**Lab 5**

Lukas Kmitas

C00252129

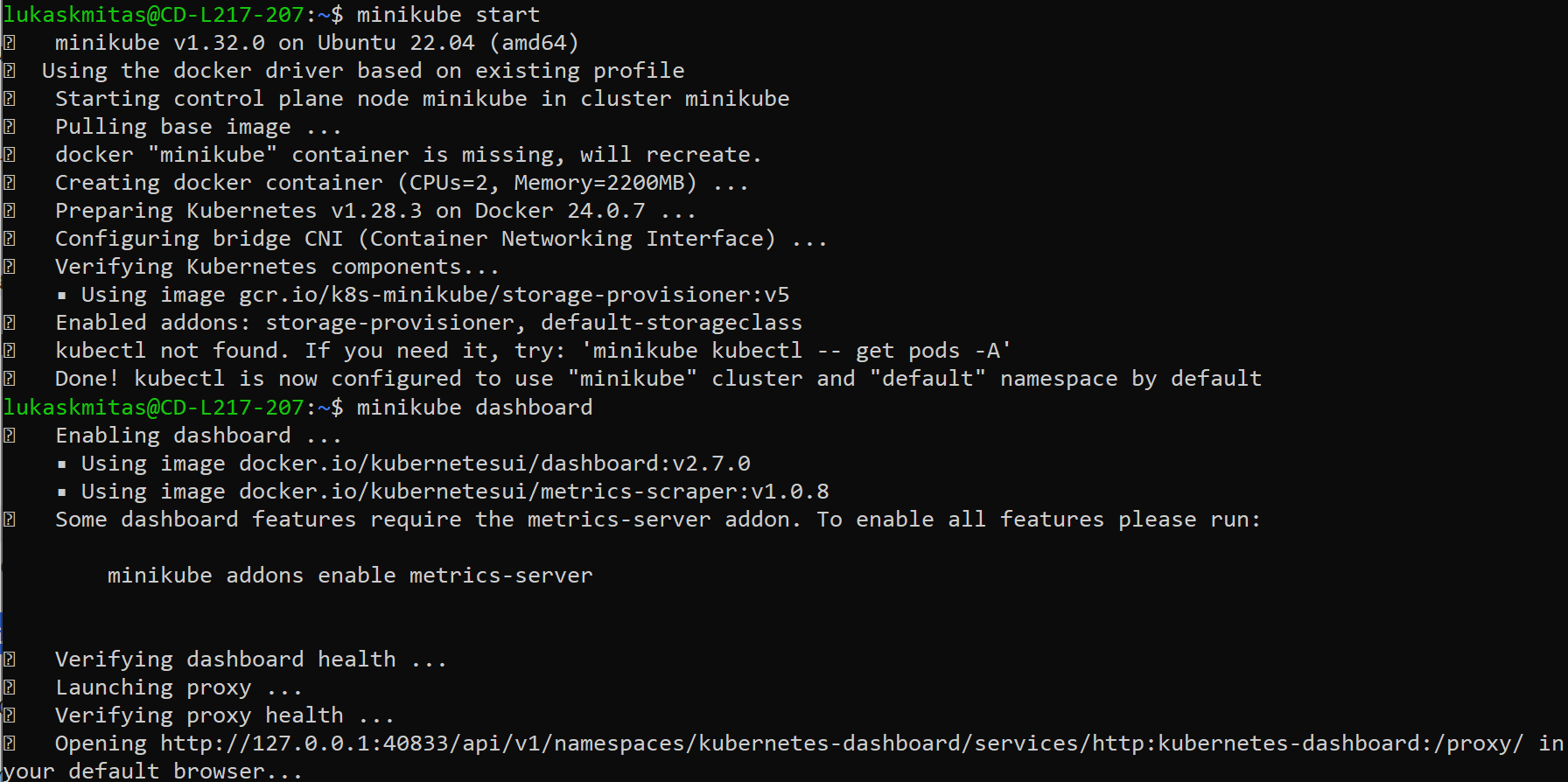
**Introduction**

Kubernetes is an open-source platform designed to automate deploying, scaling, and operating application containers. Originally developed by Google, Kubernetes is now maintained by the Cloud Native Computing Foundation (CNCF). It provides a container-centric management environment, orchestrating computing, networking, and storage infrastructure on behalf of user workloads.

Kubernetes provides a powerful platform for deploying and managing containerized applications at scale, offering features for automation, scalability, reliability, and resilience. It has become the de facto standard for container orchestration in the cloud-native ecosystem.

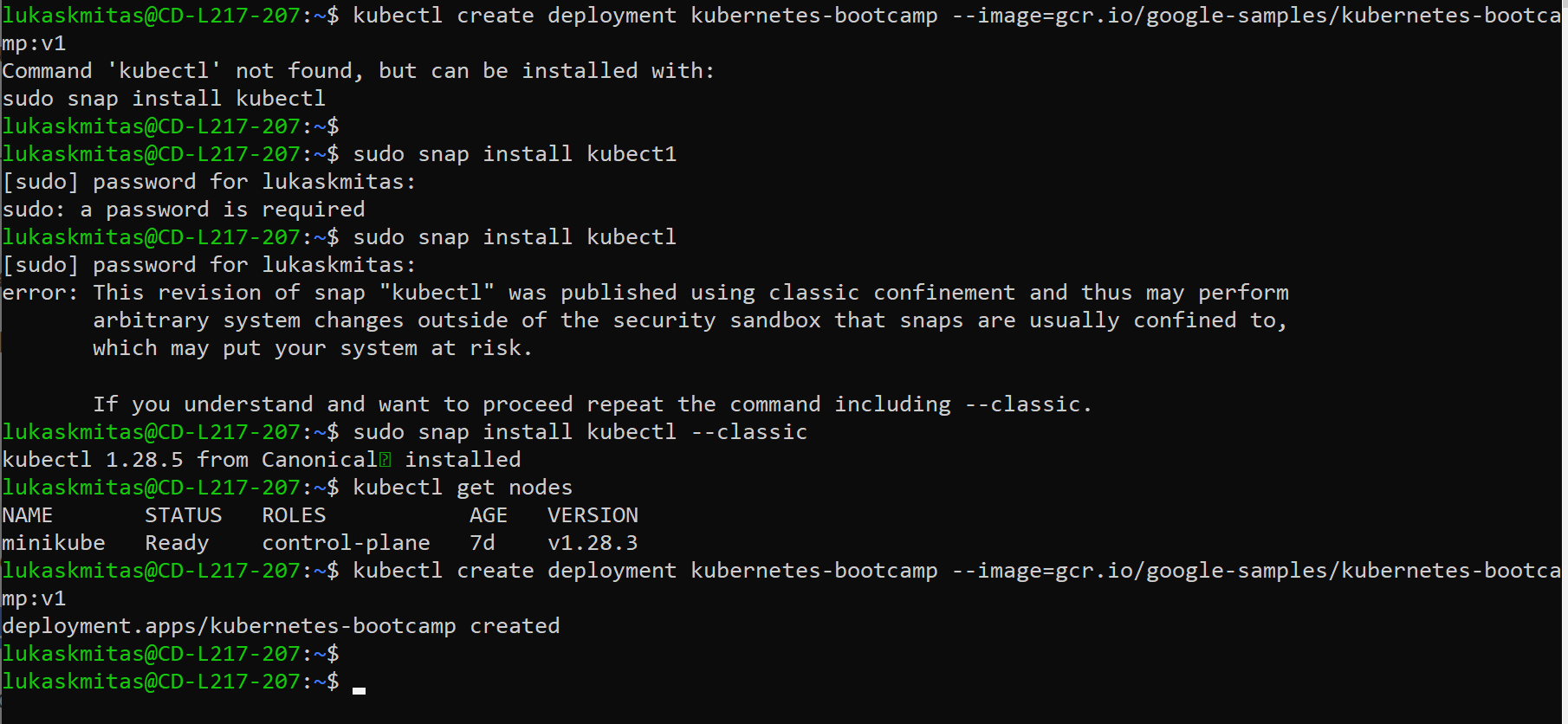
A black and grey logo

Description automatically generated



**minikube start** is a command used to launch a local Kubernetes cluster using Minikube. It initializes a virtual machine on your local system and configures Kubernetes components within it, enabling you to develop and test Kubernetes applications locally.

