DevOps Challenge

Name: Hamidullah Muslih

1. **What’s the difference between vertical and horizontal scaling?**

   Answer: Both scaling approaches are basically adding more resources to the existing infrastructure. But they are different in the case of implementation.

   - **Vertical scaling (scaling up):** This is an approach to adding more resources on top of an existing machine, for example, most of the relational SQL servers are vertically scalable. This means making the server stronger by increasing the capacity of the RAM, CPU, SSD, etc.
   - **Horizontal scaling (scaling out):** this is an approach to adding more machines to the pool of servers such that the load will be distributed. For example, No-SQL databases support horizontal scaling where the admin can add more machines to the pool of resources in the infrastructure.

   My experience: I implemented horizontal scaling through sharding in MongoDB and EC2 autoscaling on AWS. I also used vertical and horizontal scaling in K8s.

2. **If we want to execute a predefined script every day at a specific time, how can we achieve that**?

   Answer: We can schedule a predefined script as a cron job.

   My experience:

   - I used the Cron job in the Jenkins pipeline to check every minute for any changes in the main branch of the GitHub repo and trigger the following jobs, 1-fetch and build, 2-unit test, 3-integration test, 4-static code analysis, 5-quality gate analysis, and 6-Uploading to docker hub.
   - Moreover, I usually use Cron job to backup Jenkins configuration from /var/lib/jenkins to S3 buckets every week.

3. **Provide steps to set up Nginx reverse proxy, as well as reasons for it.**

   Answer: Reverse proxies are used to isolate internal servers from the internet or the outside world. It receives the request of the outside users and redirects them to the specific server. Below is some basic use case:

   - Used as SSL/TLS terminator and help the backend applications to work smoothly.
   - Acts as a load balancer between backend servers.

- Cache content and increase the performance.
- Isolate servers for security purposes.

Below are the steps to set up the Nginx reverse proxy:

In the below example we suppose to isolate Jenkins (<localhost>:<8080>) behind the Nginx reverse proxy.

a. Install Nginx server

```
sudo apt update
sudo apt install nginx
sudo systemctl stop nginx
```

b. Delete the default Nginx config file from /etc/Nginx/sites-enabled/

```
Sudo rm /etc/nginx/sites-enabled/default
```

c. Create a reverse proxy config file for the Jenkins server with the below content.

```
Sudo nano /etc/nginx/sites-available/jk-proxy
```

```
upstream jenkins {
  server 127.0.0.1:8080;
}
server {
  listen 80;
  server_name www.jenkins.yourdomain.com jenkins.yourdomain.com;
location / {
        return 301 https://$host$request_uri;
}
}

server {
    listen 443 ssl;

    server_name www.jenkins.yourdomain.com jenkins.yourdomain.com;

    ssl_certificate /etc/letsencrypt/live/jenkins.yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ jenkins.yourdomain.com /privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
            location / {
                        include /etc/nginx/proxy_params;
                              proxy_pass http://jenkins;
                        proxy_read_timeout  60s;
                        proxy_redirect      http://127.0.0.1:8080 https://
jenkins.yourdomain.com;
            }
}
```

d. Make a soft link of config file /etc/nginx/sites-available/jk-proxy to /etc/nginx/sites-enabled/jk-proxy

e. Check Nginx configuration

   nginx -t

f. Start the Nginx server

   sudo systemctl reload nginx

Note: if you don't have DNS server record for jenkins.yourdomain.com, use /etc/hosts file.

g. Access the URL jenkins.yourdomain.com from the browser.

My experience:

- I used a reverse proxy to isolate the Jenkins server.
- I used the Nginx ingress controller in Elastic Kubernetes Service, to reduce the cost of using ELB. And implemented rules/path with regex for backend services.

4. When would you use k8s, and when would you not? Support your answer with an example.

Answer: K8s is one of the leading container orchestration tools.
You should use K8s if your project is based on containerized microservice, where you will have multiple services running in containers and communicating with each other. It is very difficult to manage a multi-container environment. K8s manage container's lifecycle, resource allocation, request distribution, scale up and scale down.

You should not use K8s, if your application is based on monolith architecture and you intended to have a simple application that is not going to be scaled then K8s is not the choice. You can choose cloud-managed services which takes care of your application.

5. What is the most common file format to configure and run k8s workloads?

Answer: The common file type is YAML or YML format that we pass to Kubernetes API server to apply our workload on the cluster, although we can use JSON also it is not user-friendly.

