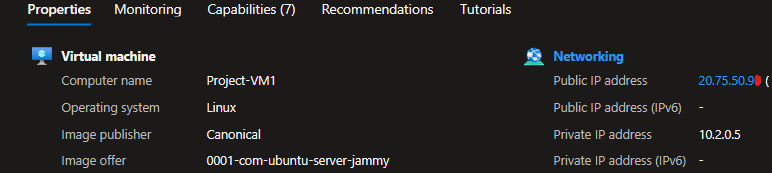
After successful deployment, the terraform output block we specified in our code will list out the IP addresses of all the VMs created in serial order.   
**Example:** In this lab, I deployed 2 VMs with name Project-VM. From the screenshots below, we can see VM 1 took the first IP address and VM 2 took the second IP address.

**Terraform Output**

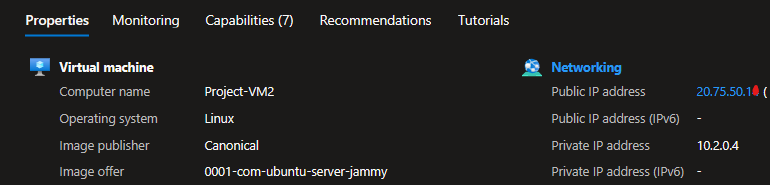


**VMs on Azure Portal**

**VM 1**

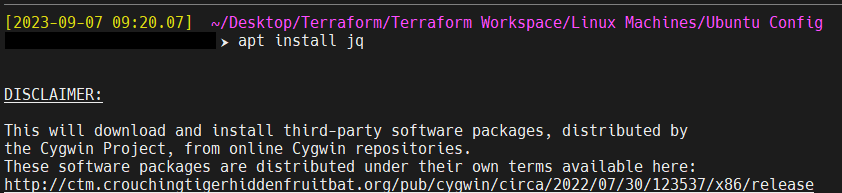


**VM 2**



**CREATING ANSIBLE INVENTORY FILE**

To create an inventory file, we will need to install a JSON processor that will enable us to execute our script to create our ansible inventory file. To do this, we need to run the command **apt/yum install jq -y**. However, for this lab, the command used was **apt install jq -y** since Mobaxterm uses the apt package manager.



Next, we create our script named **generate-inventory.sh** that will be used to create our ansible inventory file for this lab.

***#!/bin/bash***

***# Create the inventory file***

***echo "[servers]"***

***# Use Terraform output directly to get the list of public IP addresses***

***terraform output -json public\_ip\_addresses | jq -r '.[]' | while read ip; do***

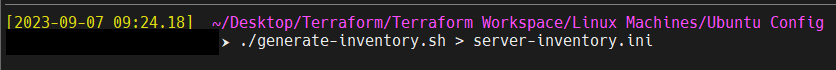
***echo "server$((++i)) ansible\_host=$ip ansible\_user=Nezie ansible\_ssh\_private\_key\_file=/home/Nezie/Ansible/VMSSHprivatekey\_openSSH"***

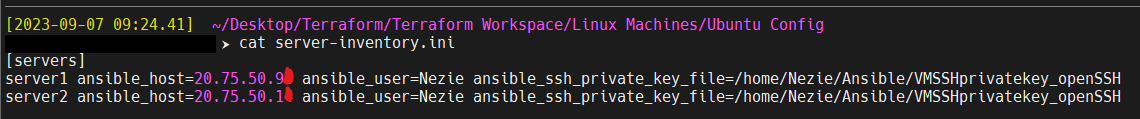
***done***

In this script, we use the **public\_ip\_addresses output variable** defined in our Terraform configuration to obtain the list of public IP addresses of our Azure VMs and loop through the public IP addresses using **jq** to parse the JSON output.

Also, this script will write the user created during our VM deployment and the path of the private key uploaded to our master VM. This would ensure Ansible is using the correct key and username to access the VMs which we want to make our configurations on.

Next, we proceed to execute the **generate-inventory.sh** script and write the output to our ansible inventory file named **server-inventory.ini**. the command and output of the file will look like the screenshot below.





After creating our inventory file, we will upload it to our Master VM using Mobaxterm or WinSCP.

**EXECUTING CONFIGURATIONS WITH ANSIBLE**

On our master VM, I have created a script and playbook for each of the tasks I want to execute with Ansible.

For this Lab, I am executing 3 tasks.

1. **Custom-vmconfig task** - This will make the default user Nezie to execute sudo commands without password prompt, set the time of the VMs to West Africa Time zone (WAT), install nmap and net-tools. (Check custom-vmconfig directory for script and playbook).
2. **Apps-installation task** – This will install GUI and docker on each of the VMs. After completing this task, manually reboot the VMs.
3. **GUI-Config task** – This will change the GUI display username from Ubuntu to the default username Nezie.

