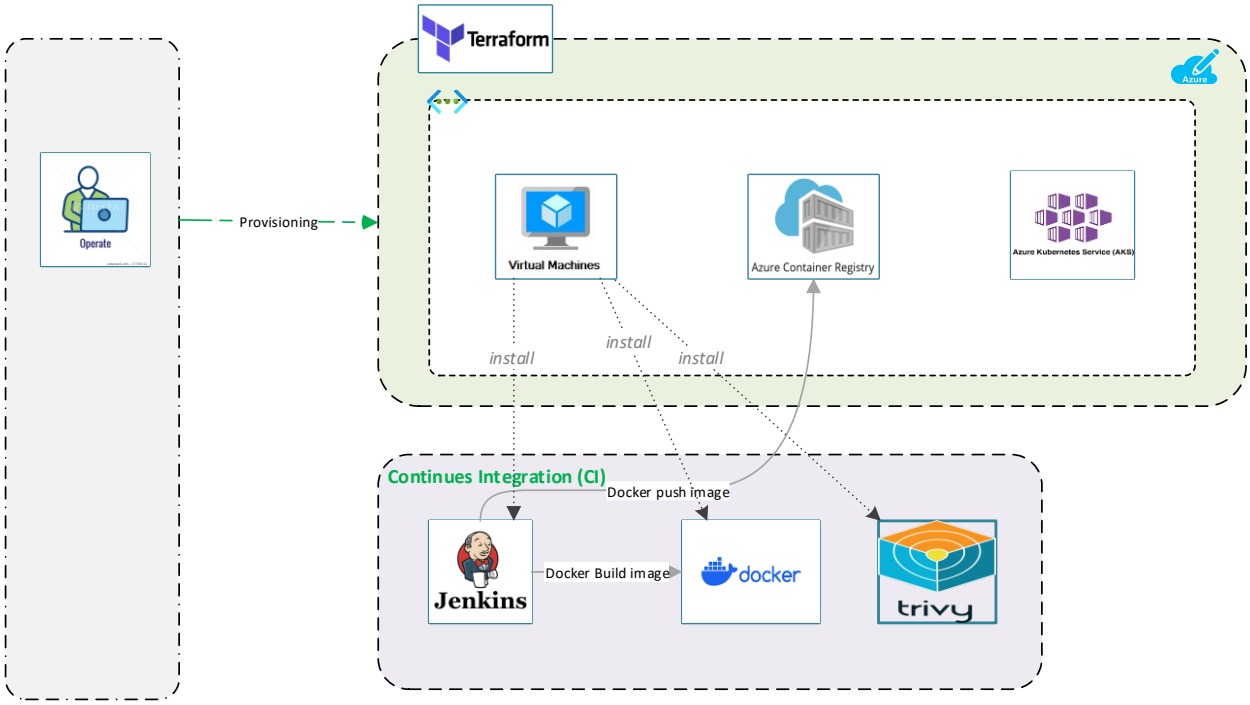
Overview

This document will help you to provision Azure resources (AKS, ACR, VM) using the terraform.

This also includes the installation of Jenkins, Docker, Kubectl CLI, Trivy into the Virtual Machine via a custom data script included when provisioning the Virtual Machine.



Prerequisites

- You must have an Aure account with an Active subscription.
- Azure Command-line Interface (CLI) tool is installed on your machine.
- Kubectl CLI tool is installed on your machine.
- Terraform CLI tool is installed on your machine.
- You must clone this source code from GitHub https://github.com/sieunhantanbao/sd2411_azure_infrastructure
    - git clone https://github.com/sieunhantanbao/sd2411_azure_infrastructure.git
    - cd sd2411_azure_infrastructure/

1. Provision Azure Kubernetes Service (AKS) in Multi-AZs

- Change directory to */iac/terraform/aks/ha*
- Update the values from the ***variables.tf*** if required
- Run the below commands
    - `terraform init`

```
[redacted]@k8s-node-0:~/projects/sd2411_azure_infrastructure/iac/terraform/aks/ha$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of azure/azapi from the dependency lock file
- Reusing previous version of hashicorp/azurerm from the dependency lock file
- Using previously-installed azure/azapi v1.9.0
- Using previously-installed hashicorp/azurerm v2.36.0

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.
```

