# 

[**Kubernetes Hands On Lab**](#_7m4e2v45li7n) **1**

[Docker Desktop Installation](#_r3ggud8yvld0) 1

[Prerequisites](#_foot0ee4iean) 1

[Running your first container](#_nt80my2jux0u) 2

[Minikube](#_mz6ylqf51yxw) 3

[Running your first pod](#_y9b1gav45yud) 3

[Deploying - Nginx Service](#_w6zjky6uc1ug) 3

[Updating - Nginx Service](#_3fzdt74reody) 5

[Scaling - Nginx Service](#_9625g2uqv60x) 6

[Deleting the Service](#_pmj0i5t23ux6) 7

[Deploying WordPress and MySQL with Persistent Volumes](#_eb23lxp1eap5) 8

[Kompose](#_magx5a5mqfy1) 13

[Blue/Green Deployments](#_3l0oyppofhk3) 15

# Kubernetes Hands On Lab

## Docker Desktop Installation

### Prerequisites

Windows

* Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later)
* Hyper-V and Containers Windows features must be enabled
* 4GB system RAM
* BIOS-level hardware virtualization support must be enabled in the BIOS settings

Mac

* Mac hardware must be a 2010 or newer model
* macOS must be version 10.12 or newer. We recommend upgrading to the latest version of macOS

Installation

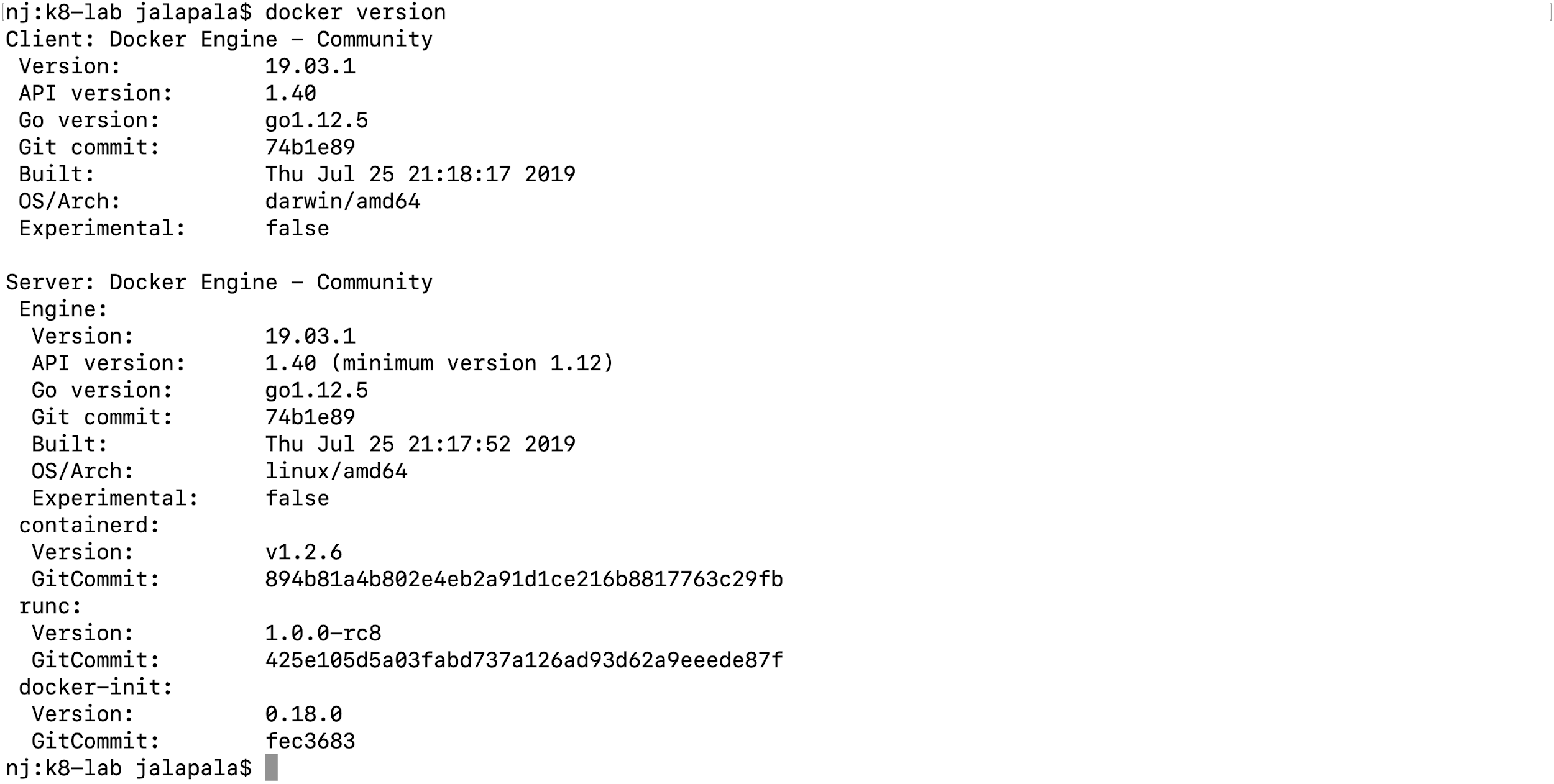
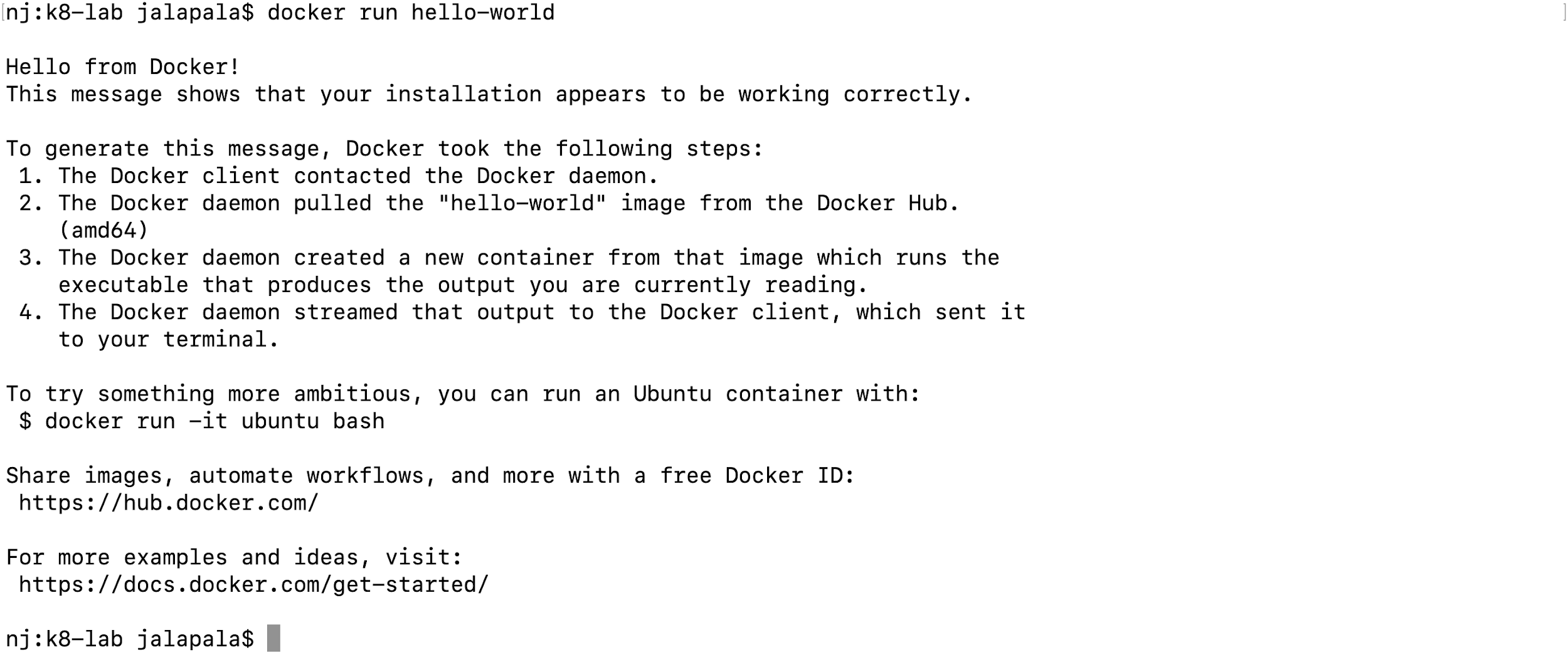
* Windows docker desktop download

