**Create a Cluster**

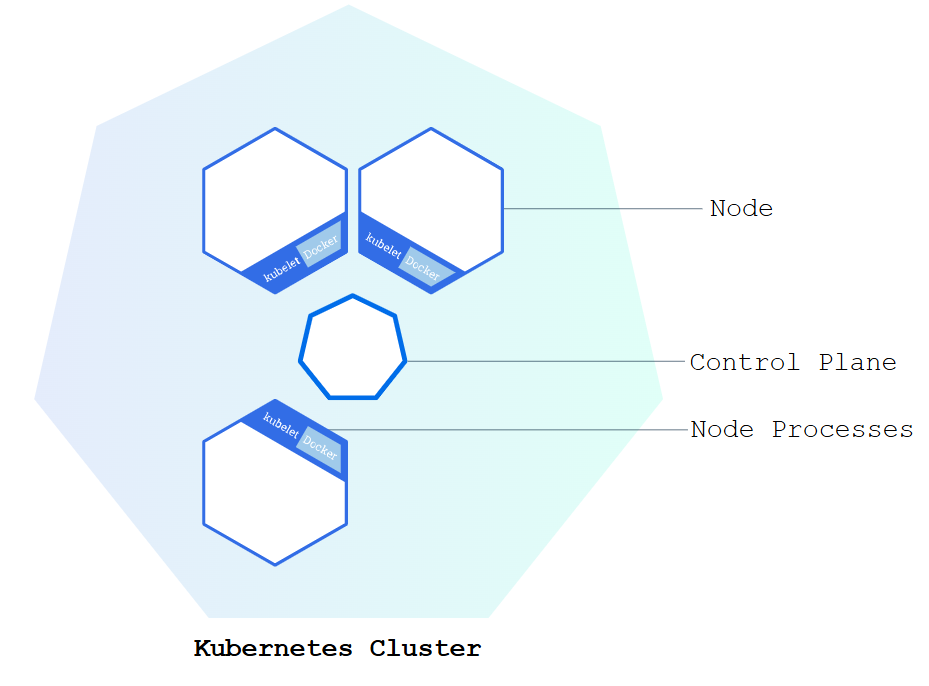
*Kubernetes is a production-grade, open-source platform that orchestrates the placement (scheduling) and execution of application containers within and across computer clusters.*

**Kubernetes coordinates a highly available cluster of computers that are connected to work as a single unit.** The abstractions in Kubernetes allow you to deploy containerized applications to a cluster without tying them specifically to individual machines. To make use of this new model of deployment, applications need to be packaged in a way that decouples them from individual hosts: they need to be containerized. Containerized applications are more flexible and available than in past deployment models, where applications were installed directly onto specific machines as packages deeply integrated into the host. **Kubernetes automates the distribution and scheduling of application containers across a cluster in a more efficient way.** Kubernetes is an open-source platform and is production-ready.

A Kubernetes cluster consists of two types of resources:

* The **Control Plane** coordinates the cluster
* **Nodes** are the workers that run applications

Cluster Diagram



**The Control Plane is responsible for managing the cluster.** The Control Plane coordinates all activities in your cluster, such as scheduling applications, maintaining applications' desired state, scaling applications, and rolling out new updates.

*Control Planes manage the cluster and the nodes that are used to host the running applications.*

**A node is a VM or a physical computer that serves as a worker machine in a Kubernetes cluster.** Each node has a Kubelet, which is an agent for managing the node and communicating with the Kubernetes control plane. The node should also have tools for handling container operations, such as [containerd](https://containerd.io/docs/" \o "" \t "_blank) or [CRI-O](https://cri-o.io/#what-is-cri-o). A Kubernetes cluster that handles production traffic should have a minimum of three nodes because if one node goes down, both an [etcd](https://kubernetes.io/docs/concepts/architecture/" \l "etcd) member and a control plane instance are lost, and redundancy is compromised. You can mitigate this risk by adding more control plane nodes.

When you deploy applications on Kubernetes, you tell the control plane to start the application containers. The control plane schedules the containers to run on the cluster's nodes. **Node-level components, such as the kubelet, communicate with the control plane using the**[**Kubernetes API**](https://kubernetes.io/docs/concepts/overview/kubernetes-api/), which the control plane exposes. End users can also use the Kubernetes API directly to interact with the cluster.

A Kubernetes cluster can be deployed on either physical or virtual machines. To get started with Kubernetes development, you can use Minikube. Minikube is a lightweight Kubernetes implementation that creates a VM on your local machine and deploys a simple cluster containing only one node. Minikube is available for Linux, macOS, and Windows systems. The Minikube CLI provides basic bootstrapping operations for working with your cluster, including start, stop, status, and delete.

