Overview of Kubernetes

What is Kubernetes[180]? It is an open-source orchestration system for containers developed by Google and open-sourced in 2014. Kubernetes is a useful tool for working with containerized applications. Given our previous work with Docker containers and containerizing an app, working with Kubernetes is the next logical step. Kubernetes is born out of the lessons learned in scaling containerized apps at Google[181]. It works for automating deployment, scaling, and the management of containerized applications.

Learn an overview of Kubernetes in the following screencast.

Video Link: https://www.youtube.com/watch?v=94DTEQp0giY[182]

- What are the benefits of using Kubernetes?

Kubernetes is the standard for container orchestration. All major cloud providers support Kubernetes. Amazon through Amazon EKS[183], Google through Google Kubernetes Engine GKE[184] and Microsoft through Azure Kubernetes Service (AKS)[185].

Kubernetes is also a framework for running distributed systems at "planet-scale"[186]. Google uses it to run billions of containers a week.

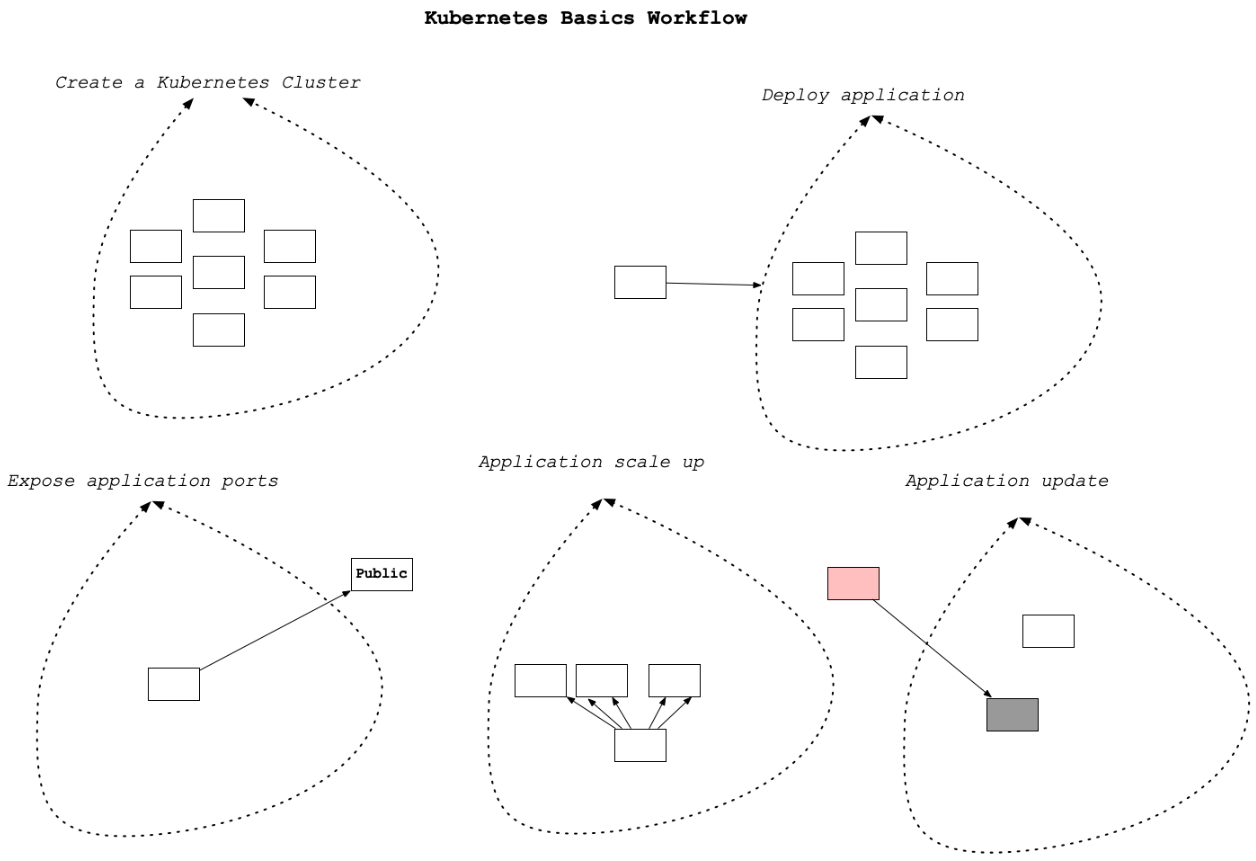
- A few of the Capabilities of kubernetes include:
  - High availability architecture
  - Auto-scaling
  - Rich Ecosystem
  - Service discovery
  - Container health management
  - Secrets and configuration management

The downside of these features is the high complexity and learning curve of Kubernetes. You can read more about the features of Kubernetes through the official documentation[187].