- terraform plan --out tfplan.out

```
[redacted]@k8s-node-0:~/projects/sd2411_azure_infrastructure/iac/terraform/aks/ha$ terraform plan -out tfplan.out

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_kubernetes_cluster.k8s will be created
  + resource "azurerm_kubernetes_cluster" "k8s" {
      + dns_prefix              = "sd2411-my-todo-dns-ha"
      + fqdn                    = (known after apply)
      + id                      = (known after apply)
      + kube_admin_config       = (known after apply)
      + kube_admin_config_raw   = (sensitive value)
      + kube_config             = (known after apply)
      + kube_config_raw         = (sensitive value)
      + kubelet_identity        = (known after apply)
      + kubernetes_version      = (known after apply)
      + location                = "japaneast"
      + name                    = "sd2411_k8s_cluster_ha"
      + node_resource_group     = (known after apply)
      + private_cluster_enabled = (known after apply)
      + private_fqdn            = (known after apply)
      + private_link_enabled    = (known after apply)
      + resource_group_name     = "rg_sd2411_aks_ha"
      + sku_tier                = "Free"
      + tags                    = {
          + "Environment" = "Development"
        }

      + default_node_pool {
          + availability_zones   = [
              + "1",
              + "2",
            ]
          + max_pods             = (known after apply)
          + name                 = "agentpool"
          + node_count           = 2
          + orchestrator_version = (known after apply)
          + os_disk_size_gb      = (known after apply)
          + type                 = "VirtualMachineScaleSets"
          + vm_size              = "Standard_B2s_v2"
          + vnet_subnet_id       = (known after apply)
        }

      + identity {
          + principal_id = (known after apply)
```

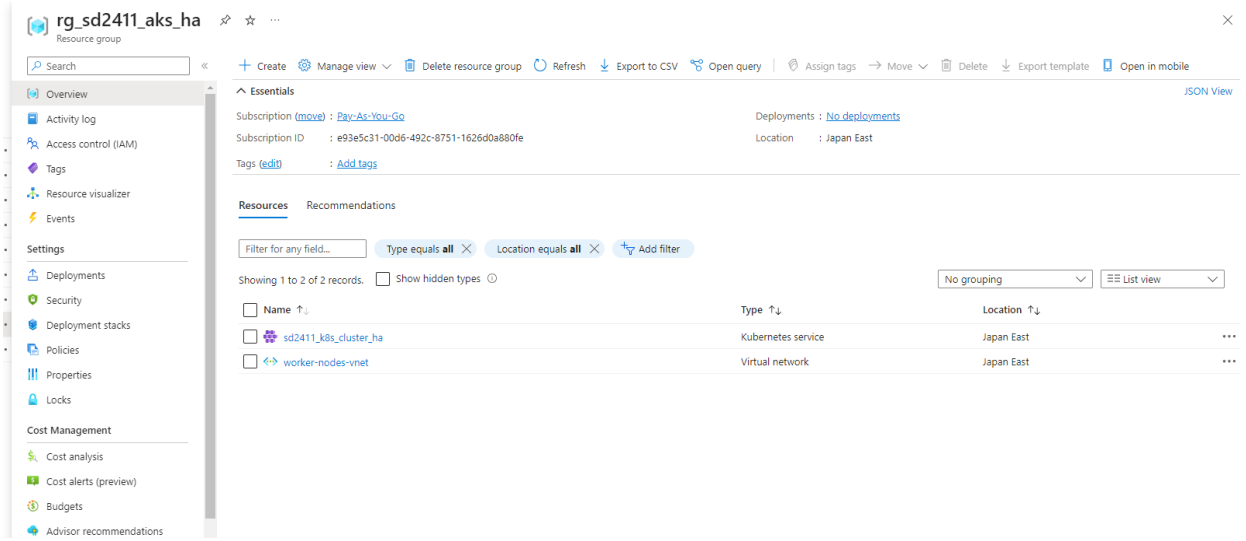
```
  # azurerm_subnet.worker_nodes_subnet will be created
  + resource "azurerm_subnet" "worker_nodes_subnet" {
      + address_prefix                                 = (known after apply)
      + address_prefixes                               = [
          + "10.1.1.0/24",
        ]
      + enforce_private_link_endpoint_network_policies = false
      + enforce_private_link_service_network_policies  = false
      + id                                             = (known after apply)
      + name                                           = "worker-nodes-subnet"
      + resource_group_name                            = "rg_sd2411_aks_ha"
      + virtual_network_name                           = "worker-nodes-vnet"
    }

  # azurerm_virtual_network.worker_nodes will be created
  + resource "azurerm_virtual_network" "worker_nodes" {
      + address_space         = [
          + "10.1.0.0/16",
        ]
      + guid                  = (known after apply)
      + id                    = (known after apply)
      + location              = "japaneast"
      + name                  = "worker-nodes-vnet"
      + resource_group_name   = "rg_sd2411_aks_ha"
      + subnet                = (known after apply)
      + vm_protection_enabled = false
    }

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

Changes to Outputs:
  + client_certificate     = (known after apply)
  + client_key             = (known after apply)
  + cluster_ca_certificate = (known after apply)
  + cluster_password       = (known after apply)
  + cluster_username       = (known after apply)
  + host                   = (known after apply)
  + kube_config            = (sensitive value)

─────────────────────────────────────────────────────────────────────────────

Saved the plan to: tfplan.out

To perform exactly these actions, run the following command to apply:
    terraform apply "tfplan.out"
```