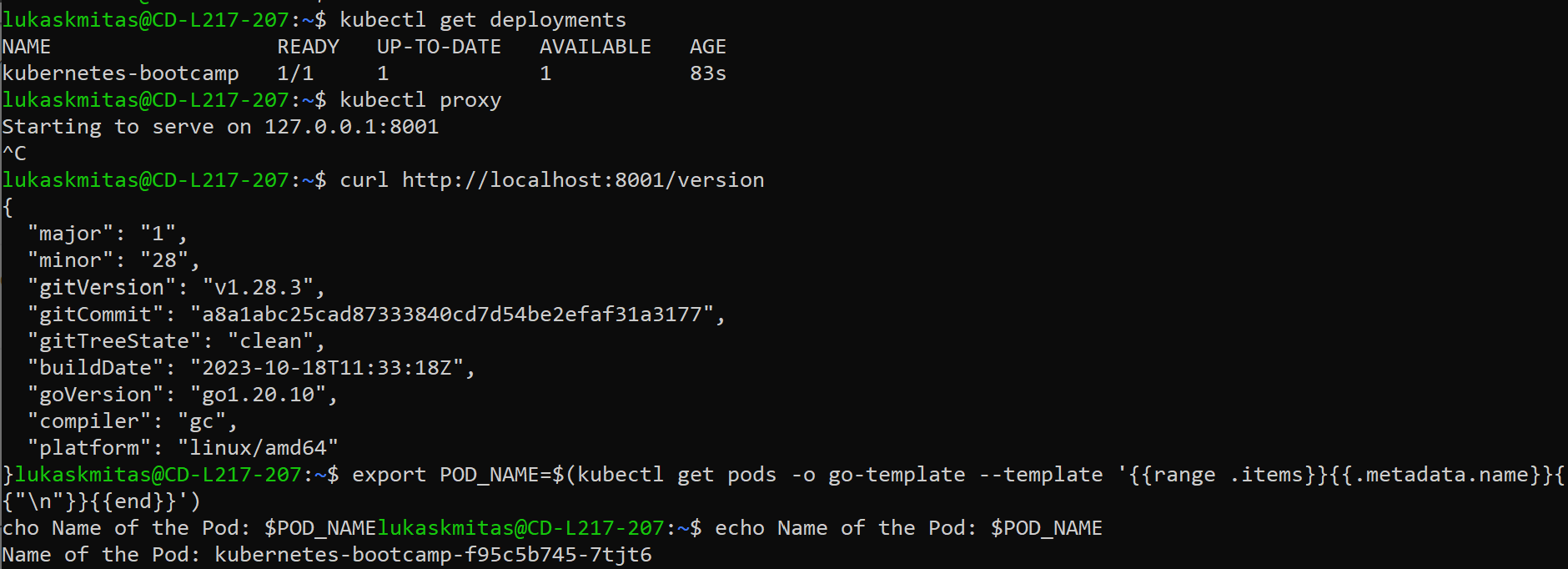
The command **minikube dashboard** is used to open the Kubernetes Dashboard for the Minikube cluster in a web browser. This dashboard provides a graphical user interface (GUI) for interacting with and managing the resources deployed in the local Kubernetes cluster. It offers features such as viewing cluster metrics, exploring deployed applications, and managing Kubernetes objects like pods, services, and deployments.



The command **kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1** is used to create a new deployment in a Kubernetes cluster.

* **kubectl**: This is the command-line interface for interacting with Kubernetes clusters.
* **create deployment**: This part of the command tells Kubernetes to create a new deployment.
* **kubernetes-bootcamp**: This is the name assigned to the deployment. You can replace it with any desired name for your deployment.
* **--image=gcr.io/google-samples/kubernetes-bootcamp:v1**: This flag specifies the Docker image to use for the deployment. In this case, it's pulling the image **gcr.io/google-samples/kubernetes-bootcamp** with the tag **v1** from the Google Container Registry (gcr.io). You can replace this image with any Docker image you want to deploy.

Overall, this command creates a deployment named **kubernetes-bootcamp** in the Kubernetes cluster, using the specified Docker image.



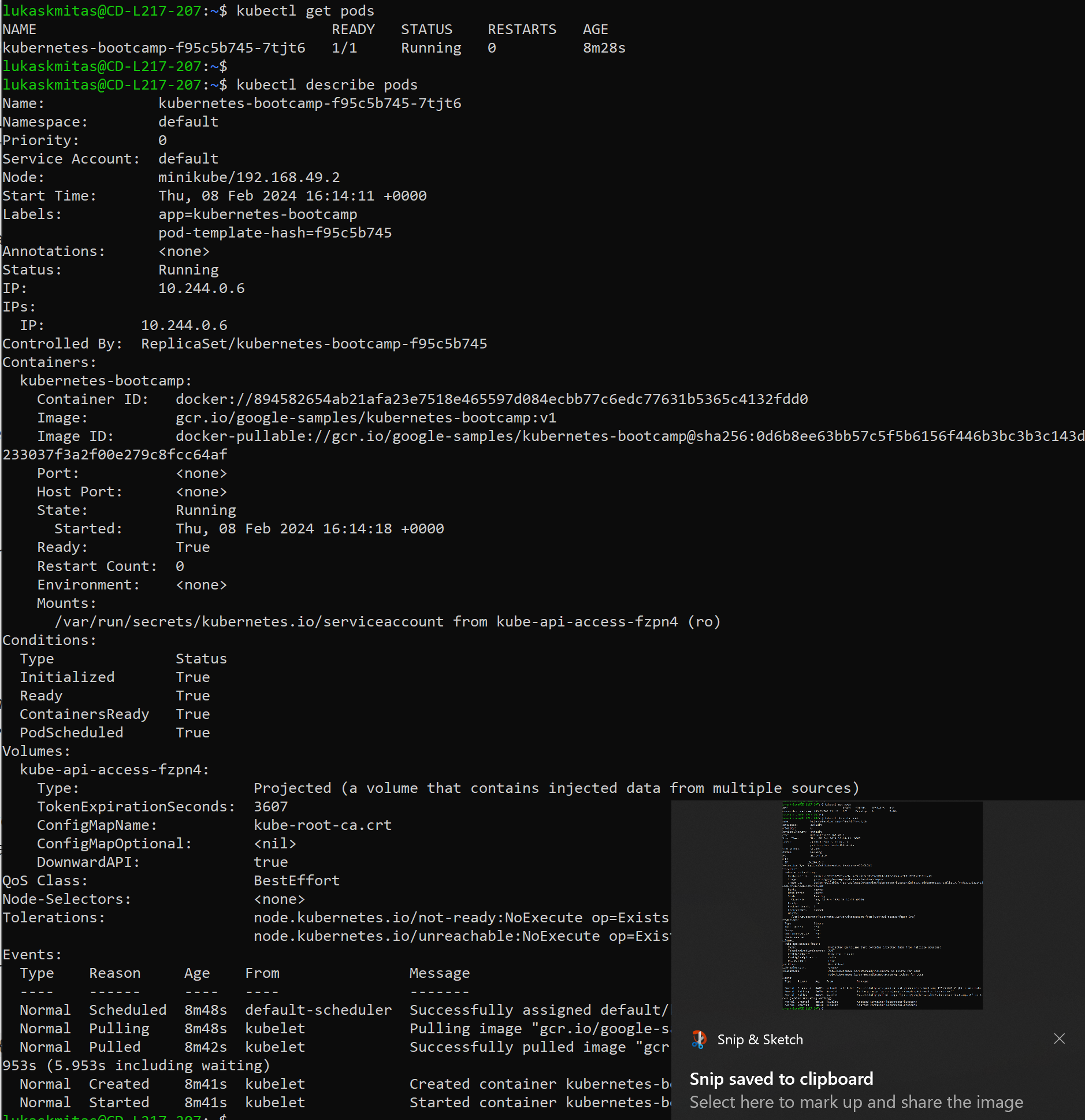
The command **kubectl get deployments** is used to retrieve information about the deployments running in the Kubernetes cluster.

* **kubectl**: The command-line interface for Kubernetes.
* **get deployments**: This part of the command specifies that you want to retrieve information about deployments.



The command - curl http://localhost:8001/api/v1/namespaces/default/pods/$POD\_NAME/

This command uses **curl**, a tool to transfer data from or to a server, to make a HTTP request to the Kubernetes API server running locally on port 8001. The URL targets the API endpoint for pods within the 'default' namespace. The **$POD\_NAME** variable should be replaced by the actual name of the pod you want to query.



The command “kubectl get pods” this command retrieves a list of all pods in the default namespace, displaying their status, restart count, and uptime.

Command: “kubectl describe pods” this command provides detailed information about a specific pod, showing configuration, status, and operational events. This includes IP addresses, the controlling ReplicaSet, container images used, and health status.

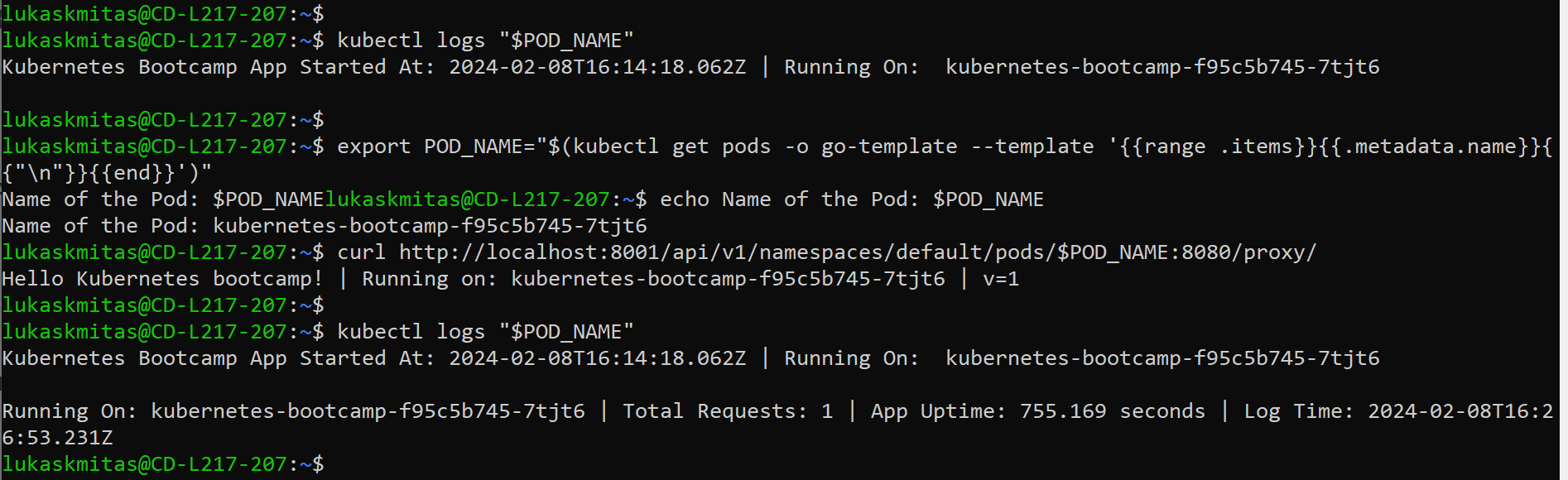
Key Details in the Output:

Pod Status: The pod named "kubernetes-bootcamp-f95cb5745-7tjt6" is running and has not restarted, indicating stable operation.

