Introduction to Kubernetes
Kubernetes 简介

Objectives 目标

In this lab, you will:
在本实验中，您将：

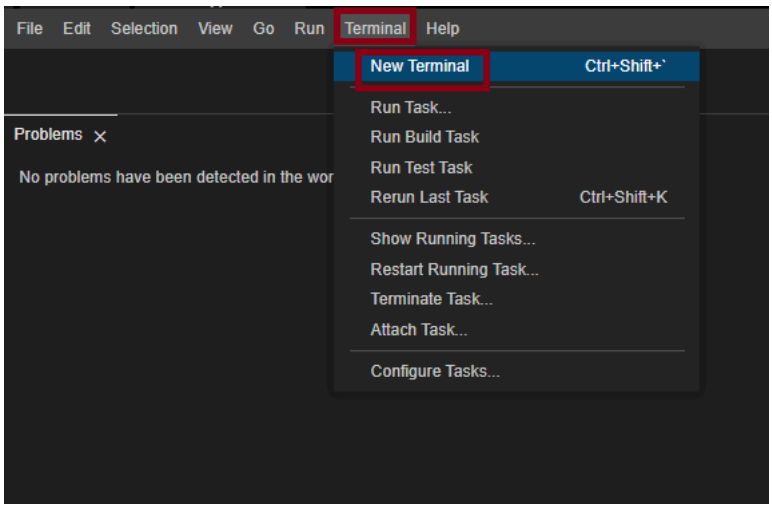
- Use the `kubectl` CLI 使用 `kubectl` 命令行界面
- Create a Kubernetes Pod 创建 Kubernetes Pod
- Create a Kubernetes Deployment
  创建 Kubernetes Deployment
- Create a ReplicaSet that maintains a set number of replicas
  创建一个维持一定数量副本的 ReplicaSet
- Witness Kubernetes load balancing in action
  见证 Kubernetes 负载均衡的实际应用

> **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/errors during the lab process, please logout from 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/错误，请从实验室环境中注销。然后清除系统缓存和 Cookie，并尝试完成实验室。**

Verify the environment and command line tools
验证环境和命令行工具

1. If a terminal is not already open, open a terminal window by using the menu in the editor: `Terminal > New Terminal`.
   如果终端尚未打开，请通过编辑器中的菜单打开一个终端窗口： `Terminal > New Terminal` 。

   > **Note: Please skip this step if the terminal already appears.**
   > **注意：如果终端已经显示，请跳过此步骤。**



2. Verify that `kubectl` CLI is installed.
   验证 `kubectl` CLI 是否已安装。

   ```
   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：

```
theia@theiadocker                /home/project$ kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:"c92036820499fedefec0f847e2054d824aea6cd1", GitTr
"2021-10-27T18:41:28Z", GoVersion:"go1.16.9", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.11+IKS", GitCommit:"7d30e1c191e870ff995f9b6ba21452d0325db2ad",
Date:"2022-03-17T16:12:51Z", GoVersion:"go1.16.15", Compiler:"gc", Platform:"linux/amd64"}
theia@theiadocker                /home/project$
```

3. Change to your project folder.
   切换到您的项目文件夹。

   Note: Please skip this step if you are already on the '/home/project' directory
   注意：如果您已经在'/home/project'目录下，请跳过此步骤。

```
cd /home/project
```

4. Clone the git repository that contains the artifacts needed for this lab, if it doesn't already exist.
   复制包含本实验所需工件的 git 仓库，如果它还不存在的话。

```
[ ! -d 'CC201' ] && git clone https://github.com/ibm-developer-skills-network/CC201.git
```

```
theia@theiadocker                /home/project$ [ ! -d 'CC201' ] && git clone https://github.com/ibm-developer-skills-network/CC201.git
Cloning into 'CC201'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 20 (delta 6), reused 19 (delta 6), pack-reused 0
Unpacking objects: 100% (20/20), done.
theia@theiadocker                /home/project$
```

5. Change to the directory for this lab by running the following command. `cd` will change the working/current directory to the directory with the name specified, in this case **CC201/labs/2_IntroKubernetes**.
   通过运行以下命令切换到本实验的目录。 `cd` 将会将工作目录/当前目录更改为指定名称的目录，在本例中为 CC201/labs/2_IntroKubernetes。

```
cd CC201/labs/2_IntroKubernetes/
```

6. List the contents of this directory to see the artifacts for this lab.
   列出此目录的内容，以查看本实验的工件。

