Let's implement the changes you requested: modifying the replica count, changing the service type to ClusterIP, and then deploying a new NGINX deployment and service, both using ClusterIP.

**Prerequisites:**

* You should have a running Kubernetes cluster (Minikube is recommended for local testing). If you've followed the previous examples, you should have a cluster and potentially an existing NGINX deployment and NodePort service.

**Steps:**

1. **Modify the Existing Deployment (Scaling to 5 Replicas):**

If you have the nginx-deployment.yaml file from the previous example, edit it to change the replicas field:

YAML

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment # Keep the same name to update the existing deployment

spec:

replicas: 5 # Changed to 5 replicas

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Apply the 1 changes:

[1. www.howtogeek.com](https://www.howtogeek.com/devops/how-to-scale-docker-containers-across-servers-using-kubernetes/" \t "_blank)

[www.howtogeek.com](https://www.howtogeek.com/devops/how-to-scale-docker-containers-across-servers-using-kubernetes/" \t "_blank)

Bash

kubectl apply -f nginx-deployment.yaml

Verify the change:

Bash

kubectl get deployments nginx-deployment

kubectl get pods -l app=nginx # List pods with the label app=nginx

You should see 5 running pods.

1. **Change the Service Type to ClusterIP:**

Edit the nginx-service.yaml file:

YAML

apiVersion: v1

kind: Service

metadata:

name: nginx-service # Keep the same name to update the existing service

spec:

type: ClusterIP # Changed to ClusterIP

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

Apply the changes:

Bash

kubectl apply -f nginx-service.yaml

Verify the change:

Bash

kubectl get services nginx-service

You'll see that the TYPE is now ClusterIP. Note the CLUSTER-IP address assigned to the service. This IP is only accessible from within the Kubernetes cluster.

1. **Deploy a *New* NGINX Deployment (3 Replicas, Different Name):**

Create a new file named nginx-deployment-new.yaml:

YAML

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment-new # Different name

spec:

replicas: 3

selector:

matchLabels:

app: nginx-new # Different label

template:

metadata:

labels:

app: nginx-new # Different label

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Apply the new deployment:

Bash

kubectl apply -f nginx-deployment-new.yaml

1. **Create a *New* NGINX Service of Type ClusterIP:**

Create a new file named nginx-service-new.yaml:

YAML

apiVersion: v1

kind: Service

metadata:

name: nginx-service-new # Different name

spec:

type: ClusterIP

selector:

matchLabels:

app: nginx-new # Matches the new deployment's label

ports:

- protocol: TCP

port: 80

targetPort: 80

Apply the new service:

Bash

kubectl apply -f nginx-service-new.yaml

Verify the new deployment and service:

Bash

kubectl get deployments nginx-deployment-new

kubectl get pods -l app=nginx-new

kubectl get services nginx-service-new

**Accessing ClusterIP Services (from within the cluster):**

ClusterIP services are *not* directly accessible from outside the cluster. To access them, you need to be inside the cluster. Here are common ways:

* **Using kubectl exec (for testing/debugging):**

Bash

# Get the name of a pod in the deployment:

POD\_NAME=$(kubectl get pods -l app=nginx-new -o jsonpath='{.items[0].metadata.name}')

# Execute a command inside the pod to access the service:

kubectl exec -it $POD\_NAME -- curl http://nginx-service-new # Use the service name as the hostname

* **Using a Pod with curl:** Create a temporary pod with curl installed and use it to access the service.
* **Using kubectl port-forward (for local development):** This creates a local port forwarding to the service:

Bash

kubectl port-forward service/nginx-service-new 8080:80 # Forward local port 8080 to service port 80

Then you can access the service at http://localhost:8080 in your browser.

**Key Changes and Explanations:**

* **Scaling Replicas:** Demonstrates how to scale an existing deployment.
* **Changing Service Type:** Shows how to modify a service to use ClusterIP.
* **New Deployment and Service:** Creates a completely new deployment and service to illustrate the process from scratch.
* **Accessing ClusterIP:** Explains how to access ClusterIP services from within the cluster and using port forwarding for local development.

This detailed explanation should help you understand how to manage deployments and services in Kubernetes, including scaling and different service types. Remember that ClusterIP is intended for internal cluster communication, and you'll need other methods (like Ingress or LoadBalancer) for external access in production.