KUBERNETES - APPLICATION EXAMPLE INCLUDING LEGAL AND TECHNICAL BASICS AS WELL AS IMPLEMENTATION

BY

BRIGHT CHUKWUEBUKA JIWUEZE
(M.SC. CYBER SECURITY WINTER SEMESTER 22/23)

ABSTRACT

This research paper was conducted for the award of semester grade in the course **DATA ASPECT & PRIVACY**. This paperwork will give a general description of what Kubernetes is all about, it's components and architecture, the paper will also elaborate the technical basics and give practical examples of what Kubernetes can do, we would discuss the legal basics as well as the implementation of Kubernetes, finally, we do the summary and conclusion pointing out the advantages of this wonderful tool over other container orchestration tools.

INTRODUCTION AND SCOPE

DEFINTION: Kubernetes, or k8s for short, is an open-source container orchestration tool which was originally developed by google for management of containers from docker and also from other technologies, this simply implies that Kubernetes helps us mange containerized applications made of hundreds and thousands containers and helps us manage them in different environments, be it physical machine, virtual machine and cloud environment.

THE NEED FOR A CONTAINER ORCHESTRATION TOOL:

Organizations used to run on physical servers. The problem was, when performing multiple tasks on a single server, one application could take up most of the resources and cause others to underperform. One solution was to have more servers, but as you can imagine, *this got expensive pretty quickly.* Then things shifted towards virtualization. Multiple Virtual Machines (VMs) could run on a single physical server's CPU, which meant that several applications could run simultaneously without performance issues. VMs also allowed developers to isolate applications, adding an extra layer of security and flexibility. The overall system remained unaffected when one application had to be added, updated, or repaired. However, large memory usage was a main issue with VMs.

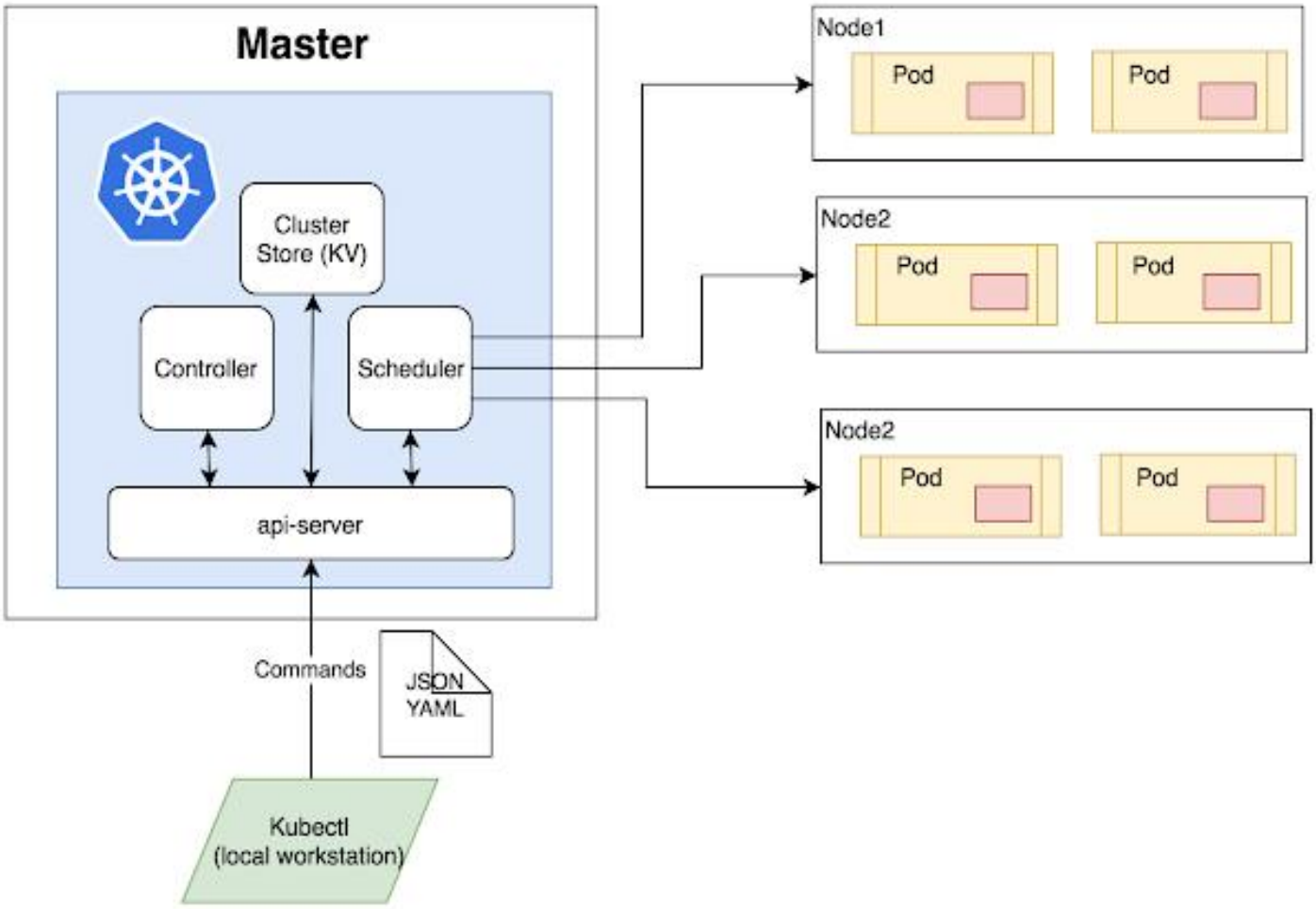
Now, lets discuss about **containers**, what are **containers**?