```
ls
```

```
theia@theiadocker                /home/project/CC201/labs/2_IntroKubernetes$ ls
app.js  Dockerfile  hello-world-apply.yaml  hello-world-create.yaml  package.json
theia@theiadocker                /home/project/CC201/labs/2_IntroKubernetes$
```

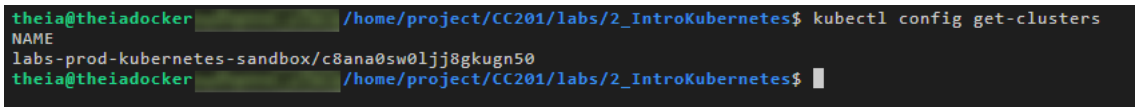
Use the `kubectl` CLI 使用 `kubectl` 命令行界面

Recall that Kubernetes namespaces enable you to virtualize a cluster. You already have access to one namespace in a Kubernetes cluster, and `kubectl` is already set to target that cluster and namespace.
回想一下，Kubernetes 命名空间使您能够虚拟化集群。您已经可以访问 Kubernetes 集群中的一个命名空间，并且 `kubectl` 已经设置为针对该集群和命名空间。

Let's look at some basic `kubectl` commands.
让我们看看一些基本的 `kubectl` 命令。

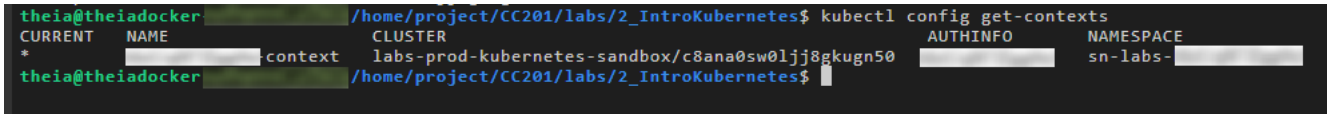
1. `kubectl` requires configuration so that it targets the appropriate cluster. Get cluster information with the following command:
   `kubectl` 需要配置以便它能够针对适当的集群。使用以下命令获取集群信息：

   ```
   kubectl config get-clusters
   ```



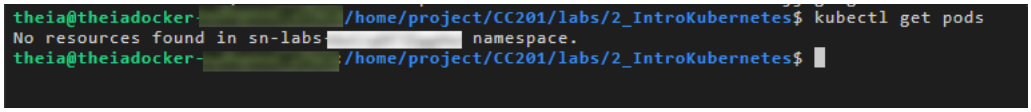
2. A `kubectl` context is a group of access parameters, including a cluster, a user, and a namespace. View your current context with the following command:
   一个 `kubectl` 上下文是一组访问参数，包括一个集群、一个用户和一个命名空间。使用以下命令查看您当前的上下文：

   ```
   kubectl config get-contexts
   ```



3. List all the Pods in your namespace. If this is a new session for you, you will not see any Pods.
   列出您命名空间中的所有 Pod。如果您是首次使用此会话，您将看不到任何 Pod。

   ```
   kubectl get pods
   ```



Create a Pod with an imperative command
创建一个使用祈使命令的 Pod

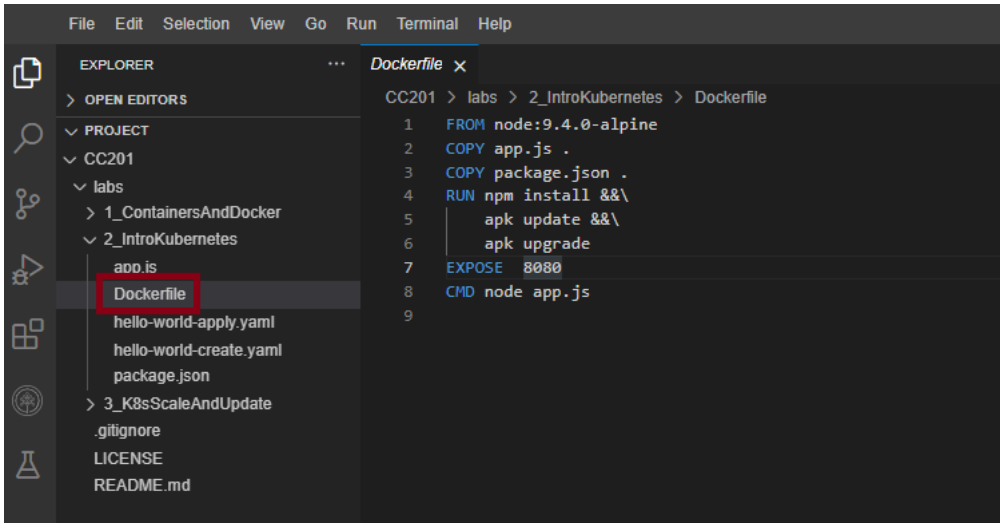
Now it's time to create your first Pod. This Pod will run the `hello-world` image you built and pushed to IBM Cloud Container Registry in the last lab. As explained in the videos for this module, you can create a Pod imperatively or declaratively. Let's do it imperatively first.
现在，是时候创建您的第一个 Pod 了。这个 Pod 将运行您在上一实验室中构建并推送到 IBM Cloud Container Registry 的 `hello-world` 镜像。正如本模块的视频中所解释的，您可以命令式地或声明式地创建 Pod。让我们先进行命令式创建。

1. Export your namespace as an environment variable so that it can be used in subsequent commands.
   将您的命名空间导出为环境变量，以便在后续命令中使用。

   ```
   export MY_NAMESPACE=sn-labs-$USERNAME
   ```

```
theia@theiadocker-[redacted] /home/project/CC201/labs/2_IntroKubernetes$ export MY_NAMESPACE=sn-labs-$USERNAME
theia@theiadocker-[redacted] /home/project/CC201/labs/2_IntroKubernetes$
```

2. Click the Explorer icon (it looks like a sheet of paper) on the left side of the window, and then navigate to the directory for this lab: `CC201 > labs > 2_IntroKubernetes`. Click on `Dockerfile`. This is the file that will be used to build our image.
   点击窗口左侧的 Explorer 图标（看起来像一张纸），然后导航到本实验室的目录： `CC201 > labs > 2_IntroKubernetes` 。点击 `Dockerfile` 。这是将用于构建我们镜像的文件。



3. Build and push the image again, as it may have been deleted automatically since you completed the first lab.
   再次构建和推送镜像，因为它可能在您完成第一个实验室后自动被删除。