<https://download.docker.com/win/stable/Docker%20Desktop%20Installer.exe>

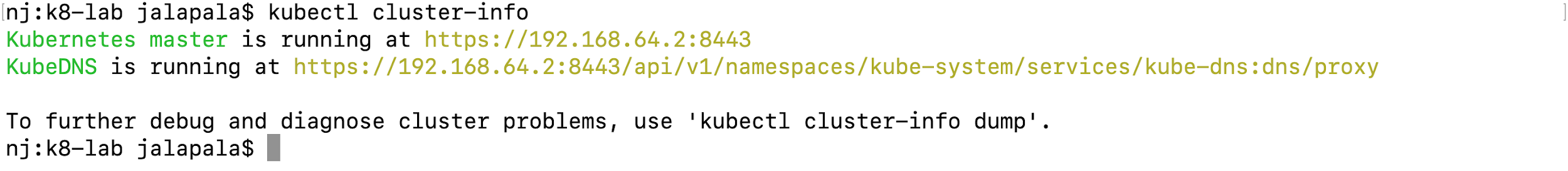
* Mac docker desktop download

<https://download.docker.com/mac/stable/Docker.dmg>

## Running your first container

* Once the script execution is completed open the powershell/bash terminal and un docker version to check installed docker version  
    
  
* Run docker run hello-world to spin up the first container  
    
  

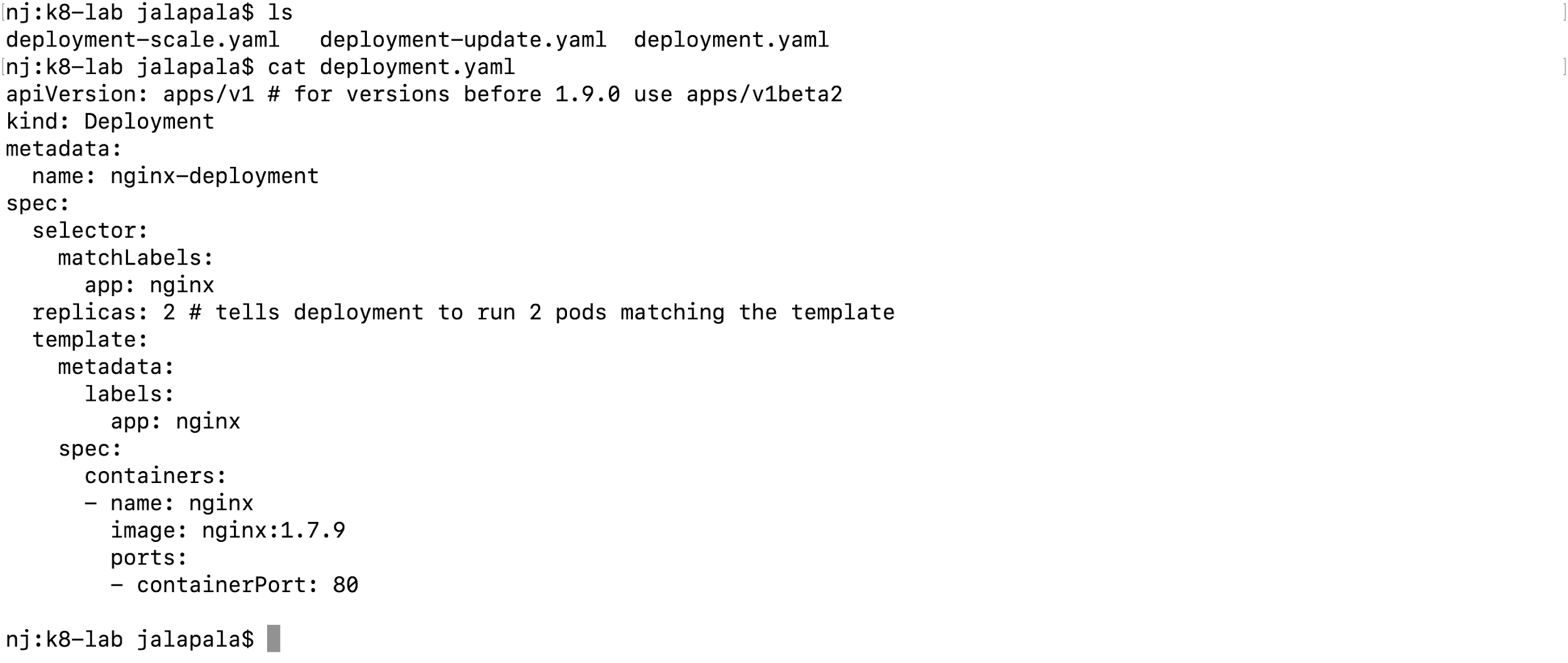
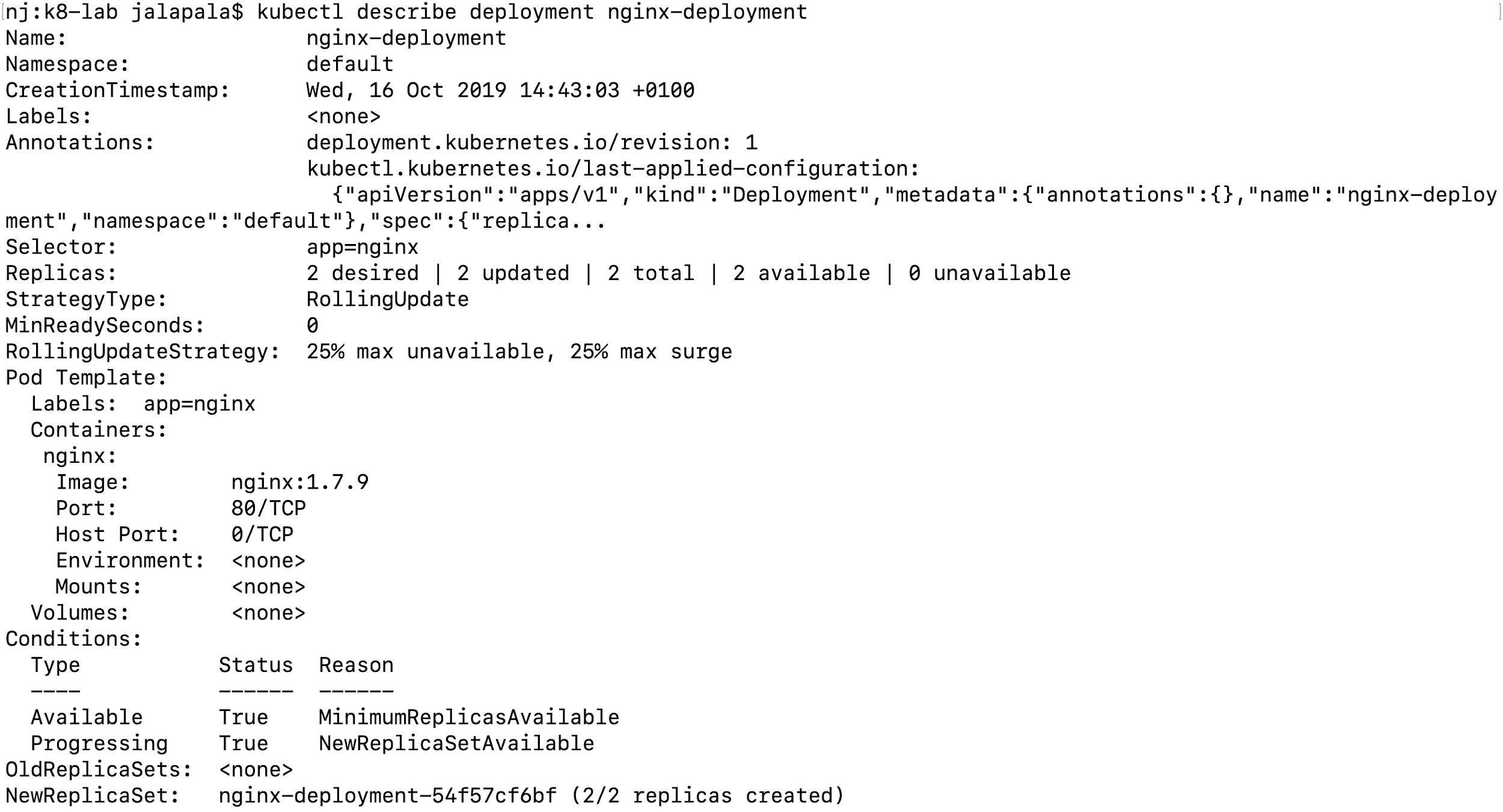
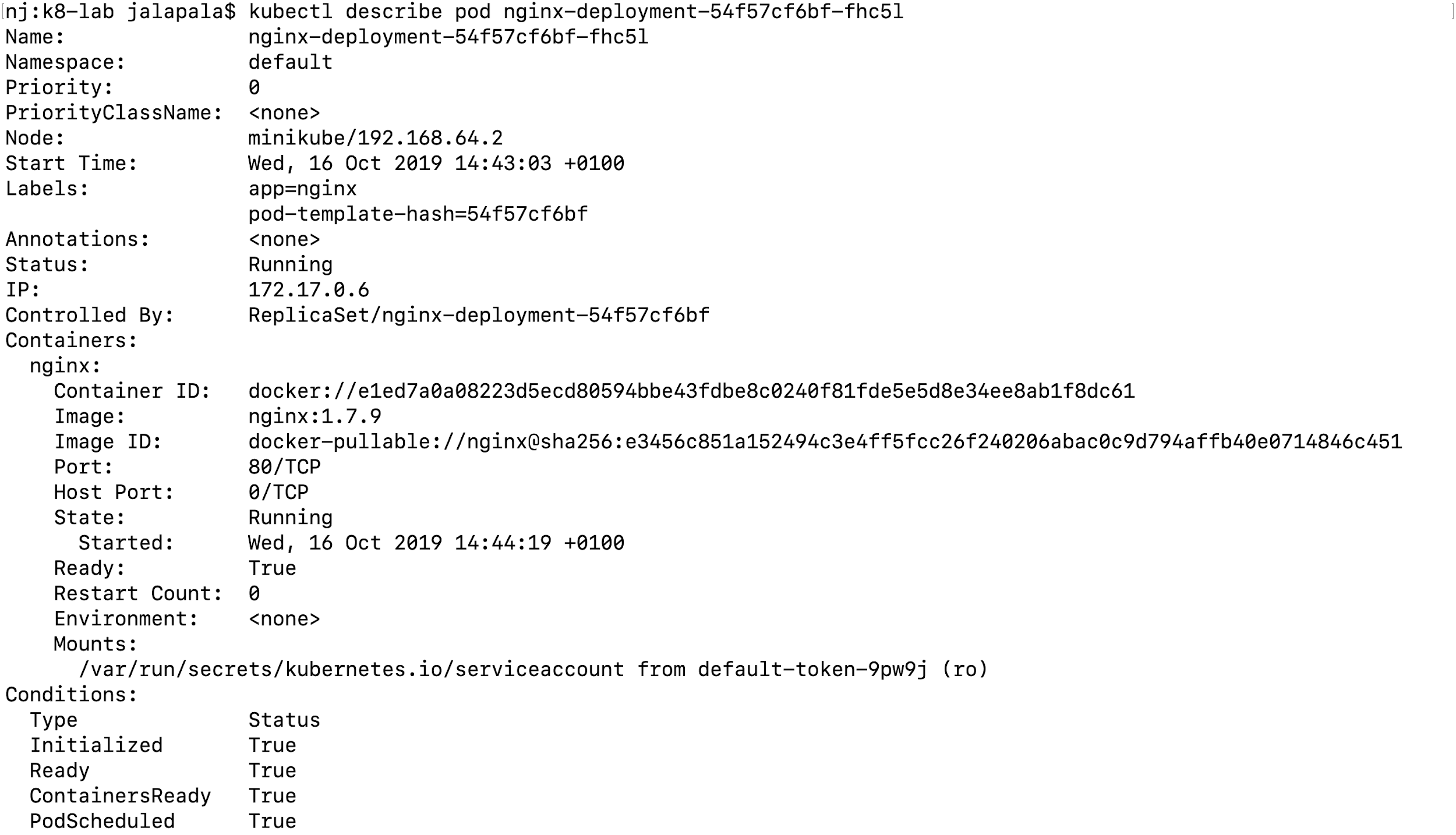
## Minikube

