Simulation Transcript
Deploy a Docker container

Clara, the senior engineer, welcomes you to the project. You've got VS code open and you're ready to start. Clara points out the areas you'll be using. Notice the following three sections and hover over the I icons. Then, elect the **Next arrow** to continue.

1. Clara explains that the first step is to create a container registry on IBM Cloud. This registry stores the Docker image you'll be creating. By using the registry, Kubernetes has a secure place from which to find and deploy the Docker image. Select the **Next arrow** to continue.

2. The Code pad provides all the code you need to complete your coding work. You will refer to it throughout these steps. Select each field and then type each line of code exactly as you see it in the Code pad. For example, in the field in the terminal, type **ibmcloud cr namespace-add utilities-cloud-registry**. Select the **Next arrow** to continue.

3. Clara asks you to create a registry using the name "utilities-cloud-registry". This command uses the IBM Cloud command line utility that communicates with IBM Cloud. To create the registry, place your cursor at the terminal window prompt and type the following command. Then, press **Enter**.

   Code pad code:
   ibmcloud cr namespace-add utilities-cloud-registry

4. Clara asks you to verify that you successfully created the registry. She tells you that after you run the command, the new registry should display in the namespace list. To verify that you created the registry, place your cursor at the terminal window prompt and type the following command. Then, press **Enter**.

   Code pad code:
   ibmcloud cr namespace-list -v

5. Clara points to the result of the command you ran which shows the registry in the output of the namespace-list command. You created the registry successfully. Nice work! Select the **Next arrow** to continue.

6. The next step is to build a Docker image that is ready for IBM Cloud. The Docker image must include the operating system files that match the operating system used on IBM Cloud. Clara asks you to build an image for Linux. To build an image that targets the Linux operating system, place your cursor at the terminal window prompt and type the following command. Then, press **Enter**.

Code pad code:
docker buildx build --platform linux/amd64 -t todo-app .

7. Clara tells you that it's time to create a copy of the image you just created that can be used in Kubernetes on the cloud. You'll use the Docker **tag** command that will create a version of the image that Kubernetes can reference from the registry you created earlier. To tag the image, place your cursor at the terminal window prompt and type the following command. Then, press **Enter**.

   Code pad code:
   docker tag todo-app:latest us.icr.io/utilities-cloud-registry/todo-app

8. Clara thanks you for tagging the image and explains that you now can push the image to the registry. The Docker image is currently on the hard disk on the laptop. For Kubernetes to be able to use it, you must push it to the registry you created earlier. Select the **Next arrow** to continue.

9. Clara explains that you will use a Docker command to push the image to the registry. Before you can do that, you must give Docker permissions to communicate with IBM Cloud. Clara asks you to run the command that will give Docker the required permissions. Place your cursor at the terminal window prompt and type the following command. Then press **Enter**.

   Code pad code:
   ibmcloud cr login --client docker

10. Now you're ready to push the image using the Docker "push" command. Clara asks you to notice that everything after the word "push" in the command is a reference to the image you tagged earlier. To tell Docker to push the image to the registry, place your cursor at the terminal window prompt and type the following command. Then, press **Enter**.

    Code pad code:
    docker push us.icr.io/utilities-cloud-registry/todo-app

11. Clara asks you to review the output in the terminal window and verify that all the files reached 100%. She congratulates you on a successful push to the registry. The image is now in the container registry on IBM Cloud and is ready for your Kubernetes cluster to use. Nice work! Select the **Next arrow** to continue.

12. Clara tells you that you're making good progress and explains that you can now deploy the image to the Kubernetes cluster. You're getting closer to making the app available to employees worldwide! Select the **Next arrow** to continue.

13. To deploy the app to Kubernetes, you need a special file that tells Kubernetes how to deploy the image. Clara creates a file called app-deploy.yaml in which you'll write the deployment instructions. Select the **Next arrow** to continue.

14. Clara has already included the code in this file to tell Kubernetes how to deploy the image from the registry. She explains that the file also needs to create a service on Kubernetes through which the to-do app will communicate with the outside world. She asks you to write that section of the file. Place your cursor on line 21 in the app-deploy.yaml file and type the following code. Then, press **Enter.**

    Code pad code:

```
apiVersion: v1
 kind: Service
 metadata
  name: todo-app
 spec:
  type: NodePort
  ports:
  - port: 3000
   targetPort: 3000
  selector:
   app: todo-app
```

15. Clara compliments you on your work. The deployment file is complete. Nice job! Next, it's time to deploy the image to Kubernetes. Clara explains that you'll use a command that will send the instructions in the deployment file to Kubernetes. Kubernetes will use those instructions to make the app live on the internet. You'll be using a special Kubernetes command line tool, kubectl, to run the command. Select the **Next arrow** to continue.

16. To deploy the image to Kubernetes, place your cursor at the terminal window prompt and type the following command. Then press **Enter**.

    Code pad code:

```
kubectl apply -f app-deploy.yaml
```

17. The app is now available on the internet and Kubernetes is delivering the app using a Docker container. Clara tells you that before you can call the deployment a success, you must perform a test to determine if the app is running and available for employees to access. To do this, you need to retrieve the public IP address and a port number which you can use in a browser to test the application. Select the **Next arrow** to continue.

18. First, you'll get the public IP address. Place your cursor at the terminal window prompt and type the following command. Then, press **Enter**.

Code pad code:
ibmcloud ks worker ls --cluster mycluster-free

19. Clara tells you to look for the column called **Public IP** in the output from the command you just ran. She asks you to copy and paste that into a browser window, which you do. Select the **Next arrow** to continue.

20. Then, you need to get the port number for the service you created. Place your cursor at the terminal window prompt and type the following command. Then, press **Enter**.

    Code pad code:
    kubectl describe service todo-app

21. Clara tells you to look for the row in the output called **NodePort**. She asks you to copy the value for **NodePort** and paste that after the public IP address you pasted in your browser window with the two values separated by a colon. You do this. Select the **Next arrow** to continue.

22. Clara tells you to browse to the location you put into your browser's address bar by pressing **Enter** on your keyboard. This will load the Node.js web application in the browser. Place your cursor after the port number in the address bar in the browser and press **Enter**.

23. Success! The application loads and you have successfully deployed it using a Docker container and Kubernetes. Notice that the app running in the container through Kubernetes appears exactly as it would if it were running on a native web server. This is because the web server is running in a container, which makes it easier to deploy to multiple regions or to scale as needed. Clara compliments you on your success with this project and thanks you for your assistance. Select the **Next arrow** to continue.

You successfully created a deployment file for your application and used command line tools to push a Docker image to the cloud. You also used the command line to deploy the app through Kubernetes on IBM Cloud. Finally, you tested that the app is running on the public cloud using an IP address and port number that employees of your company can use to get to the application.