Practice Lab: Introduction to Kubernetes Objects
练习实验室：Kubernetes 对象简介

Skills Network

Estimated time needed: 45 minutes
预计所需时间：45 分钟

This practice lab is designed to provide hands-on experience with Kubernetes, focusing on creating services, using various kubectl commands, and deploying StatefulSets and DaemonSets.
该实践实验室旨在提供 Kubernetes 的实际操作经验，重点在于创建服务、使用各种 kubectl 命令以及部署 StatefulSets 和 DaemonSets。

Objectives 目标

After completing this lab, you will be able to:
完成此实验室后，您将能够：

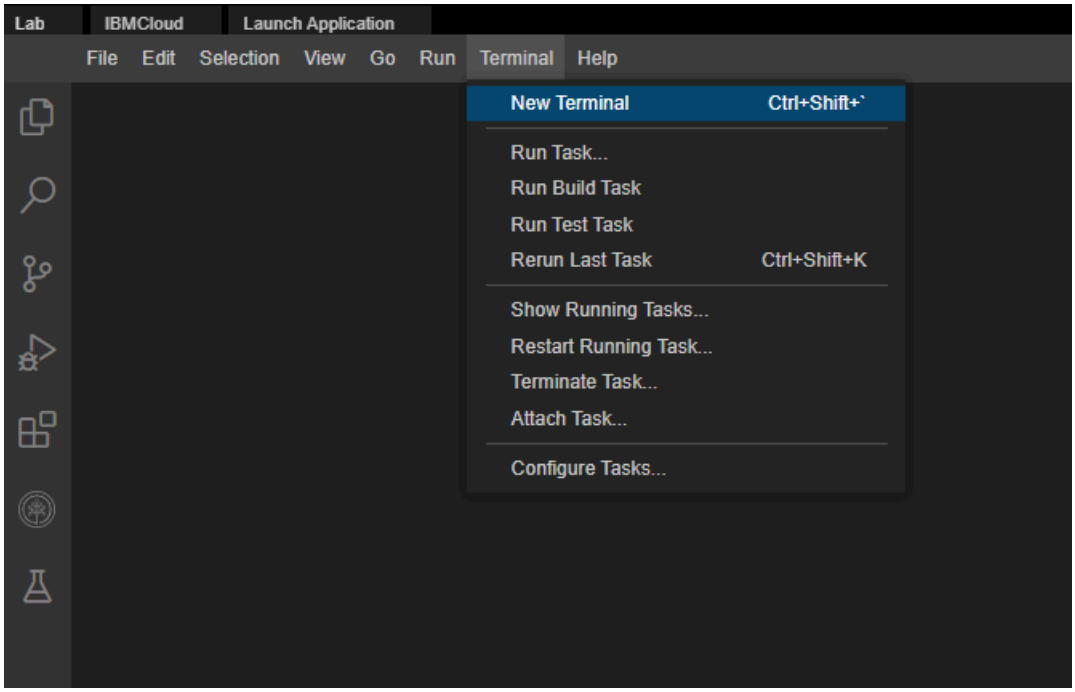
- Create a Kubernetes Service
  创建 Kubernetes 服务
- Use various `kubectl` commands 使用各种 `kubectl` 命令
- Deploy a StatefulSet for managing stateful applications
  为管理有状态应用部署 StatefulSet
- Implement a DaemonSet for running a single pod on all nodes
  为在所有节点上运行单个 Pod 实现 DaemonSet

> **Note: Kindly complete the lab in a single session without any break because the lab may go on offline mode and may cause errors. If you face any issues or errors during the lab process, please log out of the lab environment. Then clear your system cache and cookies and try to complete the lab.**
> **注意：请在一个会话中完成实验室，不要中断，因为实验室可能会进入离线模式并导致错误。如果在实验室过程中遇到任何问题或错误，请退出实验室环境。然后清除系统缓存和 Cookie，并尝试完成实验室。**

Setup Environment 设置环境

Open a terminal window using the menu: `Terminal > New Terminal`.
使用菜单打开终端窗口： `Terminal > New Terminal` 。

