# IAC-1 Deploying a Kubernetes Cluster to ONE with Ansible and Terraform

## Installing Terraform

To install Terraform, find the appropriate package for your system and download it

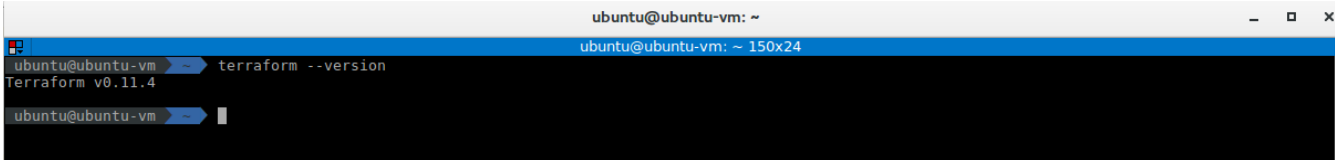
```
$ curl -O https://releases.hashicorp.com/terraform/0.11.4/terraform_0.11.4_linux_amd64.zip
```

After downloading Terraform, unzip the package

```
$ sudo mkdir /bin/terraform
$ sudo unzip terraform_0.11.4_linux_amd64.zip -d /bin/terraform
```

After installing Terraform, verify the installation worked by opening a new terminal session and checking that terraform is available.

```
$ export PATH=$PATH:/bin/terraform
$ terraform --version
```




## Installing Terraform provider Opennebula

You need to install go first: *https://golang.org/doc/install*

Install Prerequisites

```
$ sudo apt install bzr
```

Use the `wget` command and the link from Go to download the tarball:

```
$ wget https://dl.google.com/go/go1.10.linux-amd64.tar.gz
```

The installation of Go consists of extracting the tarball into the `/usr/local`

```
$ sudo tar -C /usr/local -xvzf  go1.10.linux-amd64.tar.gz
```

We will call our workspace directory projects, but you can name it anything you would like. The `-p` flag for the `mkdir` command will create the appropriate directory tree

```
$ mkdir -p ~/projects/{bin,pkg,src}
```

To execute Go like any other command, we need to append its install location to the $PATH variable.

```
$ export PATH=$PATH:/usr/local/go/bin
```

Additionally, define the GOPATH and GOBIN Go environment variables:

```
$ export GOBIN="$HOME/projects/bin"
$ export GOPATH="$HOME/projects/src"
```

After go is installed and set up, just type:

```
$ go get github.com/runtastic/terraform-provider-opennebula
$ go install github.com/runtastic/terraform-provider-opennebula
```

## Optional post-installation Step

Copy your **terraform-provider-opennebula** binary in a folder, like `/usr/local/bin`, and write this in `~/.terraformrc`:

```
 $ sudo cp ~/projects/bin/terraform-provider-opennebula /usr/local/bin/terraform-provider-opennebula
```

```
providers {
  opennebula = "$YOUR_PROVIDER_PATH"
}
```

Example for `/usr/local/bin`:

```
providers {
 opennebula = "/usr/local/bin/terraform-provider-opennebula"
}
```

## Install Ansible

We can add the Ansible PPA by typing the following command:

```
$ sudo apt-add-repository ppa:ansible/ansible
```

Next, we need to refresh our system's package index so that it is aware of the packages available in the PPA. Afterwards, we can install the software:

```
$ sudo apt-get update
$ sudo apt-get install ansible
```

## Deploy a Kubernetes cluster

Terraform code is written in a language called HCL in files with the extension “.tf”. It is a declarative language, so your goal is to describe the infrastructure you want, and Terraform will figure out how to create it.

This repository provide an Ansible playbook to Build a Kubernetes cluster with kubeadm. The goal is easily install a Kubernetes cluster on machines running `CentOS 7`

```
$ git clone https://github.com/CSUC/terransible-kubernetes-cluster
```

First, initialize Terraform for your project. This will read your configuration files and install the plugins for your provider:

```
$ terraform init
```

```
ubuntu@ubuntu-vm  ~/git/terransible-kubernetes-cluster  master ●
ubuntu@ubuntu-vm  ~/git/terransible-kubernetes-cluster  master ●  terraform init

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.null: version = "~> 1.0"
* provider.template: version = "~> 1.0"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
ubuntu@ubuntu-vm  ~/git/terransible-kubernetes-cluster  master ●
```

In a terminal, go into the folder where you created *main.tf*, and run the `terraform plan` command:

```
data.template_file.tf-kube-template: Refreshing state...

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + null_resource.kubernetes
      id:          <computed>

  + opennebula_template.tf-kube-template
      id:          <computed>
      description: "CONTEXT = [\n  NETWORK = \"YES\",\n  SSH_PUBLIC_KEY = \"$USER[SSH_PUBLIC_KEY]\",\n  SET_HOSTNAME=\"$NAME\",\n  USERNAME = \"$UN
AME\"\n]\nCPU = \"0.25\"\nDISK = [\n  IMAGE_ID = \"4\" ]\nGRAPHICS = [\n  LISTEN = \"0.0.0.0\",\n  TYPE = \"VNC\" ]\nINPUTS_ORDER = \"\"\nLOGO = \"
images/logos/centos.png\"\nMEMORY = \"1024\"\nMEMORY_UNIT_COST = \"MB\"\nNIC = [\n  NETWORK = \"default\",\n  NETWORK_UNAME = \"oneadmin\" ]\nOS =
[\n  ARCH = \"x86_64\",\n  BOOT = \"\" ]\n\n\n"
      gid:         <computed>
      gname:       <computed>
      name:        "terraform-kube-template"
      permissions: "600"
      reg_time:    <computed>
      uid:         <computed>
      uname:       <computed>

  + opennebula_vm.kube-master-vm
      id:          <computed>
      gid:         <computed>
      gname:       <computed>
      instance:    <computed>
      ip:          <computed>
      lcmstate:    <computed>
      name:        "terraform-kube-master"
      permissions: "600"
      state:       <computed>
      template_id: "0"
      uid:         <computed>
      uname:       <computed>

  + opennebula_vm.kube-node-vm
      id:          <computed>
      gid:         <computed>
      gname:       <computed>
      instance:    <computed>
      ip:          <computed>
      lcmstate:    <computed>
      name:        "terraform-kube-node0"
      permissions: "600"
      state:       <computed>
      template_id: "0"
      uid:         <computed>
      uname:       <computed>
```

The plan command lets you see what Terraform will do before actually doing it.