* Run kubectl get nodes to check connectivity to the Kubernetes cluster  
    
  
* Run kubectl cluster-info to get some information about the Minikube  
    
  

## Running your first pod

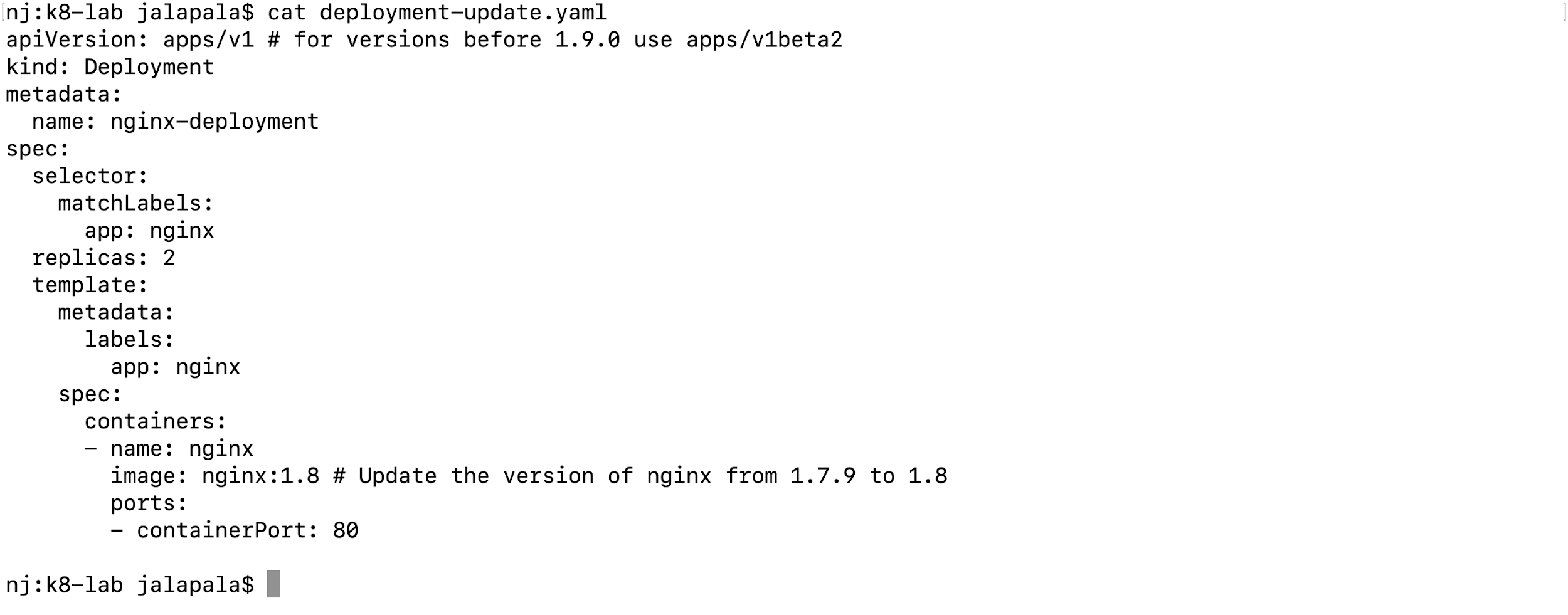
* Run kubectl create deployment hello-node --image=gcr.io/helloto command to create a Deployment that manages a Pod. The Pod runs a Container based on the provided Docker image.

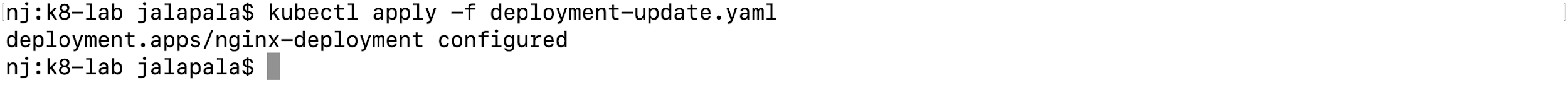
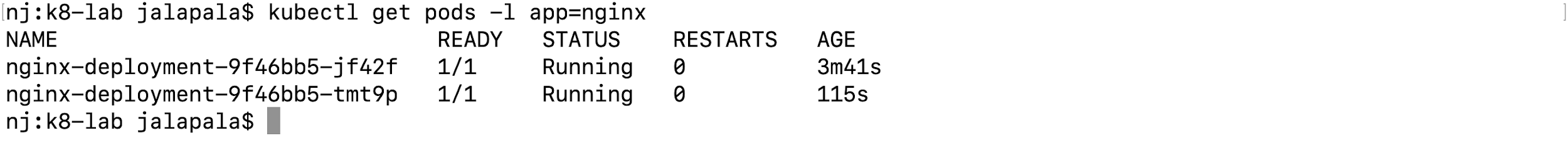
## Deploying - Nginx Service

* In the Lab1 folder the file “deployment.yaml” describes nginx deployment that runs on the nginx Docker image  
    
  
* Run kubectl apply -f deployment.yaml to create a deployment based on the YAML file:  
    
  
* Run kubectl describe deployment nginx-deployment to display information about the Deployment:  
    
  
* Run kubectl get pods -l app=nginx to list the pods created by the deployment:  
    