```
docker build -t us.icr.io/$MY_NAMESPACE/hello-world:1 . && docker push us.icr.io/$MY_NAMESPACE/hello-world:1
```

```
theia@theiadocker-[redacted] /home/project/CC201/labs/2_IntroKubernetes$ docker build -t us.icr.io/$MY_NAMESPACE/hello-world:1 . &&
ACE/hello-world:1
Sending build context to Docker daemon  6.656kB
Step 1/6 : FROM node:9.4.0-alpine
9.4.0-alpine: Pulling from library/node
605ce1bd3f31: Pull complete
fe58b30348fe: Pull complete
46ef8987ccbd: Pull complete
Digest: sha256:9cd67a00ed111285460a83847720132204185e9321ec35dacec0d8b9bf674adf
Status: Downloaded newer image for node:9.4.0-alpine
 ---> b5f94997f35f
Step 2/6 : COPY app.js .
 ---> 28350e465969
Step 3/6 : COPY package.json .
 ---> 45bf6db4af5f
Step 4/6 : RUN npm install &&    apk update &&    apk upgrade
 ---> Running in a37db9ced1bc
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN hello-world-demo@0.0.1 No repository field.
npm WARN hello-world-demo@0.0.1 No license field.

added 50 packages in 2.085s
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.6/community/x86_64/APKINDEX.tar.gz
v3.6.5-44-gda55e27396 [http://dl-cdn.alpinelinux.org/alpine/v3.6/main]
v3.6.5-34-gf0ba0b43d5 [http://dl-cdn.alpinelinux.org/alpine/v3.6/community]
OK: 8448 distinct packages available
Upgrading critical system libraries and apk-tools:
(1/1) Upgrading apk-tools (2.7.5-r0 -> 2.7.6-r0)
Executing busybox-1.26.2-r9.trigger
Continuing the upgrade transaction with new apk-tools:
(1/7) Upgrading musl (1.1.16-r14 -> 1.1.16-r15)
(2/7) Upgrading busybox (1.26.2-r9 -> 1.26.2-r11)
Executing busybox-1.26.2-r11.post-upgrade
(3/7) Upgrading libressl2.5-libcrypto (2.5.5-r0 -> 2.5.5-r2)
(4/7) Upgrading libressl2.5-libssl (2.5.5-r0 -> 2.5.5-r2)
(5/7) Installing libressl2.5-libtls (2.5.5-r2)
```

4. Run the `hello-world` image as a container in Kubernetes.
   在 Kubernetes 中以容器方式运行 `hello-world` 图像。

```
kubectl run hello-world --image us.icr.io/$MY_NAMESPACE/hello-world:1 --overrides='{"spec":{"template":{"spec":{"imagePullSecrets":[
```

The `--overrides` option here enables us to specify the needed credentials to pull this image from IBM Cloud Container Registry. Note that this is an imperative command, as we told Kubernetes explicitly what to do: run `hello-world`.
这里的 `--overrides` 选项使我们能够指定从 IBM 云容器注册中心拉取此图像所需的凭证。请注意，这是一个祈使命令，因为我们明确告诉 Kubernetes 要做什么：运行 `hello-world` 。

```
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$ kubectl run hello-world --image us.icr.io/$MY_NAMESPACE/h
{"template":{"spec":{"imagePullSecrets":[{"name":"icr"}]}}}}'
pod/hello-world created
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$
```

5. List the Pods in your namespace.
   列出您命名空间中的 Pod。

```
kubectl get pods
```

```
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
hello-world   1/1     Running   0          34s
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$
```

Great, the previous command indeed created a Pod for us. You can see an auto-generated name was given to this Pod.
太好了，之前的命令确实为我们创建了一个 Pod。您可以看到这个 Pod 被赋予了自动生成的名称。

You can also specify the wide option for the output to get more details about the resource.
您也可以为输出指定 wide 选项以获取更多关于资源的详细信息。

```
kubectl get pods -o wide
```

```
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE           NOMINATED NODE   READINESS GATES
hello-world   1/1     Running   0          59s   172.17.183.177   10.241.64.24   <none>           <none>
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$
```

6. Describe the Pod to get more details about it.
   描述 Pod 以获取更多关于它的详细信息。

```
kubectl describe pod hello-world
```

```
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$ kubectl describe pod hello-world
Name:                 hello-world
Namespace:            sn-labs-[redacted]
Priority:             1
Priority Class Name:  normal
Node:                 10.241.64.24/10.241.64.24
Start Time:           Fri, 08 Apr 2022 05:15:40 +0000
Labels:               run=hello-world
Annotations:          cni.projectcalico.org/containerID: c89fd419d56a582514d497f0b01b939cf745343036e9a45f135235e7d5bc528e
                      cni.projectcalico.org/podIP: 172.17.183.177/32
                      cni.projectcalico.org/podIPs: 172.17.183.177/32
                      kubernetes.io/limit-ranger:
                        LimitRanger plugin set: cpu, ephemeral-storage, memory request for container hello-world; cpu, ephemeral-sto
                      kubernetes.io/psp: ibm-privileged-psp
Status:               Running
IP:                   172.17.183.177
IPs:
  IP:  172.17.183.177
Containers:
  hello-world:
    Container ID:   containerd://31c934f489c232a36729b3e3f013a5619f11fc8f95ee8a1007f9f540dc4d420a
    Image:          us.icr.io/sn-labs-[redacted]/hello-world:1
    Image ID:       us.icr.io/sn-labs-[redacted]/hello-world@sha256:a04a56181ae9136e4b7033d5284ce9d68fe812c21b28592ffb292d8b496b6b
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 08 Apr 2022 05:15:46 +0000
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:                500m
      ephemeral-storage:  5Gi
      memory:             512Mi
    Requests:
      cpu:                200m
      ephemeral-storage:  512Mi
      memory:             128Mi
    Environment:          <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-bjdzp (ro)
Conditions:
  Type              Status
```

Note: The output shows the pod parameters like **Namespace, Pod Name, IP address, the time when the pod started running** and also the container parameters like **container ID, image name & ID, running status and the memory/CPU limits.**
注意：输出显示了 Pod 参数，如命名空间、Pod 名称、IP 地址、Pod 开始运行的时间，以及容器参数，如容器 ID、镜像名称 & ID、运行状态以及内存/CPU 限制。

7. Delete the Pod. 删除 Pod。

