Azure devops end to end project

Azure portal login

Creating azure virtual machine through terraform or ARM template

Creating Azure Subscription, Azure network, azure subnet

Creating Azure ACR

Creating Azure SQL(Data base and Sql Server)

Azure Devops login –and setting some organsation level

Added some tasks

Azure token

Azure docker

Azure ACR

Azure SQL

1.What is private cluster?

In a private cluster, the control plane or API server has internal IP addresses that are defined in the [RFC1918 - Address Allocation for Private Internet](https://tools.ietf.org/html/rfc1918) document. By using a private cluster, you can ensure network traffic between **your API server and your node pools remains on the private network only.**

2.What is service connection?

Service Connections are used in Azure DevOps Pipelines to connect to external services, like Azure, GitHub, Docker, Kubernetes, and many other service

What is Azure storage?

An Azure storage account contains all of your Azure Storage data objects, including blobs, file shares, queues, tables, and disks. The storage account provides a unique namespace for your Azure Storage data that's accessible from anywhere in the world over HTTP or HTTPS.

Data in your storage account is durable and highly available, secure, and massively scalable.

1. **What is Azure Kubernetes Service (AKS)?**
   * AKS is a managed container orchestration service provided by Microsoft Azure. It simplifies deploying, managing, and scaling containerized applications using Kubernetes.
2. **What are the key benefits of using AKS?**
   * Some benefits include automatic updates, automated scaling, simplified management, integrated monitoring, and strong security features.
3. **Explain the main components of AKS.**
   * Key components include Master Nodes (Control Plane), Node Pools (Virtual Machines), and etcd (distributed key-value store for storing cluster data).
4. **How does AKS handle node scaling?**
   * AKS allows you to scale node pools to handle varying workloads. You can manually scale node pools or set up an auto-scaling configuration based on CPU or memory usage.
5. **What is a Kubernetes Pod?**
   * A Pod is the smallest deployable unit in Kubernetes. It represents a single instance of a containerized application and can consist of one or more containers that share the same network namespace.
6. **What is a Kubernetes Deployment?**
   * A Deployment is a Kubernetes resource used to manage the deployment and scaling of a set of Pods. It ensures that a specified number of replica Pods are running at any given time.
7. **What is a Service in Kubernetes?**
   * A Service is an abstraction that defines a set of Pods and a policy by which to access them. It provides a stable endpoint for accessing your application.
8. **How does AKS manage container networking?**
   * AKS uses Azure CNI (Container Network Interface) to manage container networking. Each Pod gets its own IP address from the Azure Virtual Network.
9. **Explain the concept of Helm in Kubernetes.**
   * Helm is a package manager for Kubernetes that allows you to define, install, and upgrade even the most complex Kubernetes applications.
10. **What are Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) in Kubernetes?**
    * PVs are storage resources provisioned by an administrator, while PVCs are requests for storage by a user. PVCs consume PV resources.
11. **How do you handle secrets in AKS?**
    * Kubernetes Secrets provide a way to securely store sensitive information, such as passwords or API keys. They can be mounted as files or used as environment variables in Pods.
12. **What is Azure Container Registry (ACR) and how does it integrate with AKS?**
    * ACR is a private container registry in Azure. It allows you to store and manage Docker images. AKS can be configured to pull container images from ACR.
13. **Explain how you would perform a rolling update in AKS.**
    * A rolling update in AKS can be achieved by updating the deployment manifest with the new version, and then applying the update. Kubernetes will manage the rollout of the new version.
14. **How do you handle logging and monitoring in AKS?**
    * AKS integrates with Azure Monitor, which provides comprehensive monitoring and logging capabilities. It supports tools like Azure Log Analytics, Application Insights, and more.