Containers are similar to VMs. They have their own operating systems, file systems, CPU, memory, and process space — and they can be decoupled from the main infrastructure. The main difference is: they have a much lower memory footprint due to their relaxed isolation properties, container can be built with some technologies, the most widely used mechanism to build container recently is docker using images.

Containers has a lower memory usage advantage over VMs, this gave rise to the high usage of containers in todays web services and application building, an application can use hundreds and thousands of containers in an environment, managing such environment using scripts can be more complex or not possible, because of this, there is a high demand for a proper way of managing different number of containers, this gave rise to an orchestration tool known as **Kubernetes** which guarantees the following;

- HIGH AVAILABILITY: The application has no downtime, it is always accessible by the users, if one container fails, another needs to take its place. Kubernetes handles this changeover automatically and efficiently by restarting, replacing, and killing failed containers that don't respond to a health check
- SCALABILITY: Application has high performance, kubernetes allows users to horizontally scale the total containers used based on the application requirements.
- DISASTER RECOVERY: Backups and restore, if infrastructure has a problem like data loss, server explode or something bad happens to the server, the infrastructure has some mechanism to back up the data and to restore it at the latest state, the application cannot lose any data.
- FUTURE-PROOFED SYSTEMS: As your system grows, you can scale both your software and the teams working on it because Kubernetes favours decoupled architectures. It can handle massive growth because it was designed to support large systems. Furthermore, all major cloud vendors support it, offering you more choice.
- POTENTIALLY CHEAPER THAN THE ALTERNATIVES: It's not suitable for small applications, but when it comes to large systems, it's often the most cost-effective solution because it can automatically scale your operations. It also leads to high utilization, so you don't end up paying for features you don't use. Most of the tools in the K8s ecosystem are open-source and, therefore, free to use.

KUBERNETIS ARCHTECTURE



Kubernetes has one master node , connected to it, is a couple of nodes where each node has a kublet process running on it, kublet process is a kubernete process that makes it possible for the clusters to communicate to each other and execute some task like running applications, these nodes also have pods in them which is the smallest unit of kubernetes and this is where containers and applications are running.

The master actually runs several kubernetes processes that are absolutely necessary to run and manage cluster properly, those processes include;

API SERVER: This is the entry point for a kubernetes cluster, it determines how users interact with cluster also it determines if a request is valid and then processes it.

SCHEDULAR: This is responsible for responsible for scheduling containers on different nodes base on workload and available server resources on each node.

THE CONTROLLER: This basically keeps an overview of what is happening in the cluster, whether something need to be repaired or may be if a container died and need to be restarted.

KUBECTL: A command line tool for Kubernetes cluster, where you can give commands to the server.

JSON & YAML: The coding language written in an editor Kubernetes uses to perform its functions, e.g., managing containers, deployment, stateful set etc.

BASIC COMPONENTS OF KUBERNETES

POD: This is the smallest unit of kubernetes, abstraction over container, it creates running environment on top of the container, usually one application per pod, each pod have an ip address that it uses to communicate to other pods linked to it e.g. in a node, a pod containing a container that runs an app and another pod with a container that runs the database for the app communicate using ip adress, if any of the pods or container crashes and get restarted on built newly, it will require a new ip adress except for the use **service**.

