

## kubectl get pods

The screenshot shows a VS Code window with a terminal running a deployment script. The script sets up a Docker environment, builds an image, and deploys it to a Kubernetes cluster. After the deployment, the user runs `kubectl get pods` to check the status of the pods. The output shows three pods:  `rider-db-6c76dcb9c7-ghw5` ,  `rider-service-56bbc6cb6b-2k4wh` , and  `rider-service-56bbc6cb6b-2k4wh` . The  `rider-service`  pods are in a  `Running`  state.

```
(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % ./deploy_minikube.sh
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/6] FROM docker.io/library/python:3.10-slim@sha256:e8c4fae70d559834a40f6c3e8326e02cfe239c23510922e1fb1577a3c6ebde02
=> [internal] load build context
=> transferring context: 454.68kB
=> CACHED [2/6] MONGODB /app
=> [3/6] COPY . /app
=> [4/6] RUN mkdir -p /app/logs && chmod -R 777 /app/logs
=> [5/6] RUN pip install --no-cache-dir -r requirements.txt
=> [6/6] RUN chmod +x /app/entrypoint.sh
=> exporting to image
=> exporting layers
=> writing image sha256:d643f06aab9f42112b0dbfaaf22795a4a3ac0515854250878ffdf884789a57e0
=> naming to docker.io/library/rider-service:latest
Deploying Kubernetes resources...
configmap "rider-config" deleted from default namespace
deployment.apps "rider-db" deleted from default namespace
service "rider-db" deleted from default namespace
persistentvolumeclaim "mysql-pvc" deleted from default namespace
deployment.apps "rider-service" deleted from default namespace
service "rider-service" deleted from default namespace
secret "rider-secret" deleted from default namespace
configmap/rider-config created
deployment.apps/rider-db created
Warning: spec.SessionAffinity is ignored for headless services
service/rider-db created
persistentvolumeclaim/mysql-pvc created
deployment.apps/rider-service created
service/rider-service created
secret/rider-secret created
Waiting for Rider Service pod to start...
pod/rider-service-56bbc6cb6b-2k4wh condition met
Pods status:
NAME                READY   STATUS    RESTARTS   AGE   IP              NODE   NOMINATED NODE   READINESS GATES
rider-db-6c76dcb9c7-ghw5    1/1     Running   0           12s   10.244.0.13     minikube   <none>           <none>
rider-service-56bbc6cb6b-2k4wh 1/1     Running   0           11s   10.244.0.12     minikube   <none>           <none>
Services status:
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   None          <none>         443/TCP          95m
rider-db             ClusterIP   None          <none>         3306/TCP         12s
rider-service        NodePort    10.106.26.143 <none>         5001:30001/TCP   11s
Rider Service logs (showing last 10 lines):
Seeding from CSV: /app/rhfd_riders.csv
Inserted 80 riders
DB init complete
[2025-10-31 16:48:41 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2025-10-31 16:48:41 +0000] [1] [INFO] Listening at: http://0.0.0.0:5001 (1)
[2025-10-31 16:48:41 +0000] [1] [INFO] Using worker: sync
[2025-10-31 16:48:41 +0000] [9] [INFO] Booting worker with pid: 9
/usr/local/lib/python3.10/site-packages/flask_limiter/extension.py:364: UserWarning: Using the in-memory storage for tracking rate limits a
s no storage was explicitly specified, this is not recommended for production use. See: https://flask-limiter.readthedocs.io/configuring-a-s
storage-backend-for documentation about configuring the storage backend.
  warnings.warn(
2025-10-31 16:48:45,812 INFO: Incoming GET request: /health [in /app/app.py:51]
Setting up permanent local access...
Checking and cleaning old port-forward processes...
Rider Service now available permanently at: http://localhost:5001/health

Deployment completed successfully!
Access Rider Service at: http://localhost:5001/health
To view live logs: kubectl logs -f -l app=rider-service

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP              NODE   NOMINATED NODE   READINESS GATES
rider-db-6c76dcb9c7-ghw5    1/1     Running   0           58s   10.244.0.13     minikube   <none>           <none>
rider-service-56bbc6cb6b-2k4wh 1/1     Running   0           57s   10.244.0.12     minikube   <none>           <none>
```

