- **ReplicaSets:** It is part of the controller in the master node. It specifies the number of replicas of a pod that should be running at any given time. It ensures that the specified number of pods is maintained in the cluster.
- **Deployments:** It automatically creates `ReplicaSets`. It is also part of the `controller` in the master node. It ensures that the cluster's current state matches the desired state.
- **Namespaces:** It partitions the cluster into sub-clusters to organize users into groups.
- **Service:** It is a logical group of pods with a policy to access them.
    - *ServiceTypes:* It specifies the type of service, for example, ClusterIP, NodePort, LoadBalancer, and ExternalName. As an example, LoadBalancer exposes the service externally using a cloud provider's load balancer.

Other important tags in writing a Kubernetes deployment file

- **spec:** It describes the desired state of the cluster
- **metadata:** It contains information of the object
- **labels:** It is used to specify attributes of objects as key-value pairs
- **selector:** It is used to select a subset of objects based on their label values

The deployment file is specified as a yaml file.

Deploying Kubernetes on Google Kubernetes Engine

Google Kubernetes engine (GKE) provides a managed environment for deploying application containers. To create and deploy resources on GCP from the local shell, the Google command-line SDK gcloud will have to be installed and configured. If this is not the case on your machine, follow the instructions at `https://cloud.google.com/sdk/gcloud/`. Otherwise, a simpler option is to use the Google Cloud Shell which already has gcloud and kubectl (the Kubernetes command-line interface) installed.

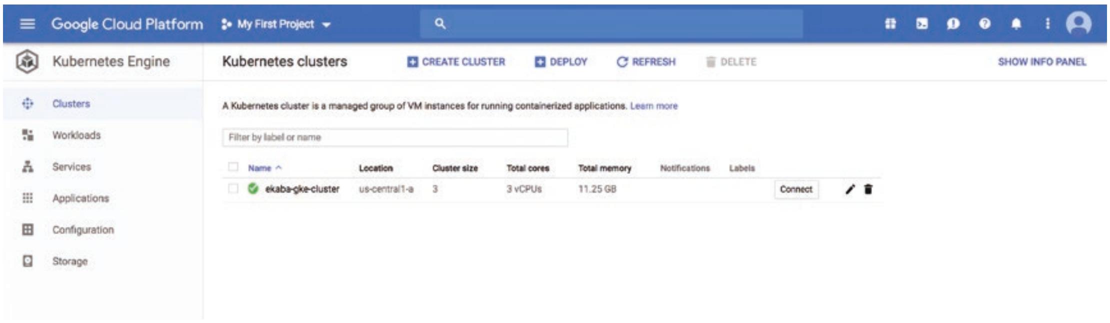
Creating a GKE Cluster

Run the following command to create a cluster of containers on GKE. Assign the cluster name.

```
# create a GKE cluster
gcloud container clusters create my-gke-cluster-name
```

A Kubernetes cluster is created on GCP with three nodes (as default). The GKE dashboard on GCP is shown in Figure 45-7.

```
Creating cluster ekaba-gke-cluster in us-central1-a... Cluster is being
deployed...done.
Created [https://container.googleapis.com/v1/projects/oceanic-sky-230504/
zones/us-central1-a/clusters/ekaba-gke-cluster].
To inspect the contents of your cluster, go to: https://console.
cloud.google.com/kubernetes/workload_/gcloud/us-central1-a/ekaba-gke-
cluster?project=oceanic-sky-230504
kubeconfig entry generated for ekaba-gke-cluster.
NAME               LOCATION       MASTER_VERSION  MASTER_IP     MACHINE_
TYPE   NODE_VERSION  NUM_NODES  STATUS
ekaba-gke-cluster  us-central1-a  1.11.7-gke.4    35.226.72.40  n1-
standard-1  1.11.7-gke.4  3          RUNNING
```



A Kubernetes cluster is a managed group of VM instances for running containerized applications. Learn more

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels
✓ ekaba-gke-cluster	us-central1-a	3	3 vCPUs	11.25 GB		

Figure 45-7. *Google Kubernetes Engine dashboard*

To learn more about creating clusters with Google Kubernetes Engine, visit https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-cluster.

Run the following command to display the nodes of the provisioned cluster on GKE.

```
# get the nodes of the kubernetes cluster on GKE
kubectl get nodes
```

```
NAME                                                STATUS   ROLES    AGE
VERSION
gke-ekaba-gke-cluster-default-pool-e28c64e0-8fk1   Ready    <none>   45m
v1.11.7-gke.4
gke-ekaba-gke-cluster-default-pool-e28c64e0-fmck   Ready    <none>   45m
v1.11.7-gke.4
gke-ekaba-gke-cluster-default-pool-e28c64e0-zzz1   Ready    <none>   45m
v1.11.7-gke.4
```

Delete the Kubernetes Cluster on GKE

Run the following command to delete a cluster on GKE.

```
# delete the kubernetes cluster
gcloud container clusters delete my-gke-cluster-name
```

Note Always remember to clean up cloud resources when they are no longer needed.

This chapter introduced the concepts of a microservice architecture and provided an overview of working with Docker containers for building applications in isolated environments/sandboxes. In the event that many of such containers are deployed in production, this chapter introduces Kubernetes as a container orchestrator for managing the concerns of deploying, scaling, and monitoring containers.

The next chapter will discuss on Kubeflow and Kubeflow Pipelines for deploying machine learning components into production on Kubernetes.

CHAPTER 46

Kubeflow and Kubeflow Pipelines

Machine learning is often and rightly viewed as the use of mathematical algorithms to teach the computer to learn tasks that are computationally infeasible to program as a set of specified instructions. However, it turns out that these algorithms constitute only a small fraction of the overall learning pipeline from an engineering perspective. Building high-performant and dynamic learning models includes a number of other critical components. These components actually dominate the space of concerns for delivering an end-to-end machine learning product.

A typical machine learning production pipeline looks like the illustration in Figure 46-1.