- What are the basics of Kubernetes?

The core operations involved in Kubernetes include creating a Kubernetes Cluster, deploying an application into the cluster, exposing application ports, scaling an application, and updating an application.

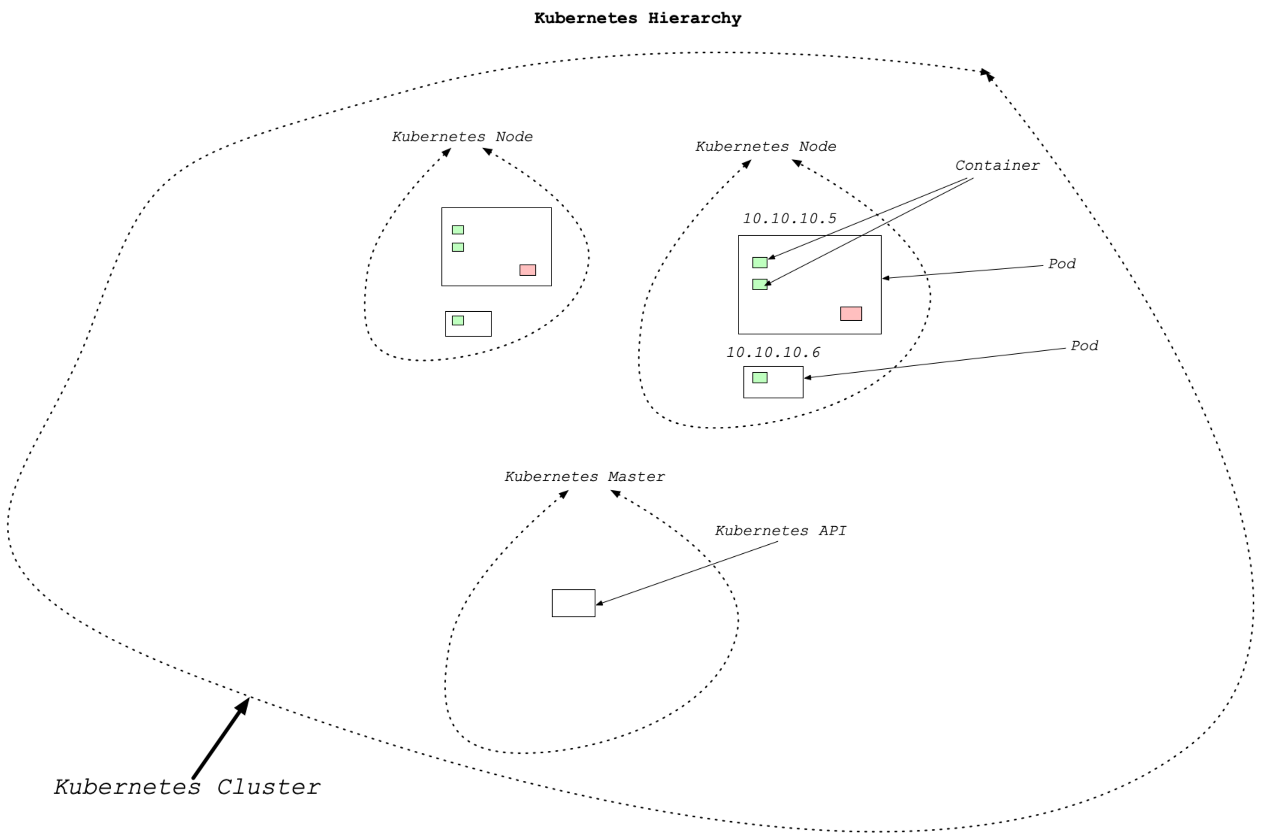
[180]https://github.com/kubernetes/kubernetes
[181]https://queue.acm.org/detail.cfm?id=2898444
[182]https://www.youtube.com/watch?v=94DTEQp0giY
[183]https://aws.amazon.com/eks/
[184]https://cloud.google.com/kubernetes-engine
[185]https://azure.microsoft.com/en-us/services/kubernetes-service/
[186]https://kubernetes.io/
[187]https://kubernetes.io/docs/home/



kubernetes-basic-workflow

- What are the Kubernetes (cluster) architecture?