  
* Run kubectl describe pod nginx-deployment-54f57cf6bf-fhc5l to display information about a pod:  
    
  

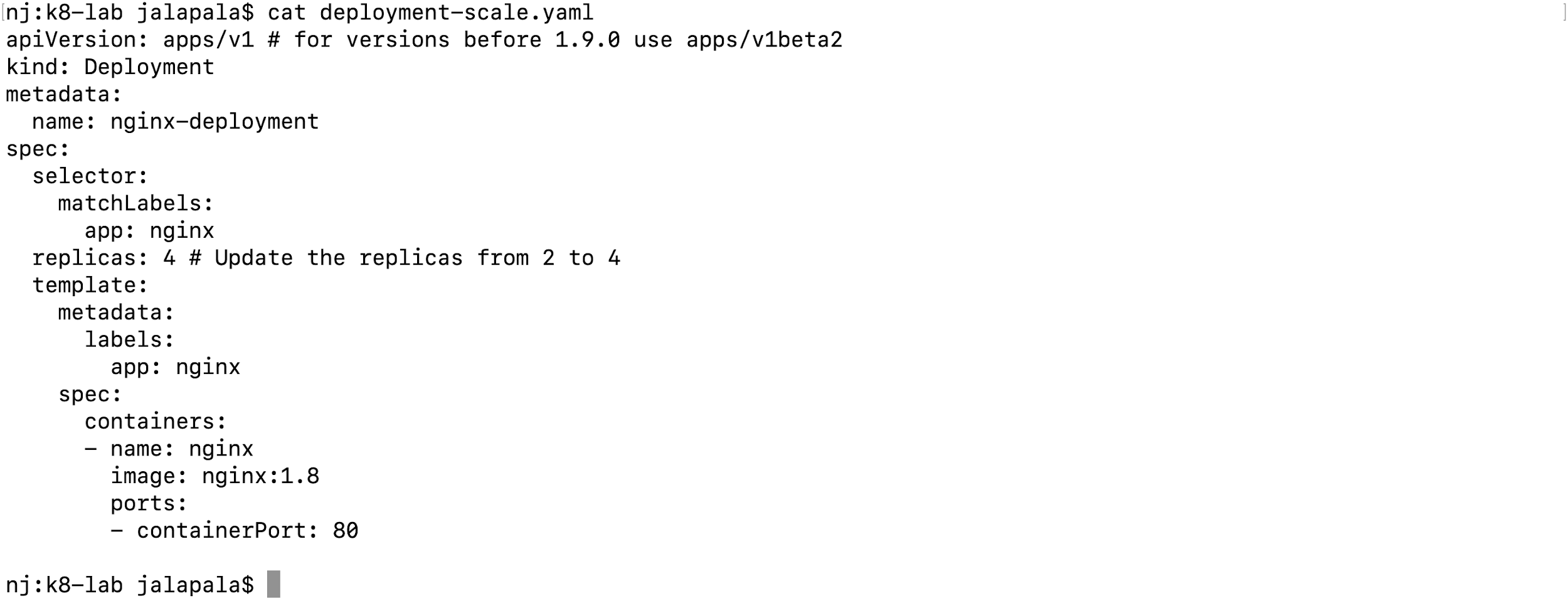
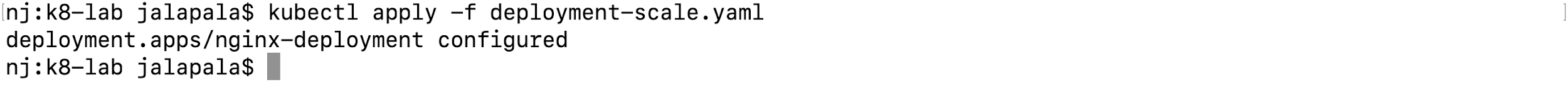
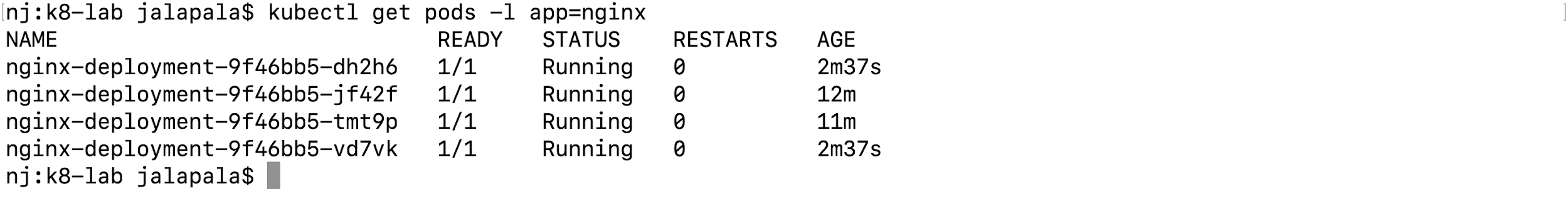
## Updating - Nginx Service

* The file “deployment-update.yaml” specifies that the deployment should be updated to use nginx 1.8

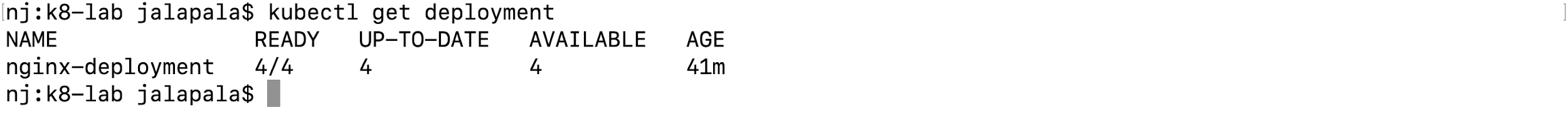
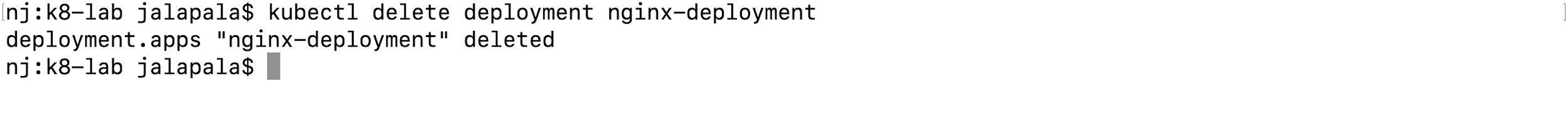


* Run kubectl apply -f deployment-update.yaml to update the deployment  
    
  
* Run kubectl get pods -l app=nginx to watch the deployment create pods with new names and delete the old pods:  
    
  