- `terraform apply tfplan.out`

```
ntgadmin@k8s-node-0:~/projects/sd2411_azure_infrastructure/iac/terraform/aks/ha$ terraform apply tfplan.out
azurerm_resource_group.rg: Creating...
azurerm_resource_group.rg: Creation complete after 2s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha]
azurerm_virtual_network.control_plane: Creating...
azurerm_virtual_network.worker_nodes: Creating...
azurerm_route_table.control_plane_route_table: Creating...
azurerm_route_table.control_plane_route_table: Creation complete after 5s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/providers/Microsoft.Networ
k/routeTables/control-plane-route-table]
azurerm_virtual_network.control_plane: Creation complete after 7s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/providers/Microsoft.Network/virtua
lNetworks/control-plane-vnet]
azurerm_subnet.control_plane_subnet: Creating...
azurerm_virtual_network.worker_nodes: Creation complete after 7s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/providers/Microsoft.Network/virtual
Networks/worker-nodes-vnet]
azurerm_virtual_network_peering.control_plane_to_worker: Creating...
azurerm_virtual_network_peering.worker_to_control_plane: Creating...
azurerm_subnet.worker_nodes_subnet: Creating...
azurerm_subnet.control_plane_subnet: Creation complete after 6s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/providers/Microsoft.Network/virtualN
etworks/control-plane-vnet/subnets/control-plane-subnet]
azurerm_subnet_route_table_association.control_plane_association: Creating...
azurerm_subnet.worker_nodes_subnet: Creation complete after 6s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/providers/Microsoft.Network/virtualNe
tworks/worker-nodes-vnet/subnets/worker-nodes-subnet]
azurerm_kubernetes_cluster.k8s: Creating...
azurerm_virtual_network_peering.worker_to_control_plane: Still creating... [10s elapsed]
azurerm_virtual_network_peering.control_plane_to_worker: Still creating... [10s elapsed]
azurerm_subnet_route_table_association.control_plane_association: Creation complete after 5s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/provide
rs/Microsoft.Network/virtualNetworks/control-plane-vnet/subnets/control-plane-subnet]
azurerm_virtual_network_peering.control_plane_to_worker: Creation complete after 12s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/providers/Micro
soft.Network/virtualNetworks/control-plane-vnet/virtualNetworkPeerings/control-plane-to-worker]
azurerm_kubernetes_cluster.k8s: Still creating... [10s elapsed]
azurerm_virtual_network_peering.worker_to_control_plane: Still creating... [20s elapsed]
azurerm_virtual_network_peering.worker_to_control_plane: Creation complete after 24s [id=/subscriptions/e93e5c31-00d6-492c-8751-1626d0a880fe/resourceGroups/rg_sd2411_aks_ha/providers/Micro
soft.Network/virtualNetworks/worker-nodes-vnet/virtualNetworkPeerings/worker-to-control-plane]
azurerm_kubernetes_cluster.k8s: Still creating... [20s elapsed]
azurerm_kubernetes_cluster.k8s: Still creating... [30s elapsed]
azurerm_kubernetes_cluster.k8s: Still creating... [40s elapsed]
```