The core of Kubernetes is the cluster. Inside of this, the Containers run in the collection. The core components of the cluster include a cluster master and nodes. Inside the nodes, there is yet another hierarchy. This order displays in the diagram. A Kubernetes node can contain multiple pods, containing multiple containers and volumes.



kubernetes-hierarchy

- How do you set up a Kubernetes cluster?

There are two main methods: set up a local cluster (preferably with Docker Desktop) or provision a cloud cluster: Amazon through Amazon EKS[188], Google through Google Kubernetes Engine GKE[189] and Microsoft through Azure Kubernetes Service (AKS)[190].

If you are using Docker and have enabled kubernetes[191], you already have a standalone Kubernetes server running. This step would be the recommended way to get started with Kubernetes clusters.

- How do you launch containers in a Kubernetes cluster?

Now that you have Kubernetes running via Docker desktop, how do you launch a container? One of the easiest ways is via[192] the `docker stack deploy --compose-file` command.

The `yaml` example file looks like the following:

[188]https://aws.amazon.com/eks/
[189]https://cloud.google.com/kubernetes-engine
[190]https://azure.microsoft.com/en-us/services/kubernetes-service/
[191]https://docs.docker.com/docker-for-mac/#kubernetes
[192]https://docs.docker.com/docker-for-mac/kubernetes/

```
version: '3.3'

services:
  web:
    image: dockersamples/k8s-wordsmith-web
    ports:
     - "80:80"

  words:
    image: dockersamples/k8s-wordsmith-api
    deploy:
      replicas: 5
      endpoint_mode: dnsrr
      resources:
        limits:
          memory: 50M
        reservations:
          memory: 50M

  db:
    image: dockersamples/k8s-wordsmith-db
```

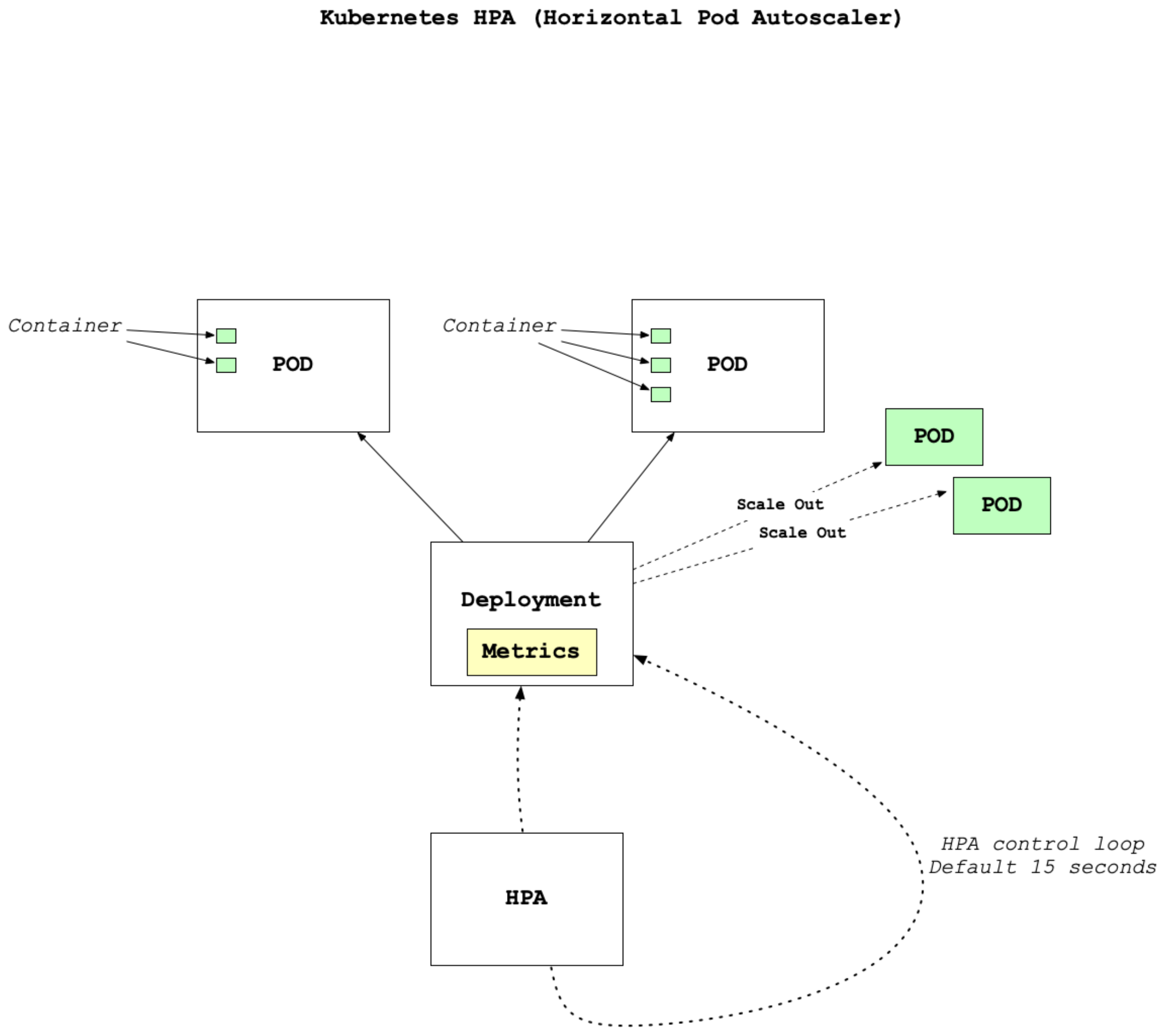
This application deploys with the following command:

```
docker stack deploy --namespace my-app --compose-file /path/to/docker-compose.yml my\
stack
```