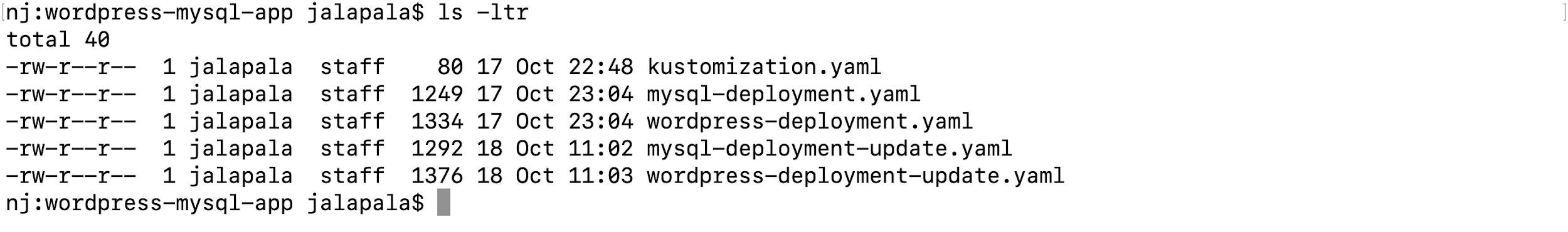
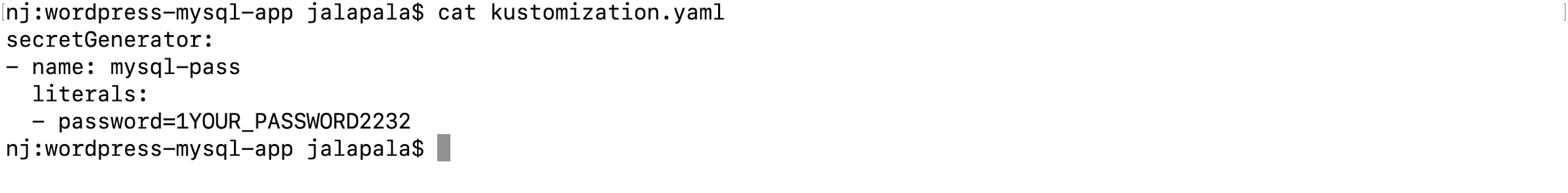
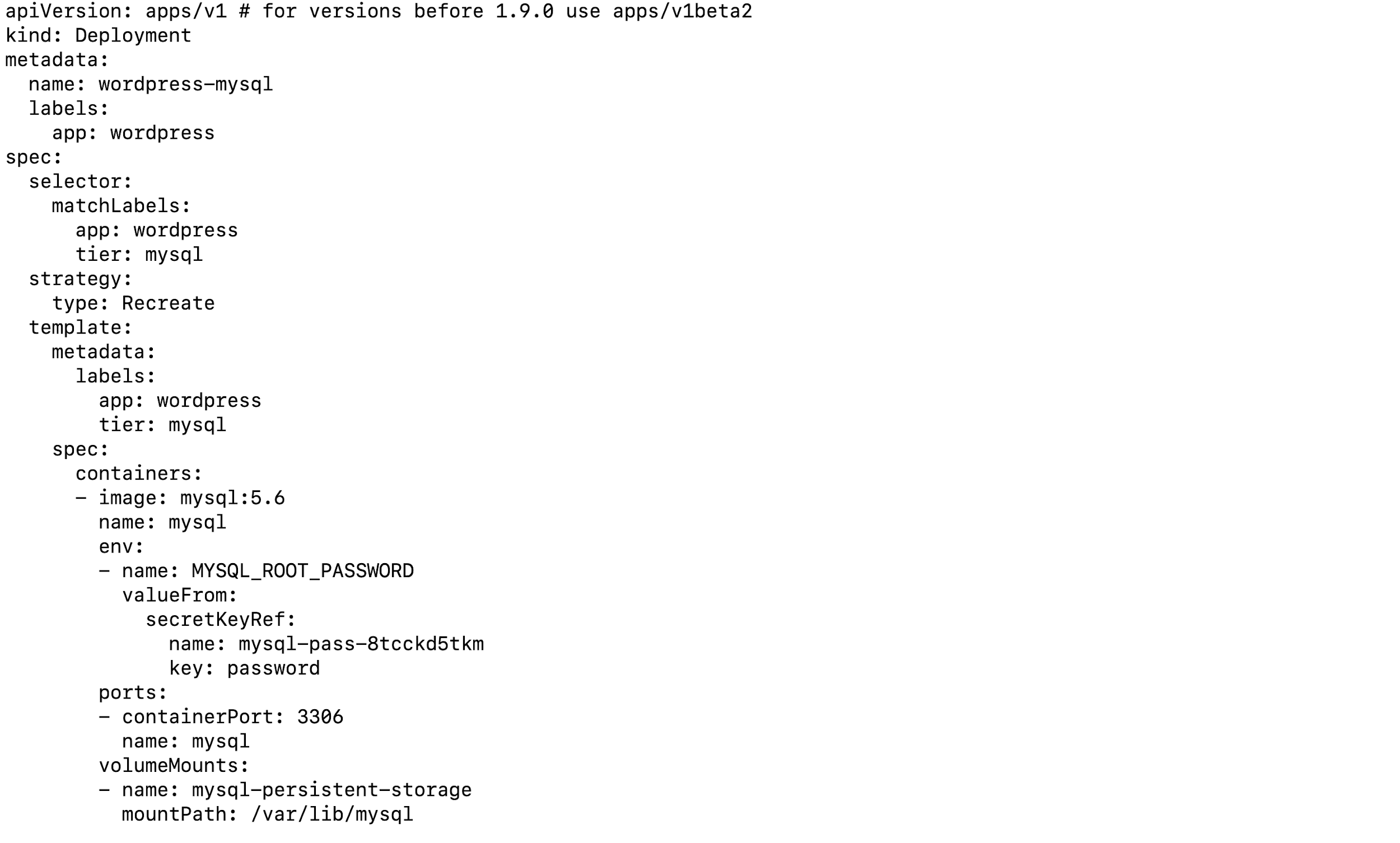
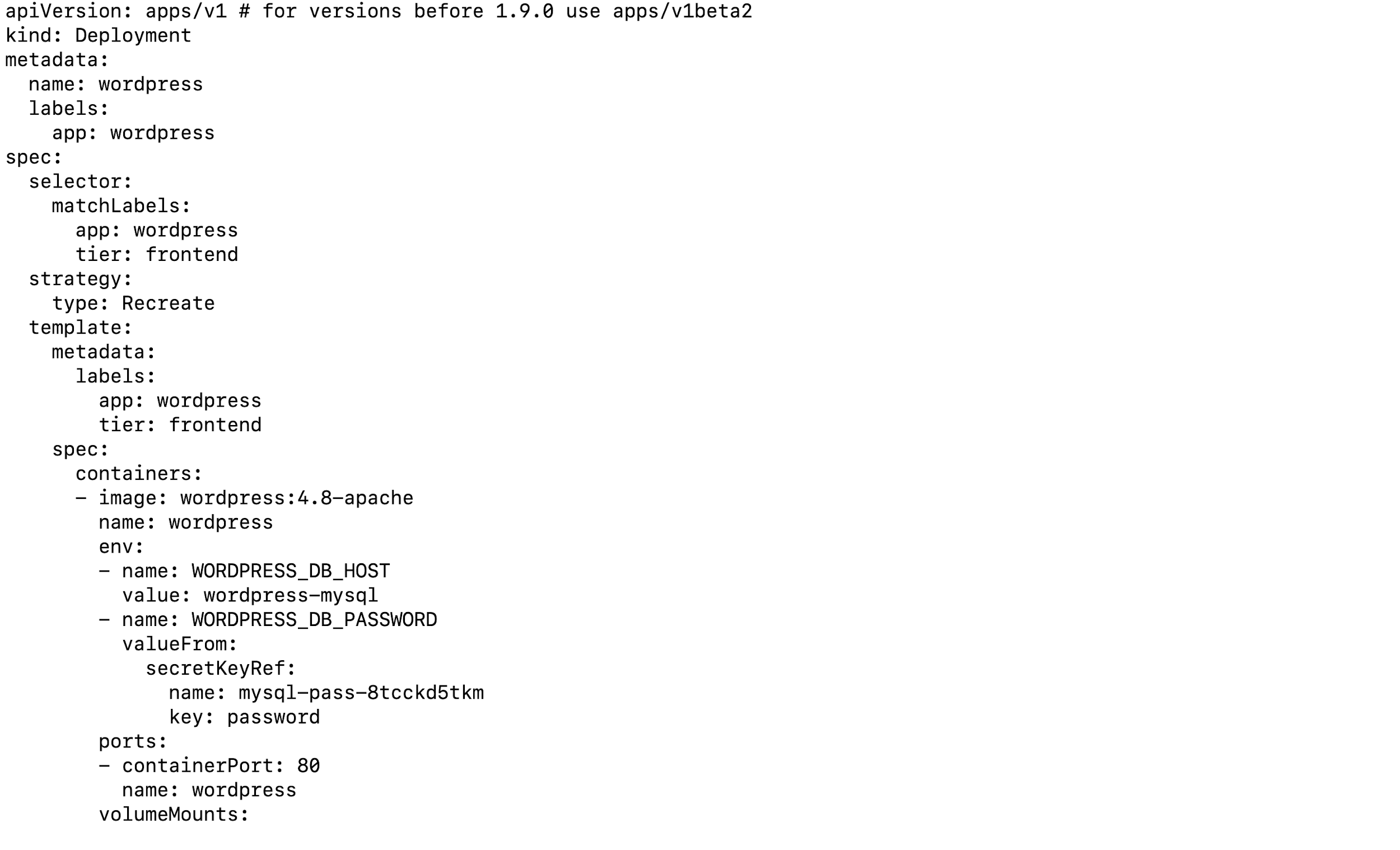
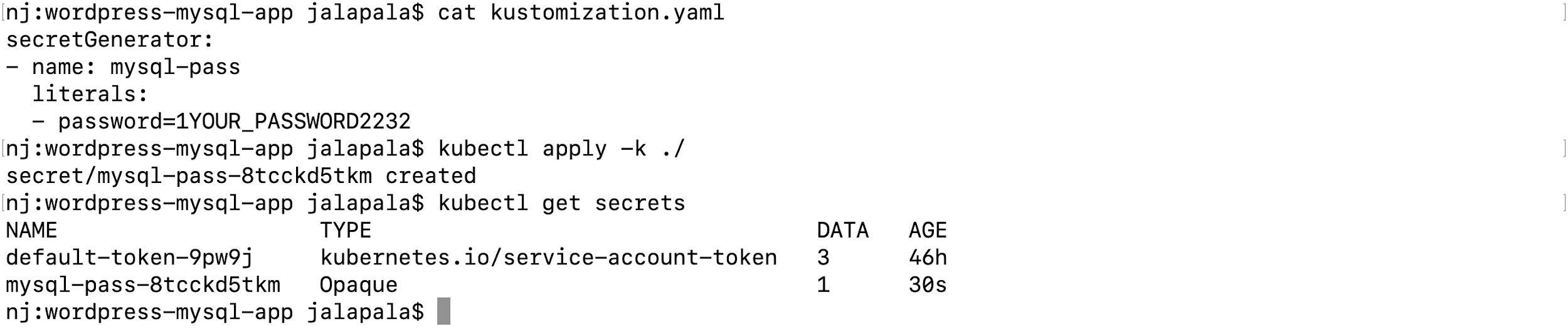
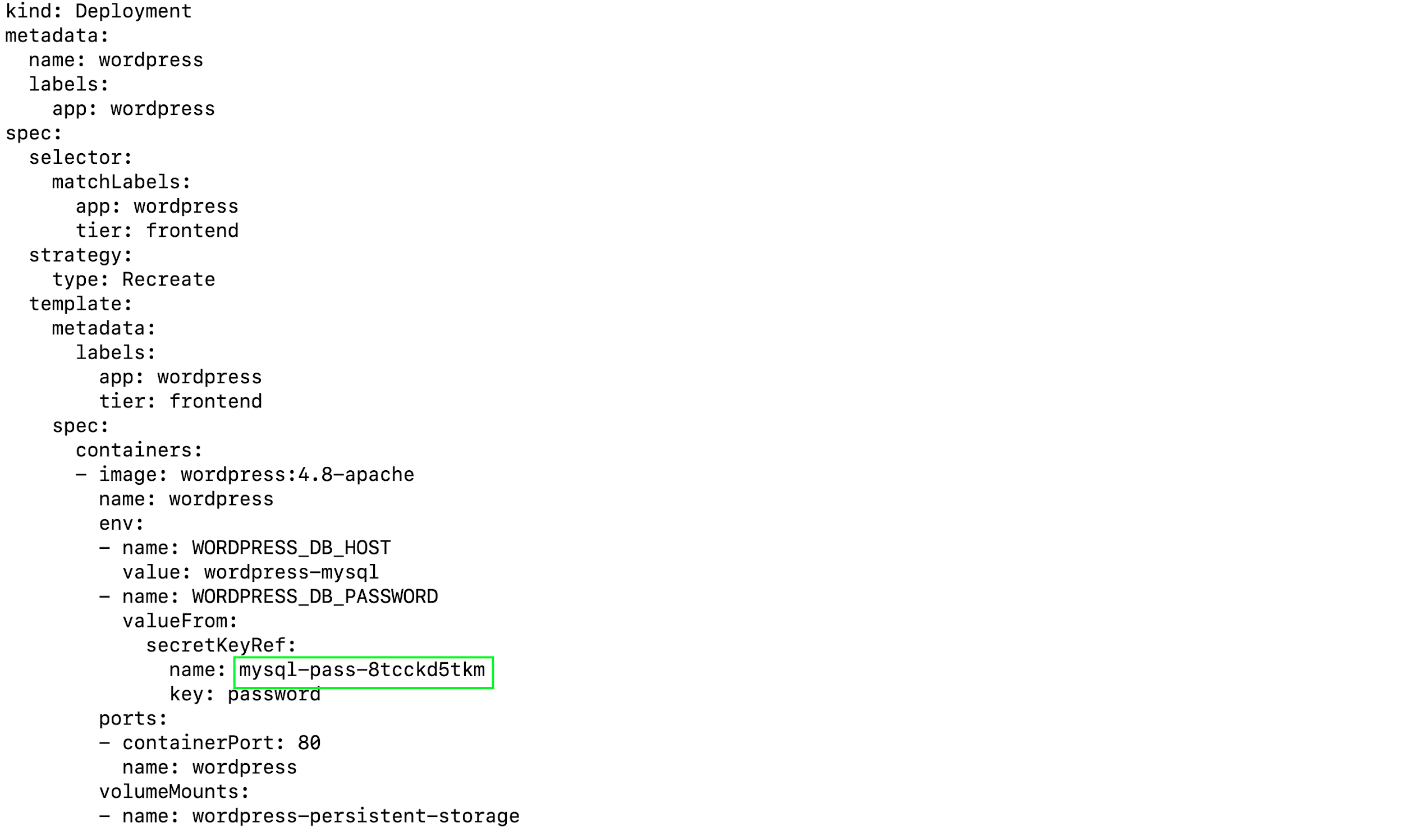