```
kubectl delete pod hello-world
```

```
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$ kubectl delete pod hello-world
pod "hello-world" deleted
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$
```

This command takes a while to execute the deletion of the pod. Please wait till the terminal prompt appears again.
该命令执行删除 Pod 的操作需要一些时间，请等待终端提示符再次出现。

8. List the Pods to verify that none exist.
列出 Pods 以验证是否不存在。

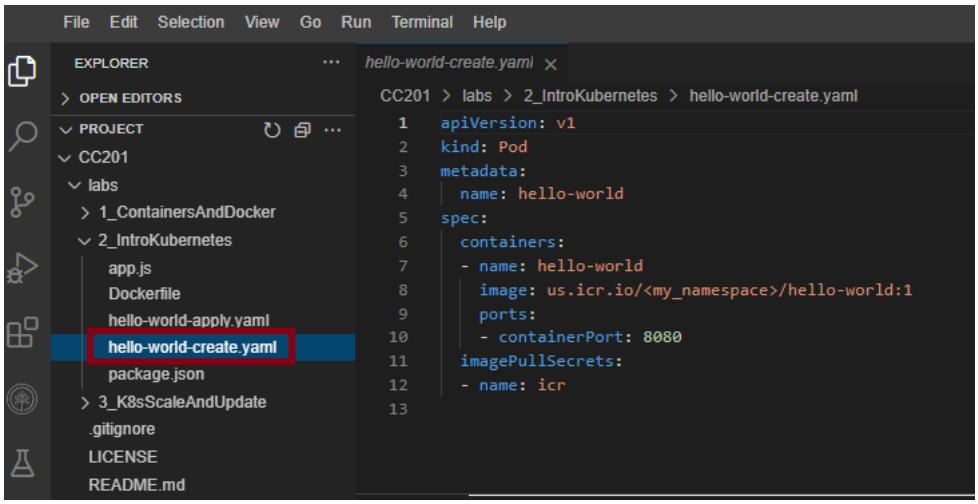
```
kubectl get pods
```

```
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$ kubectl get pods
No resources found in sn-labs-[redacted] namespace.
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$
```

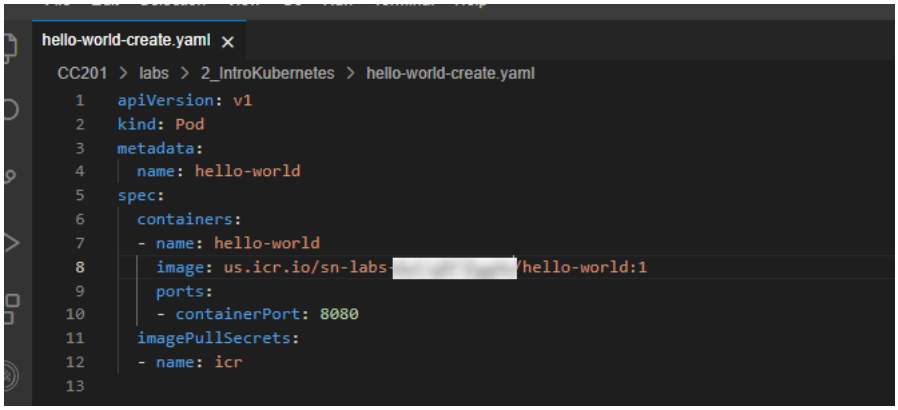
Create a Pod with imperative object configuration
创建一个具有命令式对象配置的 Pod

Imperative object configuration lets you create objects by specifying the action to take (e.g., create, update, delete) while using a configuration file. A configuration file, `hello-world-create.yaml`, is provided to you in this directory.
命令式对象配置允许您通过指定要执行的操作（例如，创建、更新、删除）并使用配置文件来创建对象。在此目录中为您提供了一个配置文件， `hello-world-create.yaml` 。

1. Use the Explorer to view and edit the configuration file. Click the Explorer icon (it looks like a sheet of paper) on the left side of the window, and then navigate to the directory for this lab: `CC201 > labs > 2_IntroKubernetes`. Click `hello-world-create.yaml` to view the configuration file.
   使用资源管理器查看和编辑配置文件。点击窗口左侧的资源管理器图标（看起来像一张纸），然后导航到此实验的目录： `CC201 > labs > 2_IntroKubernetes` 。点击 `hello-world-create.yaml` 查看配置文件。



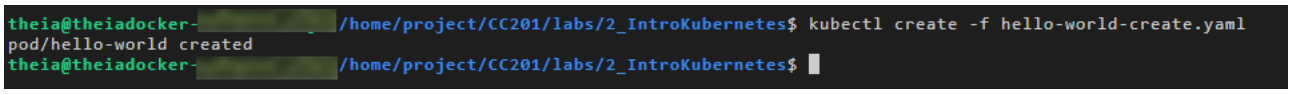
2. Use the Explorer to edit `hello-world-create.yaml`. You need to insert your namespace where it says `<my_namespace>`. Make sure to save the file when you're done.
   使用资源管理器编辑 `hello-world-create.yaml` 。您需要在 `<my_namespace>` 处插入您的命名空间。完成后请保存文件。



3. Imperatively create a Pod using the provided configuration file.
   使用提供的配置文件命令式地创建一个 Pod。

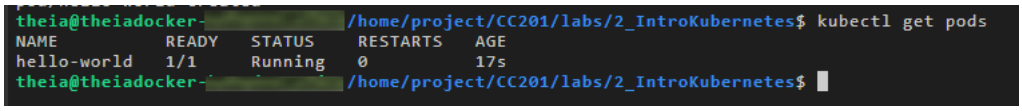
```
kubectl create -f hello-world-create.yaml
```

Note that this is indeed imperative, as you explicitly told Kubernetes to *create* the resources defined in the file.
请注意，这确实是必要的，因为你明确告诉了 Kubernetes 创建文件中定义的资源。



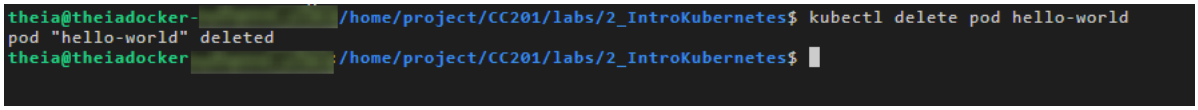
4. List the Pods in your namespace.
   列出您命名空间中的 Pod。