Container Details: It uses an image from Google’s container registry, specifying the exact version and repository.

Health and Readiness: The pod and its containers are reported as ready, meaning they are functioning correctly and able to handle requests.

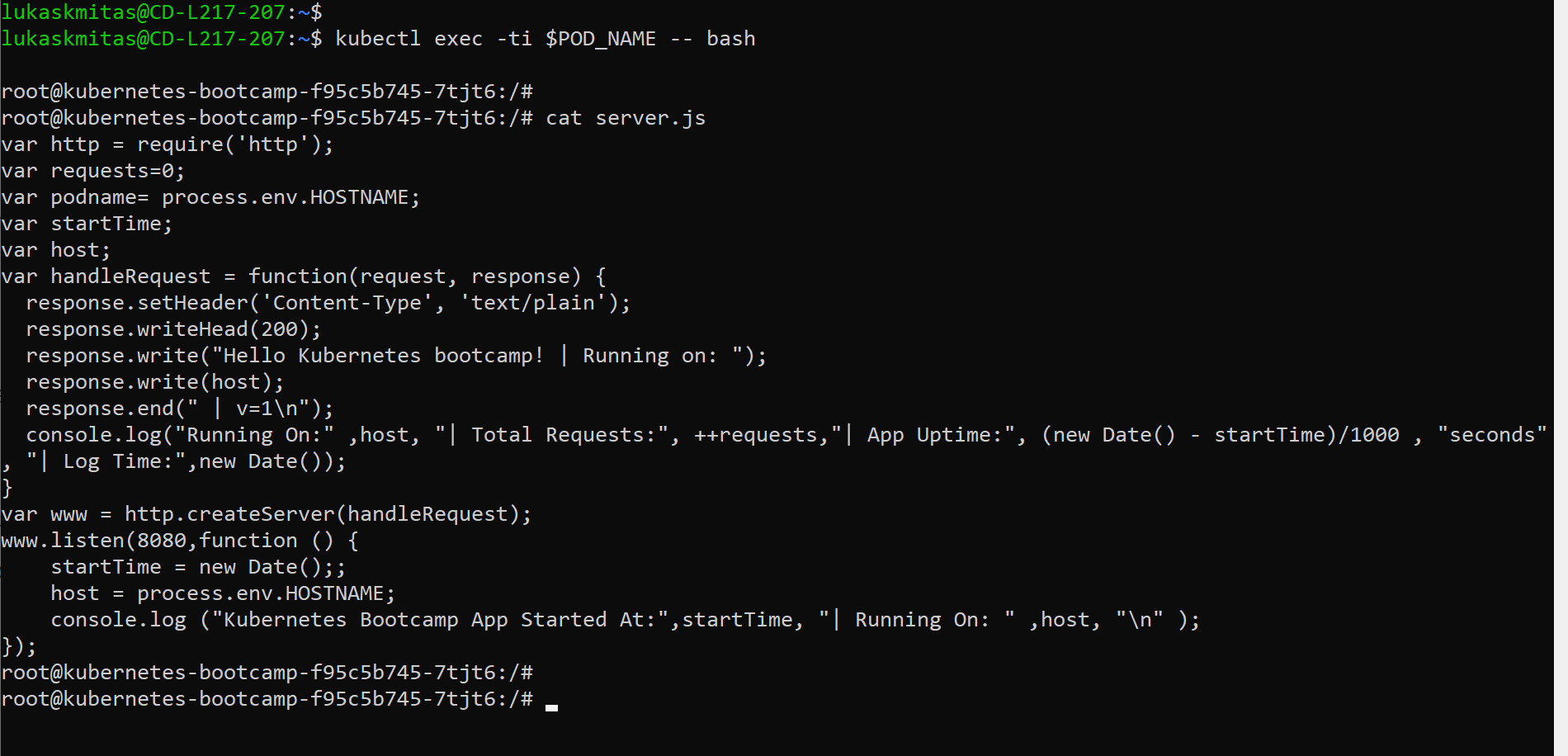
Events: These show the sequence of steps taken by Kubernetes to deploy the pod, such as scheduling, pulling the container image, and starting the container.



Retrieving and Setting Pod Name**: export POD\_NAME="$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')**"

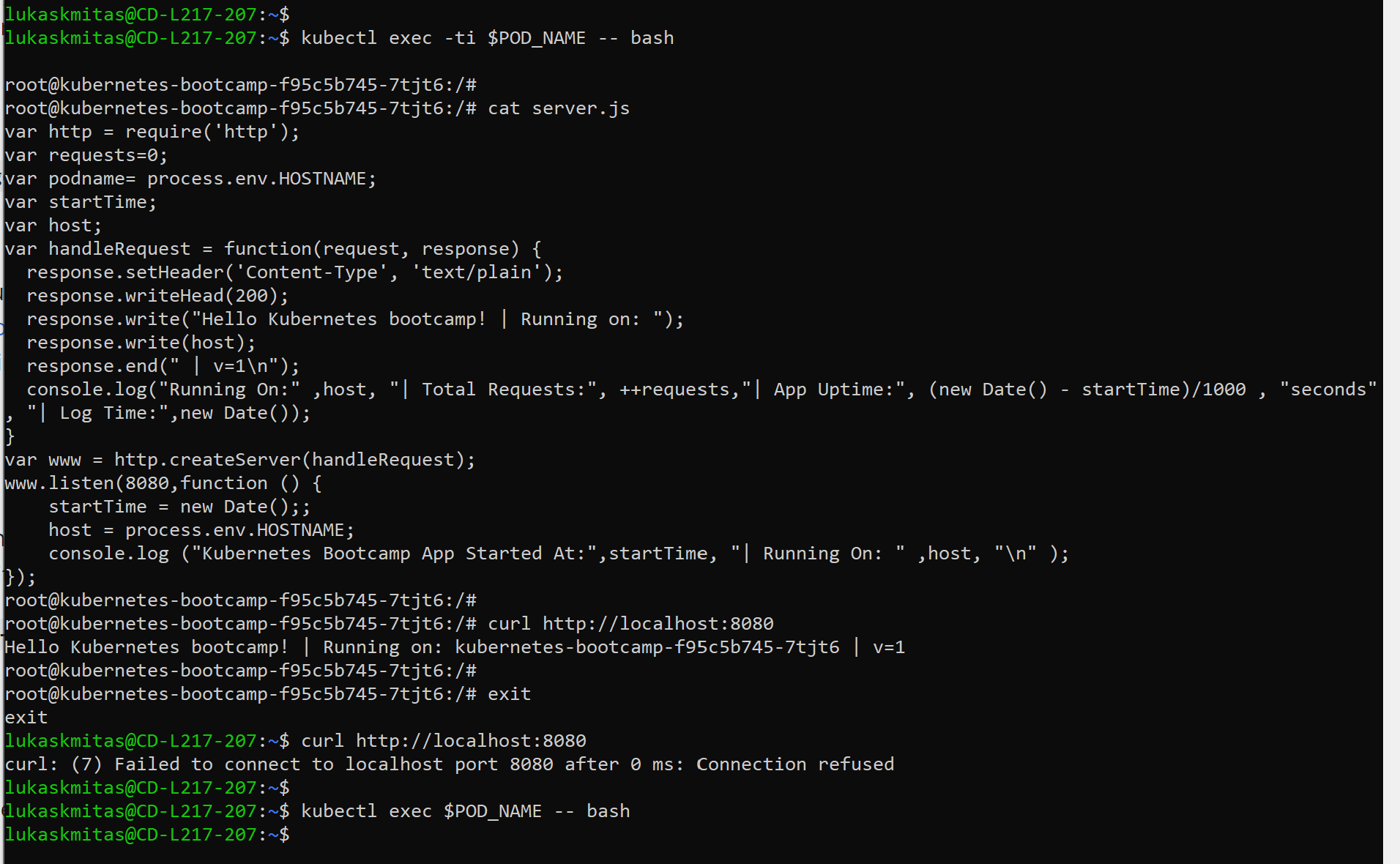
This command fetches the names of all pods in the current namespace, selects the first name, and assigns it to the environment variable POD\_NAME. The template formatting helps extract just the pod names from the output.