My experience: I worked with K8s ingress, deployment, pods, replica sets, and many more.

6. What is a k8s pod?

Answer: Pod is the basic unit of the Kubernetes. We run our container/s inside a pod. Pods can support to run many types of container engines but the most widely used is the docker engine. When we run more than one container in a pod, the containers are so-called tightly coupled and share the resources on that specific pod.

My experience: I worked with K8s pods, ingress, deployment, replica sets, and many more.

7. What is a k8s ingress resource and what is it used for?

Answer: The ingress controller is used to filter incoming traffic to your cluster. It can act as a load balancer and routes the traffic based on the rules/paths you define in your ingress manifest file.
If you have many services running in your cluster and each of them should be exposed to the outside world by a cloud-managed load balancer. Using many load balancers leads to high operational costs therefore using ingress and exposing one endpoint to the world could save lots of money.
Since ingress is also a reverse proxy so it can act as SSL/TLS terminator and services can work smoothly in the cluster.

My experience: I used the Nginx ingress controller where I needed to expose an API and a front-end web app. I used regex to filter the traffic and then route them to a specific service.

8. How to manage environment variables and secrets under k8s.

Answer: There are two basic objects to use for environment variables and secrets in K8s.
Configmaps: is used to decouple the environment-specific configuration and make your application portable. Although you can set environmental variables in your deployment manifest it will be coupled with your deployment.
Moreover, Configmaps can also save sensitive data but it is not it does not provide security.

Secrets: For dealing with secrets we can use the secret object in k8s to keep the sensitive data such as SSH keys, username and passwords, and tokens. Secrets could be used as environmental variables or as mounted data by containers in a pod.

Secrets are stored unencrypted under API/ETCD storage although anyone who has access to the cluster can get the sensitive data.
Also if we are using the manual approach of creating secret (yaml manifest) we should not push it to online repo even if it is private. We can also use third-party software to be on the safe side such as Hashicorp Vault, AWS Secrets Manager, etc.

My experience: I used the secrets object for MariaDB and RabbitMq authentication in the K8s cluster.

9. **From your local terminal, how can you remotely connect to a k8s cluster in the cloud and have full management access?**

Answer: To connect to the K8s cluster you should install kubectl and get the kubeconfig file of your cloud-managed K8s cluster. Kubeconfig file is a YAML file and by default located in $HOME/.kube/config file. It contains basic configuration like certificate, username and password, cluster URL to connect.

My experience: I use the below command to get kubeconfig file after deploying EKS using terraform.

aws eks --region $(terraform output -raw region) update-kubeconfig --name $(terraform output -raw cluster_name)

10. **What are the benefits of using IaaC infrastructures like Terraform? Can you briefly describe the flow working with this tool?**

Answer: The rapid growth of the IT industry demands speed and consistency for its infrastructure. Thanks to the API of the cloud providers such as AWS, Azure, and GCP let the other tools and IT admins to create or destroy their resources. We can interact by two methods.

h. Imperative (what and how to do): where the admin manually types commands to create or destroy any resource. Or can write code for it in sequence such as Ansible Playbooks.
i. Declarative (what to do): where the admin writes the code and apply to the cloud provider. This approach is less error-prone.
   - IAAC allows us to repetitively create and destroy our infrastructure.
   - Less error-prone.

- Portable and reusable, imagine if the admin leaves the job you can hire a new one and he/she can use the same code to spin up and maintain your infrastructure.

If we take the example of Terraform which is one of the most popular tools out there. Terraform keeps the state of your infrastructure in terraform.tfstate file. Where each time when you apply new changes to your infrastructure, terraform compares the changes and tells you whether to apply or not.
In the case of the many terraform code developers, you can store terraform.tfstate file in remote storage, such that all the developers will know what is the current state of the infrastructure.

My experience: I deployed many services with Terraform on AWS cloud such as VPC, EKS, EC2, EIP, RDS, SG, and many more. Since we are a team, we use S3 bucket as backend to hold terraform.tfstate file.

11. Assuming we have an EC2 *t3a.small* instance running that is managed with Terraform, the local configuration file is edited and the instance type is changed to *t3a.large* , if the terraform apply command is run, what would happen?

Answer: In simple words, It will destroy ***t3a.small*** and create ***t3a.large***.
Basically, Terraform will compare the desired state with the current state (terraform.tfstate) file and it will see that ***t3a.small*** is no longer in the desired state and assumes that we don't need ***t3a.small*** it will destroy it and create ***t3a.large***. Therefore all the data or running service will be lost.