```
kubectl get pods
```



5. Delete the Pod. 删除 Pod。

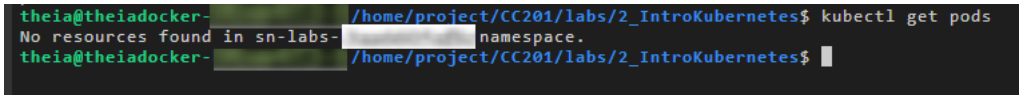
```
kubectl delete pod hello-world
```



This command takes a while to execute the deletion of the pod. Please wait till the terminal prompt appears again.
该命令执行删除 Pod 的操作需要一些时间，请等待终端提示符再次出现。

6. List the Pods to verify that none exist.
列出 Pods 以验证是否不存在。

```
kubectl get pods
```

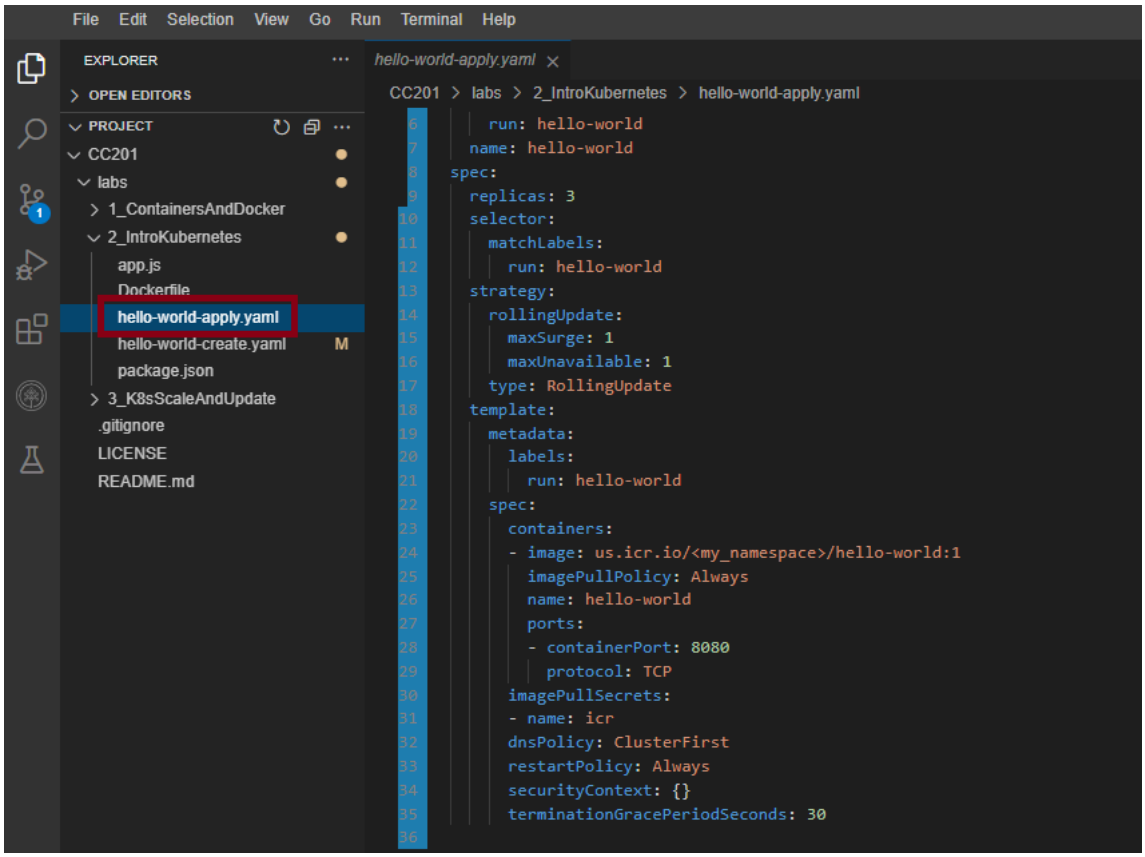


Create a Pod with a declarative command
创建一个使用声明式命令的 Pod

The previous two ways to create a Pod were imperative — we explicitly told `kubectl` what to do. While the imperative commands are easy to understand and run, they are not ideal for a production environment. Let's look at declarative commands.
创建 Pod 的前两种方法都是命令式的——我们明确告诉 `kubectl` 该做什么。虽然命令式指令易于理解和执行，但它们并不适合生产环境。让我们看看声明式指令。

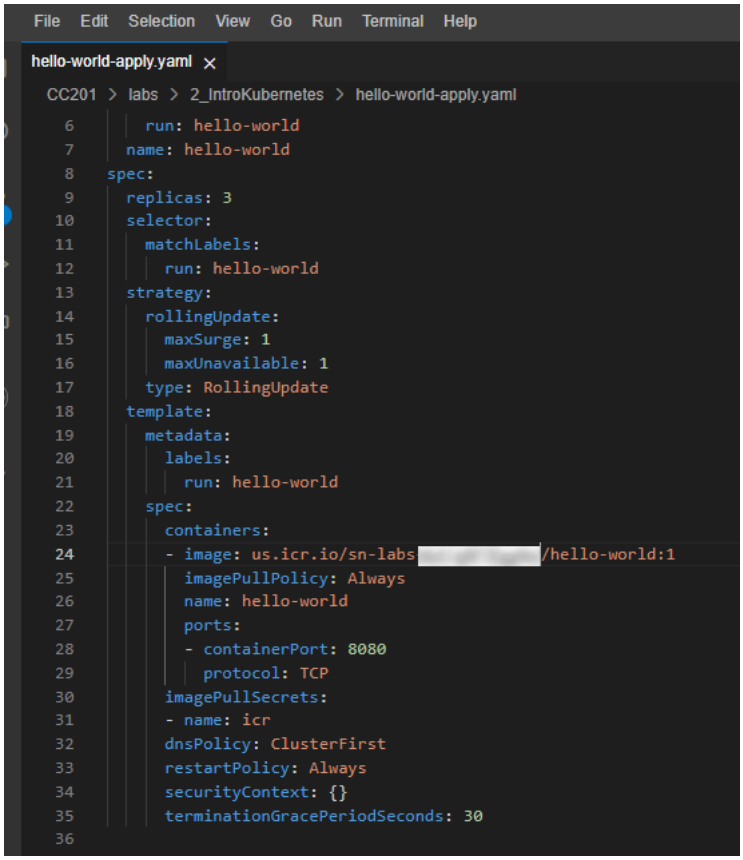
1. A sample `hello-world-apply.yaml` file is provided in this directory. Use the Explorer again to open this file. Notice the following:
本目录中提供了一个 `hello-world-apply.yaml` 文件。再次使用资源管理器打开此文件。请注意以下几点：

- We are creating a Deployment (`kind: Deployment`).
我们正在创建一个 Deployment (`kind: Deployment`)。
- There will be three replica Pods for this Deployment (`replicas: 3`).
这个 Deployment (`replicas: 3`) 将会有三个副本 Pods。