SERVICE: A permenent ip adress that can be used by pods, incase loss or damage container, another is created, the ip adress remains, the life cycle of pod and services are not connected even if the pod dies, the service and the ip adress stays.

INGRESS: Kubernetes Ingress is an API object that provides routing rules to manage access to the service within a Kubernetes cluster. This typically uses HTTPS and HTTP protocols to facilitate the routing.

ConfigMap: A ConfigMap is an API object used to store non-confidential data in key-value pairs, ConfigMap does not provide secrecy or encryption

SECRET: this is like configMap but use to store secrete data and credentials in base 64 encoded e.g. username and password.

VOLUME: This is specifically for data storage in the Kubernetes cluster, it contains data accessible containers, it works by physically attaching a storage medium to the hard drive of the local machine.

DEPLOYMENT: A Kubernetes Deployment tells Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can help to efficiently scale the number of replica pods, it provides declarative updates for Pods and ReplicaSets.

STATEFULSET: This is the workload API object used to manage stateful applications. Manages the deployment and scaling of a set of Pods, and provides guarantees about the ordering and uniqueness of these Pods. Like a Deployment, a StatefulSet manages Pods that are based on an identical container spec. It is mainly use in duplication of database pods to avoid data mismanagement.

Kubernetes provides you with:

- **Service discovery and load balancing** Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
- **Storage orchestration** Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
- **Automated rollouts and rollbacks** You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.
- **Automatic bin packing** You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- **Self-healing** Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Secret and configuration management** Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application

configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

TECHNICALITY/TECHNICAL BASICS OF KUBERNETES

Kubernetes automates operational tasks of container management and includes built-in commands for deploying applications, rolling out changes to your applications, scaling your applications up and down to fit changing needs, monitoring your applications, and more making it easier to manage applications.

Kubernetes runs on a virtual machine e.g. VMware, hyperkit, virtual box, docker e.t.c. once you have a virtual machine running in the system, you can install minikube and kubectl using command line or manual installation depending on the operating system you are using. Minikube and kubectl commands are used in a command line interface in Windows, Mac, and Linux.

PRACTICAL APPLICATION EXAMPLES AND IMPLEMENTATION OF KUBERNETES

The application examples of Kubernetes are too many, but for the purpose and straight to the point understanding of this project work, we will look at only two application examples in this paper.

1. deploying a web-server with an nginx image from docker hub using kubernetes
2. deploying a mongo database-server yaml configuration file using kubernetes

DEPLOYING A WEB-SERVER WITH AN NGINX IMAGE FROM DOCKER HUB USING KUBERNETES

In this practical section, we would deploy a web server with nginx image from docker hub.com, the version of nginx we would use nginx:latest.

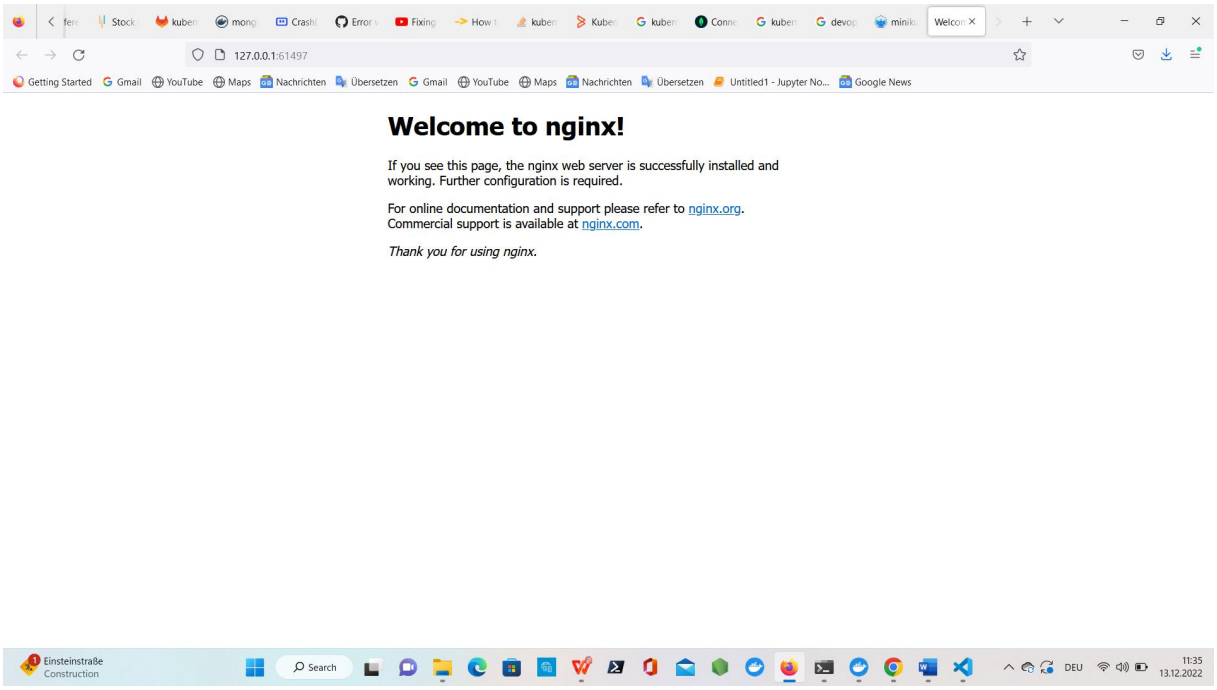
First, we create a .yaml configuration file in an editor, then deploy it using kubectl and minikube commands in a command line tool in other to manage the container to run in a web browser, we would also give the container a port number and also create a service that will give the container an external IP address.