## kubectl get svc

The screenshot shows the same VS Code window as before, but with the `kubectl get svc` command executed. The output shows a single service named  `rider-service`  of type  `NodePort` , with a cluster IP of  `10.106.26.143`  and an external IP of  `5001:30001/TCP` .

```
(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % ./deploy_minikube.sh
=> [3/6] COPY . /app
=> [4/6] RUN mkdir -p /app/logs && chmod -R 777 /app/logs
=> [5/6] RUN pip install --no-cache-dir -r requirements.txt
=> [6/6] RUN chmod +x /app/entrypoint.sh
=> exporting to image
=> exporting layers
=> writing image sha256:d643f06aab9f42112b0dbfaaf22795a4a3ac0515854250878ffdf884789a57e0
=> naming to docker.io/library/rider-service:latest
Deploying Kubernetes resources...
configmap "rider-config" deleted from default namespace
deployment.apps "rider-db" deleted from default namespace
service "rider-db" deleted from default namespace
persistentvolumeclaim "mysql-pvc" deleted from default namespace
deployment.apps "rider-service" deleted from default namespace
service "rider-service" deleted from default namespace
secret "rider-secret" deleted from default namespace
configmap/rider-config created
deployment.apps/rider-db created
Warning: spec.SessionAffinity is ignored for headless services
service/rider-db created
persistentvolumeclaim/mysql-pvc created
deployment.apps/rider-service created
service/rider-service created
secret/rider-secret created
Waiting for Rider Service pod to start...
pod/rider-service-56bbc6cb6b-2k4wh condition met
Pods status:
NAME                READY   STATUS    RESTARTS   AGE   IP              NODE   NOMINATED NODE   READINESS GATES
rider-db-6c76dcb9c7-ghw5    1/1     Running   0           12s   10.244.0.13     minikube   <none>           <none>
rider-service-56bbc6cb6b-2k4wh 1/1     Running   0           11s   10.244.0.12     minikube   <none>           <none>
Services status:
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   None          <none>         443/TCP          95m
rider-db             ClusterIP   None          <none>         3306/TCP         12s
rider-service        NodePort    10.106.26.143 <none>         5001:30001/TCP   11s
Rider Service logs (showing last 10 lines):
Seeding from CSV: /app/rhfd_riders.csv
Inserted 80 riders
DB init complete
[2025-10-31 16:48:41 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2025-10-31 16:48:41 +0000] [1] [INFO] Listening at: http://0.0.0.0:5001 (1)
[2025-10-31 16:48:41 +0000] [1] [INFO] Using worker: sync
[2025-10-31 16:48:41 +0000] [9] [INFO] Booting worker with pid: 9
/usr/local/lib/python3.10/site-packages/flask_limiter/extension.py:364: UserWarning: Using the in-memory storage for tracking rate limits a
s no storage was explicitly specified, this is not recommended for production use. See: https://flask-limiter.readthedocs.io/configuring-a-s
storage-backend-for documentation about configuring the storage backend.
  warnings.warn(
2025-10-31 16:48:45,812 INFO: Incoming GET request: /health [in /app/app.py:51]
Setting up permanent local access...
Checking and cleaning old port-forward processes...
Rider Service now available permanently at: http://localhost:5001/health

Deployment completed successfully!
Access Rider Service at: http://localhost:5001/health
To view live logs: kubectl logs -f -l app=rider-service

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP              NODE   NOMINATED NODE   READINESS GATES
rider-db-6c76dcb9c7-ghw5    1/1     Running   0           58s   10.244.0.13     minikube   <none>           <none>
rider-service-56bbc6cb6b-2k4wh 1/1     Running   0           57s   10.244.0.12     minikube   <none>           <none>

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl get svc
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes           ClusterIP   None          <none>         443/TCP          95m
rider-db             ClusterIP   None          <none>         3306/TCP         85s
rider-service        NodePort    10.106.26.143 <none>         5001:30001/TCP   84s
```