Kubernetes commands:

<https://blog.devgenius.io/deployments-daemonsets-and-statefulsets-in-kubernetes-which-one-to-use-for-your-application-5d837f738ff9>

1. **Cluster Management**:
   * **kubectl cluster-info**: Displays cluster information.
   * **kubectl get nodes**: Lists all nodes in the cluster.
   * **kubectl get pods --all-namespaces**: Lists all pods in all namespaces.
2. **Namespace Management**:
   * **kubectl get namespaces**: Lists all namespaces.
   * **kubectl create namespace <namespace-name>**: Creates a new namespace.
   * **kubectl delete namespace <namespace-name>**: Deletes a namespace and all its resources.
3. **Pod Management**:
   * **kubectl get pods**: Lists all pods in the current namespace.
   * **kubectl describe pod <pod-name>**: Provides detailed information about a specific pod.
   * **kubectl delete pod <pod-name>**: Deletes a pod.
4. **Deployment Management**:
   * **kubectl get deployments**: Lists all deployments in the current namespace.
   * **kubectl describe deployment <deployment-name>**: Provides detailed information about a specific deployment.
   * **kubectl scale deployment <deployment-name> --replicas=<replica-count>**: Scales a deployment to the specified number of replicas.
5. **Service Management**:
   * **kubectl get services**: Lists all services in the current namespace.
   * **kubectl describe service <service-name>**: Provides detailed information about a specific service.
   * **kubectl expose deployment <deployment-name> --port=<port> --type=LoadBalancer**: Exposes a deployment as a service.
6. **ConfigMap and Secret Management**:
   * **kubectl get configmaps**: Lists all ConfigMaps in the current namespace.
   * **kubectl get secrets**: Lists all secrets in the current namespace.
7. **Persistent Volume (PV) and Persistent Volume Claim (PVC) Management**:
   * **kubectl get pv**: Lists all persistent volumes.
   * **kubectl get pvc**: Lists all persistent volume claims.
   * **kubectl describe pv <pv-name>**: Provides detailed information about a specific persistent volume.
   * **kubectl describe pvc <pvc-name>**: Provides detailed information about a specific persistent volume claim.
8. **Role-Based Access Control (RBAC)**:
   * **kubectl get roles**: Lists all roles in the current namespace.
   * **kubectl get rolebindings**: Lists all role bindings in the current namespace.
   * **kubectl describe role <role-name>**: Provides detailed information about a specific role.
9. **Namespace-Specific Commands**:
   * **kubectl apply -f <yaml-file>**: Creates or updates resources defined in a YAML file in the current namespace.
10. **Cleaning Up**:
    * **kubectl delete <resource-type> <resource-name>**: Deletes a specific resource.

Docker Commands:

1. **Building Images**:
   * **docker build**: Builds an image from a Dockerfile.
   * **docker image ls**: Lists all images.
   * **docker image rm <image-name>**: Removes one or more images.
2. **Managing Containers**:
   * **docker run**: Creates and starts a container from an image.
   * **docker ps**: Lists running containers.
   * **docker ps -a**: Lists all containers (including stopped ones).
   * **docker start <container-id>**: Starts a stopped container.
   * **docker stop <container-id>**: Stops a running container.
   * **docker rm <container-id>**: Removes a container.
   * **docker exec -it <container-id> <command>**: Runs a command inside a running container.
3. **Managing Volumes**:
   * **docker volume ls**: Lists all volumes.
   * **docker volume create <volume-name>**: Creates a volume.
   * **docker volume rm <volume-name>**: Removes a volume.
4. **Managing Networks**:
   * **docker network ls**: Lists all networks.
   * **docker network create <network-name>**: Creates a network.
   * **docker network rm <network-name>**: Removes a network.
5. **Managing Images**:
   * **docker pull <image-name>**: Downloads an image from a registry.
   * **docker push <image-name>**: Uploads an image to a registry.
6. **Inspecting Docker Objects**:
   * **docker inspect <object-id>**: Displays detailed information about an object (container, image, etc.).
   * **docker logs <container-id>**: Displays the logs of a container.
   * **docker diff <container-id>**: Shows changes made to a container's filesystem.
7. **Cleaning Up**:
   * **docker system prune**: Removes all unused data (containers, networks, volumes, images not attached to a container).
8. **Registry and Repository**:
   * **docker login**: Log in to a Docker registry.
   * **docker logout**: Log out from a Docker registry.
   * **docker tag <source-image> <target-image>**: Assigns a new name to an image.
   * **docker push <image-name>**: Uploads an image to a registry.
9. **Docker Compose**:
   * **docker-compose up**: Starts services defined in a Compose file.
   * **docker-compose down**: Stops and removes containers defined in a Compose file.