Autoscaling Kubernetes

One of the “killer” features of Kubernetes is the ability to set up auto-scaling via the Horizontal Pod Autoscaler[193]. How does this work? The Kubernetes HPA (Horizontal Pod Autoscaler) will automatically scale the number of pods (*remember they can contain multiple containers*) in a replication controller, deployment or replica set. The scaling uses CPU utilization, memory, or custom metrics defined in the Kubernetes Metrics Server.

[193]https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/



kubernetes-hpa

There is a Docker articleKubernetes autoscaling in UCP[194] that is an excellent guide to go more in-depth on this topic and experiment with it yourself.

Learn Kubernetes by watching this demo in the following screencast.

Video Link: https://www.youtube.com/watch?v=ZUyNDfZP1eQ[195]

Kubernetes in the Cloud

The ideal and most common location to use Kubernetes is in a cloud environment. In particular, the cloud solves a major problem in that running a Kubernetes cluster is non-trivial. By leveraging a

[194]https://success.docker.com/article/kubernetes-autoscaling-in-ucp
[195]https://www.youtube.com/watch?v=ZUyNDfZP1eQ

cloud provider, it can make complex Kubernetes tasks very straightforward.

GKE (Google Kubernetes Engine)

The Google Cloud is a great way to explore Kubernetes because they are the open-source framework's originator. Two common ways to use Kubernetes on Google are via GKE and via Google Cloud Run[196]. Google Cloud Run is a very compelling method to deploy prototypes using containers and has much of the same simplicity as Google App Engine.

Learn to deploy Kubernetes on GKE in the following screencast.

Video Link: https://www.youtube.com/watch?v=5pTQJxoK47I[197]

Hybrid and Multi-cloud Kubernetes

Running Kubernetes locally with Docker Desktop and sklearn flask

Learn to run Kubernetes locally via Docker Desktop in the following screencast.

Video Link: https://www.youtube.com/watch?v=LcunlZ_T6Ks[198]

Note how `kubectl` is doing the work as the master node.

```bash
#!/usr/bin/env bash

dockerpath="noahgift/flasksklearn"

# Run in Docker Hub container with kubernetes
kubectl run flaskskearlndemo\
    --generator=run-pod/v1\
    --image=$dockerpath\
    --port=80 --labels app=flaskskearlndemo

# List kubernetes pods
kubectl get pods

# Forward the container port to host
kubectl port-forward flaskskearlndemo 8000:80
```

[196]https://cloud.google.com/run
[197]https://www.youtube.com/watch?v=5pTQJxoK47I
[198]https://www.youtube.com/watch?v=LcunlZ_T6Ks

You can see a walkthrough of how Kubernetes could run a flask sklearn app locally in Docker desktop here[199].

Kubernetes Summary

There are many compelling reasons to use Kubernetes. Let's summarize them:

- Created, Used, and Open Sourced by Google
- High availability architecture
- Auto-scaling is built-in
- Rich Ecosystem
- Service discovery
- Container health management
- Secrets and configuration management

Another advantage is that Kubernetes is cloud-agnostic, and it could be a solution for companies that are willing to take on the additional complexity to protect against "vendor lockin."

[199]https://github.com/noahgift/container-revolution-devops-microservices