**Overview of Kubernetes**

What is [Kubernetes](https://github.com/kubernetes/kubernetes)? 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 was born out of the lessons learned in [scaling containerized apps at Google](https://queue.acm.org/detail.cfm?id=2898444), and is used for automating deployment, scaling and managing such containerized applications.

**What are the Benefits of using Kubernetes?**

Kubernetes is the standard for container orchestration. All major cloud providers support Kubernetes. Amazon through [Amazon EKS](https://aws.amazon.com/eks/), Google through [Google Kubernetes Engine GKE](https://cloud.google.com/kubernetes-engine) and Microsoft through [Azure Kubernetes Service (AKS)](https://azure.microsoft.com/en-us/services/kubernetes-service/).

Kubernetes is also a framework for running distributed systems at ["planet scale"](https://kubernetes.io/). Google uses it to run billions of containers a week.

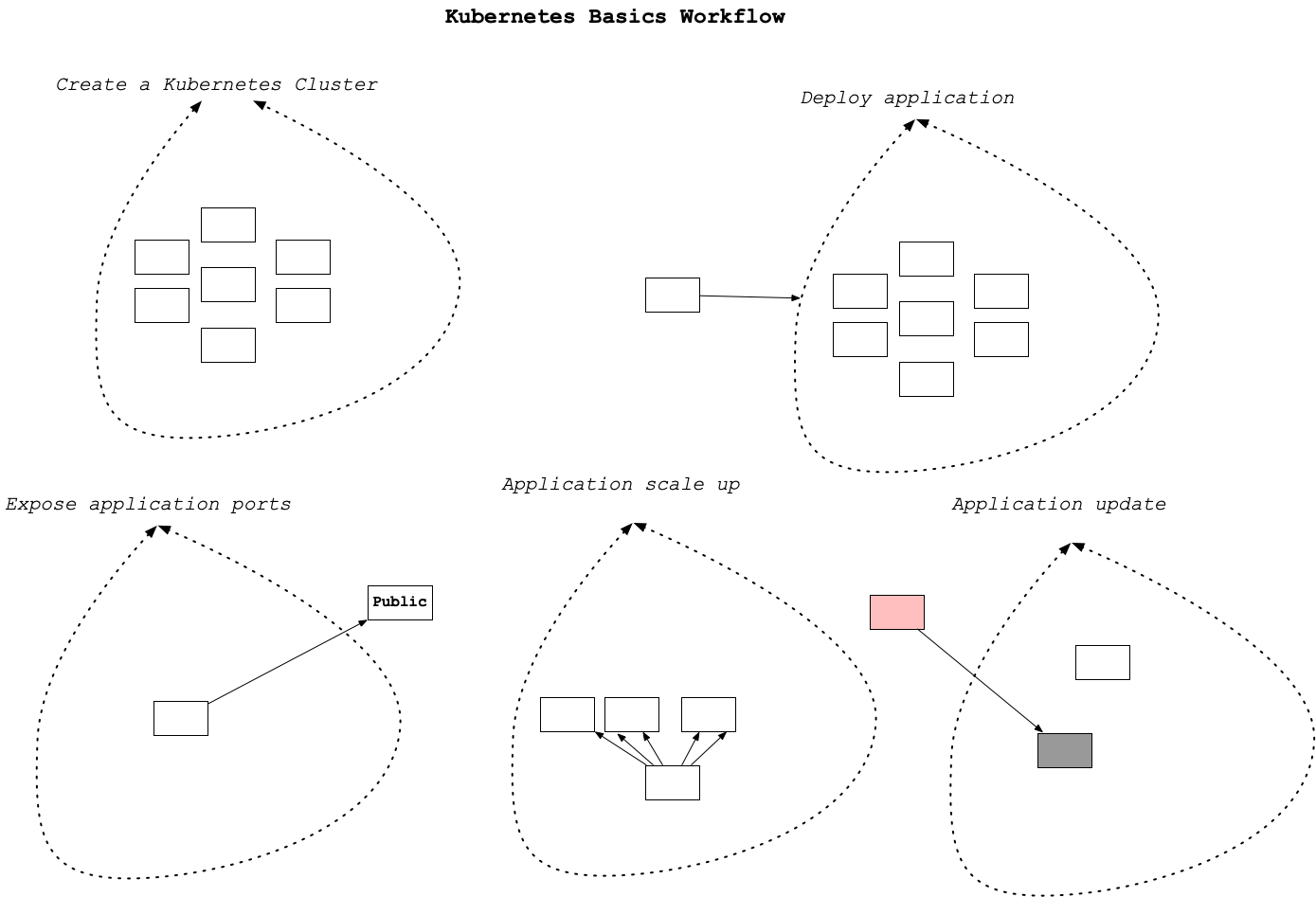
A few of the Capabilities of Kubernetes include:

* 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](https://kubernetes.io/docs/home/).

