



CS777 Big Data Analytics

TERM PAPER REPORT

KUBERNETES

**Aditya
Maheshwari**

Dr. Farshid Alizadeh

BU MET, Fall 2023

INTRODUCTION

DEFINITION

Kubernetes, at its core, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. It provides a framework for efficiently managing containers, abstracting away the underlying infrastructure, and enabling the seamless orchestration of application components across a distributed environment. In essence, Kubernetes enables the "containerization" of the cloud, allowing organizations to abstract their applications into lightweight, portable containers that can run consistently across diverse environments, from on-premises data centers to multi-cloud and hybrid cloud settings.



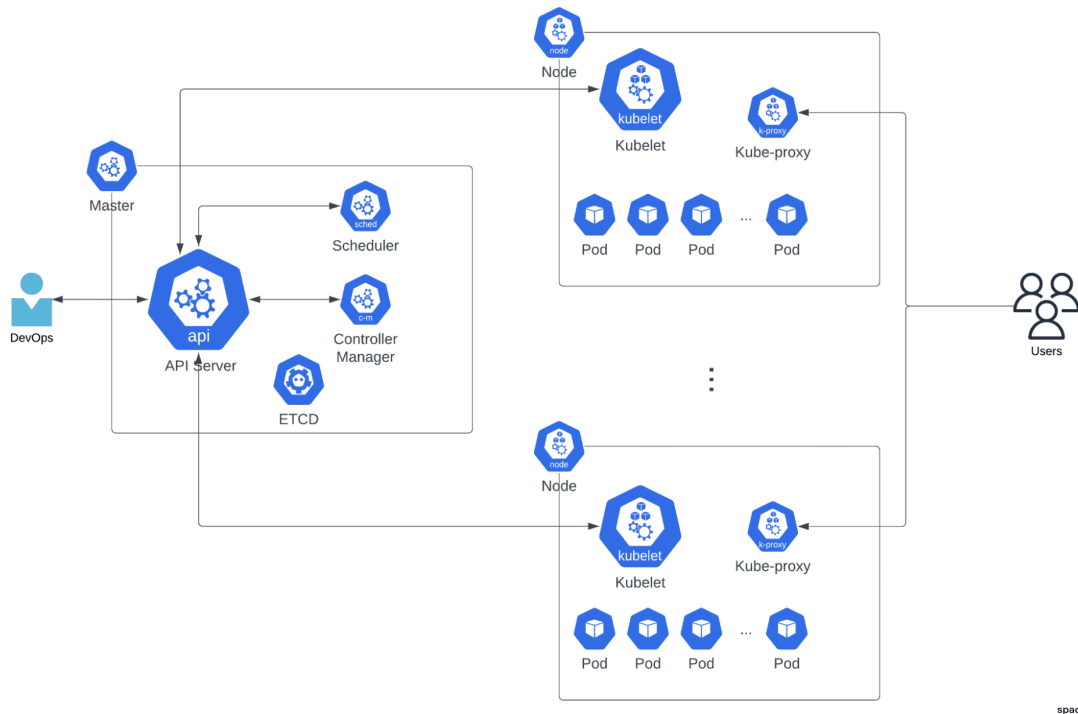
kubernetes

SIGNIFICANCE

The significance of Kubernetes in the realm of containerization and cloud computing cannot be overstated. With the advent of container technology, such as Docker, the need for a sophisticated orchestration system became evident. Kubernetes stepped in to fill this void, offering a comprehensive solution to many of the challenges that arose as containers proliferated. It simplifies the management of complex, microservices-based applications, which can consist of multiple containers and require dynamic scaling, fault tolerance, and load balancing. The project, originally developed by Google, has seen rapid adoption by both enterprises and cloud providers. Its vibrant open-source community ensures that Kubernetes continually evolves, adding new features and addressing real-world use cases.

ARCHITECTURE

Kubernetes follows a distributed architecture designed to manage containerized applications at scale. At the heart of this architecture is a cluster, which is a collection of nodes that work together to run container workloads efficiently. The architecture of Kubernetes consists of a distributed control plane responsible for managing the state of the cluster, and nodes that run containers within Pods.