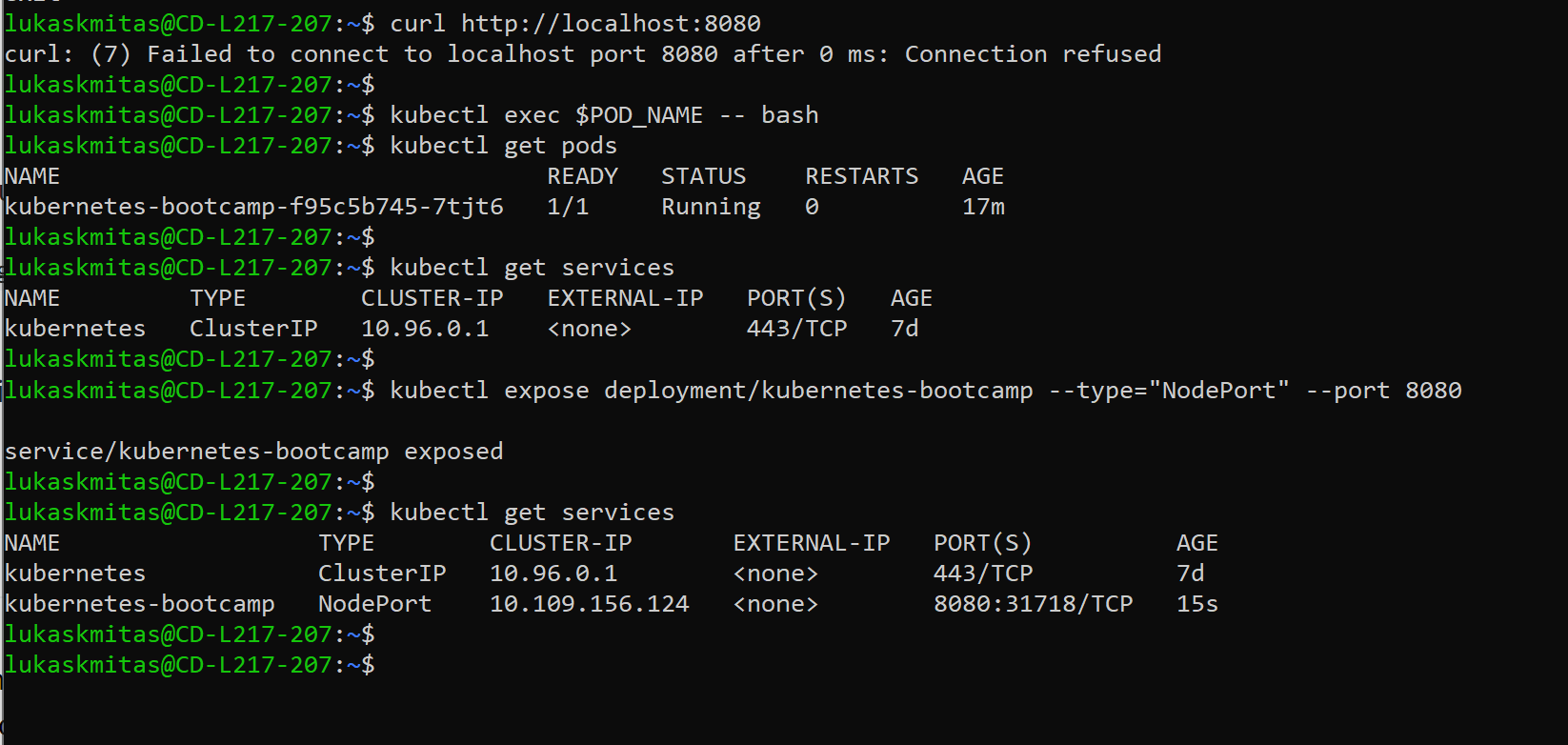
Displaying the Pod Name: “echo Name of the Pod: $POD\_NAME” This simply prints the name of the pod stored in POD\_NAME.



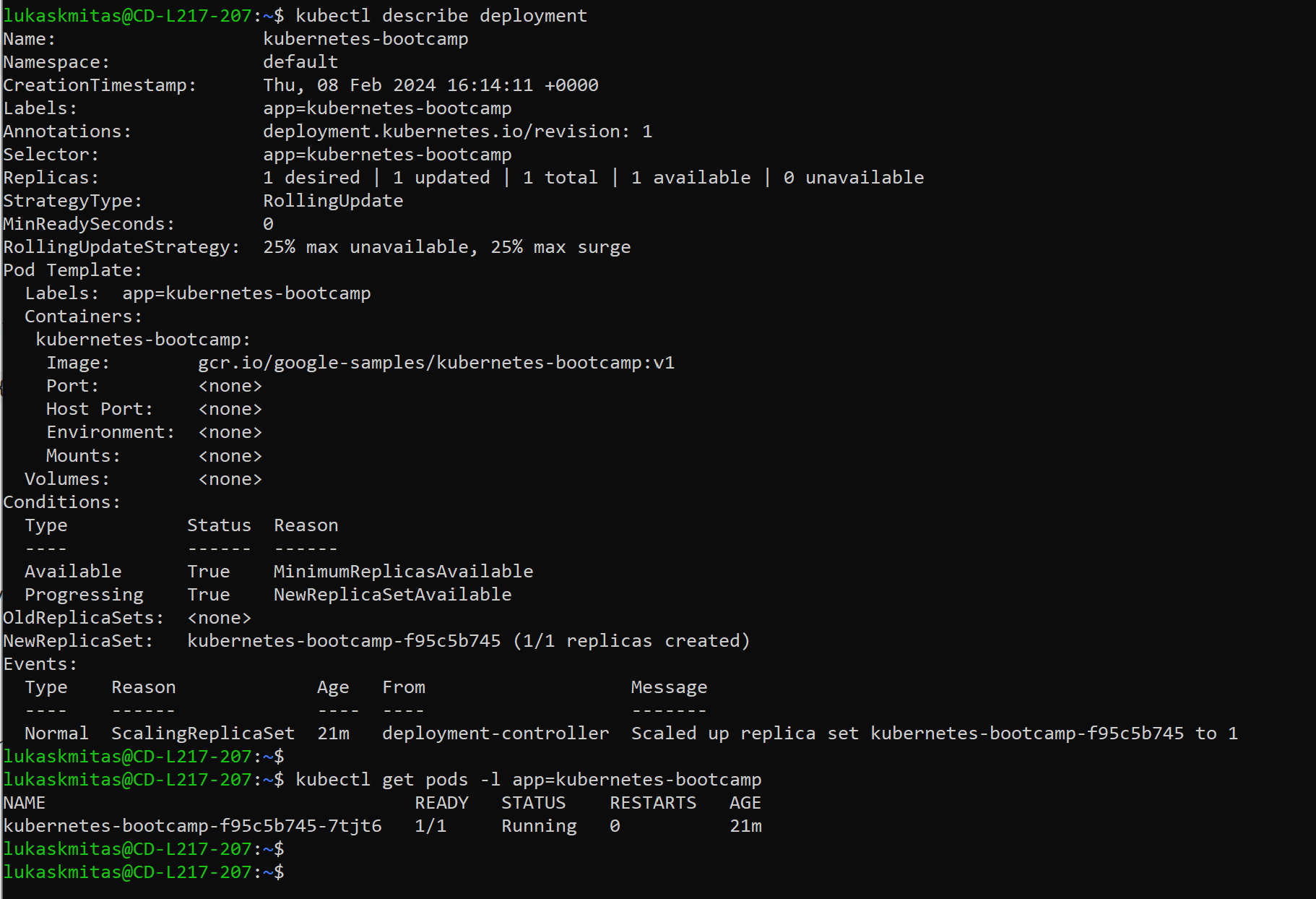
**cat**: This is a Unix/Linux command used to read and display the content of a file directly in the terminal.

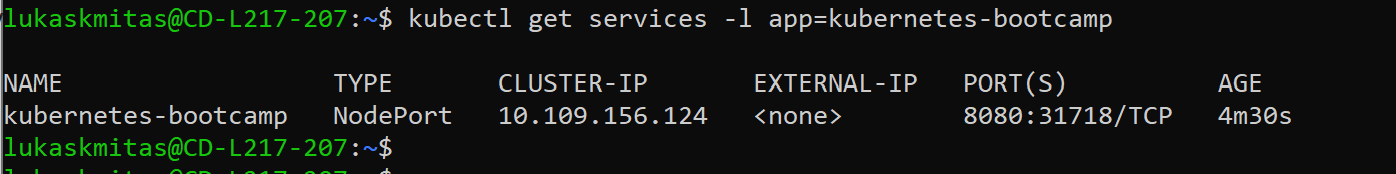
**server.js**: This is the name of the file being read by the cat command.