12. Can you describe what a monitoring system is and expand with some examples on how it can be used and for what reasons/benefits and what's the difference between metrics & logs?

Answer: A monitoring system is used to know the live state of the infrastructure or application resources. If you have many different microservice it is difficult to log in to each service and monitor them. The easiest way is to set up a monitoring service such as Prometheus + Grafana such that you will have a central place to know what is going on in your application service.
The monitory system helps you know when you should scale up/down your resources such as RAM, CPU, disk, and networking.

You can set up alerts, which means when there is a problem with a service for example a web server or database server, it can alert the admin on email, phone, and any other place.
We have two different monitoring techniques:

a. **Log**: is used to know what happened at a specific time. Even based monitoring.
b. **Metrics**: are about numbers, thresholds, and statistics. Number-based monitoring.

For example, we put a threshold if the CPU usage is above 70% to perform auto-scaling or alert the admin.

My experience: I worked with Prometheus + Grafana server, where I configured the Nginx server as the target to monitor its resource. I also set up alerts to notify to my email.

13. **How would you open a shell to get into a running container? (what would be the exact command you would run to achieve that)**

Answer: For a docker container in Docker engine environment I used the below command

docker exec -it <container name> /bin/bash *or* *docker-compose run <container name> <command>*

For a docker container in K8s environment, I use the below command.
kubectl exec --stdin --tty <pod name> -- /bin/bash

14. **Can you explain the basic differences between relational and non-relational DBs, and provide some use cases when one architecture would be better than the other?**

Answer:

a. Relational databases: are also called SQL-based databases, the data is structured in tables and the tables have the relation with each other. If we have structured data such as bank detail of the users where availability and consistency is the priority then we should go with Relational database. Most fintech companies use relational databases. Although it is very expensive to scale up the database and almost impossible to change the schema.

b. Non Relational databases: are also called NoSQL databases. There are many types such as key-value pair, document-based, graph-based, and column-based. This type of database is cheaper to scale and it supports horizontal scaling. We should use NoSQL database when we have unstructured data it means we don’t have a predefined schema. These unstructured data could be video, audio, logs file, and social media posts.

My experience: In one of our projects we used MariaDB with memcachd. Also, I implemented sharding in MongoDB.

15. You have ssh access to an Ubuntu or Debian Linux server, the disk is full and you need to clean it up, what would you do and which specific commands should you run?

Answer: After accessing the machine I will remove unnecessary files and folders:

a. Inspecting big files and folders and delete unused files:
```
$ cd / && sudo du -h --max-depth=1
```

b. Removing useless packages:
```
sudo apt-get autoremove
```

c. Deleting cache of package manager
```
sudo apt-get autoclean
```

d. Cleaning logs before 5 days:
```
sudo journalctl --vacuum-time=3d
```

e. Remove snap cache:
```
Sudo rm -rf /var/lib/snapd/cache
```

f. Find and delete unnecessary packages:

16. Assuming you have a Kubernetes workload running a set of microservices in production, the cluster was provisioned with Terraform. The k8s cluster is running under version 1.20. The version 1.21 has been released and we want to upgrade the cluster. What's the best approach you may take to upgrade the cluster without disturbing the entire workload and without any downtime?

Answer: Below are the steps to perform for upgrading the cluster version from 1.20 to 1.21.
Our cluster is made of two parts, control plan and node plane. We should first upgrade the control plane then node plane, and clients (kubectl), and finally manifest files.

In case of EKS below are the steps:

a. Upgrade EKS cluster version as below code show and run:
   Terraform plan/apply

```
module "eks" {
source  = "terraform-aws-modules/eks/aws"
version = "18.17.0"
cluster_name                    = "my-cluster"
cluster_version                 = "1.21"
}
```

b. Update system component versions kube-proxy, CoreDNS, and Amazon VPC CNI.
c. Select a new AMI for worker nodes.
d. Turn off autoscaler && creating a new worker group.
e. Drain pods from the old worker nodes.
f. Clean up extra resources which is the old worker group.
g. Update the manifest files based on the new API.

17. The following nodeJs app has been developed, and the team is requesting a complete CI/CD workflow under Gitlab.

Answer: I have very less experience with Gitlab, but I am a quick learner. I worked with Jenkins to create CI/CD. Below is the complete pipeline as Jekninsfile.
The following link has the complete CI/CD project finished task:
https://github.com/HamidullahMuslih/nrg.git