**Deploy an App**

**Using kubectl to Create a Deployment**

*A Deployment is responsible for creating and updating instances of your application.*

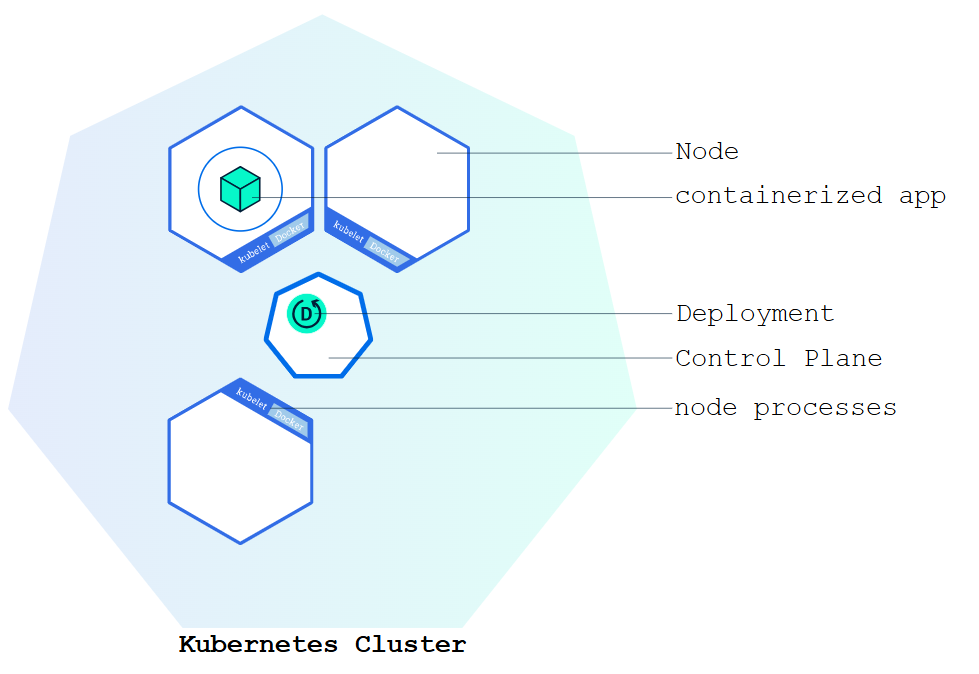
Once you have a [running Kubernetes cluster](https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/), you can deploy your containerized applications on top of it. To do so, you create a Kubernetes **Deployment**. The Deployment instructs Kubernetes how to create and update instances of your application. Once you've created a Deployment, the Kubernetes control plane schedules the application instances included in that Deployment to run on individual Nodes in the cluster.

Once the application instances are created, a Kubernetes Deployment controller continuously monitors those instances. If the Node hosting an instance goes down or is deleted, the Deployment controller replaces the instance with an instance on another Node in the cluster. **This provides a self-healing mechanism to address machine failure or maintenance.**

In a pre-orchestration world, installation scripts would often be used to start applications, but they did not allow recovery from machine failure. By both creating your application instances and keeping them running across Nodes, Kubernetes Deployments provide a fundamentally different approach to application management.

Deploying your first app on Kubernetes

*Applications need to be packaged into one of the supported container formats in order to be deployed on Kubernetes.*



You can create and manage a Deployment by using the Kubernetes command line interface, [kubectl](https://kubernetes.io/docs/reference/kubectl/). kubectl uses the Kubernetes API to interact with the cluster. In this module, you'll learn the most common kubectl commands needed to create Deployments that run your applications on a Kubernetes cluster.

When you create a Deployment, you'll need to specify the container image for your application and the number of replicas that you want to run. You can change that information later by updating your Deployment; [Module 5](https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/) and [Module 6](https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/) of the bootcamp discuss how you can scale and update your Deployments.

For your first Deployment, you'll use a hello-node application packaged in a Docker container that uses NGINX to echo back all the requests. (If you didn't already try creating a hello-node application and deploying it using a container, you can do that first by following the instructions from the [Hello Minikube tutorial](https://kubernetes.io/docs/tutorials/hello-minikube/).

You will need to have installed kubectl as well. If you need to install it, visit [install tools](https://kubernetes.io/docs/tasks/tools/#kubectl) install tools.

Now that you know what Deployments are, let's deploy our first app!

**kubectl basics**

The common format of a kubectl command is: kubectl action resource.

This performs the specified *action* (like create, describe or delete) on the specified *resource* (like node or deployment. You can use --help after the subcommand to get additional info about possible parameters (for example: kubectl get nodes --help).

Check that kubectl is configured to talk to your cluster, by running the kubectl version command.

Check that kubectl is installed and that you can see both the client and the server versions.

To view the nodes in the cluster, run the kubectl get nodes command.

You see the available nodes. Later, Kubernetes will choose where to deploy our application based on Node available resources.

Deploy an app

Let’s deploy our first app on Kubernetes with the kubectl create deployment command. We need to provide the deployment name and app image location (include the full repository url for images hosted outside Docker Hub).