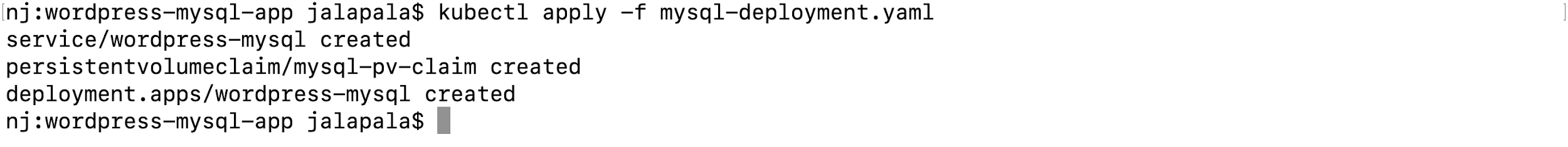
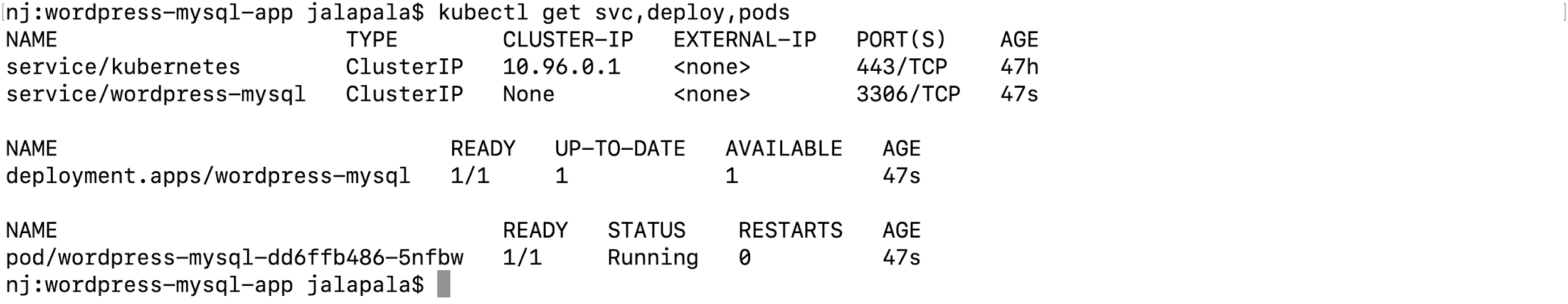
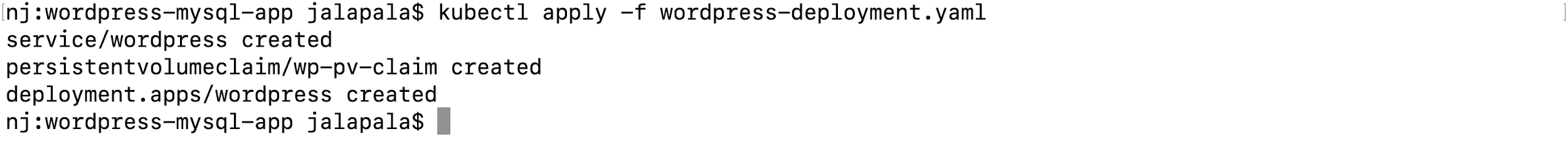
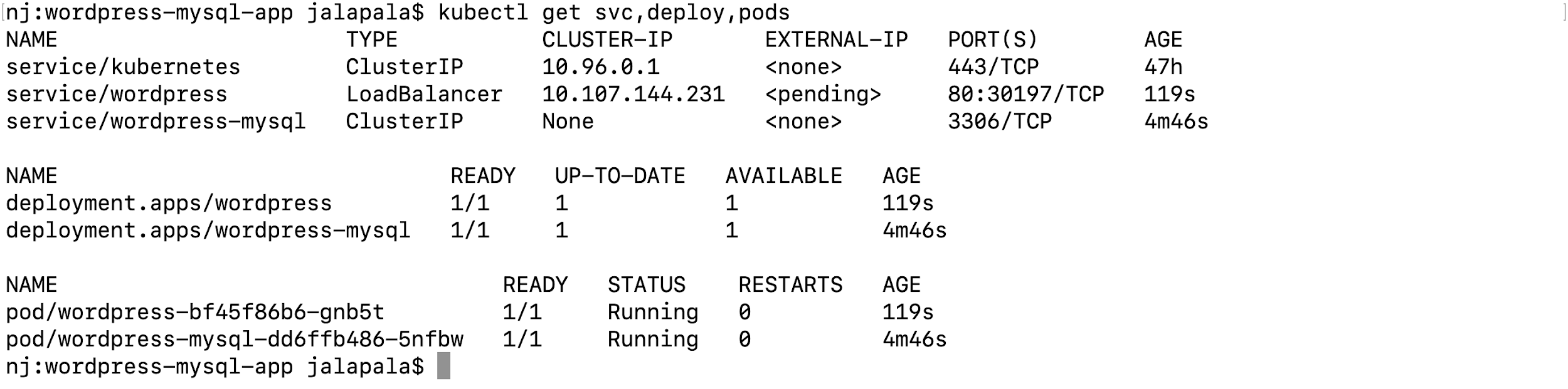
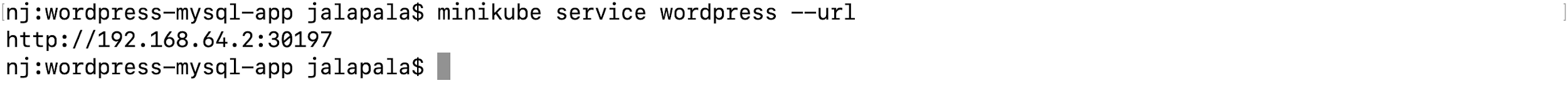
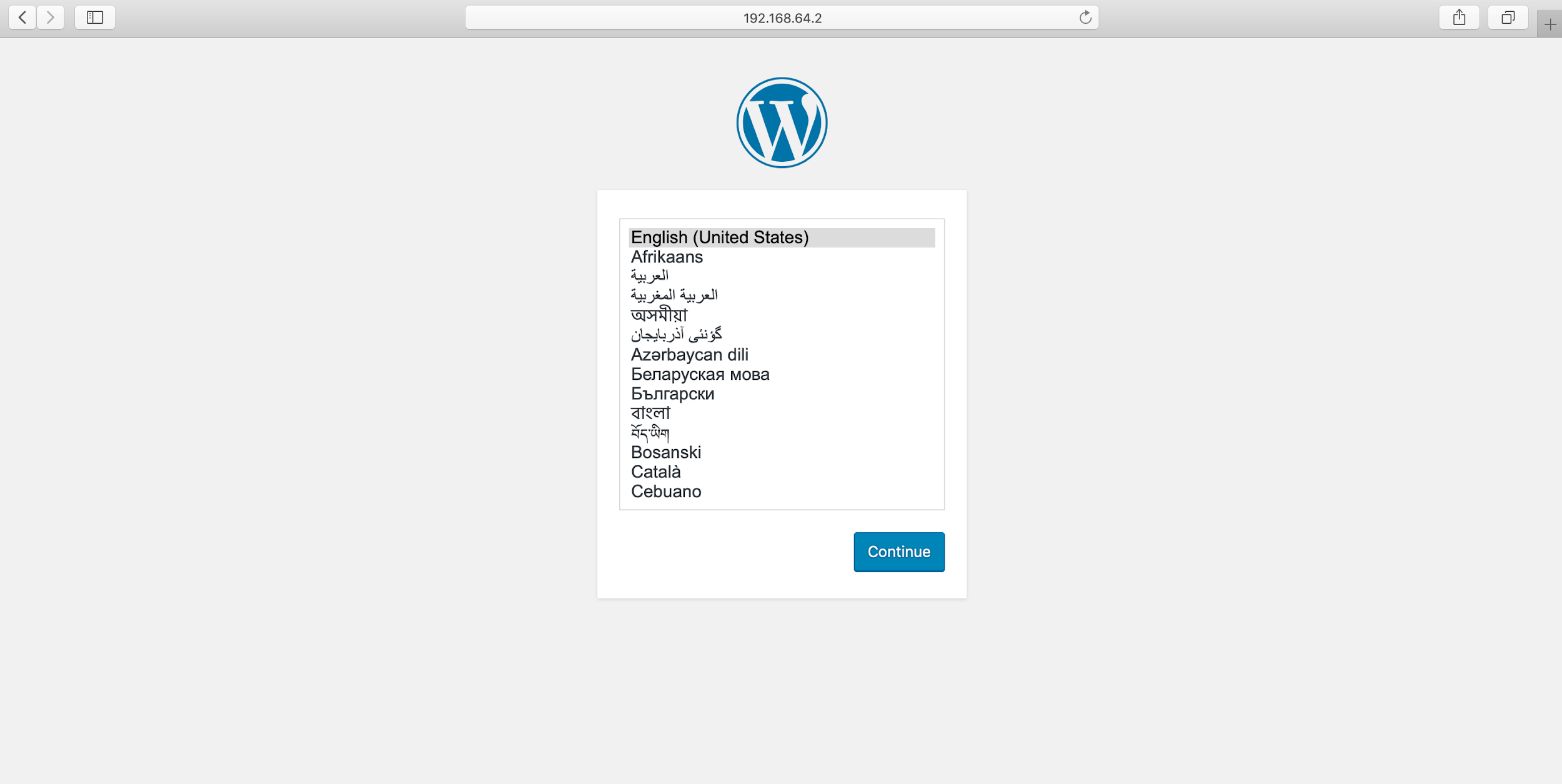
## Scaling - Nginx Service