> Note: If the terminal is already opened, please skip this step.
> 注意：如果终端已经打开，请跳过此步骤。

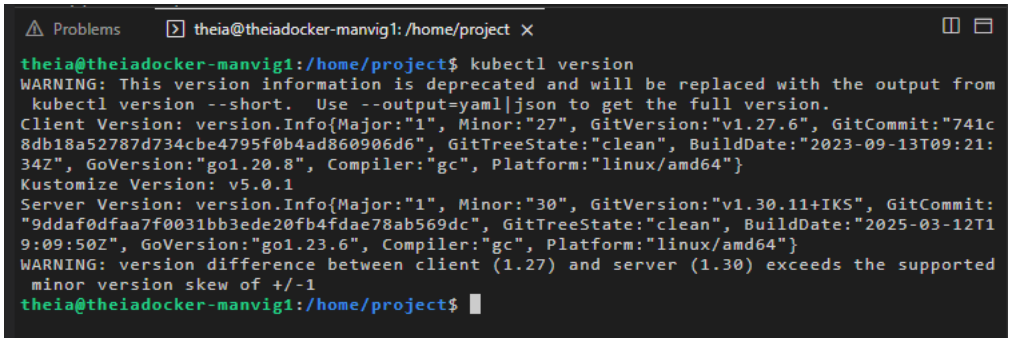


Step 1: Verify kubectl Version
步骤 1：验证 kubectl 版本

Before proceeding, ensure that you have kubectl installed and properly configured. To check the version of kubectl, run the following command:
在进行下一步之前，请确保您已安装 kubectl 并正确配置。要检查 kubectl 的版本，请运行以下命令：

```
kubectl version
```

You should see the following output, although the versions may be different:
你应该看到以下输出，尽管版本可能不同：



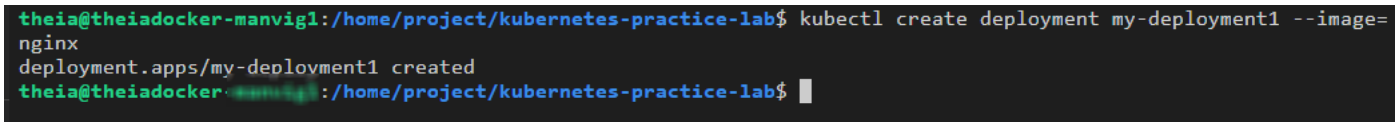
Task 1: Create a Kubernetes Service using nginx image
任务 1：使用 nginx 镜像创建 Kubernetes 服务

A popular open-source web server, **nginx** is known for its high performance, stability, and low resource usage. It can also function as a reverse proxy, load balancer, and HTTP cache.
一个流行的开源 Web 服务器，nginx 以其高性能、稳定性和低资源消耗而闻名。它还可以作为反向代理、负载均衡器和 HTTP 缓存使用。

Creating a Kubernetes Service using an nginx image involves setting up a networking layer that allows other components within the Kubernetes cluster or external users to access the nginx application running in pods. To run nginx as a service in Kubernetes, you can follow these steps:
使用 nginx 镜像创建 Kubernetes 服务涉及设置一个网络层，允许 Kubernetes 集群内的其他组件或外部用户访问在 Pod 中运行的 nginx 应用程序。要在 Kubernetes 中运行 nginx 作为服务，您可以按照以下步骤操作：

1. **Create a Deployment named `my-deployment1` using the nginx image**
   创建一个名为 `my-deployment1` 的 Deployment，使用 nginx 镜像

   ```
   kubectl create deployment my-deployment1 --image=nginx
   ```



`kubectl`: The command-line tool for interacting with the Kubernetes API.
`kubectl`： 用于与 Kubernetes API 交互的命令行工具。

`create deployment`: Tells Kubernetes that you want to create a new Deployment. A Deployment is a Kubernetes object that manages a set of replicated Pods, ensuring that the specified number of replicas are running and updated.
`create deployment`：告知 Kubernetes 您希望创建一个新的 Deployment。Deployment 是 Kubernetes 中的一个对象，用于管理一组复制的 Pods，确保指定的副本数量正在运行并得到更新。

`my-deployment1`: It is the name of the Deployment being created. In this case, the Deployment is named my-deployment1.
`my-deployment1`：这是正在创建的 Deployment 的名称。在这种情况下，Deployment 的名称为 my-deployment1。

`--image=nginx`: It specifies the container image used for the Pods managed by this Deployment. The nginx image is a popular web server and reverse proxy server.