Control Plane

The control plane, often referred to as the master node, is responsible for making global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (e.g., scaling). It is the brain of the Kubernetes cluster and includes the following components:

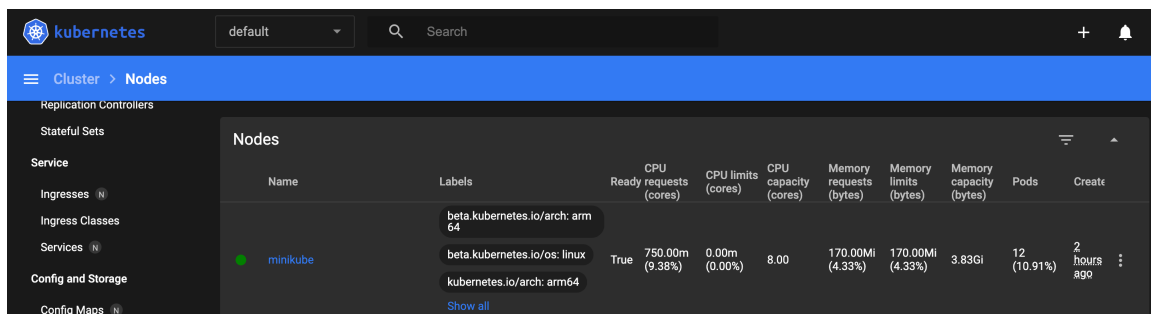
1. **API Server**: The API server is the entry point for all interactions with the Kubernetes cluster. It exposes the Kubernetes API, which is used by both users and other Kubernetes components to perform various operations. It validates and processes API requests, then communicates with other components to manage the cluster.
2. **etcd**: etcd is a distributed key-value store that serves as the cluster's primary data store. It stores all configuration data and state information of the cluster. This is critical for maintaining consistency and ensuring the cluster's high availability.

3. Controller Manager: The controller manager enforces the desired state of the system. It includes various controllers (e.g., Replication Controller, Endpoints Controller) that handle tasks like scaling, replication, and endpoints management. These controllers watch the API server for changes and make adjustments to the cluster accordingly.
4. Scheduler: The scheduler is responsible for placing containers onto available nodes in the cluster. It evaluates various factors, including resource requirements and constraints, to make optimal decisions about where to run containers.

Nodes

Nodes are the worker machines in the Kubernetes cluster, responsible for running containers and workloads. Each node typically runs a container runtime, such as Docker, containerd, or CRI-O. Here are the core components on a node:

1. Kubelet: The Kubelet is an agent that runs on each node and communicates with the control plane. It is responsible for ensuring that containers are running in a Pod. It receives Pod specifications from the API server and takes care of their execution.
2. Kube Proxy: Kube Proxy is responsible for network traffic management within the cluster. It maintains network rules and forwards traffic to the appropriate services or pods. This helps in load balancing and service discovery.
3. Container Runtime: The container runtime, such as Docker, is responsible for pulling and running containers based on the Pod specifications provided by the Kubelet.



Name	Labels	CPU Ready	CPU requests (cores)	CPU limits (cores)	CPU capacity (cores)	Memory requests (bytes)	Memory limits (bytes)	Memory capacity (bytes)	Pods	Create
minikube	beta.kubernetes.io/arch: arm64 beta.kubernetes.io/os: linux kubernetes.io/arch: arm64	True	750.00m (9.38%)	0.00m (0.00%)	8.00	170.00Mi (4.33%)	170.00Mi (4.33%)	3.83Gi	12 (10.91%)	2 hours ago

Pods

A Pod is the smallest deployable unit in Kubernetes. It represents a single instance of a running process in the cluster. Pods are used to encapsulate one or more containers that are tightly coupled and share the same network namespace and storage volumes. Key points about Pods include:

- Containers within the same Pod share the same IP address and port space, making it easy for them to communicate.

- Pods are ephemeral and can be scheduled, scaled, and replaced easily.
- Pods provide a mechanism for grouping related containers and ensuring they run on the same node.

kubernetes									
default									
Search									
Workloads > Pods									
Pods									
Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created	
square-job-9clmm	gettyimages/spar	batch.kubernetes.io/controller-uid: fb4cb4f0-aa25-4579-a7f3-39df83650b84 batch.kubernetes.io/job-name: square-job controller-uid: fb4cb4f0-aa25-4579-a7f3-39df83650b84	minikube	Completed	0	-	-	an hour ago	
nginx-deployment-55f598f8d-52k78	nginx	app: nginx pod-template-hash: 55f598f8d	minikube	Running	0	-	-	2 hours ago	
nginx-deployment-55f598f8d-6rj66	nginx	app: nginx pod-template-hash: 55f598f8d	minikube	Running	0	-	-	2 hours ago	
nginx-deployment-55f598f8d-xlkrf	nginx	app: nginx pod-template-hash: 55f598f8d	minikube	Running	0	-	-	2 hours ago	

INSTALLATION AND SETUP

Kubernetes:

Install with Homebrew on macOS

If you are on macOS and using [Homebrew](#) package manager, you can install kubectl with Homebrew.