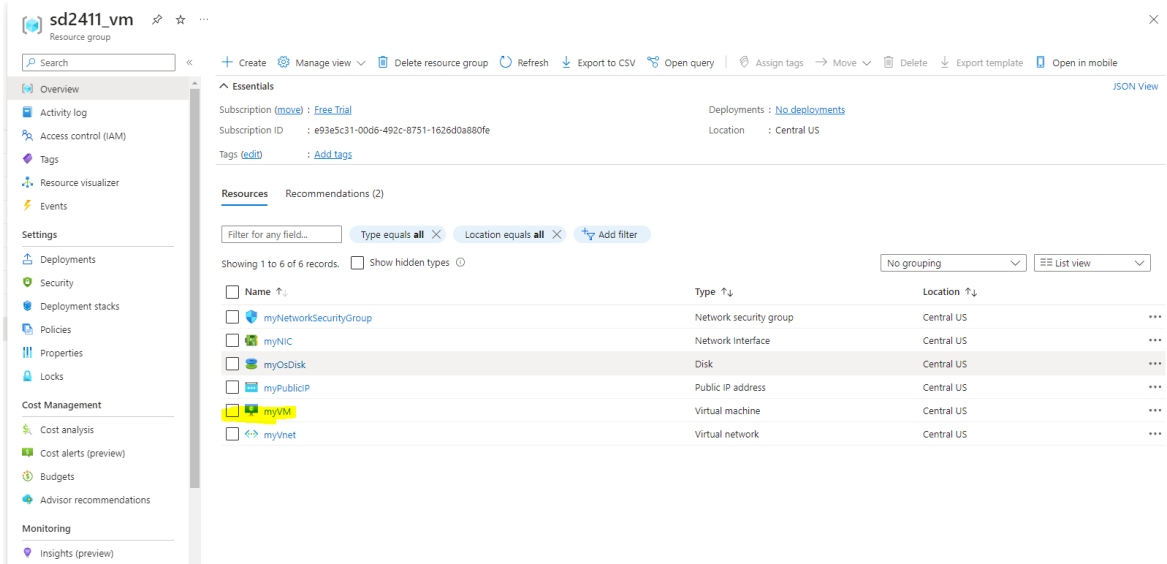
```
Apply complete! Resources: 10 added, 0 changed, 0 destroyed.

Outputs:

client_certificate = "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZIakNDQXdhZ0F3SUJBZ0lSQUtaRUwvQ0xvTG83QVBaa2tuU2I5bkV3RFFZSktvWklodmNOQVFFTEJRQXcK
VNRFkwTlRFNFdoY05NalV4TURFNU1EWTFOVEU0V2pBdwpNUmN3RlFZRFZRUUtFdzV6ZVhOMFpXMDZiV0Z6ZEdWeWN6RVZNQk1HQTFVRUF4TU1iV0Z6ZEdWeVkyeHBaVzUwCk1JSUNJakFOQmdr
VBNmN4UmdSSUtlYlRldTlqaUl5RzQKSGppMjI4ODI1bEVyNDFnQ1J6R2VSbzhCK2d0RjRWVndRaWZGaWhaQ1pmM1pzdEgxZG9LczU0eGFoUUo1Wi8zcgpUODNUUFhFYnQrUU0zWjdjV2NleVU2
dKd2pzcU45MGxqClpNK1lidGVQcHZkT29oUnNzWWZDT3owQWNTZEZ1OC9SWUN3VXlRcStQUy9DVFNzanliVmdqMGtMR3Y3akpQb3YKMko2Nk1DVFZNcE53K2RyODllc2ZrOW5Bb0xXU3VQWlNp
pOVFg2TGdPdnBSZCtQYlZvMjVQWUs0SmZ6dm5uaUwwcEFrNW13S0d4QlJ5L1NpRnY5UllNZ2E2eUw0b2EwWTFWCmROObWk2RjE5WUNUUzdUak9DMmpGdzZIYTV6V1R6SWVQeGdHQVdyb2VzakdM
NHdlJ4NFdkTUtKRUdrQ1ZXQ1kvSlk5enI1SHZUTEQ4SjYrVElaOXFEeldmVk9RRTQydkh5agpmNk1kejFleFV1Zk5HZ252aFhUUVZRTENSNlRQZUdwRkwvVHFMYWxjZUxXUk0zRWxncStFa1N3