# kubectl logs

The screenshot shows the VS Code interface with the Explorer, Problems, Output, Debug Console, and Terminal panels. The Explorer panel on the left shows a project structure for 'rider-service' with files like 'deploy\_minikube.sh', 'docker-compose.yml', 'Dockerfile', 'entrypoint.sh', 'init\_db.py', 'init\_db.sql', 'LocallyMySQL\_session.sql', 'requirements.txt', 'rhfd\_riders.csv', 'rider\_portforward.log', and 'riders\_backup.sql'. The Terminal panel shows the output of the 'kubectl logs' command for the 'rider-service' pod. The output includes the pod status, service status, and a list of logs showing incoming GET requests to the '/health' endpoint. A red box highlights the 'kubectl logs' command and its output.

```
(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % ./deploy_minikube.sh
deployment.apps/rider-service created
service/rider-service created
secret/rider-secrets created
Waiting for Rider Service pod to start...
pod/rider-service-56bcb6b6-2k4wh condition met
Pods status:
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE     NOMINATED NODE   READINESS GATES
rider-db-6c76dc9c7-qhw5s            1/1     Running   0           12s   10.244.0.13     minikube <none> <none>
rider-service-56bcb6b6-2k4wh        1/1     Running   0           11s   10.244.0.12     minikube <none> <none>
Services status:
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes ClusterIP    10.96.0.1     <none>         443/TCP     95m
rider-db  ClusterIP    None         <none>         3386/TCP    12s
rider-service NodePort    10.106.26.143 <none>         5001:30001/TCP 11s
Rider Service logs (showing last 10 lines):
Seeding from CSV: /app/rhfd_riders.csv
Inserted 80 riders
DB init complete
2025-10-31 16:48:41 +0000 [1] [INFO] Starting gunicorn 20.1.0
2025-10-31 16:48:41 +0000 [1] [INFO] Listening at http://0.0.0.0:5001 (1)
2025-10-31 16:48:41 +0000 [1] [INFO] Using worker: sync
2025-10-31 16:48:41 +0000 [9] [INFO] Booting worker with pid: 9
/usr/local/lib/python3.10/site-packages/flask_limiter/extension.py:364: UserWarning: Using the in-memory storage for tracking rate limits a
s no storage was explicitly specified. This is not recommended for production use. See: https://flask-limiter.readthedocs.io/configuring-a-s
storage-backend-for-documentation about configuring the storage backend.
warnings.warn(
2025-10-31 16:48:45.812 INFO: Incoming GET request: /health [in /app/app.py:51]
Setting up permanent local access...
Checking and cleaning old port-forward processes...
Rider Service now available permanently at: http://localhost:5001/health
Deployment completed successfully!
Access Rider Service at: http://localhost:5001/health
To view live logs: kubectl logs -f -l app=rider-service

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE     NOMINATED NODE   READINESS GATES
rider-db-6c76dc9c7-qhw5s            1/1     Running   0           58s   10.244.0.13     minikube <none> <none>
rider-service-56bcb6b6-2k4wh        1/1     Running   0           57s   10.244.0.12     minikube <none> <none>

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl get svc
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes ClusterIP    10.96.0.1     <none>         443/TCP     96m
rider-db  ClusterIP    None         <none>         3386/TCP    85s
rider-service NodePort    10.106.26.143 <none>         5001:30001/TCP 84s

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl logs
error: expected 'logs [-f] [-p] (POD | TYPE/NAME) [-c CONTAINER]',
POD or TYPE/NAME is a required argument for the logs command

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl logs -l app=rider-service --tail=15
2025-10-31 16:50:44.332 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:45.815 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:50.844 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:54.329 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:55.814 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:00.815 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:04.336 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:05.813 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:10.814 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:14.330 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:15.813 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:20.816 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:24.347 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:25.813 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:30.825 INFO: Incoming GET request: /health [in /app/app.py:51]
```