**What are the Basics of Kubernetes?**

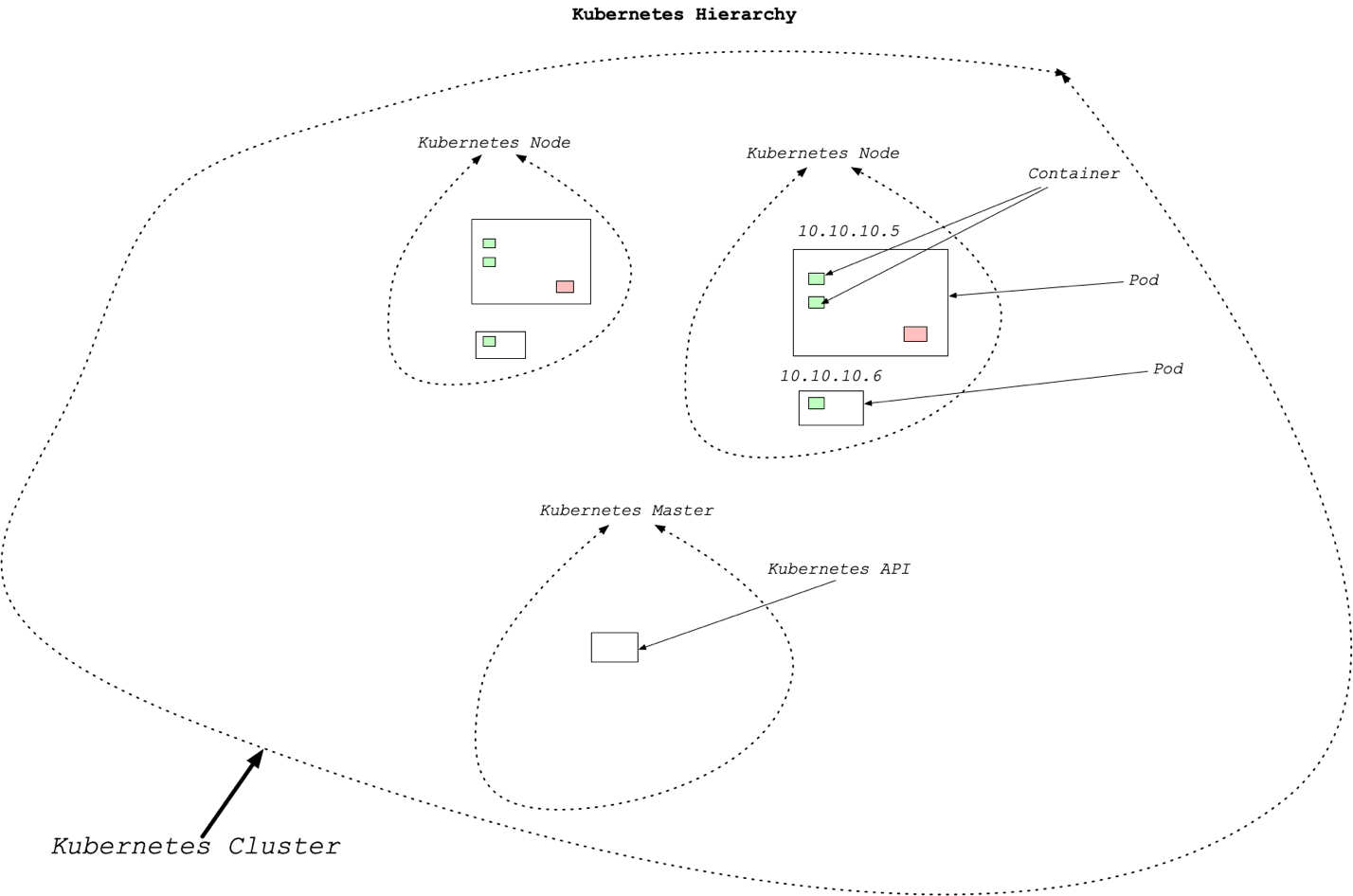
The core operations involved in Kubernetes include creating a Kubernetes Cluster, deploying an application into the cluster, exposing an application ports, scaling an application and updating an application.



**Kubernetes Basics Workflow**

**What is the Kubernetes (Cluster) Architecture?**

The core of Kubernetes is the cluster. Containers run in the cluster. The core components of the cluster include a cluster master and nodes. Inside nodes there is another hierarchy. This is shown in the diagram. A Kubernetes node can contain multiple pods, which in turn can contain multiple containers and/or volumes.