* The file “deployment-scale.yaml” sets replicas to 4, which specifies that the Deployment should have four pods:  
    
  
* Run kubectl apply -f deployment-scale.yaml to scale the replica to 4  
    
  
* Run kubectl get pods -l app=nginx to verify the deployments of 4 pods  
    
  

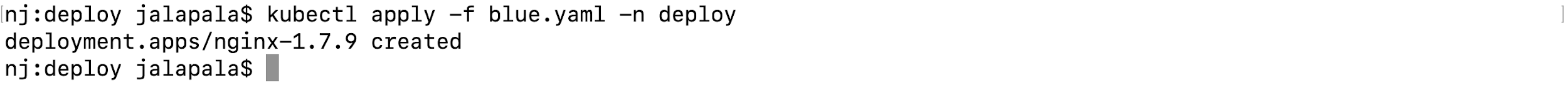
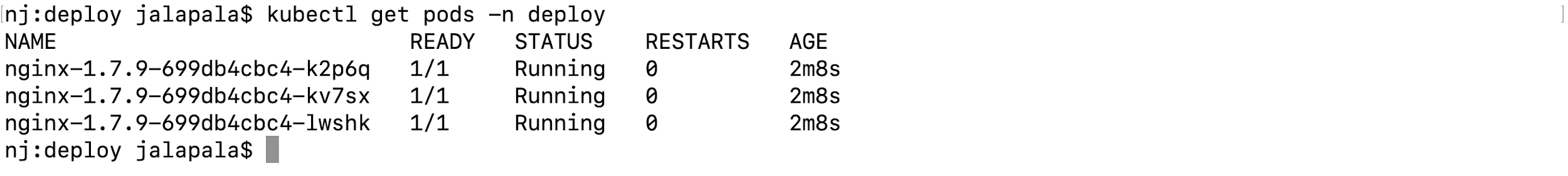
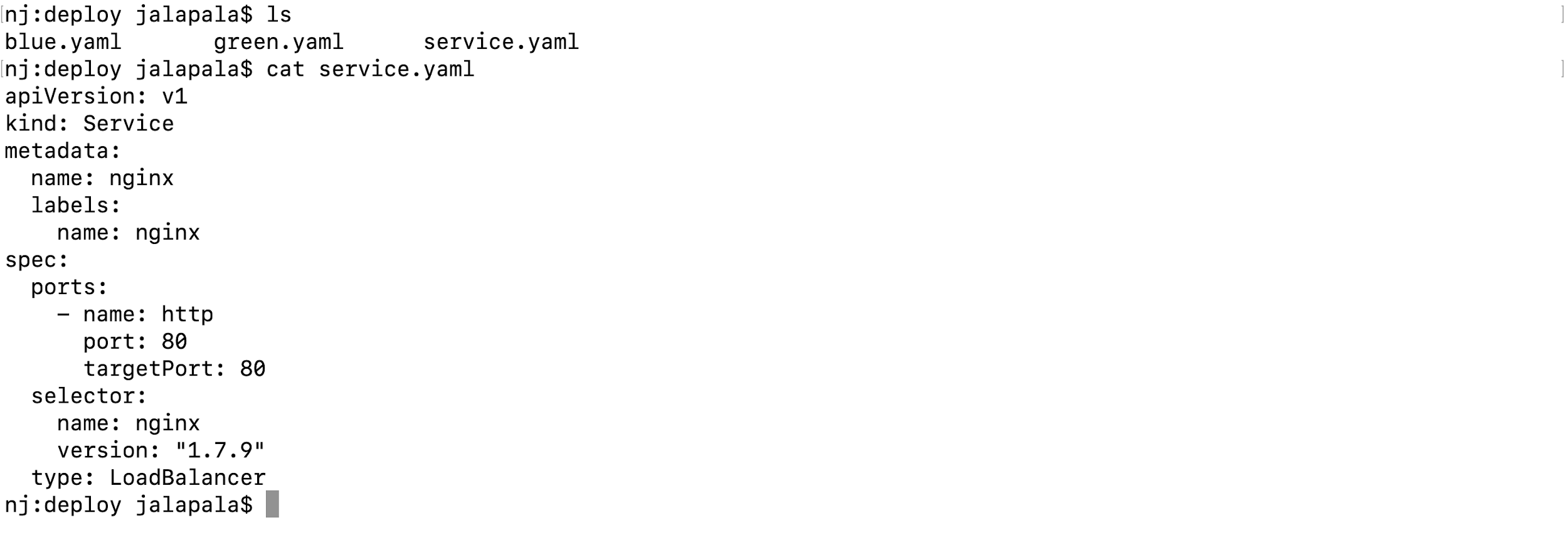
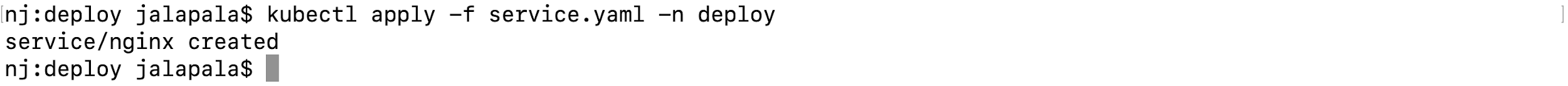
## Deleting the Service