## curl or Postman test (against NodePort)

The screenshot shows the VS Code interface with the Explorer, Problems, Output, Debug Console, and Terminal panels. The Explorer panel on the left shows the same project structure as the previous screenshot. The Terminal panel shows the output of the 'kubectl logs' command for the 'rider-service' pod. The output includes the pod status, service status, and a list of logs showing incoming GET requests to the '/health' endpoint. A red box highlights the 'kubectl logs' command and its output. A green box highlights the 'curl' command and its output.

```
(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % ./deploy_minikube.sh
Pods status:
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE     NOMINATED NODE   READINESS GATES
rider-db-6c76dc9c7-qhw5s            1/1     Running   0           12s   10.244.0.13     minikube <none> <none>
rider-service-56bcb6b6-2k4wh        1/1     Running   0           11s   10.244.0.12     minikube <none> <none>
Services status:
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes ClusterIP    10.96.0.1     <none>         443/TCP     95m
rider-db  ClusterIP    None         <none>         3386/TCP    12s
rider-service NodePort    10.106.26.143 <none>         5001:30001/TCP 11s
Rider Service logs (showing last 10 lines):
Seeding from CSV: /app/rhfd_riders.csv
Inserted 80 riders
DB init complete
2025-10-31 16:48:41 +0000 [1] [INFO] Starting gunicorn 20.1.0
2025-10-31 16:48:41 +0000 [1] [INFO] Listening at http://0.0.0.0:5001 (1)
2025-10-31 16:48:41 +0000 [1] [INFO] Using worker: sync
2025-10-31 16:48:41 +0000 [9] [INFO] Booting worker with pid: 9
/usr/local/lib/python3.10/site-packages/flask_limiter/extension.py:364: UserWarning: Using the in-memory storage for tracking rate limits a
s no storage was explicitly specified. This is not recommended for production use. See: https://flask-limiter.readthedocs.io/configuring-a-s
storage-backend-for-documentation about configuring the storage backend.
warnings.warn(
2025-10-31 16:48:45.812 INFO: Incoming GET request: /health [in /app/app.py:51]
Setting up permanent local access...
Checking and cleaning old port-forward processes...
Rider Service now available permanently at: http://localhost:5001/health
Deployment completed successfully!
Access Rider Service at: http://localhost:5001/health
To view live logs: kubectl logs -f -l app=rider-service

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE     NOMINATED NODE   READINESS GATES
rider-db-6c76dc9c7-qhw5s            1/1     Running   0           58s   10.244.0.13     minikube <none> <none>
rider-service-56bcb6b6-2k4wh        1/1     Running   0           57s   10.244.0.12     minikube <none> <none>

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl get svc
NAME      TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
kubernetes ClusterIP    10.96.0.1     <none>         443/TCP     96m
rider-db  ClusterIP    None         <none>         3386/TCP    85s
rider-service NodePort    10.106.26.143 <none>         5001:30001/TCP 84s

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl logs
error: expected 'logs [-f] [-p] (POD | TYPE/NAME) [-c CONTAINER]',
POD or TYPE/NAME is a required argument for the logs command
See 'kubectl logs -h' for help and examples

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % kubectl logs -l app=rider-service --tail=15
2025-10-31 16:50:44.332 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:45.815 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:50.844 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:54.329 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:50:55.814 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:00.815 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:04.336 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:05.813 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:10.814 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:14.330 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:15.813 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:20.816 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:24.347 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:25.813 INFO: Incoming GET request: /health [in /app/app.py:51]
2025-10-31 16:51:30.825 INFO: Incoming GET request: /health [in /app/app.py:51]

(.venv) dishanibasa@Dishanis-MacBook-Air Rider-Service % curl http://localhost:5001/health
{"status": "ok"}
```