BHQ2VRc2JoRDVQd2dTRHdiTVRpNWVDbENPcU5ucXdaYVNLV2pUdHZFZksKZWlXUjFZbEJqbWV2ZjQ2cXNZWGNnODRpR2ZXU2E2Z1J0T2JSWVBhSEt5VDZDTkxlN0pwQ0dzei92eS9uSEhQUQpu
1GUXdEZ1lEVlIwUEFRSC9CQVFEQWdXZ01CTUdBMVVkCkpRUU1NQW9HQ0NzR0FRVUZCd01DTUF3R0ExVWRFd0VCL3dRQ01BQXdId1lEVlIwakJCZ3dGb0FVVFlrSWx0RTUKTDNiNmRmSmZjZjBF
JBQVNBM1V5RU5lZ21vVDdmeXVNaAozZEtlelFIWGgrK2pGbWJZQ2FXLy9TYlZxelA4dXQrSi90YTB0eVl2VHVJYmdKc2Y5bG94QStmM0VyMVJkRFpKCmtXZTluSWZ4YjM0SDIrKzhFbTd1TDhL
xxNm02WjlwL2wkL25tQkVGSVZjQjdVV1k4WW1kUWVIM0JGbWlmR1pRZFpCMTdtWkdlb0tld2lLeUxkWTd0a0lGL3JwTzU2eVJidQpsWDRzMUxiaThRTEYyQnVjbXcyTXhUNDFmR25Zc2pjWm5S
RMdUtvYWRMRmxoRS92WldVd3BUZ1k0UHRPZk8zOFVhZ3hxdWtrb1VCN25ZQU9WcVpTcGpDYnhhRkZkQjdrd3QKRGJ1b1Q2d0FPRDN4eGlFMHlHk1gwQlV2QWtrQkV5VWs1RW1TMHNGRHhQNktH
E1bVZ0clFha3FpVjZBQTg0NE5jbnF4YzFjUkNzNHJIc1BnWGdiOFY3YjJXazhRUmdLc2I3CkdyWlVpci8vaVc2Q0lYa21vc3pjRTcvSk45UnowcXVlOFdoMUVpeFR5a00zNG5obLFxWTdaM1E0
```