`--image=nginx`：它指定了由该 Deployment 管理的 Pods 使用的容器镜像。nginx 镜像是一个流行的 Web 服务器和反向代理服务器。

It creates a Deployment named my-deployment1 that uses the nginx image. Deployments manage the rollout and scaling of applications.
它创建了一个名为 my-deployment1 的 Deployment，该 Deployment 使用 nginx 镜像。Deployments 管理应用程序的发布和扩展。

2. **Expose the deployment as a service**
   将部署作为服务公开

```
kubectl expose deployment my-deployment1 --port=80 --type=NodePort --name=my-service1
```

```
theia@theiadocker-[redacted]:/home/project/kubernetes-practice-lab$ kubectl expose deployment my-deployment1 --port=80 --
t --name=my-service
service/my-service exposed
theia@theiadocker-[redacted]:/home/project/kubernetes-practice-lab$
```

It exposes the my-deployment1 Deployment as a Service named my-service1, making it accessible on port 80 through a NodePort. NodePort services allow external traffic to access the service.
它将名为 my-deployment1 的 Deployment 作为服务 my-service1 公开，并通过 NodePort 在端口 80 上使其可访问。NodePort 服务允许外部流量访问该服务。

3. **Lists all services in the default namespace. Services provide a stable IP address and DNS name for accessing a set of pods.**
   列出默认命名空间中的所有服务。服务为访问一组 Pod 提供稳定的 IP 地址和 DNS 名称。

```
kubectl get services
```

```
theia@theiadocker-[redacted]:/home/project/kubernetes-practice-lab$ kubectl get services
NAME         TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
my-service   NodePort   172.21.229.19   <none>        80:31449/TCP   43s
theia@theiadocker-[redacted]:/home/project/kubernetes-practice-lab$
```

This command lists all the services in the default namespace, including nginx-service, and provides details such as the ClusterIP, NodePort, and target port.
此命令列出默认命名空间中的所有服务，包括 nginx-service，并提供 ClusterIP、NodePort 和目标端口等详细信息。

By following these steps, you create a Kubernetes Service named `nginx`, which routes traffic to the nginx pods running in your cluster, making them accessible internally and externally via the assigned NodePort.
按照这些步骤，您将创建一个名为 `nginx` 的 Kubernetes Service，该服务将流量路由到集群中运行的 nginx pod，并通过分配的 NodePort 实现内部和外部访问。

Task 2: Manage Kubernetes Pods and Services
任务 2：管理 Kubernetes Pod 和服务

1. Get the list of pods
   获取 Pod 列表

```
kubectl get pods
```

```
theia@theiadocker-[redacted]:/home/project$  kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
my-deployment1-76974f7b8d-6xjph     1/1     Running   0          49s
theia@theiadocker-[redacted]:/home/project$
```

This command displays all pods, including those created by the my-deployment1 Deployment.
该命令显示所有 Pod，包括由 my-deployment1 Deployment 创建的 Pod。

2. Show labels 显示标签

Replace <pod-name> with the actual pod Name:
将 <pod-name> 替换为实际的 Pod 名称：

```
kubectl get pod <pod-name> --show-labels
```

```
theia@theiadocker-[redacted]:/home/project$  kubectl get pod  my-deployment1-76974f7b8d-6xjph --show-labels
NAME                                READY   STATUS    RESTARTS   AGE    LABELS
my-deployment1-76974f7b8d-6xjph     1/1     Running   0          2m9s   app=my-deployment1,pod-template-hash=76
974f7b8d
theia@theiadocker-[redacted]:/home/project$
```

This command will list the labels associated with the specified pod, helping you identify its attributes and categorization within your Kubernetes cluster.
该命令将列出与指定 Pod 关联的标签，帮助您识别其在 Kubernetes 集群中的属性和分类。

