**Install terraform on ubuntu**

sudo apt-get update && sudo apt-get install -y gnupg software-properties-common

wget -O- https://apt.releases.hashicorp.com/gpg | \

gpg --dearmor | \

sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] \

https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | \

sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update

sudo apt-get install terraform -y

**Terraform file for launching argocd**

* **vim unati.tf**
* Save and exit

terraform {

required\_providers {

helm = {

source = "hashicorp/helm"

version = ">= 2.0.0, <= 3.0.0" # Update this to the appropriate version range

}

kubernetes = {

source = "hashicorp/kubernetes"

version = ">= 1.0.0, <= 1.17.3" # Update this to the appropriate version range

}

}

}

provider "helm" {

kubernetes {

host = "https://kubernetes.default.svc"

token = var.token

cluster\_ca\_certificate = file(var.cluster\_ca\_certificate)

}

}

variable "token" {

type = string

}

variable "cluster\_ca\_certificate" {

type = string

}

resource "helm\_release" "argocd" {

name = "argocd"

repository = "https://argoproj.github.io/argo-helm"

chart = "argo-cd"

namespace = "argocd"

version = "4.9.7"

create\_namespace = true

values = [

<<-EOT

data:

secretToken: ${var.token}

users:

- name: admin

password: password

EOT

]

}

In above file look for below section code:

provider "helm" {

kubernetes {

host = "https://kubernetes.default.svc"

token = var.token

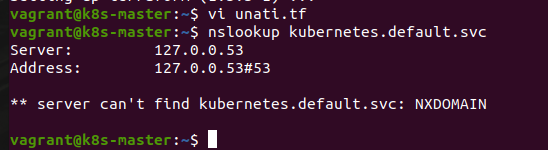
cluster\_ca\_certificate = file(var.cluster\_ca\_certificate)

}

}

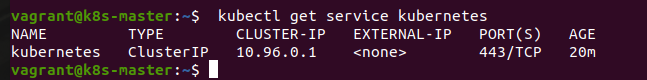
It is going to connect to the host <https://kubernetes.default.svc>

* Check whether that host reachable or not
* nslookup [kubernetes.default.svc](https://kubernetes.default.svc)



No domain existed >? Lets find out cluster ip

* kubectl get service kubernetes

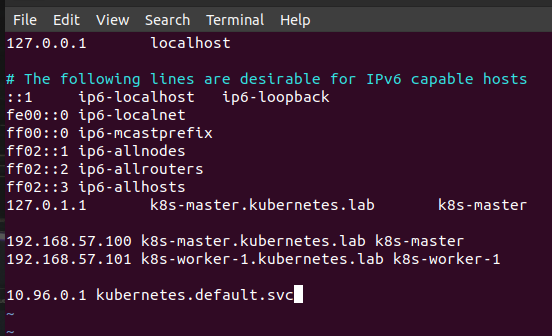


– ok now we have two option either we change host in the code with this ip or add this domain with ip in /etc/hosts file

Lets go with 2nd method:

* Cluster ip is : 10.96.0.1
* Add below line in /etc/hosts
* vi /etc/hosts

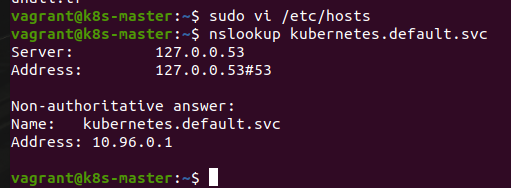
10.96.0.1 [kubernetes.default.svc](https://kubernetes.default.svc)



* Save , exit

Now check with nslookup :

* nslookup kubernetes.default.svc



Looks good 🙂

Next , we have fresh installation of k8s cluster, by default we dont have service acc.. So lets create for us

* vi service-unati.yml

apiVersion: v1

kind: ServiceAccount

metadata:

name: unati

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: unati-cluster-role-binding