1. Run the installation command:

```
brew install kubectl
```

or

```
brew install kubernetes-cli
```

2. Test to ensure the version you installed is up-to-date:

```
kubectl version --client
```

Minikube:

1 Installation

Click on the buttons that describe your target platform. For other architectures, see [the release page](#) for a complete list of minikube binaries.

Operating system	Linux	macOS	Windows
Architecture	x86-64	ARM64	
Release type	Stable	Beta	
Installer type	Binary download	Homebrew	

To install the latest minikube **stable** release on **ARM64 macOS** using **Homebrew**:

If the [Homebrew Package Manager](#) is installed:

```
brew install minikube
```

If `which minikube` fails after installation via brew, you may have to remove the old minikube links and link the newly installed binary:

```
brew unlink minikube  
brew link minikube
```

2 Start your cluster

From a terminal with administrator access (but not logged in as root), run:

```
minikube start
```

If minikube fails to start, see the [drivers page](#) for help setting up a compatible container or virtual-machine manager.

3 Interact with your cluster

If you already have `kubectl` installed (see [documentation](#)), you can now use it to access your shiny new cluster:

```
kubectl get po -A
```

Alternatively, `minikube` can download the appropriate version of `kubectl` and you should be able to use it like this:

```
minikube kubectl -- get po -A
```

[Copy](#)

You can also make your life easier by adding the following to your shell config: (for more details see: [kubectl](#))

```
alias kubectl="minikube kubectl --"
```

Initially, some services such as the storage-provisioner, may not yet be in a Running state. This is a normal condition during cluster bring-up, and will resolve itself momentarily. For additional insight into your cluster state, `minikube` bundles the Kubernetes Dashboard, allowing you to get easily acclimated to your new environment:

```
minikube dashboard
```

4 Deploy applications

Service

LoadBalancer

Ingress

Create a sample deployment and expose it on port 8080:

```
kubectl create deployment hello-minikube --image=kicbase/echo-server:1.0  
kubectl expose deployment hello-minikube --type=NodePort --port=8080
```

It may take a moment, but your deployment will soon show up when you run:

```
kubectl get services hello-minikube
```

The easiest way to access this service is to let `minikube` launch a web browser for you:

```
minikube service hello-minikube
```

Alternatively, use `kubectl` to forward the port:

```
kubectl port-forward service/hello-minikube 7080:8080
```

Tada! Your application is now available at <http://localhost:7080/>.

You should be able to see the request metadata in the application output. Try changing the path of the request and observe the changes. Similarly, you can do a POST request and observe the body show up in the output.

5 Manage your cluster

Pause Kubernetes without impacting deployed applications:

```
minikube pause
```

Unpause a paused instance:

```
minikube unpause
```

Halt the cluster:

```
minikube stop
```

Change the default memory limit (requires a restart):

```
minikube config set memory 9001
```

Browse the catalog of easily installed Kubernetes services:

```
minikube addons list
```

Create a second cluster running an older Kubernetes release:

```
minikube start -p aged --kubernetes-version=v1.16.1
```

Delete all of the minikube clusters:

```
minikube delete --all
```

Creating a cluster using terminal commands (manually):

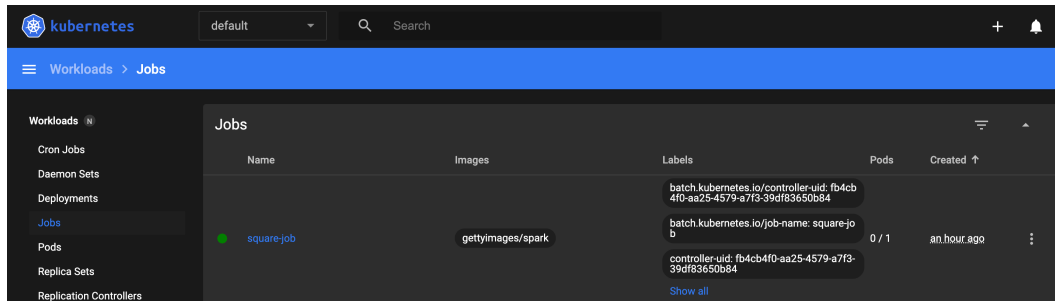
```
1 Terminal commands to create a minikube cluster
2
3 minikube status
4 minikube start --driver=docker
5 minikube ip
6 minikube ssh
7 docker ps
8 kubectl cluster-info
9 kubectl get nodes
10 kubectl get pods
11 kubectl get namespaces
12 kubectl get pods --namespace = kube-system
13 kubectl run nginx --image = nginx
14 kubectl describe pod nginx
15 minikube ssh
16 kubectl get pods -o wide
17 alias k='kubectl'
18 k create deployments
19 k get pods
20 k describe deployment nginx-deployment
21 k scale deployment nginx-deployment --replicas=5
22 k get deployments
23 k expose deployment nginx-deployment --port = 8080 --target-port = 80
24 k get services
25 k describe service nginx-deployment
```