- The Pods should run the `hello-world` image (`- image: us.icr.io/<my_namespace>/hello-world:1`).
Pods 应运行 `hello-world` 图像 (`- image: us.icr.io/<my_namespace>/hello-world:1`)。



You can ignore the rest for now. We will get to a lot of those concepts in the next lab.
你可以暂时忽略其余部分。我们将在下一个实验中接触到很多这些概念。

2. Use the Explorer to edit `hello-world-apply.yaml`. You need to insert your namespace where it says `<my_namespace>`. Make sure to save the file when you're done.
使用资源管理器编辑 `hello-world-apply.yaml` 。您需要在 `<my_namespace>` 处插入您的命名空间。完成后请保存文件。



3. Use the `kubectl apply` command to set this configuration as the desired state in Kubernetes.
使用 `kubectl apply` 命令将此配置设置为 Kubernetes 中的期望状态。

```
kubectl apply -f hello-world-apply.yaml
```

```
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$ kubectl apply -f hello-world-apply.yaml
deployment.apps/hello-world created
```

4. Get the Deployments to ensure that a Deployment was created.
   获取 Deployments 以确保已创建 Deployment。

   ```
   kubectl get deployments
   ```

```
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-world   3/3     3            3           22s
```

5. List the Pods to ensure that three replicas exist.
   列出 Pod 以确保存在三个副本。

   ```
   kubectl get pods
   ```