**Kubernetes Hierarchy**

**How do you Set Up a Kubernetes Cluster?**

There are two main methods:

1. Set up a local cluster (preferably with Docker Desktop)
2. Provision a cloud cluster:
   * Amazon through [Amazon EKS](https://aws.amazon.com/eks/)
   * Google through [Google Kubernetes Engine GKE](https://cloud.google.com/kubernetes-engine)
   * Microsoft through [Azure Kubernetes Service (AKS)](https://azure.microsoft.com/en-us/services/kubernetes-service/).

If you are using Docker and have [enabled kubernetes](https://docs.docker.com/docker-for-mac/#kubernetes) then you already have a standalone Kubernetes server running. This 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](https://docs.docker.com/docker-for-mac/kubernetes/) the docker stack deploy --compose-file command.

The yaml example file looks like the following:

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 could be deployed with the following command:

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

**Demo**

You can follow the demo yourself, or read through a [quick primer](https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/) beforehand:

* [Create a cluster](https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-interactive/)

**More on Kubernetes**

To go more in-depth with Kubernetes, we also suggest going through the remaining starter tutorials on the [Kubernetes website](https://kubernetes.io/docs/tutorials/kubernetes-basics/):

* [Deploy An App](https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/)
* [Explore Your App](https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/)
* [Expose Your App Publicly](https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/)
* [Scale Your App](https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/)
* [Update Your App](https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/)

**Additional References**

Here is a list of links to concepts in Kubernetes:

* [kubectl](https://kubernetes.io/docs/tasks/tools/install-kubectl/)
* [Pods](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/)
* [Containers](https://kubernetes.io/docs/concepts/containers/)
* [Clusters](https://kubernetes.io/docs/tutorials/clusters/)

As with coding itself, once you have launched your app with Kubernetes, it's likely you will need to do some debugging to get everything working properly. Here, you'll do some debugging with an example app to build your skills with Kubernetes.

Take a look at this sample bash script that shows how the kubectl commands would be run in the project root directory. You will want to change the dockerpath to your container name and DockerHub username.

**#!/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

**Instructions - Pod Issues**

Let's say you have deployed a Kubernetes app, but have the pod does not seem to be running.

1. First, use kubectl get pods to check the names of your running pods. You may notice the pod with an issue is shown as in a Pending status instead of Running.
2. Using the NAME of the specific pod from step 1, use kubectl describe pod {POD NAME} to get more information about that pod.
3. From the output of the above command, search until you find the Events header. This should give you a Reason and Message related to the failure, such as FailedScheduling. An issue like this could be due to the necessary resources not being available for the pod, such as CPU limits.
4. From what we have seen before, kubectl scale could be used in such a situation to correctly scale up and provide the necessary resources for our Pending pod. On the next page, you'll get to see an automated way to scale up your apps which improves on the manual functionality of kubectl scale.

**Instructions - Node Issues**

In this case, consider a Kubernetes app where the pod is working, but behaving strangely. Alternatively, you may have noticed an issue where no pod will schedule onto a particular node. In this case, there is likely an issue with the specific node that needs to be debugged. While the overall process is fairly similar to debugging issues, the syntax of commands is slightly different, so let's walk through these.

1. First, use kubectl get nodes to check the names of the available nodes. You may notice the node with an issue is shown as in a NotReady status instead of Ready.
2. Using the NAME of the specific node from step 1, use kubectl describe node {NODE NAME} to get more information about that node.
3. The outputs here can vary quite a bit, but the issue could be caused by a disconnection from the network, some other negative Event, too high of resource usage, etc.