Nginx-deployment.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: webserver
spec:
  replicas: 4
  selector:
    matchLabels:
      app: webserver
  template:
    metadata:
      labels:
        app: webserver
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Once the configuration is done, I used my windows powershell to deploy the container using this command “kubectl apply -f nginx-deployment.yml”, I got this response from Kubernetes server showing that the container has been created “deployment.apps/nginx-deployment created”.

Next, I exposed a service to give my deployed web-server an external IP address with this command “kubectl expose deployment nginx-deployment --type=LoadBalancer --name=nginx-web-server” and I got this response showing that the sever has successfully exposed “service/nginx-web-server exposed”. Running the service with minikube command “minikube service nginx-web-server”, it gave my web-server and IP address and opened it on a web browser..



With this so far, I can do some other development on the website, but for the purpose of this project, I should stop here for now and go to the next example.

DEPLOYING A MONGO DATABASE-SERVER USING WITH A MONGO IMAGE FROM DOCKER HUB USING KUBERNETES

To successfully, configure a mongodb file, we need to apply the following containers;

- Deployment/Pod which tells Kubernetes how to create or modify instances of the pods that hold a containerized application
- Service which assigns a unique Ip address to the pod
- Configmap which allows to store data as key-value pairs
- Secret which will contain sensitive data like username and password to access the database, this is coded in base 64 encoded form

Using an editor to create .yaml configurations and then applying Kubernetes using Command line tool.

First, we create the secret

Mongodb-secret.yml

Here, we first convert the username and password in base 64 encoded to be more secured, using a command line tool..

echo -n ‘username’ | base 64

echo -n ‘password’ | base 64

Then we save the output in a variable as recorded in the figure below

```yaml
apiVersion: v1
kind: Secret
metadata:
    name: mongodb-secret
type: Opaque
data:
    mongo-root-username: dXNlcm5hbWU=
    mongo-root-password: cGFzc3dvcmQ=
```

In the command line tool, we use kubectl to create the secret by typing this command “kubectl apply -f Mongodb-secret.yml”

After creating the secret, we create the deployment alongside it’s service that gives the deployment a unique Ip address

Mongodb-deployment.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo
        ports:
        - containerPort: 27017
```

```
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-username
        - name: MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-password
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

In the command line tool, we use kubectl to create the deployment by typing this command “kubectl apply -f Mongodb-deployment.yml”.

We now create the configmag that stores data in key-value pairs in the express database.

Mongodb-configmap.yml

```
apiVersion: v1
kind: ConfigMap
metadata:
    name: mongodb-configmap
data:
  database_url: mongodb-service
```

In the command line tool, we use kubectl to create the configmap by typing this command “kubectl apply -f Mongodb-configmap.yml”.

