**AKS Cluster Creation**

1. Create a Resource Group

Note: Select region Southeast Asia or North Europe.

1. Create a Virtual Network and Subnet inside the RG

Note: Enable Public IP bastion under security tab.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Create a Virtual machine with below mentioned configurations

Note: Authentication -> SSH, Public inbound ports: None

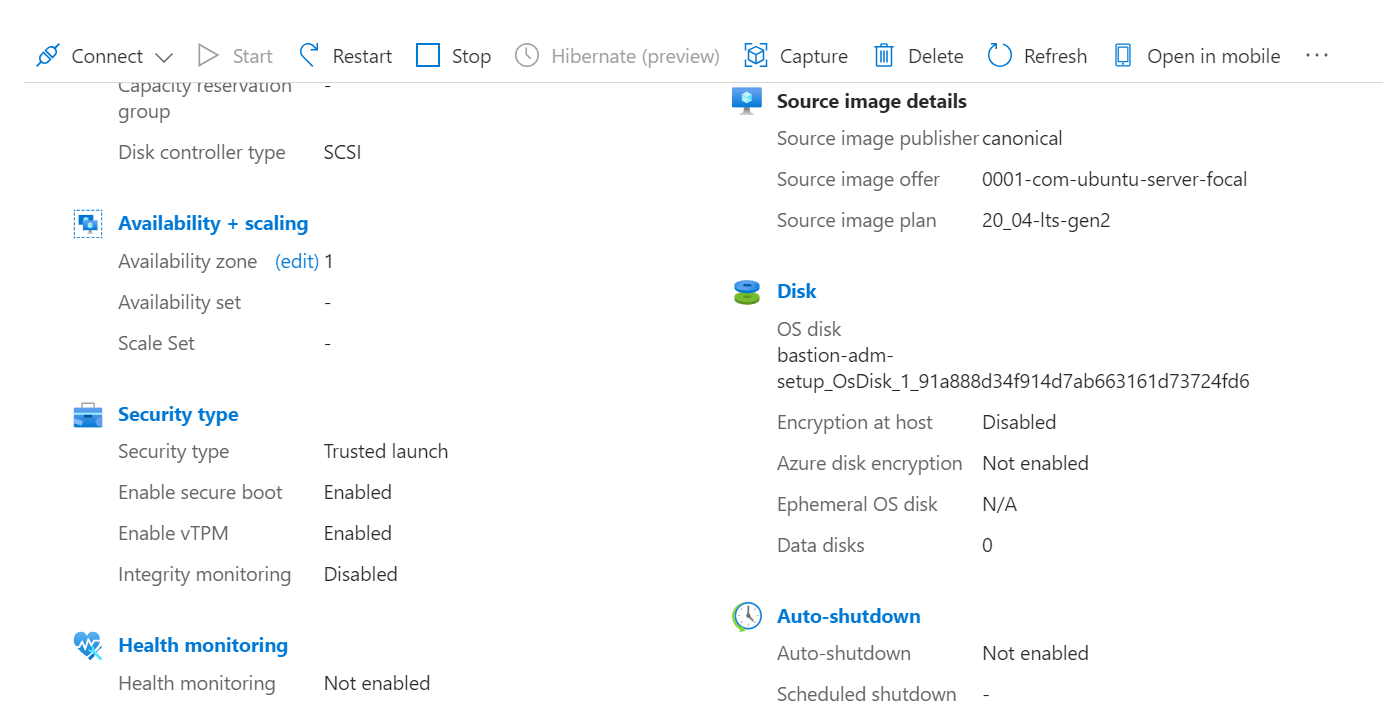
Once VM created got to NSG and add new rule in inbound for SSH.

