**Objective**: Set up a Kubernetes cluster with multiple worker nodes and deploy a sample application across the cluster.

**Prerequisites**:

1. Docker installed on each host.
2. Kubernetes installed on a control plane node.
3. Access to Kubernetes command-line tool (kubectl).

**Steps**:

1. **Provision Docker Hosts**:
   * Set up multiple Docker hosts (e.g., virtual machines) that will serve as worker nodes in your Kubernetes cluster.
   * Ensure that each host meets the minimum system requirements and has Docker installed and configured.
2. **Set Up Control Plane Node**:
   * Choose one of the Docker hosts to serve as the control plane node (master node) for your Kubernetes cluster.
   * Install Kubernetes components (kubelet, kube-proxy, container runtime) on the control plane node.
   * Initialize the Kubernetes cluster using kubeadm or another installation method.
3. **Join Worker Nodes**:
   * On each worker node, install Kubernetes components (kubelet, kube-proxy, container runtime).
   * Use kubeadm or kubectl to join the worker nodes to the Kubernetes cluster, connecting them to the control plane node.
4. **Configure Networking**:
   * Set up networking between nodes in the Kubernetes cluster to enable communication between pods and services.
   * Choose a networking solution compatible with Kubernetes (e.g., Calico, Flannel) and configure it to provide network connectivity and isolation.
5. **Deploy Sample Application**:
   * Package a sample application as a Docker container and push it to a container registry (e.g., Docker Hub).
   * Create Kubernetes manifests (YAML or JSON files) to define the deployment, service, and other resources required for the sample application.
   * Specify deployment configurations, such as the number of replicas, container image, resource requirements, and ports.
   * Apply the Kubernetes manifests using kubectl to deploy the sample application to the Kubernetes cluster.
6. **Scale Application**:
   * Experiment with scaling the sample application by adjusting the number of pod replicas in the deployment manifest.
   * Use kubectl commands to scale up or down the number of pod replicas and observe how Kubernetes automatically schedules and distributes pods across the cluster.
7. **Monitor Cluster**:
   * Monitor the health and performance of the Kubernetes cluster using built-in monitoring and logging tools or third-party solutions.
   * Use kubectl commands to retrieve cluster status, view pod logs, and inspect resource usage.
8. **Update Application**:
   * Perform a rolling update of the sample application to deploy a new version with updated features or bug fixes.
   * Update the container image tag in the deployment manifest to point to the new version.
   * Apply the updated manifest using kubectl to trigger a rolling update, allowing Kubernetes to gracefully replace existing pods with the new version.
9. **Clean Up**:
   * Once you've completed the exercise, clean up resources by deleting the sample application deployment and tearing down the Kubernetes cluster.
   * Use kubectl commands to delete Kubernetes resources (e.g., deployments, services, pods) and remove worker nodes from the cluster.

By completing this exercise, you'll gain hands-on experience in setting up a Kubernetes cluster with multiple worker nodes, deploying applications, scaling workloads, and managing resources using Kubernetes orchestration.