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: unati

namespace: default

---

apiVersion: v1

kind: Secret

metadata:

name: unati-token

annotations:

kubernetes.io/service-account.name: unati

type: kubernetes.io/service-account-token

* Save ,exit

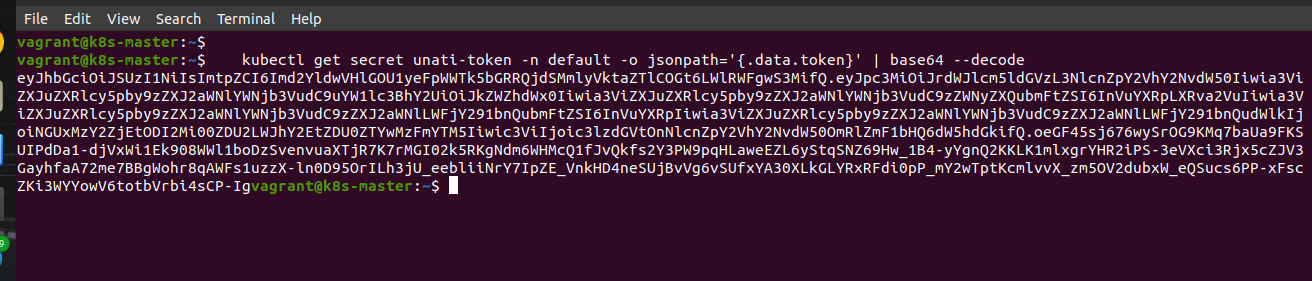
Apply this file to create service account

* kubectl apply -f service-unati.yml

**Retrieve secret token and cluster certificate :**

For token type below command :

* kubectl get secret unati-token -n default -o jsonpath='{.data.token}' | base64 --decode



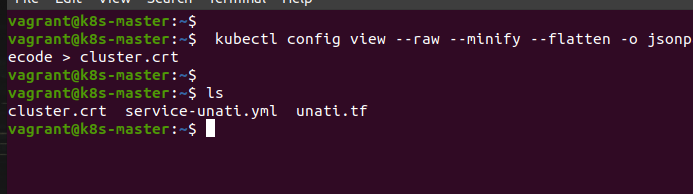
Copy token obtained and save it somewhere , we need it later

My token is:

eyJhbGciOiJSUzI1NiIsImtpZCI6Imd2YldwVHlGOU1yeFpWWTk5bGRRQjdSMmlyVktaZTlCOGt6LWlRWFgwS3MifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6InVuYXRpLXRva2VuIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6InVuYXRpIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiNGUxMzY2ZjEtODI2Mi00ZDU2LWJhY2EtZDU0ZTYwMzFmYTM5Iiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmRlZmF1bHQ6dW5hdGkifQ.oeGF45sj676wySrOG9KMq7baUa9FKSUIPdDa1-djVxWi1Ek908WWl1boDzSvenvuaXTjR7K7rMGI02k5RKgNdm6WHMcQ1fJvQkfs2Y3PW9pqHLaweEZL6yStqSNZ69Hw\_1B4-yYgnQ2KKLK1mlxgrYHR2iPS-3eVXci3Rjx5cZJV3GayhfaA72me7BBgWohr8qAWFs1uzzX-ln0D95OrILh3jU\_eebliiNrY7IpZE\_VnkHD4neSUjBvVg6vSUfxYA30XLkGLYRxRFdi0pP\_mY2wTptKcmlvvX\_zm5OV2dubxW\_eQSucs6PP-xFscZKi3WYYowV6totbVrbi4sCP-Ig

For cert type below command :

* kubectl config view --raw --minify --flatten -o jsonpath='{.clusters[].cluster.certificate-authority-data}' | base64 --decode > cluster.crt