A screenshot of a computer

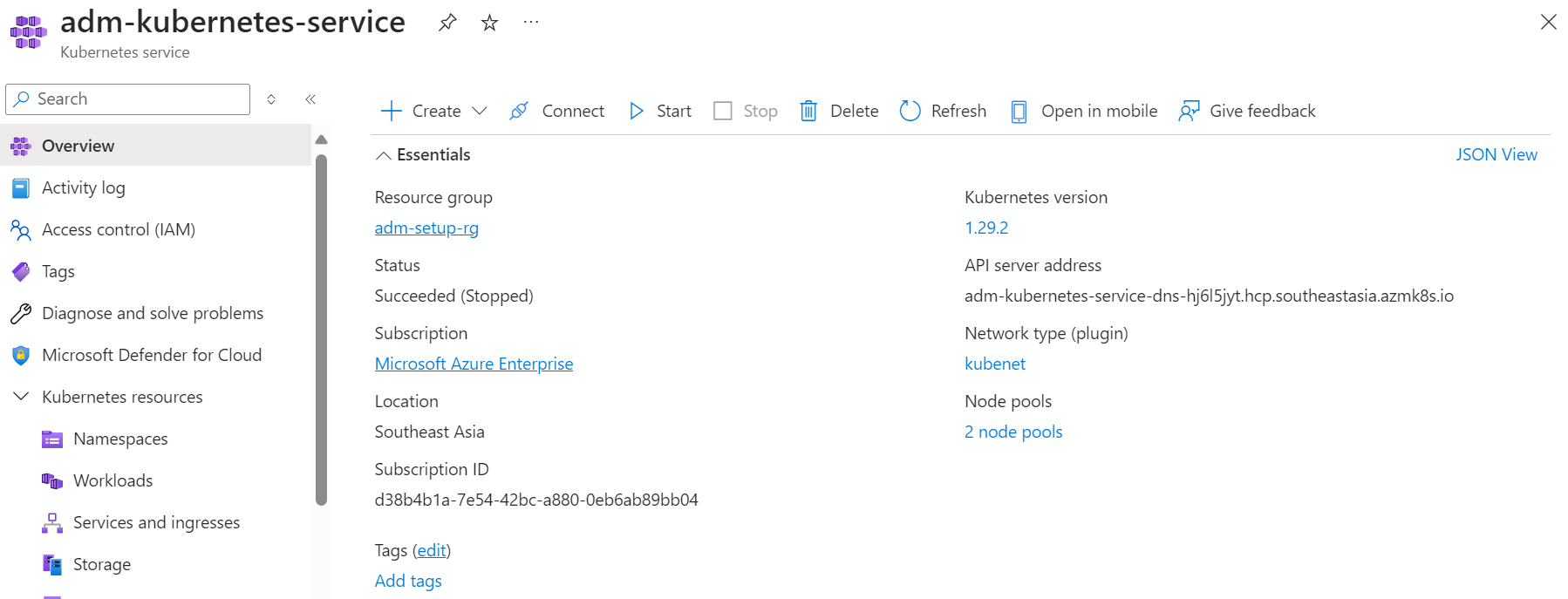
Description automatically generated

A screenshot of a computer

Description automatically generated



1. Create Azure Kubernetes service with below mentioned configurations.



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Connect to the bastion VM with Public IP and SSH key
2. Install azure-cli and login with azure creds

**sudo apt-get update**

**sudo apt-get install azure-cli**

**sudo az login --service-principal --username <client\_ID> --password <client\_secret> --tenant <tenant\_id>**

**Create empty kubconfig file**

**vi <configfile>**

**export KUBECONFIG="/home/ubuntu/<configfile>"**

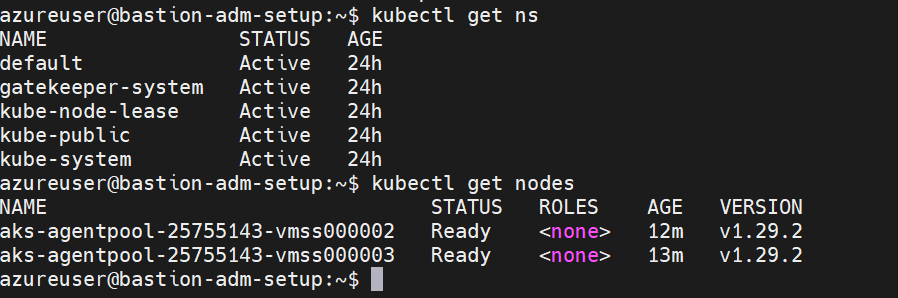
**sudo az aks get-credentials --resource-group <provision id> --name <provision id> --file <configfile>**

1. Install kubectl

**sudo snap install kubectl --classic**

1. Execute below command check whether nodes are ready

**kubectl get nodes**



1. Deploy Sample application

A computer screen with white text

Description automatically generated

Note: If application is deployed as loadbalancer and need to access open the particular port to vpn ip in aks created infra nsg.