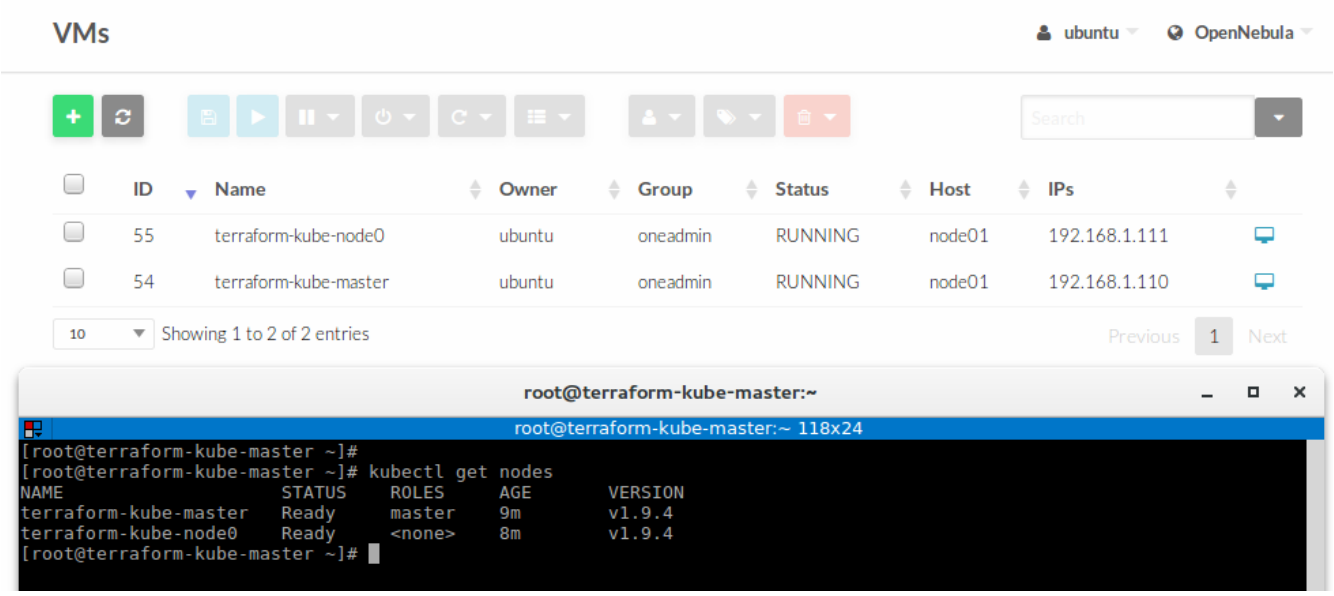
To actually create the instance, run the `terraform apply` command:

```
null_resource.kubernetes: Creation complete after 8m41s (ID: 4839363584769424432)

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

kube-master-vm_id = [
    54
]
kube-master-vm_ip = [
    192.168.1.110
]
```




You can access Dashboard using the kubectl command-line tool by running the following command:

```
$ kubectl proxy --address $MASTER_IP --accept-hosts='^*$'
```

```
The last step is to complete the cluster life cycle by removing your resources, do: `terraform destroy`
```

```
ubuntu@ubuntu-vm  ~/git/terransible-kubernetes-cluster  master ●
ubuntu@ubuntu-vm  ~/git/terransible-kubernetes-cluster  master ●  terraform destroy
data.template_file.tf-kube-template: Refreshing state...
opennebula_template.tf-kube-template: Refreshing state... (ID: 42)
opennebula_vm.kube-node-vm: Refreshing state... (ID: 55)
opennebula_vm.kube-master-vm: Refreshing state... (ID: 54)
null_resource.kubernetes: Refreshing state... (ID: 4839363584769424432)

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  - null_resource.kubernetes

  - opennebula_template.tf-kube-template

  - opennebula_vm.kube-master-vm

  - opennebula_vm.kube-node-vm


Plan: 0 to add, 0 to change, 4 to destroy.

Do you really want to destroy?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

null_resource.kubernetes: Destroying... (ID: 4839363584769424432)
null_resource.kubernetes: Destruction complete after 0s
opennebula_vm.kube-node-vm: Destroying... (ID: 55)
opennebula_vm.kube-master-vm: Destroying... (ID: 54)
opennebula_vm.kube-node-vm: Still destroying... (ID: 55, 10s elapsed)
opennebula_vm.kube-master-vm: Still destroying... (ID: 54, 10s elapsed)
opennebula_vm.kube-node-vm: Destruction complete after 10s
opennebula_vm.kube-master-vm: Destruction complete after 10s
opennebula_template.tf-kube-template: Destroying... (ID: 42)
opennebula_template.tf-kube-template: Destruction complete after 0s

Destroy complete! Resources: 4 destroyed.
ubuntu@ubuntu-vm  ~/git/terransible-kubernetes-cluster  master ●
```