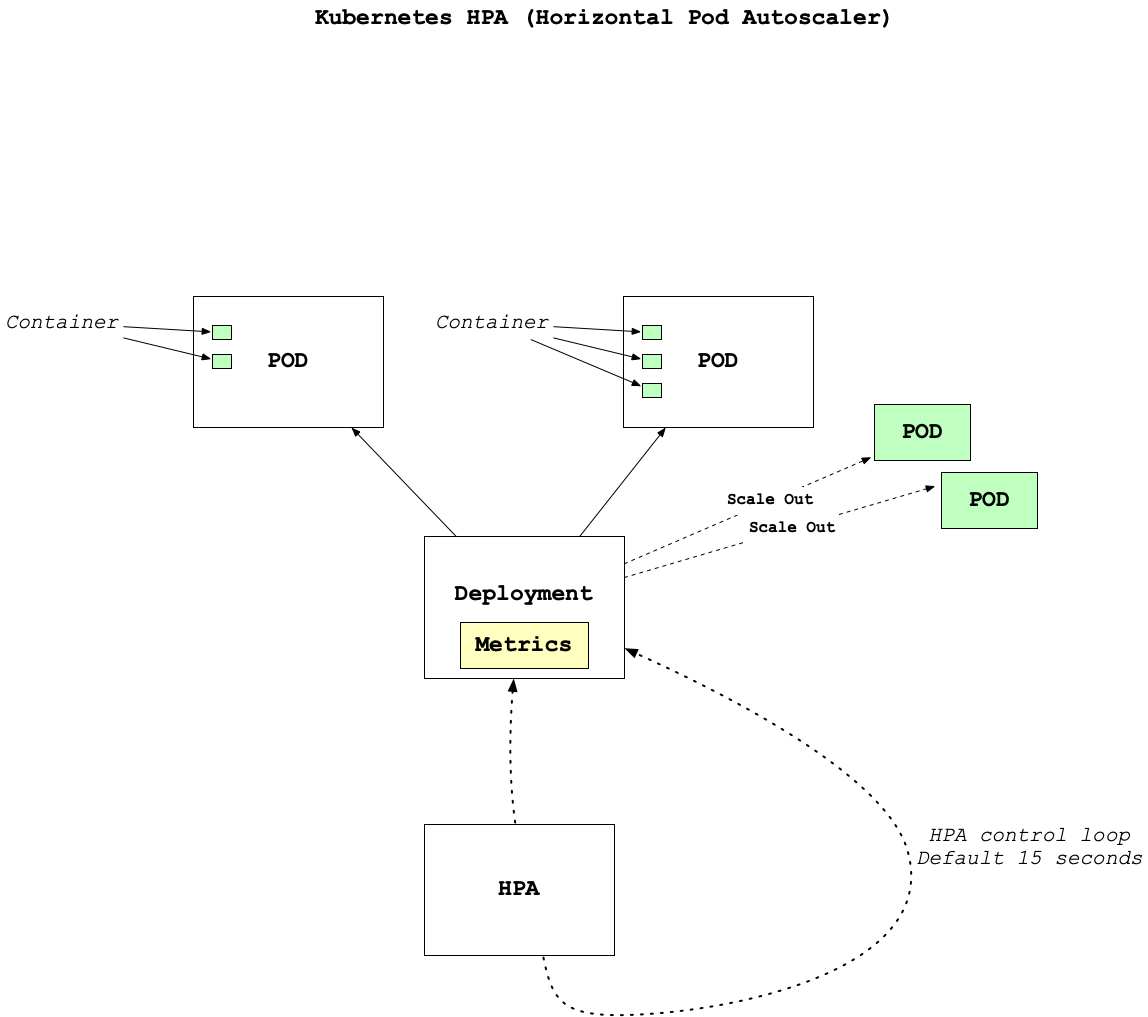
## Autoscaling with CPU or Memory

If you did the [Scaling Demo](https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/) earlier, you already saw one way to scale your apps:

kubectl scale {deployment name} --replicas={desired number of replicas}

The Horizontal Pod Autoscaler does this work for you.

One of the "killer" features of Kubernetes is the ability to set up auto-scaling via the [Horizontal Pod Autoscaler](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/). 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 is based on CPU utilization, memory or custom metrics defined in the Kubernetes Metrics Server.



**Kubernetes HPA (Horizontal Pod Autoscaler)**

There is a Docker article [Kubernetes autoscaling in UCP](https://success.docker.com/article/kubernetes-autoscaling-in-ucp) that is a good guide to go deeper on this topic and experiment with it yourself.

The Horizontal Pod Autoscaler built into Kubernetes is incredibly useful for expanding the number of Pods available based on processing or memory needs. The underlying algorithm itself, somewhat simplified, is as follows:

newNumPods = ceil(currentNumPods \* (currentMetric / desiredMetric))

This means, if by some metric, we are currently at 2.5X our desired metric level, we will scale up our number of pods by 2.5X, rounded up to the nearest one pod.

### Why Kubernetes?

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".

### Key Terms:

#### Kubernetes

Kubernetes is an open-source system for automating the operations of containerized applications. Google created it and opened-sourced it in 2014.

#### Amazon EKS

Amazon EKS is a managed Kubernetes service created by Amazon.

#### Google GKE

Google GKE is a managed Kubernetes service created by Google.

#### Azure Kubernetes Service AKS

Azure Kubernetes Service is a managed Kubernetes service created by Google.

#### YAML

YAML is a human-readable serialization format often used in configuration systems. It is easily portable to JSON format.

#### Kubernetes Pods

A Kubernetes pod is a group of one or more containers.

#### Kubernetes Containers

A Kubernetes container is a Docker image that deploys into a Kubernetes cluster.

#### Kubernetes Clusters

A Kubernetes cluster is a deployment of Kubernetes that contains the entire ecosystem of Kubernetes components, including nodes, pods, the API, and containers.

#### Prometheus

Prometheus is an open-source monitoring system with an efficient time-series database.

#### Logging

Logging is a process of creating messages about the running state of a software application.

#### Autoscaling

Autoscaling is the process of scaling load up or down automatically based on how many resources the nodes are using.