- Check the result on the Azure Portal



2. Provision Virtual Machine (VMs) and install necessary tools

- Change directory to */iac/terraform/vm*
- Update the values from the ***variables.tf*** if required
- Run the below commands
    a. `terraform init`
    b. `terraform plan -out tfplan.out`
    c. `terraform apply tfplan.out`
- Check the result on the Azure Portal



- Please be noted that, we have the **azure-user-data.sh** file (that run when the Virtual Machine is provisioned) to install the Docker, Jenkins, Kubectl CLI, and Trivy

iac > terraform > vm > $ azure-user-data.sh

```
#! /bin/bash
##########################INSTALL DOCKER#########################################################3
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common -y
# Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# The following command is to set up the stable repository
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu `lsb_release -cs` stable" -y

# Update the apt package index, and install the latest version of Docker Engine and containerd, or go to the next step to install a specific version
sudo apt update
sudo apt install -y docker-ce

# To run docker without sudo
sudo groupadd docker
sudo usermod -aG docker ubuntu
sudo usermod -a -G docker jenkins
##############################INSTALL JENKINS########################################

sudo apt-get update
sudo apt install openjdk-11-jdk -y

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]  https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt update
sudo apt install jenkins -y
sudo systemctl start jenkins

# Modify Firewall to Allow Jenkins
sudo ufw allow 8080
sudo ufw allow ssh
sudo ufw --force enable
sudo ufw status

##############################INSTALL KUBECTL CLI#############################
sudo apt update
sudo snap install kubectl --classic

##############################INSTALL TRIVY#############################
sudo apt install wget apt-transport-https gnupg lsb-release -y
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt install trivy -y

```

- SSH to the Virtual Machine (the username and password can be found in the **variables.tf** file) and check if the **Docker**, **Jenkins**, **kubectl**, and **Trivy** are installed

```
anhnguyens@VNNOT01050 ~ ssh ubuntu@172.173.112.179
The authenticity of host '172.173.112.179 (172.173.112.179)' can't be established.
ECDSA key fingerprint is SHA256:aJA8Kn5efoaYdOk9ZHBzETMPKP59107yXYi3q81i57Q.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.173.112.179' (ECDSA) to the list of known hosts.
ubuntu@172.173.112.179's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1014-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Oct 19 08:54:43 UTC 2023

  System load:  0.08837890625      Processes:                111
  Usage of /:   54.4% of 28.89GB   Users logged in:          0
  Memory usage: 37%                IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                 IPv4 address for eth0:    10.0.1.4

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

7 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Oct 11 22:37:14 2023 from 123.20.210.63
ubuntu@hostname:~$
```

```
ubuntu@hostname:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-10-15 22:40:56 UTC; 3 days ago
   Main PID: 647 (java)
      Tasks: 54 (limit: 9509)
     Memory: 2.6G
        CPU: 16min 22.914s
     CGroup: /system.slice/jenkins.service
             └─647 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

```
ubuntu@hostname:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-10-15 22:40:39 UTC; 3 days ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 832 (dockerd)
      Tasks: 12
     Memory: 242.5M
        CPU: 56.560s
     CGroup: /system.slice/docker.service
             └─832 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

```
ubuntu@hostname:~$ kubectl --help
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/

Basic Commands (Beginner):
  create          Create a resource from a file or from stdin
  expose          Take a replication controller, service, deployment or pod and expose it as a new Kubernetes service
  run             Run a particular image on the cluster
  set             Set specific features on objects

Basic Commands (Intermediate):
  explain         Get documentation for a resource
  get             Display one or many resources
  edit            Edit a resource on the server
  delete          Delete resources by file names, stdin, resources and names, or by resources and label selector

Deploy Commands:
  rollout         Manage the rollout of a resource
```

```
ubuntu@hostname:~$ trivy --help
Scanner for vulnerabilities in container images, file systems, and Git repositories, as well as for configuration issues and hard-coded secrets

Usage:
  trivy [global flags] command [flags] target
  trivy [command]

Examples:
  # Scan a container image
  $ trivy image python:3.4-alpine

  # Scan a container image from a tar archive
  $ trivy image --input ruby-3.1.tar

  # Scan local filesystem
  $ trivy fs .

  # Run in server mode
  $ trivy server
```

3. Provision Azure Container Registry (ACR)

- Change directory to */iac/terraform/acr*
- Update the values from the ***variables.tf*** if required
- Run the below commands
    a. `terraform init`
    b. `terraform plan -out tfplan.out`
    c. `terraform apply tfplan.out`
- Check the result on the Azure Portal