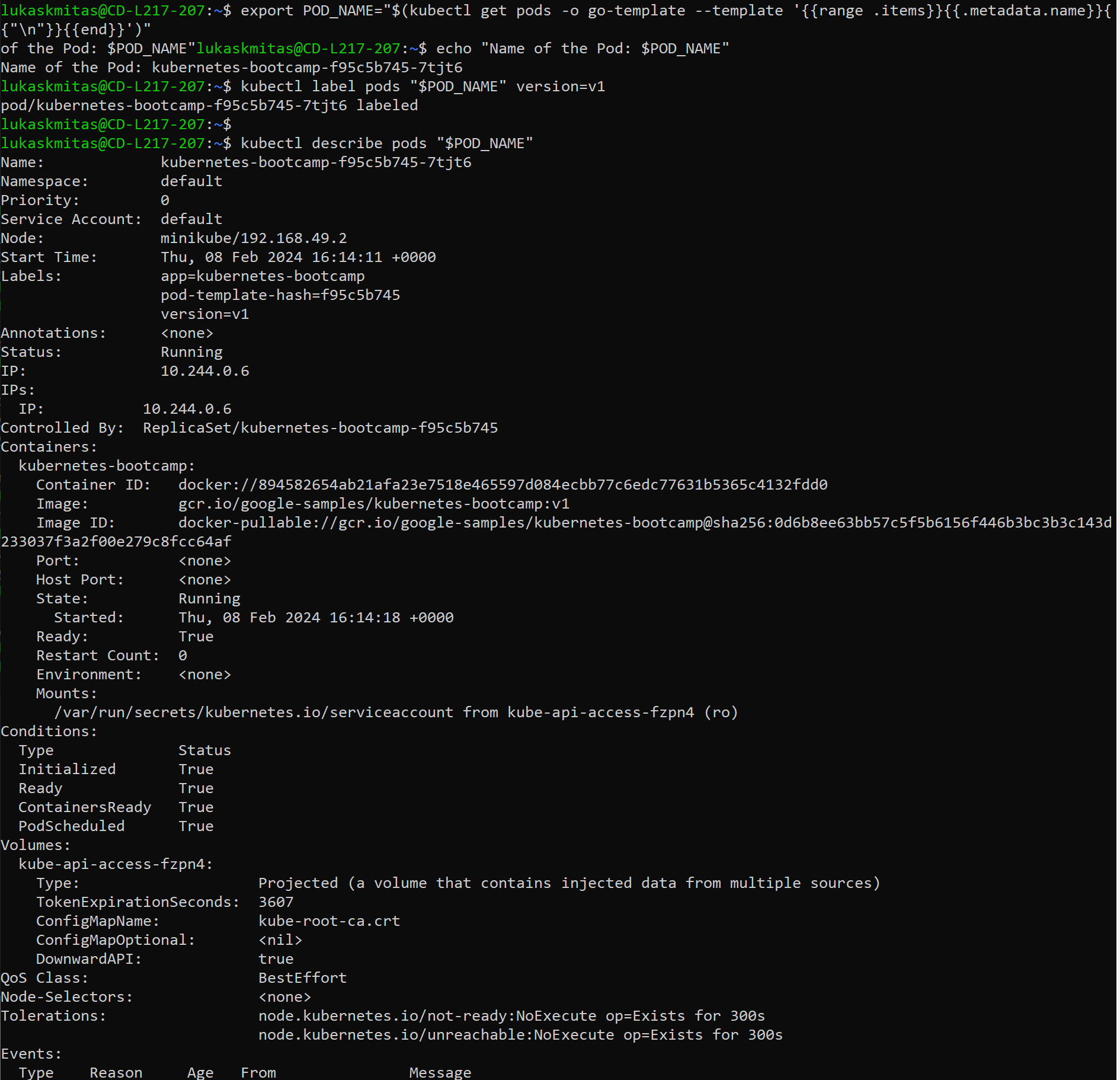


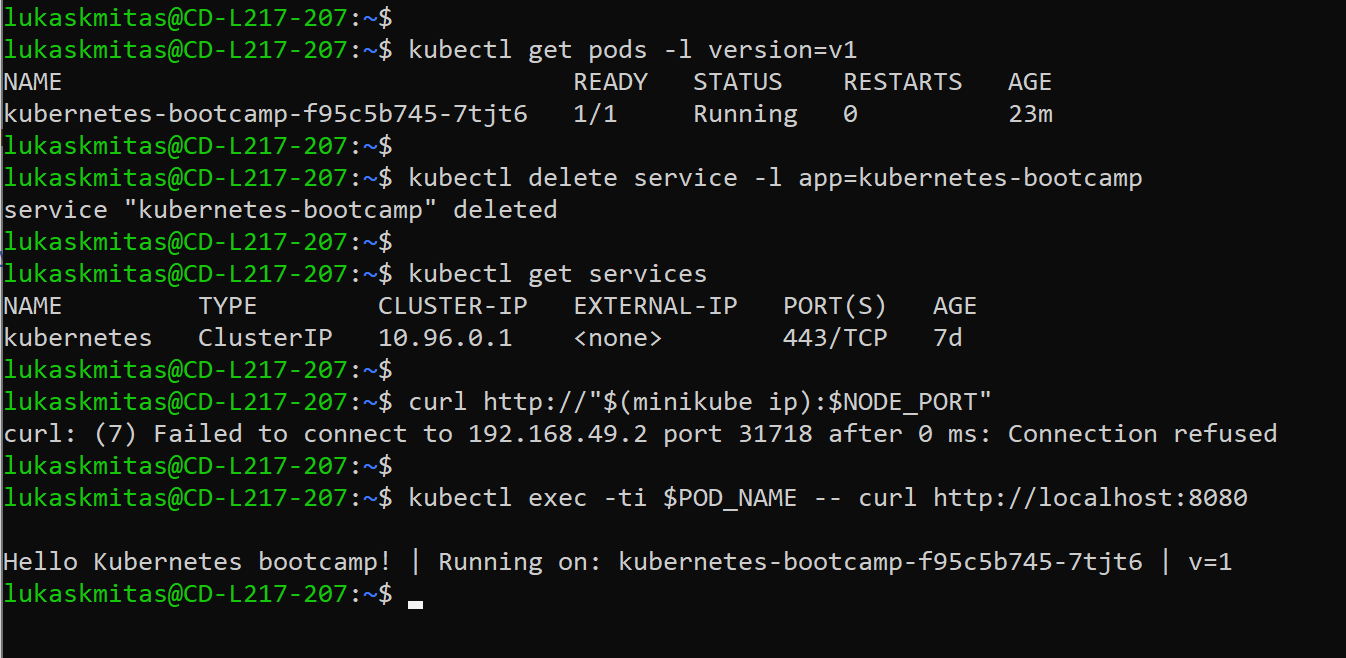
1. **Curl Command Failure**:
   * **curl http://localhost:8080**
   * This command attempts to access a service at port 8080 on the localhost. The connection is refused, indicating that nothing is currently serving requests on this port locally.
2. **Kubectl Commands**:
   * **kubectl exec $POD\_NAME -- bash**
     + This command opens a bash shell inside the specified pod, referenced by the variable **$POD\_NAME**, allowing for direct interaction with the pod's environment.
   * **kubectl get pods.**
     + Retrieves a list of all pods, showing their status and other details like readiness, restarts, and age.
   * **kubectl get services.**
     + Initially, this retrieves a list of all services, showing that only the default Kubernetes service is exposed with no external IP and serving on port 443/TCP.
3. **Exposing the Pod as a Service**:
   * **kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080**
     + This command creates a new service by exposing the deployment named "kubernetes-bootcamp". It specifies the service type as **NodePort**, which makes the service accessible on a static port on each node’s IP. The service forwards traffic to port 8080 on the pods.
4. **Verifying Service Exposure**:
   * **kubectl get services**
     + After exposing the deployment, this command is run again to show the newly created service named "kubernetes-bootcamp". It lists the internal cluster IP and the node port (**31718**) mapped to port **8080** on the container. This port mapping allows external access to the service.