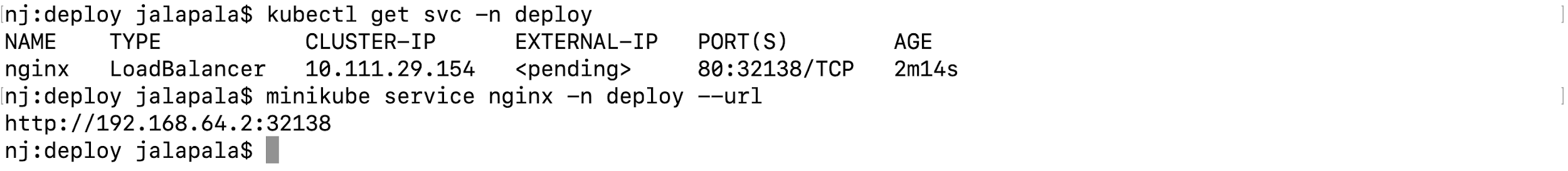
* Run kubectl get deployment to see the list of deployments  
    
  
* Run kubectl delete deployment nginx-deployment to delete the nginx-deployment  
    
  

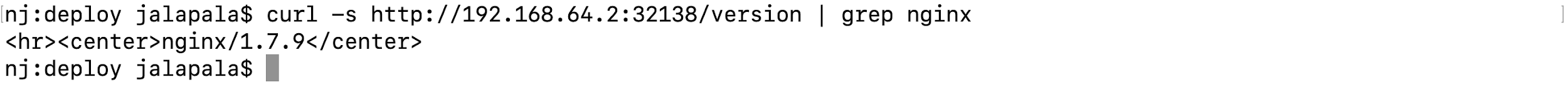
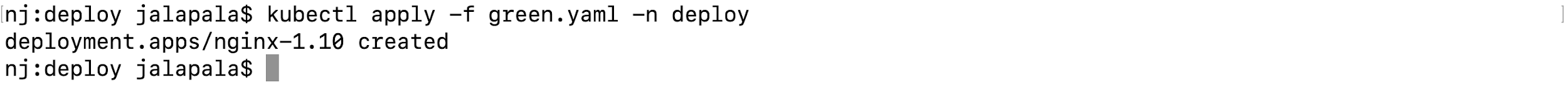
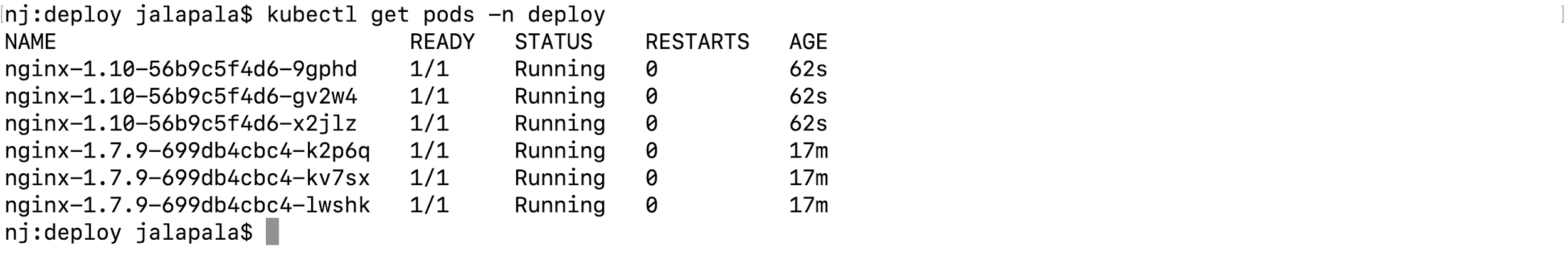
## Deploying WordPress and MySQL with Persistent Volumes

* In the wordpress-mysql-app directory you can see the following files   
    
  
* Kustomization.yaml file generates the secrets for the mysql root password  
    
  
* The following manifest describes a single-instance MySQL Deployment. The MySQL container mounts the PersistentVolume at /var/lib/mysql. The MYSQL\_ROOT\_PASSWORD environment variable sets the database password from the Secret  
    
  
* MySQL and Wordpress each require a PersistentVolume to store data. Their PersistentVolumeClaims will be created at the deployment step  
    
  
* The following manifest describes a single-instance WordPress Deployment. The WordPress container mounts the PersistentVolume at /var/www/html for website data files
* The WORDPRESS\_DB\_HOST environment variable sets the name of the MySQL Service defined above, and WordPress will access the database by Service
* The WORDPRESS\_DB\_PASSWORD environment variable sets the database password from the Secret kustomize generated  
    
  
* Run kubectl apply -k ./ to generate the secrets   
    
  
* Update the secrets reference as “mysql-pass-8tcckd5tkm” in both mysql-deployment.yaml and wordpress-deployment.yaml files   
    
  
* Run kubectl apply -f mysql-deployment.yaml to create a mysql deployment   
    
  
* Run kubectl get svc,deploy,pods to get details about the service, deployment and the pods  
    
  
* Run kubectl apply -f wordpress-deployment.yaml to create a wordpress deployment  
    
  
* Run kubectl get svc,deploy,pods to get details about the service, deployment and the pods  
    
  
* Run minikube service wordpress --url to access the wordpress site or if you are using the docker desktop access the website on the http://localhost  
    
  
* Access the website from the browser  
    
  

## Blue/Green Deployments

* Kubernetes has built in feature for deployments - Rolling Update
* We can use rolling update feature to update the new version of app
* There are legacy applications out there that don't work well with rolling updates. Some applications simply need to deploy a new version and cut over to it
* In Blue/Green deployments a copy of application green is deployed along side of existing blue version the ingress traffic is routed to new version green
* Kubernetes doesn't have support for blue/green deployments built in
* Currently the best way to do it is create a new deployment and then update the service for the application to point to the new deployment
* Under deploy folder there are three files  
    
  
* Run kubectl create namespace deploy to create a separate namespace  
    
  
* Run kubectl apply -f blue.yaml -n deploy to create the nginx deployment. In blue.yaml file we are deploying nginx version 1.7.9   
    
  
* Run kubectl get pods -n deploy to get the status of the pods of the newly created deployment  
    
  
* Run kubectl apply -f service.yaml -n deploy to create new service nginx  
    
    
    
  
* Get the service url by running minikube service nginx -n deploy --url



* Verify the version of nginx deploy by running the command   
    
  
* Run kubectl apply -f green.yaml -n deploy to create the nginx deployment. In green.yaml file we are deploying nginx version 1.10  
    
  
* Run kubectl get pods -n deploy to get the status of the pods of the newly created deployment  
    
  
* Now update the service.yaml to route the traffic from blue to green  
    
  
* Run kubectl apply -f service.yaml -n deploy to update new service nginx
* Now verify the service url by running minikube service nginx -n deploy --url  
    
  