**Terraform commands:**

1. **terraform init**: Initializes a new or existing Terraform configuration. This command downloads and installs any required providers and modules.
2. **terraform plan**: Creates an execution plan showing what actions Terraform will take to apply the desired state. It does not make any actual changes to the infrastructure.
3. **terraform apply**: Applies the changes specified in the Terraform configuration files to reach the desired state.
4. **terraform destroy**: Destroys the infrastructure managed by Terraform. Be cautious when using this command, as it will delete resources.
5. **terraform validate**: Validates the configuration files for correct syntax and semantics.
6. **terraform fmt**: Rewrites Terraform configuration files to a canonical format.
7. **terraform get**: Downloads and installs modules and their dependencies.
8. **terraform state**: Used for advanced state management operations. Examples include **terraform state list**, **terraform state show**, and **terraform state mv**.
9. **terraform output**: Prints outputs from the state file.
10. **terraform import**: Imports existing resources into the Terraform state.
11. **terraform refresh**: Updates the state file to reflect the current real-world infrastructure.
12. **terraform show**: Outputs the current state or a specific resource.
13. **terraform taint**: Manually marks a resource as tainted, forcing it to be destroyed and recreated on the next **terraform apply**.
14. **terraform untaint**: Manually untaints a resource, marking it as clean.
15. **terraform workspace**: Manages workspaces, allowing you to work with multiple environments within a single configuration.
16. **terraform graph**: Generates a visual representation of the configuration and its dependencies.
17. **terraform console**: Interactive command-line console for evaluating expressions.
18. **terraform providers**: Prints a tree of the available providers and their versions.
19. **terraform version**: Prints the Terraform version information

All Linux commands:

1. **ls** - List directory contents
   * Example: $ ls
2. **pwd** - Print working directory
   * Example: $ pwd
3. **cd** - Change directory
   * Example: $ cd /path/to/directory
4. **mkdir** - Make directory
   * Example:$ mkdir new\_directory
5. **rm** - Remove files or directories
   * Example: $ rm file.txt
6. **cp** - Copy files or directories
   * Example: $ cp file.txt /path/to/destination
7. **mv** - Move or rename files or directories
   * Example: $ mv file.txt new\_name.txt
8. **touch** - Create a new file
   * Example: $ touch new\_file.txt
9. **cat** - Concatenate and display file content
   * Example: $ cat file.txt
10. **less** - Display file content page by page
    * Example: $ less large\_file.txt
11. **head** - Display the beginning of a file
    * Example: $ head file.txt
12. **tail** - Display the end of a file
    * Example: $ tail file.txt
13. **grep** - Search for patterns in files
    * Example: $ grep "pattern" file.txt
14. **find** - Search for files and directories
    * Example: $ find /path/to/search -name "pattern"
15. **ps** - Display information about running processes
    * Example: $ ps aux
16. **kill** - Terminate processes
    * Example: $ kill <process\_id>
17. **chmod** - Change file permissions
    * Example: $ chmod +x script.sh
18. **chown** - Change file ownership
    * Example: $ chown user:group file.txt
19. **df** - Display disk space usage
    * Example: $ df -h
20. **du** - Display file and directory space usage
    * Example: $ du -h
21. **free** - Display system memory usage
    * Example: free -h
22. **top** - Display system resource usage in real time
    * Example: $ top