```
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
hello-world-774ddf45b5-86gn6   1/1     Running   0          42s
hello-world-774ddf45b5-9cbv2   1/1     Running   0          41s
hello-world-774ddf45b5-svpf7   1/1     Running   0          41s
theia@theiadocker-[redacted]/home/project/CC201/labs/2_IntroKubernetes$
```

With declarative management, we did not tell Kubernetes which actions to perform. Instead, `kubectl` inferred that this Deployment needed to be created. If you delete a Pod now, a new one will be created in its place to maintain three replicas.
通过声明式管理，我们没有告诉 Kubernetes 要执行哪些操作。相反， `kubectl` 推断出这个 Deployment 需要被创建。如果你现在删除一个 Pod，一个新的将被创建来维持三个副本。

6. Note one of the Pod names from the previous step, replace the `pod_name` in the following command with the pod name that you noted and delete that Pod and list the pods. To see one pod being terminated, there by having just 2 pods, we will follow the **delete**, immediately with **get**.
   从上一步中记下一个 Pod 名称，在以下命令中替换 `pod_name` 为你记下的 Pod 名称，并删除该 Pod，然后列出 Pod。为了看到有一个 Pod 被终止，从而只有两个 Pod，我们将跟随删除操作，立即使用 get 命令。

   ```
   kubectl delete pod <pod_name> && kubectl get pods
   ```

```
theia@theiadocker-ksundararaja:/home/project/CC201/labs/2_IntroKubernetes$ kubectl delete pod hello-world-5b5467f896-9brft && kubect
pod "hello-world-5b5467f896-9brft" deleted
NAME                           READY   STATUS    RESTARTS   AGE
hello-world-5b5467f896-6jpnd   1/1     Running   0          3m7s
hello-world-5b5467f896-wz45f   1/1     Running   0          3m6s
theia@theiadocker-ksundararaja:/home/project/CC201/labs/2_IntroKubernetes$
```

This command takes a while to execute the deletion of the pod. Please wait till the terminal prompt appears again.
该命令执行删除 Pod 的操作需要一些时间，请等待终端提示符再次出现。

7. List the Pods to see a new one being created.
   列出 Pod 以查看一个新的正在被创建。

You may have to run this command a few times as it may take a while to create the new pod.
您可能需要运行几次此命令，因为创建新的 pod 可能需要一些时间。

```
kubectl get pods
```

```
NAME                     READY   STATUS    RESTARTS   AGE
hello-world-774ddf45b5-28k7j   1/1     Running   0          36s
hello-world-774ddf45b5-9cbv2   1/1     Running   0          112s
hello-world-774ddf45b5-svpf7   1/1     Running   0          112s
```

The output should reflect three pods running.
输出应反映三个 pod 正在运行。

Load balancing the application
应用程序负载均衡

Since there are three replicas of this application deployed in the cluster, Kubernetes will load balance requests across these three instances. Let's expose our application to the internet and see how Kubernetes load balances requests.
由于该应用程序在集群中部署了三个副本，Kubernetes 将跨这三个实例进行请求负载均衡。让我们将应用程序暴露给互联网，看看 Kubernetes 如何进行请求负载均衡。

1. In order to access the application, we have to expose it to the internet using a Kubernetes Service.
   为了访问该应用程序，我们必须使用 Kubernetes 服务将其暴露给互联网。

   ```
   kubectl expose deployment/hello-world
   ```

```
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$ kubectl expose deployment/hello-world
service/hello-world exposed
```

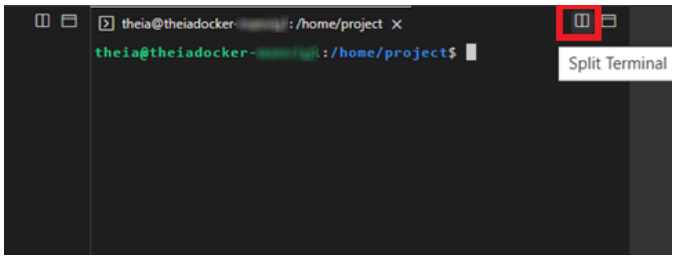
This command creates what is called a ClusterIP Service. This creates an IP address that accessible within the cluster.
该命令创建了一个称为 ClusterIP 服务的东西。这会创建一个在集群内部可访问的 IP 地址。

2. List Services in order to see that this service was created.
   列出服务，以查看该服务是否已创建。

   ```
   kubectl get services
   ```

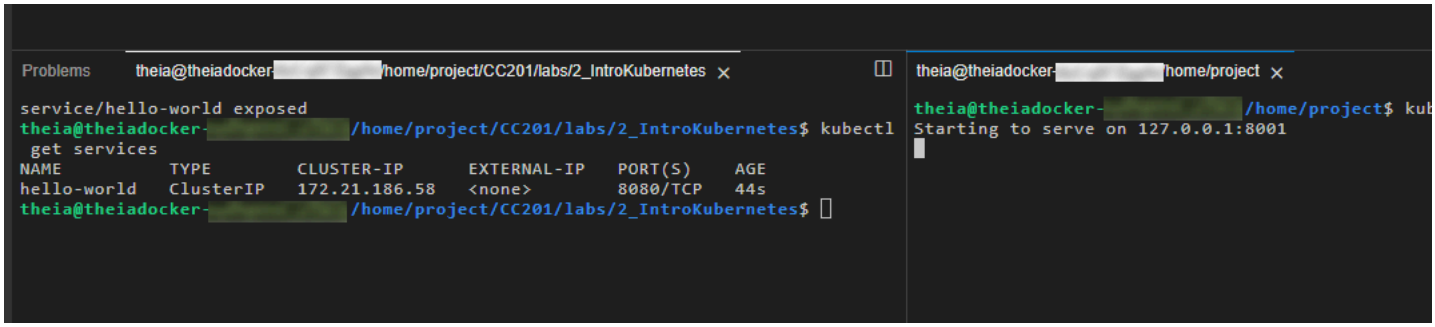
```
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$ kubectl get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
hello-world   ClusterIP   172.21.186.58   <none>        8080/TCP   44s
theia@theiadocker-                /home/project/CC201/labs/2_IntroKubernetes$
```

3. Open a new split terminal window by locate the split icon in the top-right corner of the terminal panel.
   通过在终端面板右上角找到拆分图标来打开一个新的拆分终端窗口。



4. Since the cluster IP is not accessible outside of the cluster, we need to create a proxy. Note that this is not how you would make an application externally accessible in a production scenario. Run this command in the new terminal window since your environment variables need to be accessible in the original window for subsequent commands.
由于集群 IP 无法从集群外部访问，我们需要创建一个代理。请注意，这不是在生产环境中使应用程序外部可访问的方法。在新终端窗口中运行此命令，因为您需要从原始窗口中访问环境变量以执行后续命令。

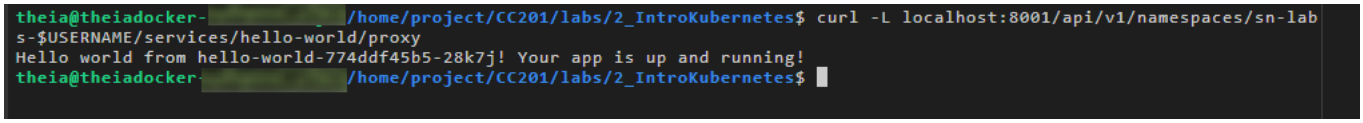
```
kubectl proxy
```



This command doesn't terminate until you terminate it. Keep it running so that you can continue to access your app.
此命令不会终止，直到您手动终止它。保持它运行，以便您可以继续访问您的应用程序。

5. In the original terminal window, ping the application to get a response.
在原始终端窗口中，对应用程序进行 ping 操作以获取响应。