3. Label the pod 为 Pod 标签

Replace <pod-name> with the actual pod Name:
将 <pod-name> 替换为实际的 Pod 名称：

```
kubectl label pods <pod-name> environment=deployment
```

```
theia@theiadocker-[redacted]:/home/project$ kubectl label pods my-deployment1-76974f7b8d-6xjph environment=deployment
pod/my-deployment1-76974f7b8d-6xjph labeled
theia@theiadocker-[redacted]:/home/project$
```

The command is used in Kubernetes to label a specific pod with the key-value pair `environment=deployment`. This label helps categorize and manage pods based on their deployment environment, making it easier to organize and select Kubernetes objects within the cluster.
该命令在 Kubernetes 中用于使用键值对 `environment=deployment` 为特定 Pod 标签。此标签有助于根据其部署环境对 Pod 进行分类和管理，从而更方便地在集群中组织和选择 Kubernetes 对象。

4. Show labels 显示标签

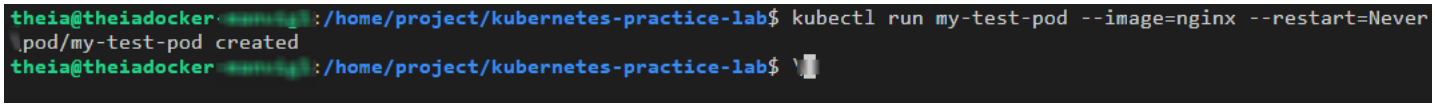
Replace <pod-name> with the actual pod Name:
将 <pod-name> 替换为实际的 Pod 名称：

```
kubectl get pod <pod-name> --show-labels
```

```
theia@theiadocker-[redacted]:/home/project$  kubectl get pod  my-deployment1-76974f7b8d-6xjph --show-labels
NAME                                READY   STATUS    RESTARTS   AGE     LABELS
my-deployment1-76974f7b8d-6xjph     1/1     Running   0          6m49s   app=my-deployment1,environment=deployment,pod-template-hash=769
theia@theiadocker-[redacted]:/home/project$
```

5. Run a test pod using the `nginx` image
使用 `nginx` 镜像运行一个测试 Pod

```
kubectl run my-test-pod --image=nginx --restart=Never
```