Lastly, we create the express for the database and also the service

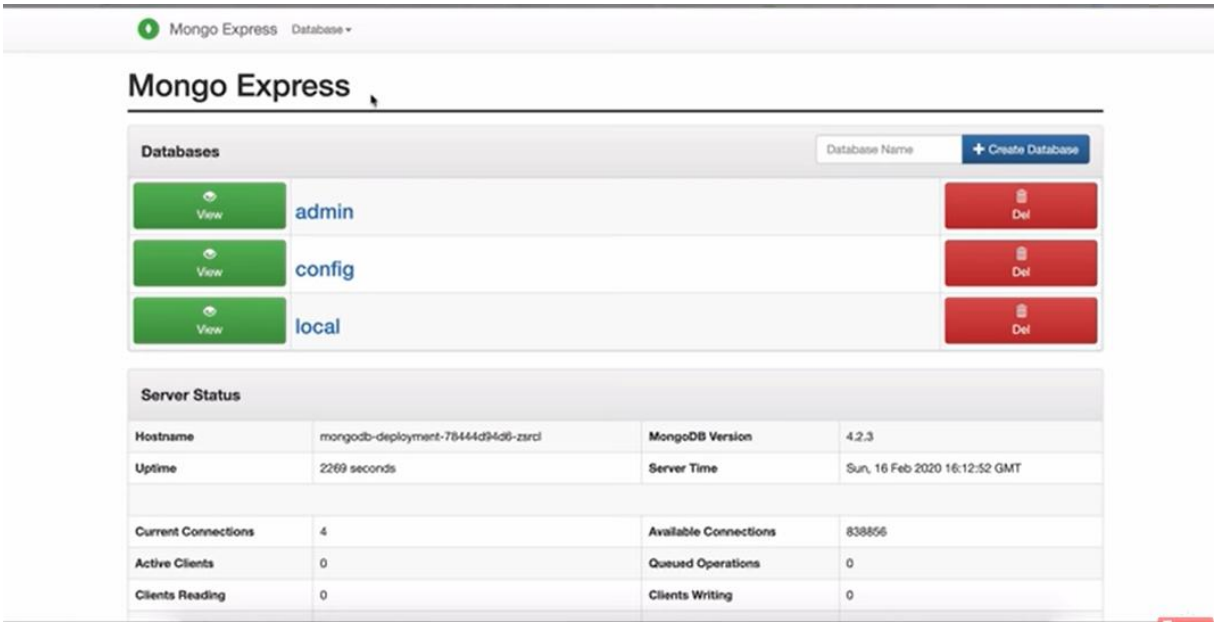
Mongodb-express.yml

```
apiVersion: apps/v1
kind: Deployment
```

```yaml
metadata:
  name: mongo-express
  labels:
    app: mongo-express
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo-express
  template:
    metadata:
      labels:
        app: mongo-express
    spec:
      containers:
      - name: mongo-express
        image: mongo-express
        ports:
        - containerPort: 8081
        env:
        - name: ME_CONFIG_MONGODB_ADMINUSERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-username
        - name: ME_CONFIG_MONGODB_ADMINPASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-password
        - name: ME_CONFIG_MONGODB_SERVER
          valueFrom:
            configMapKeyRef:
              name: mongodb-configmap
              key: database_url
---
apiVersion: v1
kind: Service
metadata:
  name: mongo-express-service
spec:
  selector:
    app: mongo-express
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8081
      targetPort: 8081
      nodePort: 30000
```

In the command line tool, we use kubectl to create the express by typing this command “kubectl apply -f Mongodb-express.yml”.

After applying this configuration files using kubectl command, we can now use the minikube command to run the express mongodb database setup using this command line “minikube service mongodb-express-service”, this command will assign a permanent Ip address to the deployed mongodp and open it in a web browser in this format...



With this Mongo Express database, you can link it to another container that deployed a software or a webserver and use it to save data and inputs from both manufacturer and users.

Other areas Kubernetes can be applied includes.

Cloud computing: Kubernetes automates operational tasks of container management and includes built-in commands for deploying applications, rolling out changes to your applications, scaling your applications up and down to fit changing needs, monitoring your applications, and more—making it easier to manage applications. Kubernetes runs on Amazon Web Services (AWS), Microsoft Azure, and the Google Cloud Platform (GCP), and you can also run it on premises.

DevOps Engineering: Kubernetes is the most popular container orchestration platform and has become an essential tool for DevOps teams. Application

teams can now deploy containerized applications to Kubernetes clusters, which can run either on-premises or in a cloud environment.