Creating a cluster using a yaml file:

```
kubernetes-dashboard.yaml x
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: nginx-deployment
5 spec:
6   replicas: 3
7   selector:
8     matchLabels:
9       app: nginx
10  template:
11    metadata:
12      labels:
13        app: nginx
14    spec:
15      containers:
16        - name: nginx
17          image: nginx
18          ports:
19            - containerPort: 80
20 ---
21 apiVersion: v1
22 kind: Service
23 metadata:
24   name: nginx-service
25 spec:
26   selector:
27     app: nginx
28   ports:
29     - protocol: TCP
30       port: 8080
31       targetPort: 80
32   type: NodePort
```

Submitting a job in Kubernetes:

Job creation in Kubernetes Dashboard:



YAML file:

```

1  apiVersion: batch/v1
2  kind: Job
3  metadata:
4    name: square-job
5  spec:
6    template:
7      metadata:
8        name: square-job
9      spec:
10     containers:
11     - name: pyspark
12       image: gettyimages/spark
13       command: ["python"]
14       args: ["/mnt/square_calculator.py"]
15       volumeMounts:
16       - name: workdir
17         mountPath: /mnt
18       restartPolicy: Never
19     volumes:
20     - name: workdir
21       hostPath:
22         path: /mnt
23

```

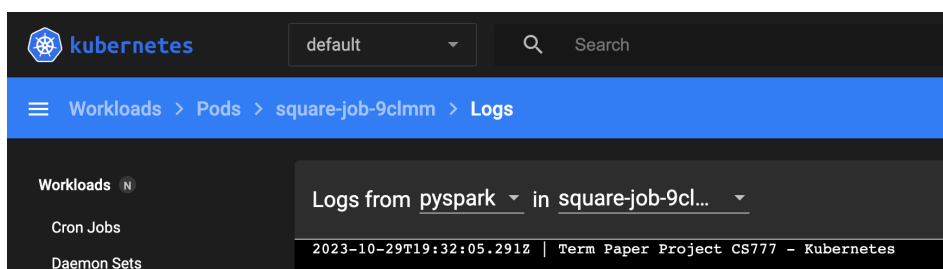
Python file:

```

print.py - /Users/adit0106/Desktop/print.py (3.10.6)
print("Term Paper Project CS777 - Kubernetes")

```

Logs in Kubernetes Dashboard in cluster pod:



KEY FEATURES

1. Automatic Scaling: Efficiently handles varying workloads, reducing costs, and improving performance.
2. Load Balancing: Enhances application availability, reliability, and simplifies traffic routing.
3. Self-Healing: Ensures application resilience by quickly recovering from failures.
4. Rolling Updates: Supports seamless, zero-downtime application updates.
5. Declarative Configuration: Simplifies application management by defining desired states.
6. Immutable Infrastructure: Improves consistency and reliability while reducing configuration drift.

CHALLENGES

1. Complexity: Kubernetes can be complex to set up and manage, particularly for small or inexperienced teams. The Kubernetes community is developing tools like Helm and Kustomize to simplify deployment and configuration management.
2. Resource Demands: Running Kubernetes clusters can be resource-intensive, both in terms of hardware and operational overhead.
3. Learning Curve: Kubernetes has a steep learning curve, requiring time and effort to master.
4. Networking Complexity: Setting up and configuring network policies and services can be complex.
5. Compatibility: Ensuring compatibility between different Kubernetes versions and configurations can be complex.

CONCLUSION

The introduction of Kubernetes has indeed transformed the landscape of Big Data, bringing with it a potent blend of efficiency, scalability, and cost-effectiveness. Among its many applications, Kubernetes has proven to be an invaluable tool for running distributed processing frameworks like Apache Hadoop and Apache Spark. Leveraging Kubernetes's built-in features, organizations can enhance the efficiency of these data processing workloads. Moreover, Kubernetes facilitates the streamlined management of data storage and data management systems. Its capabilities extend to deploying and managing distributed databases, exemplified by MongoDB, as well as data processing systems such as Apache Kafka. In essence, Kubernetes serves as a versatile, all-encompassing solution for optimizing Big Data operations, making them more agile, scalable, and cost-efficient.

REFERENCES

- [1] https://youtu.be/d6WC5n9G_sM?si=6C-iUdB4lRe6PEvQ
- [2] <https://kubernetes.io/releases/download/>
- [3] <https://kubernetes.io/docs/tasks/tools/install-kubectl-macos/#install-with-homebrew-on-macos>
- [4] <https://minikube.sigs.k8s.io/docs/start/>