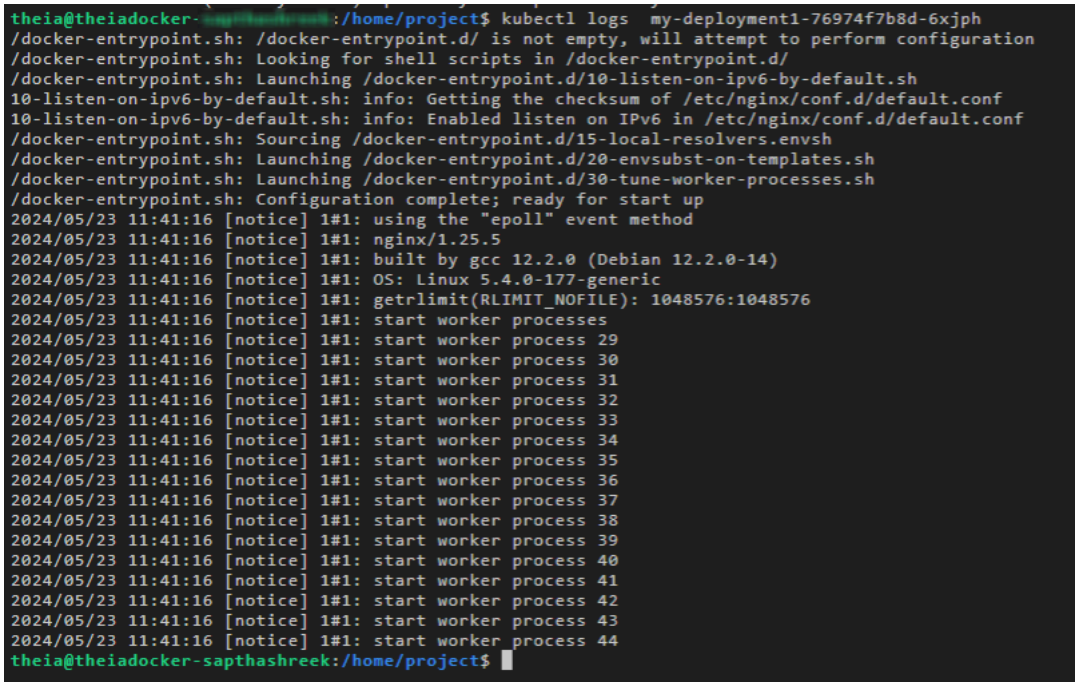


This command tells Kubernetes to create a pod named "my-test-pod" using the nginx image, and the pod will not restart automatically if it stops for any reason as we are using `--restart=Never`.
该命令指示 Kubernetes 创建一个名为"my-test-pod"的 Pod，使用 nginx 镜像，并且如果 Pod 因任何原因停止，由于我们使用的是 `--restart=Never`，它不会自动重启。

6. Show logs 显示日志

```
kubectl logs <pod-name>
```

Replace `<pod-name>` with the actual name of the pod.
将 `<pod-name>` 替换为实际的 Pod 名称。



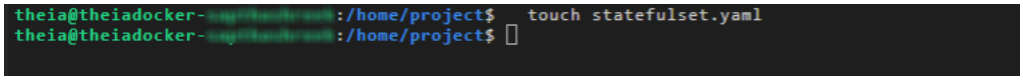
This command retrieves and displays the logs generated by the specified pod, allowing you to troubleshoot issues, monitor activity, and gather information about the pod's behavior.
该命令检索并显示指定 Pod 生成的日志，允许您解决问题、监控活动以及收集有关 Pod 行为的信息。

Task 3: Deploying a StatefulSet
任务 3：部署 StatefulSet

A StatefulSet manages the deployment and scaling of a set of pods, and maintains a sticky identity for each of their Pods, ensuring that each Pod has a persistent identity and storage.
一个 StatefulSet 管理一组 Pod 的部署和扩展，并为每个 Pod 维护一个粘性身份，确保每个 Pod 具有持久的身份和存储。

1. Create and open a file named `statefulset.yaml` in edit mode.
   创建并进入名为 `statefulset.yaml` 的文件编辑模式。

```
touch statefulset.yaml
```



2. Open `statefulset.yaml`, and add the following code, and save the file:
   打开 `statefulset.yaml` ，添加以下代码并保存文件：

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: my-statefulset
spec:
  serviceName: "nginx"
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
          name: web
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

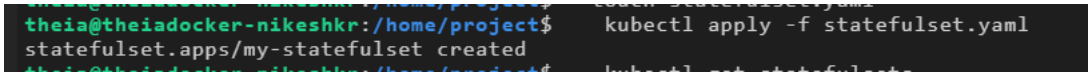
Explanation: 解释：

- `apiVersion: apps/v1` & `kind: StatefulSet`: Establishes that this resource is a StatefulSet, leveraging the stable, production-ready apps/v1 API.
  `apiVersion: apps/v1` & `kind: StatefulSet`：确立该资源为 StatefulSet，利用稳定且适用于生产的 apps/v1 API。
- `metadata.name: my-statefulset`: Assigns a human-readable identifier to the StatefulSet.
  `metadata.name: my-statefulset`：为 StatefulSet 分配一个人类可读的标识符。
- `spec.serviceName: "nginx"`: Binds the StatefulSet to a headless Service named “nginx,” ensuring each pod has a stable network identity.
  `spec.serviceName: "nginx"`：将 StatefulSet 绑定到名为“nginx”的无头服务，确保每个 Pod 具有稳定的网络身份。
- `spec.replicas: 3`: Orchestrates three pod replicas, each with its own persistent identity.
  `spec.replicas: 3`：协调三个 Pod 副本，每个副本都具有其自身的持久身份。
- `spec.selector.matchLabels`: Directs the StatefulSet to manage pods labeled app: nginx.
  `spec.selector.matchLabels`：指示 StatefulSet 管理标记为 app: nginx 的 pod。
- `spec.template`: Defines the pod blueprint:
  `spec.template`：定义 pod 蓝图：
  - `metadata.labels`: app: nginx ensures new pods carry the matching label.
    `metadata.labels`：app: nginx 确保新的 pod 携带匹配的标签。
  - `spec.containers` configures the `nginx` container on port 80, named “web.”
    `spec.containers` 在端口 80 上配置 `nginx` 容器，命名为“web。”