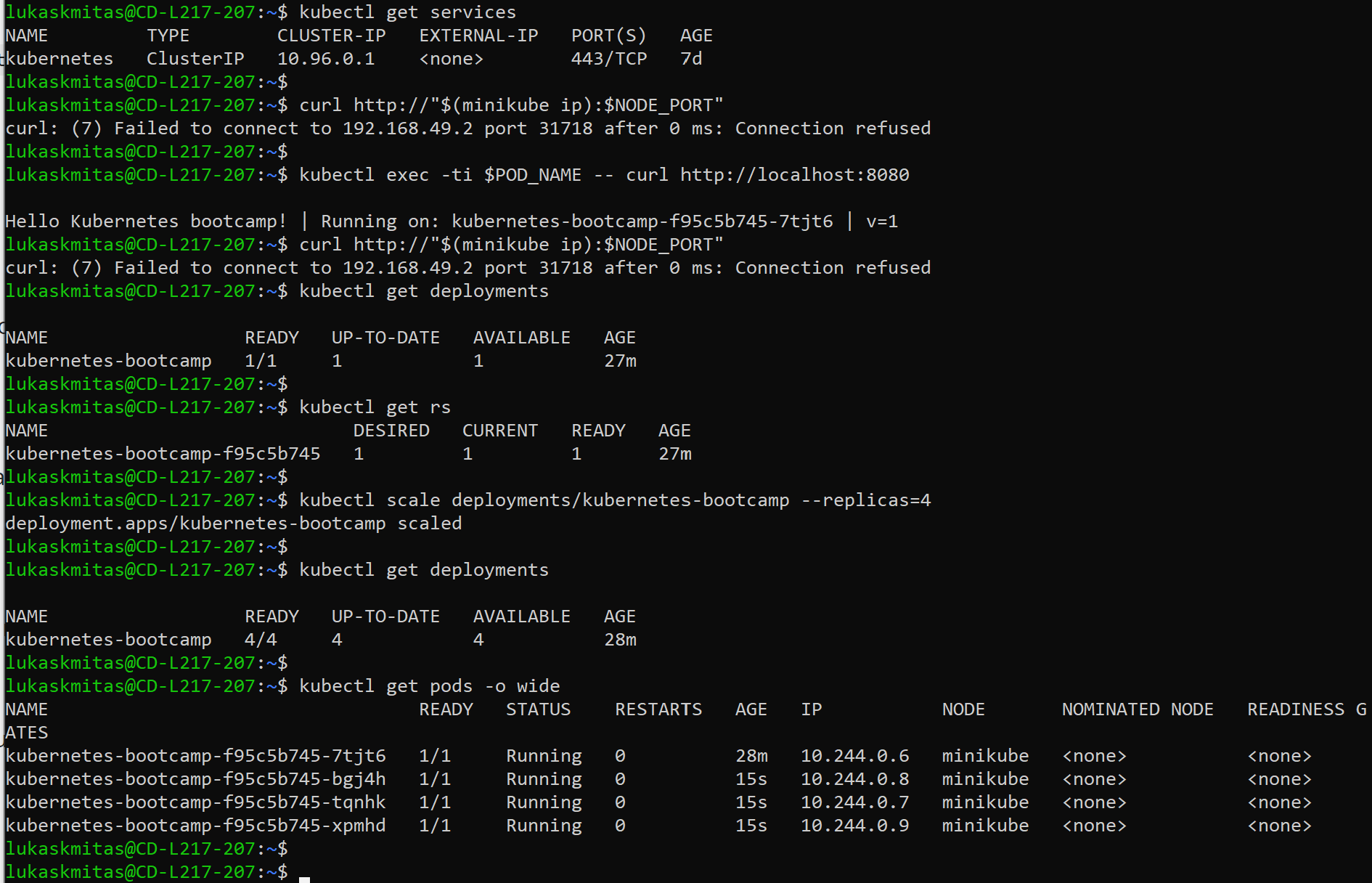


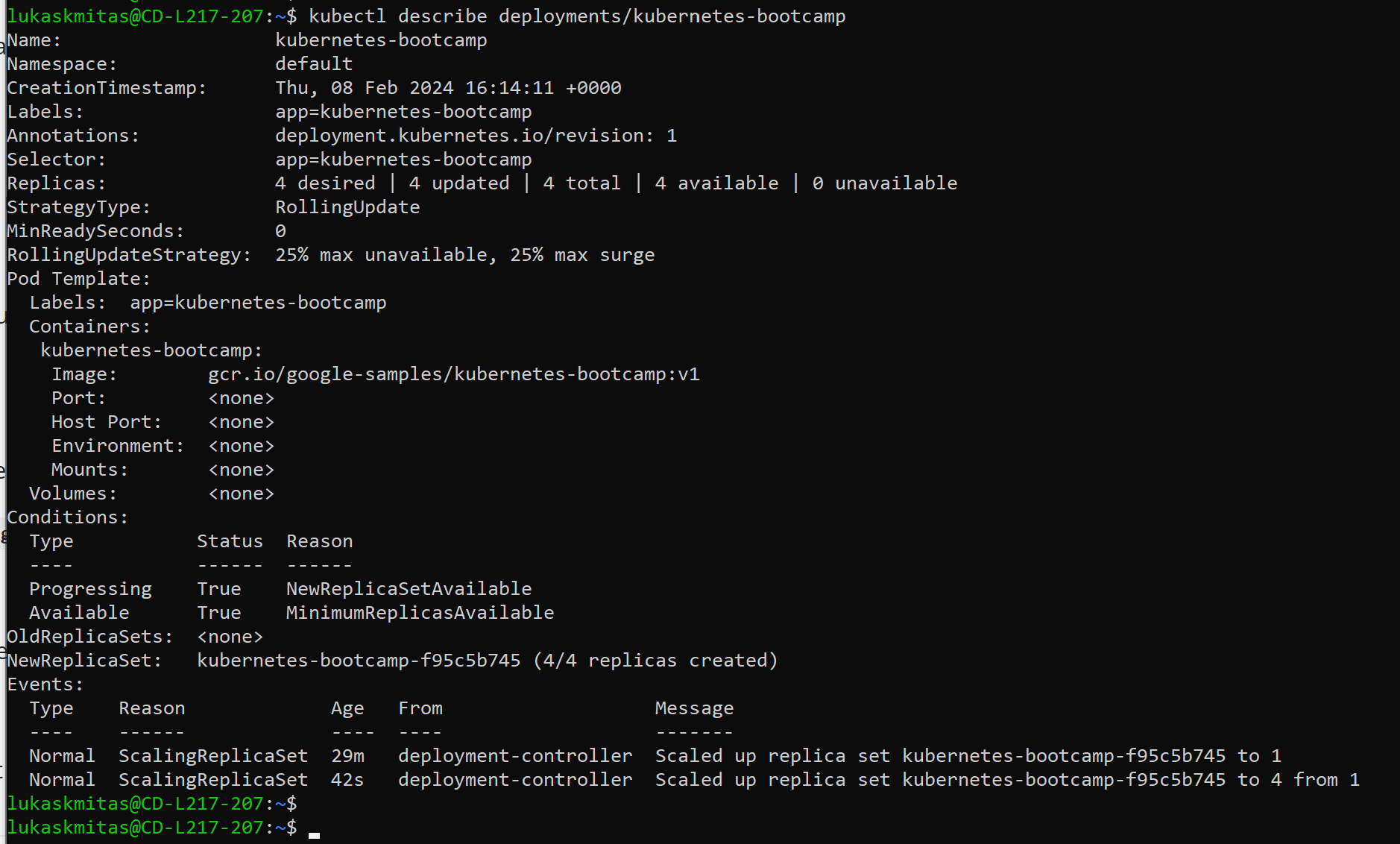


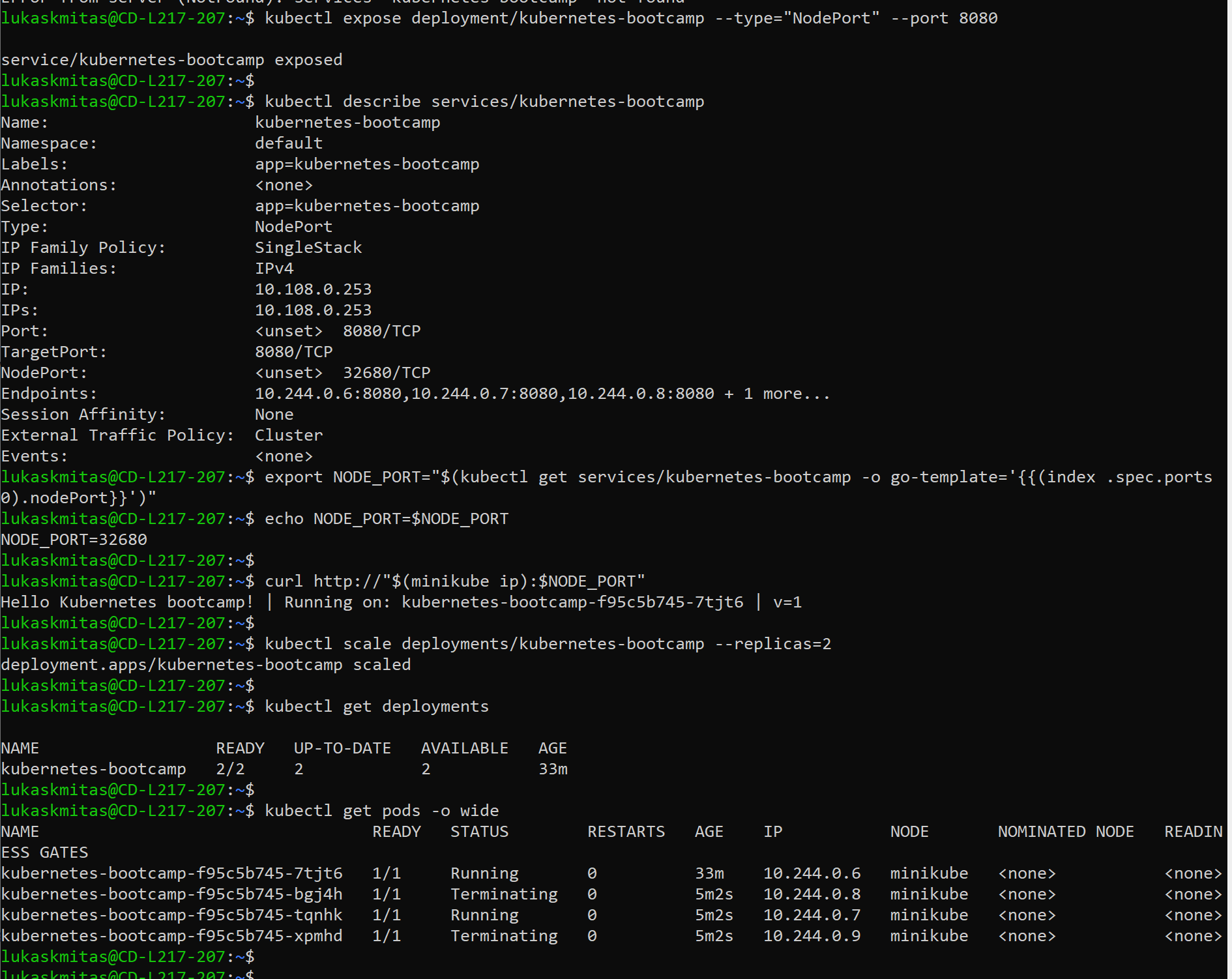
* **kubectl get services**: This command fetches and lists services managed by Kubernetes. Services are Kubernetes resources that abstract access to a set of pods, typically providing a network service.
* **-l app=kubernetes-bootcamp**: This is a label selector option. It filters the services to only show those with a label **app** that matches the value **kubernetes-bootcamp**.