```
curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy
```



Notice that this output includes the Pod name.
注意输出中包含 Pod 名称。

6. Execute the following command to send ten consecutive requests to the hello-world service via the Kubernetes API proxy. As each request is forwarded, note the pod name in the response (which shows which pod handled the request).
执行以下命令通过 Kubernetes API 代理向 hello-world 服务发送十次连续请求。每次请求转发时，注意响应中的 Pod 名称（这显示了处理请求的 Pod）。

```
for i in `seq 10`; do curl -L localhost:8001/api/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy; done
```

```
theia@theiadocker-[redacted] /home/project/CC201/labs/2_IntroKubernetes$ for i in `seq 10`; do curl -L localhost:8001/ap
i/v1/namespaces/sn-labs-$USERNAME/services/hello-world/proxy; done
Hello world from hello-world-774ddf45b5-svpf7! Your app is up and running!
Hello world from hello-world-774ddf45b5-9cbv2! Your app is up and running!
Hello world from hello-world-774ddf45b5-28k7j! Your app is up and running!
Hello world from hello-world-774ddf45b5-28k7j! Your app is up and running!
Hello world from hello-world-774ddf45b5-28k7j! Your app is up and running!
Hello world from hello-world-774ddf45b5-28k7j! Your app is up and running!
Hello world from hello-world-774ddf45b5-28k7j! Your app is up and running!
Hello world from hello-world-774ddf45b5-28k7j! Your app is up and running!
Hello world from hello-world-774ddf45b5-svpf7! Your app is up and running!
Hello world from hello-world-774ddf45b5-svpf7! Your app is up and running!
theia@theiadocker-[redacted] /home/project/CC201/labs/2_IntroKubernetes$
```

You should see more than one Pod name, and quite possibly all three Pod names, in the output. This is because Kubernetes load balances the requests across the three replicas, so each request could hit a different instance of our application.
您应该看到多个 Pod 名称，很可能全部三个 Pod 名称都会出现在输出中。这是因为 Kubernetes 将请求跨三个副本进行负载均衡，因此每次请求都可能击中我们应用程序的不同实例。

7. Delete the Deployment and Service. This can be done in a single command by using slashes.
   删除 Deployment 和 Service。这可以通过使用斜杠的单个命令来完成。

   ```
   kubectl delete deployment/hello-world service/hello-world
   ```

```
theia@theiadocker-[redacted] /home/project/CC201/labs/2_IntroKubernetes$ kubectl delete deployment/hello-world service/h
ello-world
deployment.apps "hello-world" deleted
service "hello-world" deleted
theia@theiadocker-[redacted] /home/project/CC201/labs/2_IntroKubernetes$
```

> **Note: If you face any issues in typing further commands in the terminal, press Enter.**
> **注意：如果在终端中输入后续命令时遇到问题，请按 Enter 键。**

8. Return to the terminal window running the `proxy` command and kill it using `Ctrl+C`.
   返回正在运行 `proxy` 命令的终端窗口，并使用 `Ctrl+C` 命令终止它。

```
theia@theiadocker-[redacted] home/project ×

theia@theiadocker-[redacted] /home/project$ kubectl proxy
Starting to serve on 127.0.0.1:8001
^C
theia@theiadocker-[redacted] /home/project$
```

Congratulations! You have completed the lab for the second module of this course.
恭喜！您已完成本课程第二模块的实验。

© IBM Corporation. All rights reserved.