- `volumeClaimTemplates`: Automates the creation of a PersistentVolumeClaim named www for each replica, each requesting 1Gi of storage with ReadWriteOnce access.
  `volumeClaimTemplates`：自动为每个副本创建一个名为 www 的 PersistentVolumeClaim，每个副本请求 1 Gi 的存储空间，并具有 ReadWriteOnce 访问权限。

3. Apply the StatefulSet configuration.
   应用 StatefulSet 配置。

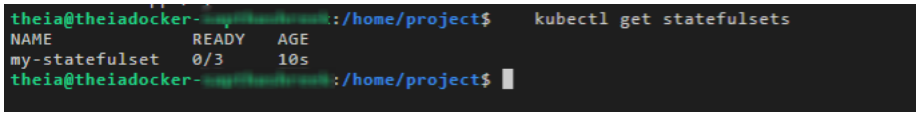
   ```
   kubectl apply -f statefulset.yaml
   ```



This command tells Kubernetes to create the resources defined in the YAML file.
该命令指示 Kubernetes 创建 YAML 文件中定义的资源。

4. **Verify that the StatefulSet is created.**
   验证 StatefulSet 是否已创建。

   ```
   kubectl get statefulsets
   ```



After applying the StatefulSet, you should verify that the StatefulSet has been created and is running. This can be done using the kubectl get command.
在应用 StatefulSet 后，您应该验证 StatefulSet 是否已创建并正在运行。这可以通过使用 kubectl get 命令来完成。

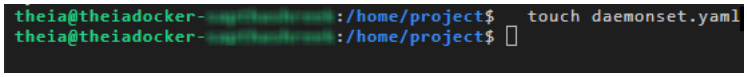
By following these steps, you can successfully apply a StatefulSet in Kubernetes. The kubectl apply command is used to create the StatefulSet, and the kubectl get command helps you verify that the StatefulSet is running as expected.
按照这些步骤，您可以在 Kubernetes 中成功应用 StatefulSet。kubectl apply 命令用于创建 StatefulSet，而 kubectl get 命令帮助您验证 StatefulSet 是否按预期运行。

Task 4: Implementing a DaemonSet
任务 4：实现 DaemonSet

A DaemonSet ensures that a copy of a specific Pod runs on all (or some) nodes in the cluster. It is particularly useful for deploying system-level applications that provide essential services across the nodes in a cluster, such as log collection, monitoring, or networking services.
一个 DaemonSet 确保特定 Pod 的副本在集群的所有（或部分）节点上运行。它特别适用于部署系统级应用，这些应用在集群节点间提供基本服务，例如日志收集、监控或网络服务。

1. **Create a file named `daemonset.yaml` and open it in edit mode:**
   创建一个名为 `daemonset.yaml` 的文件，并以编辑模式打开：

   ```
   touch daemonset.yaml
   ```



2. Create and open a file named `daemonset.yaml` in edit mode.
   创建并进入名为 `daemonset.yaml` 的文件编辑模式。

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: my-daemonset
spec:
  selector:
    matchLabels:
      name: my-daemonset
  template:
    metadata:
      labels:
        name: my-daemonset
    spec:
      containers:
      - name: my-daemonset
        image: nginx
```

Explanation 说明

- `apiVersion: apps/v1` & `kind: DaemonSet`: Declares this resource as a DaemonSet under the stable apps/v1 API.
  `apiVersion: apps/v1` & `kind: DaemonSet`：将此资源声明为稳定 apps/v1 API 下的 DaemonSet。
- `metadata.name: my-daemonset`: Provides the DaemonSet's unique name.
  `metadata.name: my-daemonset`：提供了 DaemonSet 的唯一名称。
- `spec.selector.matchLabels`: Targets pods labeled name: my-daemonset for scheduling.
  `spec.selector.matchLabels`：针对标记为 name: my-daemonset 的 Pod 进行调度。
- `spec.template.metadata.labels`: Labels the pod template so it matches the selector.
  `spec.template.metadata.labels`：标记 Pod 模板，使其与选择器匹配。
- `spec.template.spec.containers`: Defines a single container named my-daemonset using the nginx image.
  `spec.template.spec.containers`：使用 nginx 镜像定义了一个名为 my-daemonset 的单个容器。

3. Apply the DaemonSet 应用 DaemonSet