Example:

A screenshot of a computer

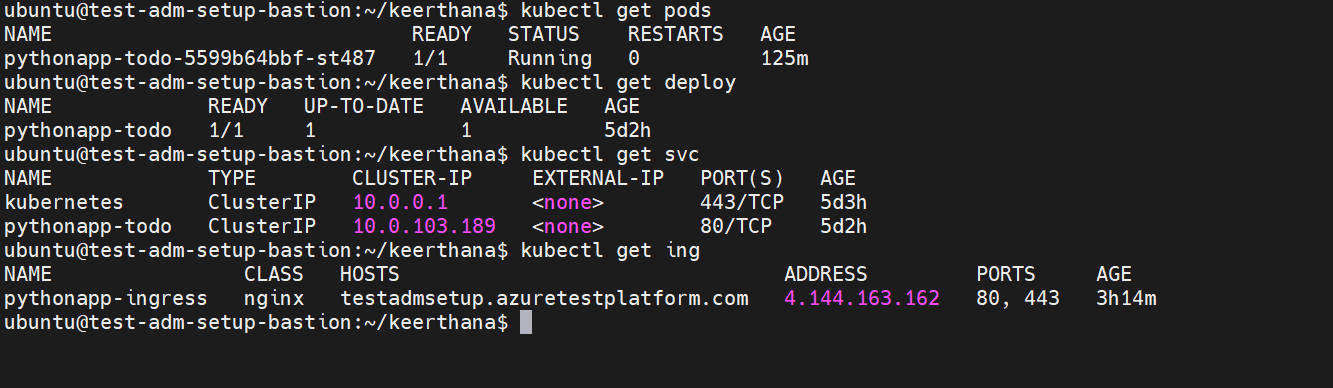
Description automatically generated

**How to Install Ingress Niginx Controller**

* kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.3.0/deploy/static/provider/cloud/deploy.yaml
* kubectl get pods --namespace ingress-nginx
* kubectl get service ingress-nginx-controller --namespace=ingress-nginx

Create a DNS record set

Create ingress.yaml and deploy



Try to access the application with DNS with http.

**How to make application secure by applying SSL (https)**

**helm repo add jetstack https://charts.jetstack.io**

**helm repo update**

**helm upgrade cert-manager jetstack/cert-manager \**

**--install \**

**--create-namespace \**

**--wait \**

**--namespace cert-manager**

This will create three Deployments and some Services and Pods in a new namespace called cert-manager. It also installs various cluster scoped supporting resources such as RBAC roles and Custom Resource Definitions.

You can view some of the resources that have been installed as follows:

**kubectl -n cert-manager get all**

**kubectl explain Certificate**

**kubectl explain CertificateRequest**

**kubectl explain Issuer**

**Create a ClusterIssuer and a Certificate**

# clusterissuer-selfsigned.yaml

apiVersion: cert-manager.io/v1

kind: ClusterIssuer

metadata:

name: letsencrypt-production

spec:

acme:

# The ACME production api URL

server: https://acme-v02.api.letsencrypt.org/directory

# Email address used for ACME registration

email: abc@gmail.com

# Name of a secret used to store the ACME account private key

privateKeySecretRef:

name: letsencrypt-production

# Enable the HTTP-01 challenge provider

solvers:

- http01:

ingress:

class: nginx

# certificate.yaml

apiVersion: cert-manager.io/v1

kind: Certificate

metadata:

name: tls

spec:

secretName: secret-tls

privateKey:

rotationPolicy: Always

commonName: testadmsetup.azuretestplatform.com

dnsNames:

- testadmsetup.azuretestplatform.com

usages:

- digital signature

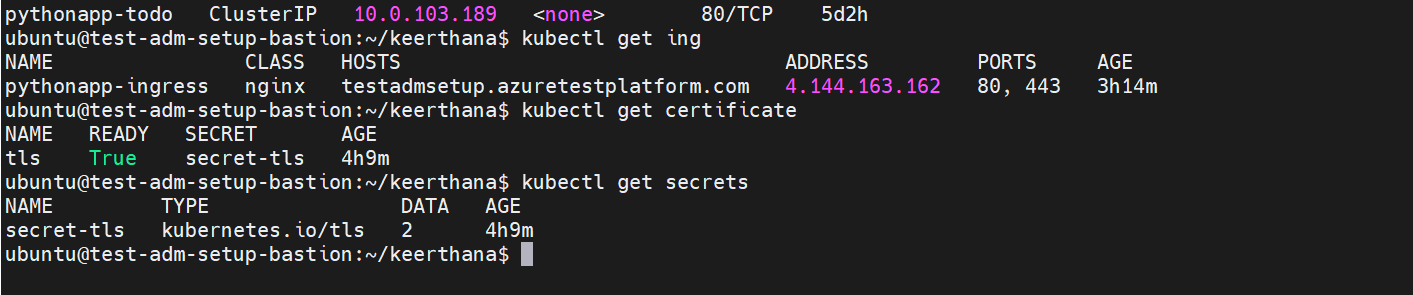
- key encipherment

- server auth

issuerRef:

name: letsencrypt-production

kind: ClusterIssuer



A screenshot of a computer

Description automatically generated