The New York Times adapts Kubernetes: The New York Times is one of the biggest publications in the world, with over 150 million monthly unique readers. The NYT like any other publication and magazine entered the digital era to cater to its audiences, who now consume content on their phones instead of the grey paper. One of the craziest problems that it faced was that of Fake News. They wanted to avoid sending out incorrect and unverified news to their readers, for which they wanted to check and cross-check each piece of information aggressively. Now, the amount of data processed in a newsroom like this would require a strict back-up. They began their journey with LAMP Stack but were broadly dissatisfied and soon moved on to a React-based front end using Apollo. A few years ago, the company decided to move out its data centres and less critical applications that shall be managed on VMs. That's when Kubernetes offered its solution GKE. There was a significant increase in the spread, and deployment, and productivity. The Legacy deployments took less than 45 minutes and are now pushed in just a few minutes. NYT has now moved on from a ticket-based system to requesting resources that deploy weekly schedules. This has enabled developers to push updates independently.

Apache Spark with Kubernetes: Apache Spark is an open-source distributed computing framework. Spark helps manage many data sets with the help of a cluster of machines. But it does not manage the machine, this is where Kubernetes helps. Spark creates its own driver which runs within the Kubernetes Pod. The driver then creates executors that also run within Kubernetes pods, connects them, and then executes the application code. Once the application is completed, the executioner pods are terminated and cleared, but the driver pods persist logs and remain in the "completed" state in the API. Kubernetes takes care of scheduling the driver and executor pod. fabric8 is used to communicate to Kubernetes API. Kubernetes enables containerization which is helpful in traditional software engineering and big data as well as Spark. The containers make the application more portable, simplifies the packaging of dependencies, and build a reliable and secure workflow. In the recent version of Spark, Spark on Kubernetes is marked as Generally Available and production-ready in the official docs.

Google Kubernetes Engine: Google Cloud is where Kubernetes was first originated in 2014. Google developed the tool to containerize their 15 years' worth of workloads. When they announced that it was an open-source tool, it paved the way for the community to make their contributions to the tool. Providing automated container orchestration, Kubernetes improves a Company's reliability and reduces the time and effort of a resource. Google still uses Borg internally and Kubernetes plays the role of 3rd party container orchestration system. The aim to build Kubernetes was to learn from their previous mistakes and improve on the design. Kubernetes was never built with the agenda to replace Borg, as it would've taken a tremendous amount of effort for the engineers to migrate. Hence developers learned from their past experiences and developed an open-sourced version for various other companies to depend on. Google Kubernetes Engine (GKE) is a secured and managed service with a 4-way auto-scaling and multi-cluster support. Companies like Pizza-Hut U.S, heavily depend on Google solutions including GKE. They used the tools to transform their E-Commerce infrastructure and increase the response time for orders.

LEGAL BASICS OF KUBERNETES

Kubernetes as an orchestration tool for deploying and managing of containers which may include database servers, web servers, and software systems, have implemented some legal basics in its deployment.

A Kubernetes deployed software, web and database server have great security assurance because most of these servers it deploys are meant to store and process individual sensitive data, Kubernetes works mostly in cloud hosting and security, that's why it is one of the widest and safest tools used in cloud security today, it always works in accordance with Art. 32 Para. 1(b) GDPR. Which is the ability to ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services, referencing mostly to its disaster recovery and high availability roles. Therefore, scanning through this project work, you would agree with me that the technicality, scope, architecture, and implementation of Kubernetes is to a greater extent taking cognizance on this section of the GDPR, also, plays a major role in other aspect of IT security by deploying secured software and servers.

CONCLUSION

With the widespread adoption of containers among organizations, Kubernetes, the container-centric management software, has become a good standard to deploy and operate containerized applications. Kubernetes has built-in commands to handle a lot of the heavy lifting that goes into application management, allowing you to automate day-to-day operations. You can make sure applications are always running the way you intended them to run. When you install Kubernetes, it handles the computing, networking, and storage on behalf of your workloads. This allows developers to focus on applications and not worry about the underlying environment. Kubernetes continuously runs health checks against your services, restarting containers that fail, or have stalled, and only making available services to users when it has confirmed they are running, it has greater advantage over other orchestration tools like Docker Swarm, and Apache Mesos because of its continuous automated deployment, currently, it is a widely used container orchestration platform among establishments today.

SUMMARY

In this paper work, we were able to explain in details about a container orchestration tool called “Kubernetes”, what it is made up of, what it can do, what it cannot to, how to use it, when to use it, where to use it, who uses it and why it ought to be used, we also talked about the role of the tool in cloud hosting/security and data protection(Confidentiality, integrity and availability). Finally, we concluded with some key points about Kubernetes and why it is mostly preferred among establishments today.

REFERENCES

- **European General Data Protection *https://gdpr-info.eu/***
- **Kubernetes tutorial by Nana, from Zero to Hero *https://www.youtube.com/watch?v=X48VuDVv0do***