```
kubectl apply -f daemonset.yaml
```

```
theia@theiadocker-[redacted]:/home/project/kubernetes-practice-lab$ kubectl apply -f daemonset.yaml
daemonset.apps/my-daemonset created
theia@theiadocker-[redacted]:/home/project/kubernetes-practice-lab$
```

This command tells Kubernetes to apply the configuration defined in the daemonset.yaml file. The apply command is used to create or update Kubernetes resources based on the configuration provided in the YAML file.
该命令指示 Kubernetes 应用 daemonset.yaml 文件中定义的配置。apply 命令用于根据 YAML 文件中提供的配置创建或更新 Kubernetes 资源。

4. Verify that the DaemonSet has been created
   验证 DaemonSet 是否已创建

```
kubectl get daemonsets
```

```
theia@theiadocker-[redacted]:/home/project$    kubectl get daemonsets
NAME           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
my-daemonset   7         6         6       6            6           <none>          3m2s
theia@theiadocker-[redacted]:/home/project$
```

This output from `kubectl get daemonsets` provides information about the DaemonSet named "my-daemonset" in your Kubernetes cluster.
`kubectl get daemonsets` 的输出提供了有关您 Kubernetes 集群中名为 "my-daemonset" 的 DaemonSet 的信息。

- **NAME**: The name of the DaemonSet, which is "my-daemonset" in this case.
  名称：DaemonSet 的名称，此处为"my-daemonset"。
- **DESIRED**: The desired number of DaemonSet pods. In your case, it's set to 7.
  期望：期望的 DaemonSet Pod 数量。在此情况下，设置为 7。
- **CURRENT**: The current number of DaemonSet pods running. It shows 6 pods are currently running.
  当前：当前正在运行的 DaemonSet Pod 数量。显示目前有 6 个 Pod 正在运行。
- **READY**: The number of DaemonSet pods that are ready and available for use. All 6 running pods are ready.
  就绪：已准备就绪且可供使用的 DaemonSet Pod 数量。所有 6 个正在运行的 Pod 均已就绪。
- **UP-TO-DATE**: The number of DaemonSet pods that are up-to-date with the latest configuration.
  最新状态：与最新配置保持一致的 DaemonSet 实例数量。
- **AVAILABLE**: The number of DaemonSet pods that are available for use.
  可用状态：可用于使用的 DaemonSet 实例数量。
- **NODE SELECTOR**: Specifies which nodes in the cluster the DaemonSet should run on. In this case, it's set to `<none>`, meaning the DaemonSet is not restricted to specific nodes.
  节点选择器：指定 DaemonSet 应该在集群中的哪些节点上运行。在此情况下，设置为 `<none>` ，表示 DaemonSet 不受特定节点的限制。
- **AGE**: The age of the DaemonSet, indicating how long it has been running.
  运行时长：DaemonSet 的运行时间，表示其已运行多久。

Conclusion 结论

Congratulations! You have completed the practice lab on Kubernetes. You created a Kubernetes Service, used various kubectl commands, deployed StatefulSets for stateful applications, and implemented DaemonSets for uniform pod deployment across cluster nodes.
恭喜！您已完成 Kubernetes 实践实验室。您创建了一个 Kubernetes 服务，使用了多种 kubectl 命令，为有状态应用部署了 StatefulSets，并为集群节点实现了统一 Pod 部署的 DaemonSets。

Author(s) 作者

Manvi Gupta

© IBM Corporation. All rights reserved.