***Check the Ansible directory for the scripts and playbooks used for this lab.***

**PLAYBOOK EXECUTION COMMAND**

The command used for executing these tasks will look this way.

**ansible-playbook /path/to/ansible-file/playbook.yaml -i /path/to/ansible-inventory/file.ini**

For this lab, see an example of the command used to apply my custom configurations on my slave VMs.

**ansible-playbook** **/home/Nezie/Ansible/custom-vmconfig/custom-vmconfig-playbook.yaml** **-i /home/Nezie/Ansible/server-inventory.ini** **-vv &> /home/Nezie/Ansible/custom-vmconfig.txt**

* **ansible-playbook** – This is the command for executing the playbook.
* **/home/Nezie/Ansible/custom-vmconfig/custom-vmconfig-playbook.yaml** – This is the path to my custom-vmconfig playbook to execute my custom configurations on the slave VMs.
* **-i /home/Nezie/Ansible/server-inventory.ini** - -**i** is used to specify the path containing our inventory file (/home/Nezie/Ansible/server-inventory.ini).
* **-vv &>** **/home/Nezie/Ansible/custom-vmconfig.txt** – This gives a verbose out put and redirects the standard error and output from the playbook execution to the file /home/Nezie/Ansible/custom-vmconfig.txt

To execute other tasks, change the paths in the ansible-playbook command accordingly.

**RESULTS**

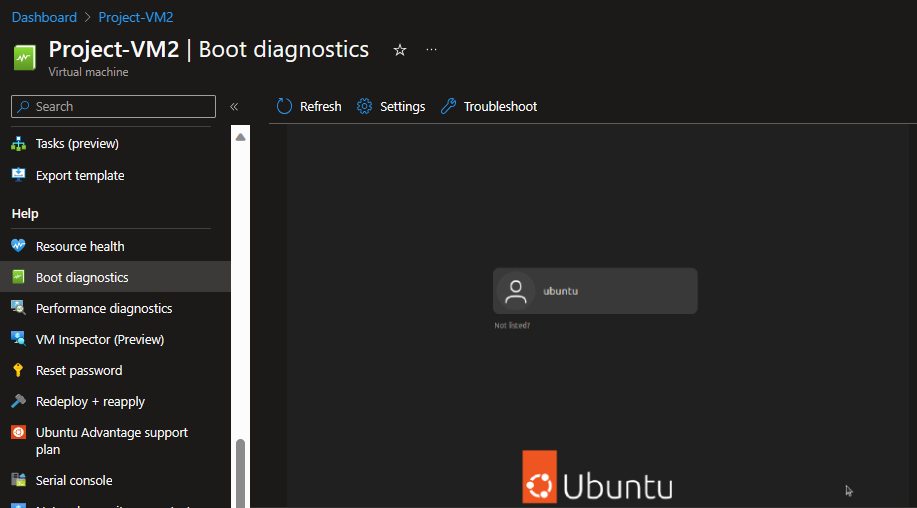
A screenshot of a computer

Description automatically generated

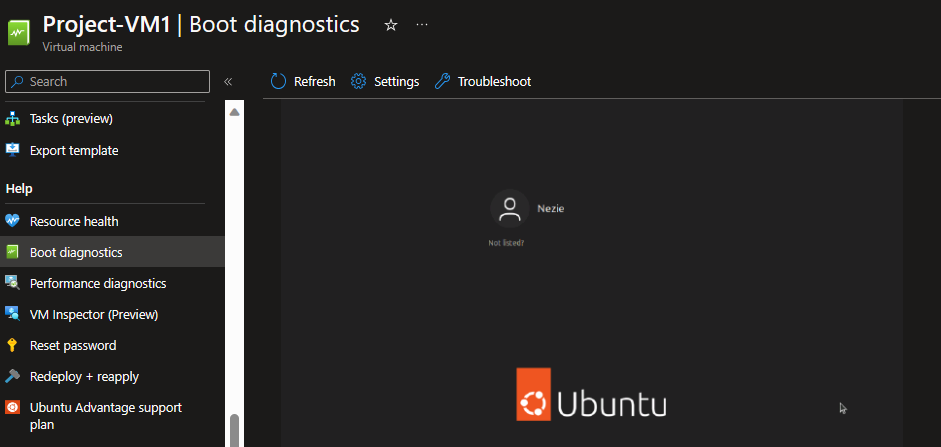
From the screenshot above, we can see that Ansible successfully made the required configurations:

* setting system default timezone to WAT
* allowing user Nezie to run sudo commands without password prompt
* installed docker
* ensured time and date are recorded for each command executed on the VM.

From the above screen screenshots, we can see that GUI is already configured on the VMs. This can also be confirmed from the boot diagnostic screenshots below.



For Task 3, I do not want the GUI display username to be ubuntu, rather I want the GUI to display the system default username – Nezie. To do this, we executed the gui-config playbook in task 3 to get the results below.



A screenshot of a computer

Description automatically generated