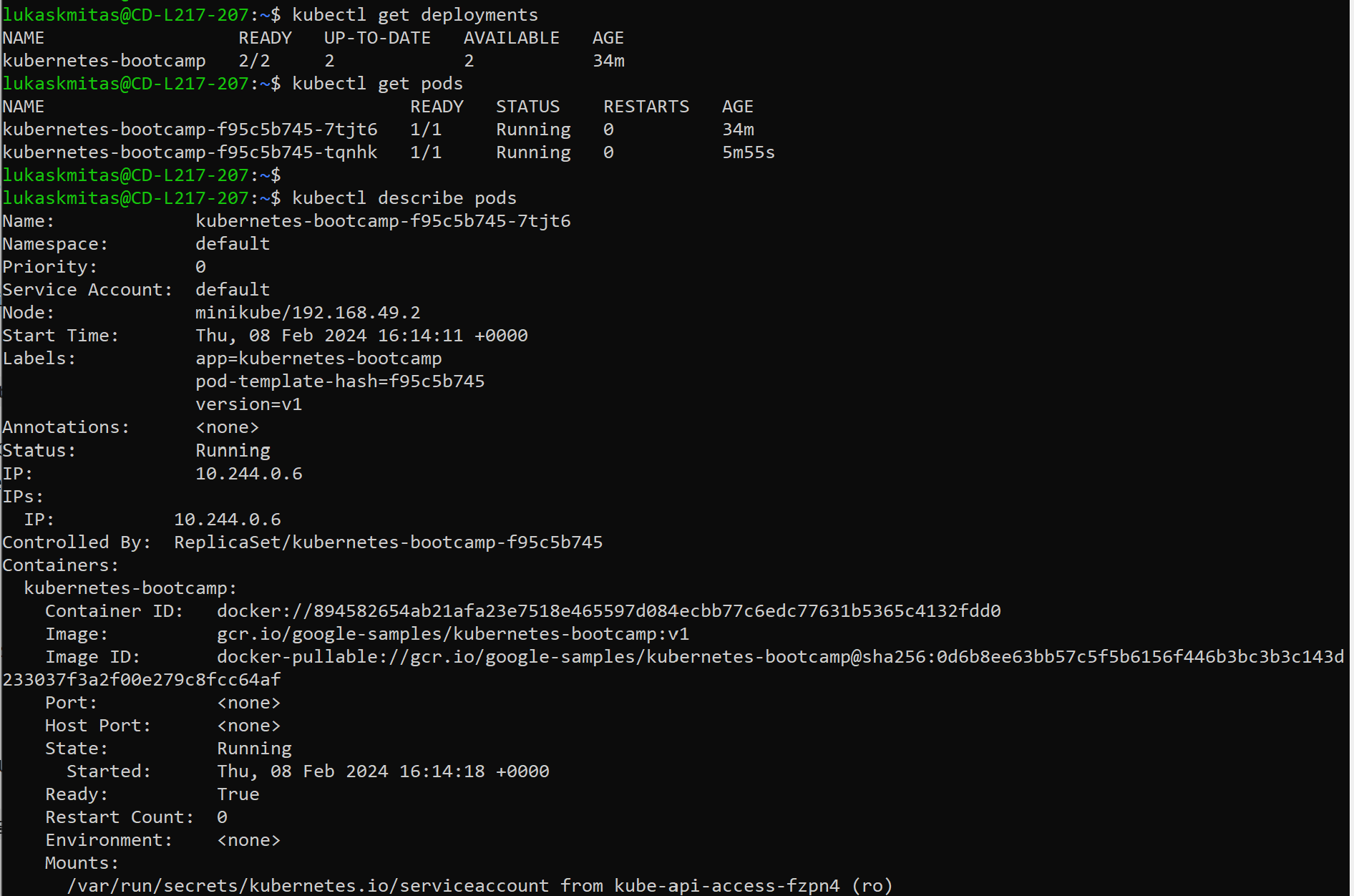


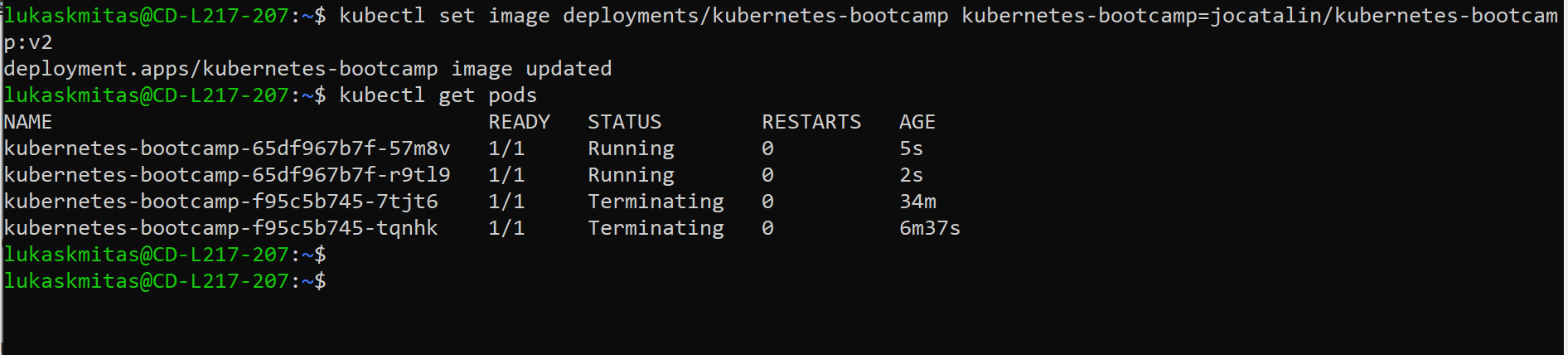




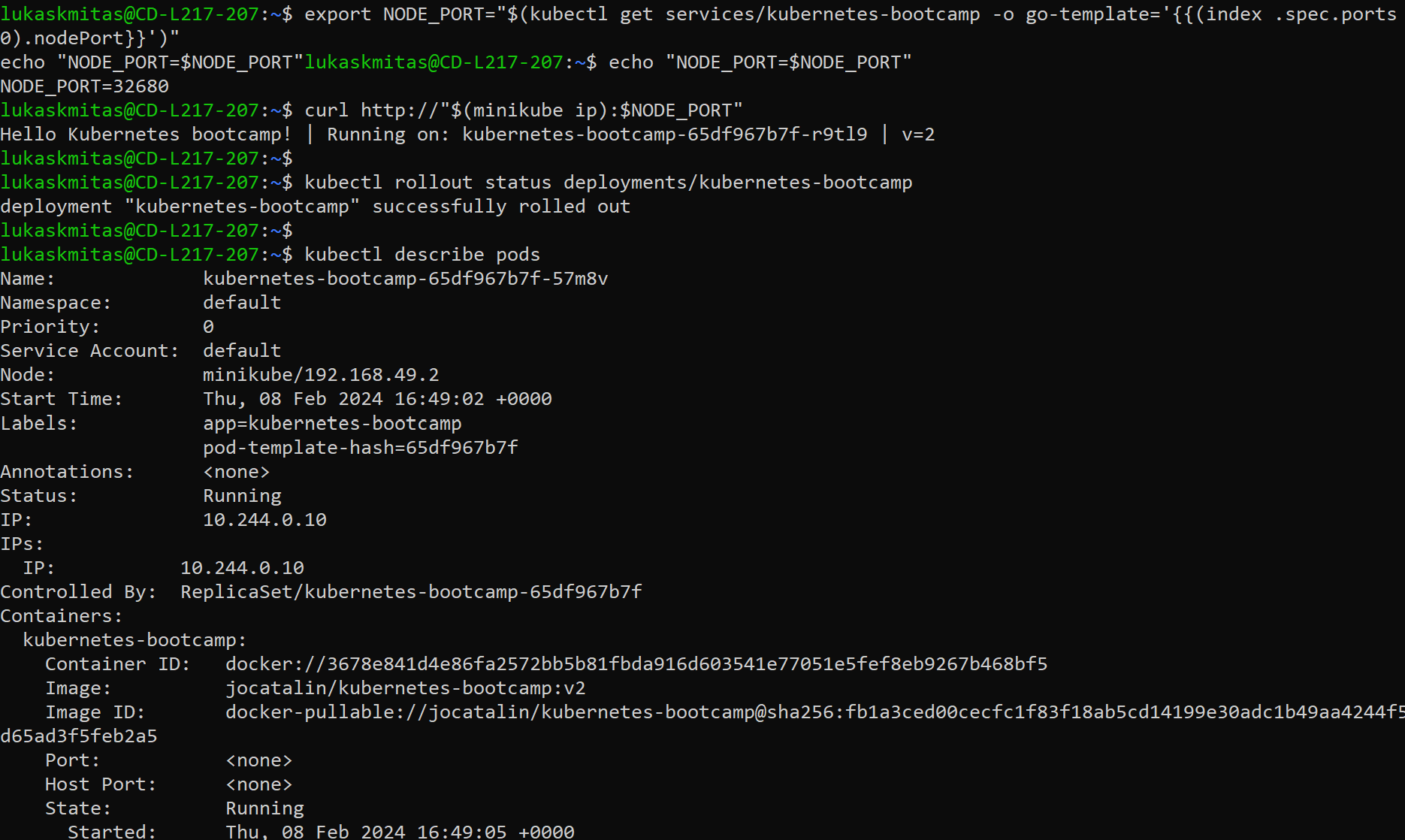


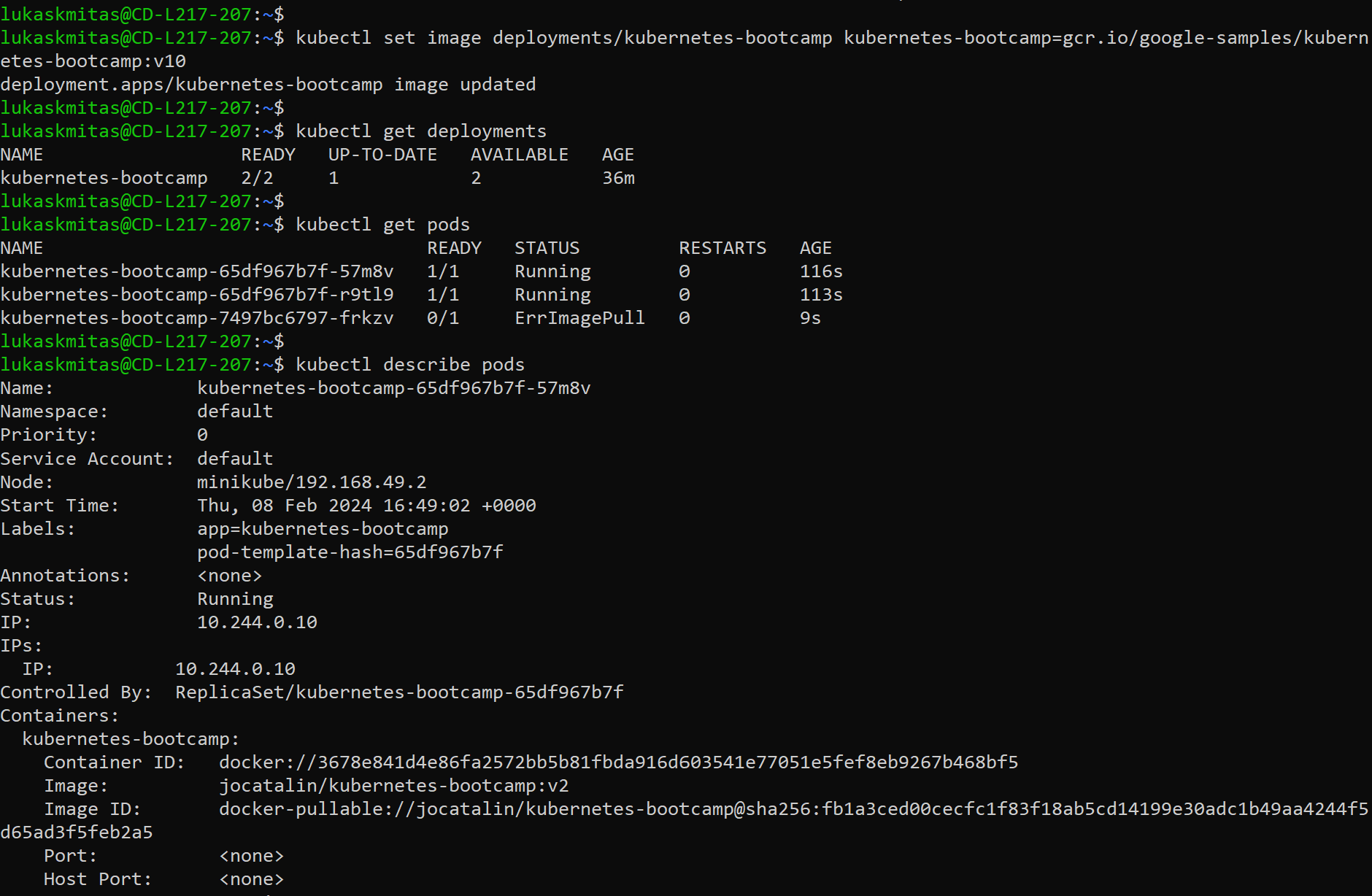


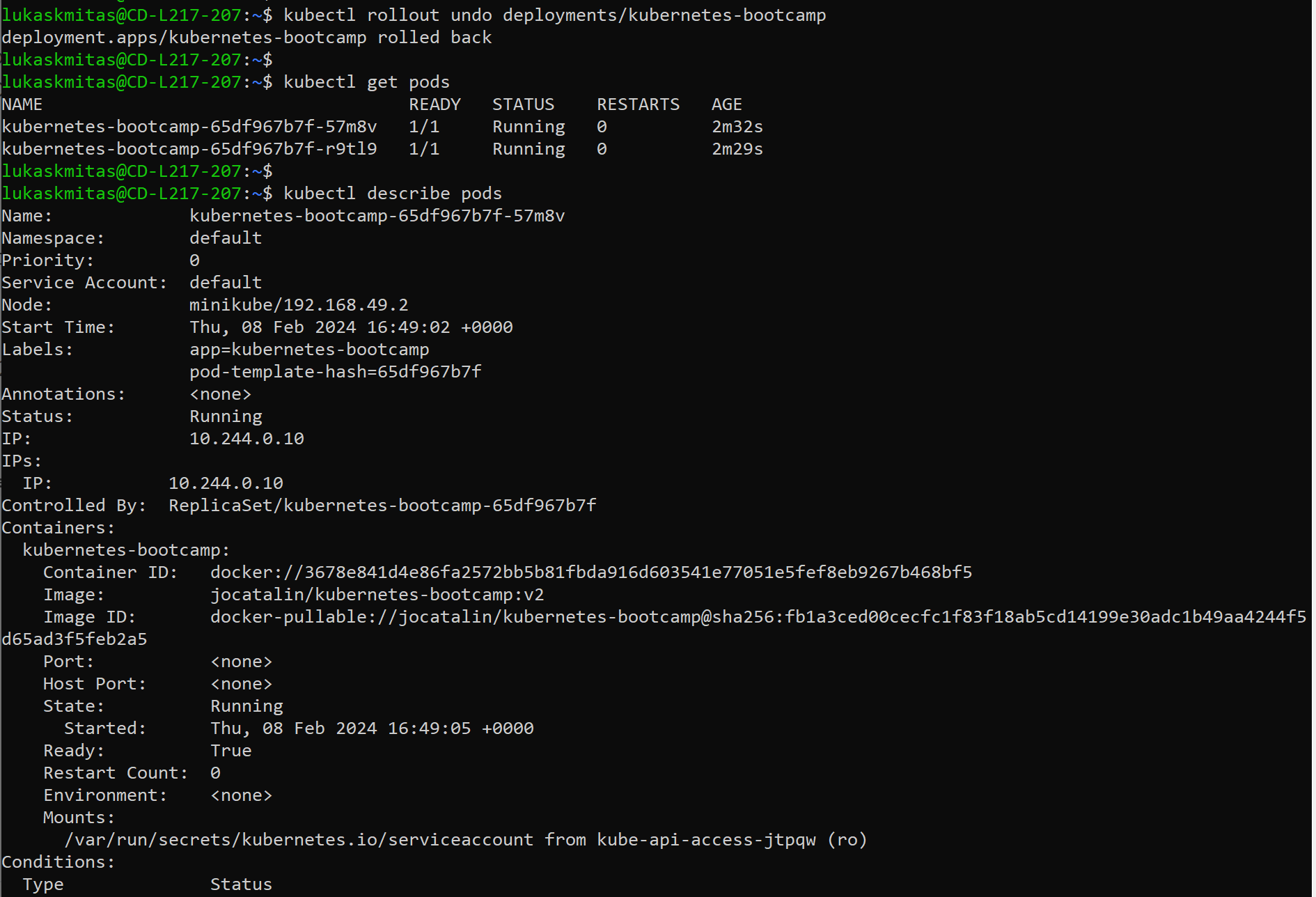




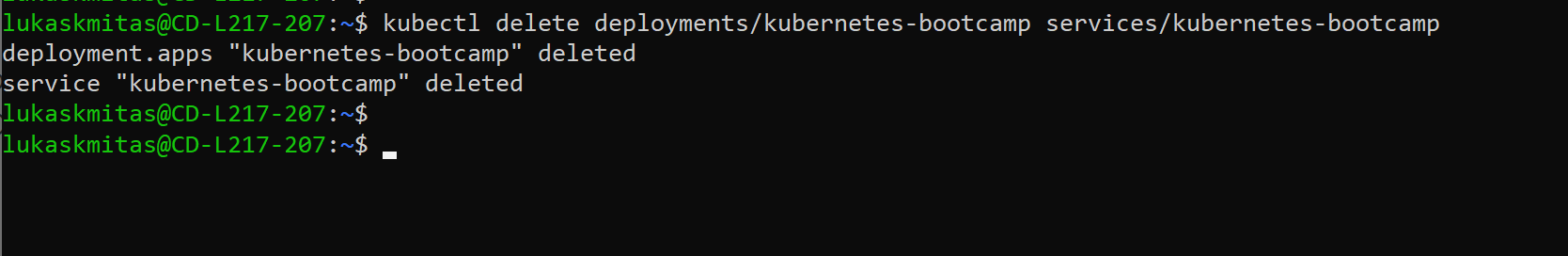
1. **Updating the Deployment Image**:
   * **kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2**
   * This command updates the container image used in the deployment named **kubernetes-bootcamp**. Specifically, it sets the container named **kubernetes-bootcamp** to use the image **jocatalin/kubernetes-bootcamp:v2**.
   * The response "deployment.apps/kubernetes-bootcamp image updated" confirms that the image has been successfully set to the new version.
2. **Listing the Pods**:
   * **kubectl get pods**
   * This command retrieves a list of all pods currently managed by the Kubernetes cluster, providing details about each pod's readiness, status, restarts, and age.
   * From the output:
     + Two new pods (**5d5967bf7-57m8v** and **5d5967bf7-r9t19**) are in the **Running** state with an age of 5 seconds and 2 seconds, respectively, indicating they are recently started as part of the rolling update triggered by the image change.
     + One old pod (**f95cb5745-7tjt6**) is in the **Terminating** state, being shut down as part of the update process.
     + Another old pod (**f95cb5745-tqnhk**) is also terminating, as shown by its status.







1. **Undo Deployment Update**:
   * **kubectl rollout undo deployments/kubernetes-bootcamp**
   * This command reverses the most recent update to the deployment **kubernetes-bootcamp**, reverting it to the previous version. It is useful for quickly undoing changes if they cause issues.
2. **List Pods**:
   * **kubectl get pods**
   * The command lists all pods, showing their readiness, status, restarts, and age. This helps verify the state of pods after an update or rollback.
3. **Describe Specific Pod**:
   * **kubectl describe pods kubernetes-bootcamp-65d9f67bf7-57m8v**
   * This command provides detailed information about a specific pod, including its configuration and current state. This is particularly useful for debugging and understanding the pod's environment and operational characteristics.



* **deployments/kubernetes-bootcamp**: This specifies that the Kubernetes deployment named "kubernetes-bootcamp" should be deleted. A deployment in Kubernetes manages a set of replicas of a container, which allows for updates and scaling.
* **services/kubernetes-bootcamp**: This part of the command specifies that the Kubernetes service named "kubernetes-bootcamp" should also be deleted. A service in Kubernetes is an abstraction which defines a logical set of pods and a policy by which to access them, often used to expose containers to the internet or other parts of the cluster.