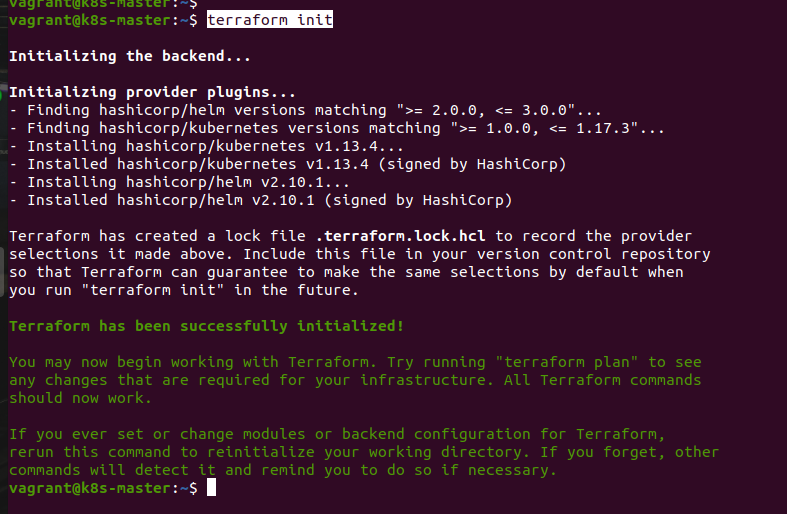
* It will create cert file name : cluster.crt

**We have enough things to deploy argocd**

* Cluster certificate
* Token

Lets go further,

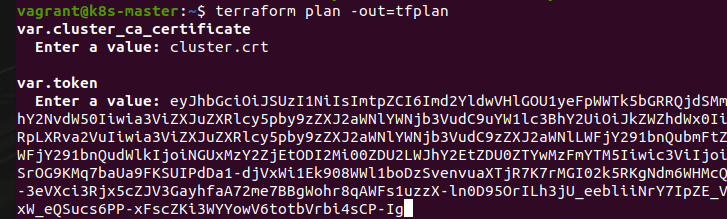
* terraform init

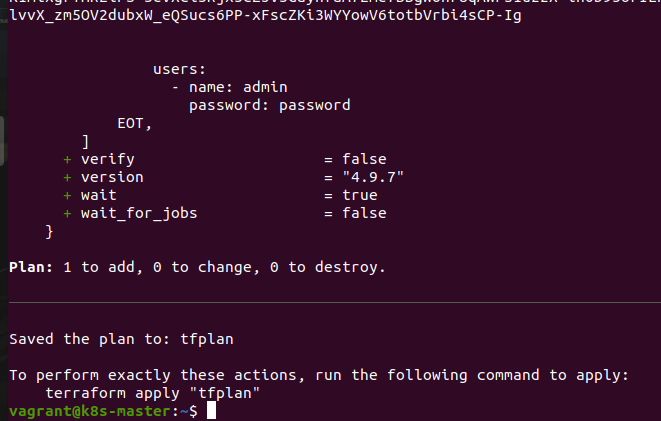


* terraform plan -out=tfplan

The terraform plan -out=tfplan command generates a plan of the changes that Terraform will make to your infrastructure. The plan is saved in a file named tfplan.

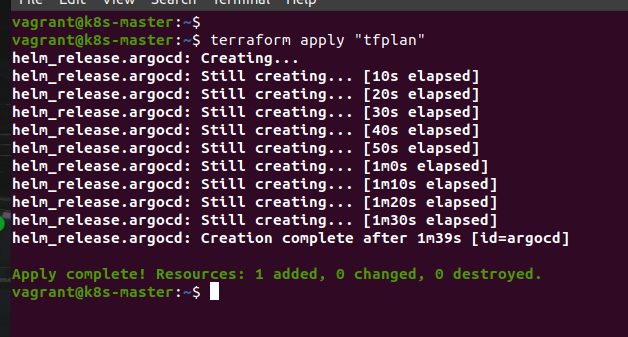
* It will ask you for cluster cert , provide path of cert, in my case it is in same dir
  + Previously Cert name is : cluster.crt
  + Provide token : copy the contents and paste there





* terraform apply "tfplan"

The terraform apply "tfplan" command applies the plan that was generated by the terraform plan -out=tfplan command. The plan is read from the file named tfplan.



* Check arogocd pods
* kubectl get pods --all-namespaces

