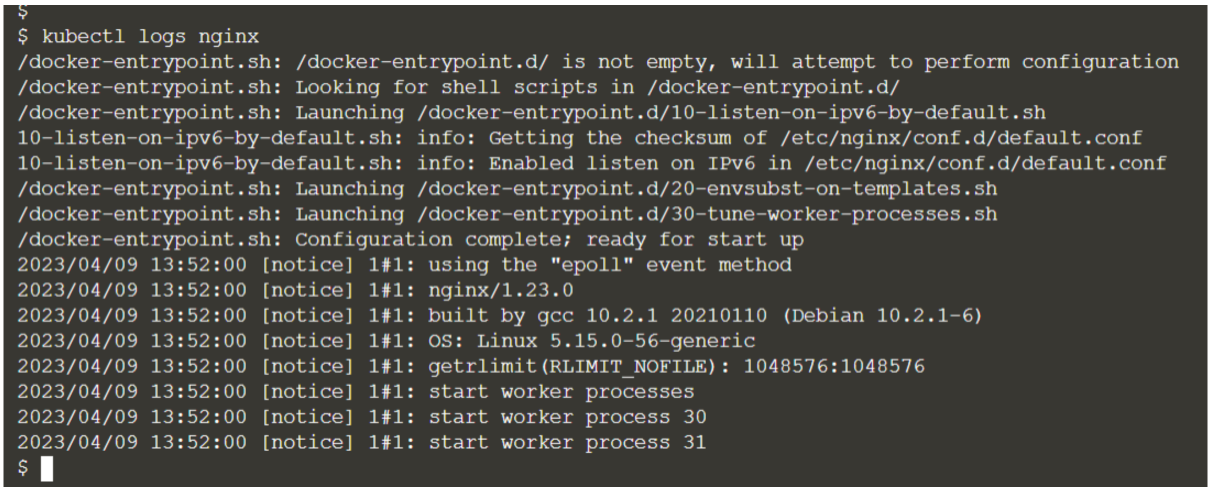
In the optimal case, you deploy an application to Kubernetes and it will run without issues. There may be many reasons why an application doesn't behave the way it should. For example, the application is misconfigured, the application exposes a runtime problem like a deadlock, or it can't connect to another microservice running inside of the cluster. kubectl offers a number of commands to troubleshoot an issue with an application running in a container. In this lab, you will try them one by one against an already running Pod.

Retrieving Container Logs
You can use the following command to see the logs for a running container:

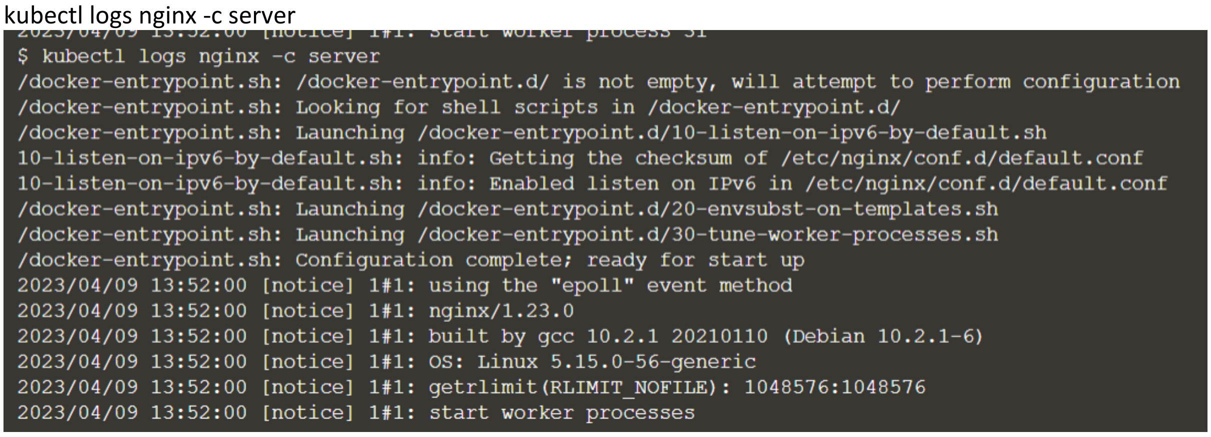
kubectl logs nginx



```
$
$ kubectl logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/04/09 13:52:00 [notice] 1#1: using the "epoll" event method
2023/04/09 13:52:00 [notice] 1#1: nginx/1.23.0
2023/04/09 13:52:00 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/04/09 13:52:00 [notice] 1#1: OS: Linux 5.15.0-56-generic
2023/04/09 13:52:00 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/04/09 13:52:00 [notice] 1#1: start worker processes
2023/04/09 13:52:00 [notice] 1#1: start worker process 30
2023/04/09 13:52:00 [notice] 1#1: start worker process 31
$ █
```

If you have multiple containers in your Pod, you can choose the container to view using the -c flag. The nginx Pod only runs a single container. Therefore, you do not have to explicitly spell out the container name. If you want, you can state the container. The following command has the same effect as the first one. We are targeting the container named server:

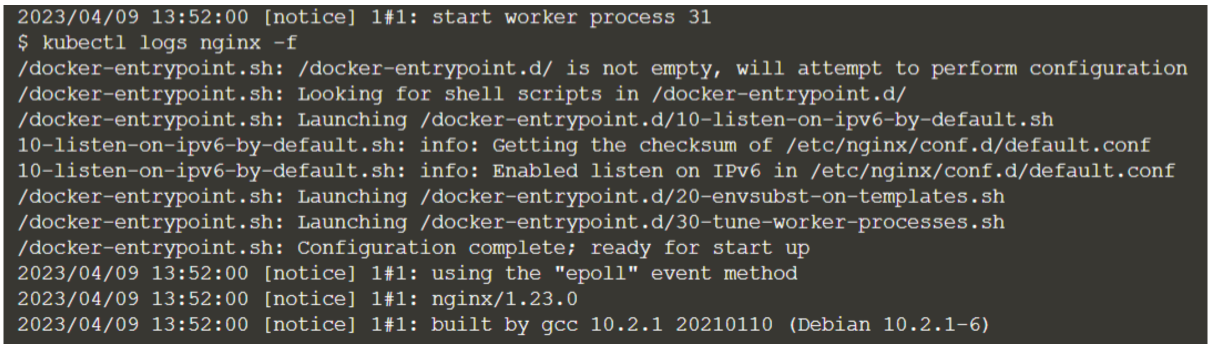
kubectl logs nginx -c server



```
2023/04/09 13:52:00 [notice] 1#1: start worker process 31
$ kubectl logs nginx -c server
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/04/09 13:52:00 [notice] 1#1: using the "epoll" event method
2023/04/09 13:52:00 [notice] 1#1: nginx/1.23.0
2023/04/09 13:52:00 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/04/09 13:52:00 [notice] 1#1: OS: Linux 5.15.0-56-generic
2023/04/09 13:52:00 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/04/09 13:52:00 [notice] 1#1: start worker processes
```

By default, kubectl logs lists the current logs and exits. If you instead want to continuously stream the logs back to the terminal without exiting, you can add the -f (follow) command-line flag.

kubectl logs nginx -f



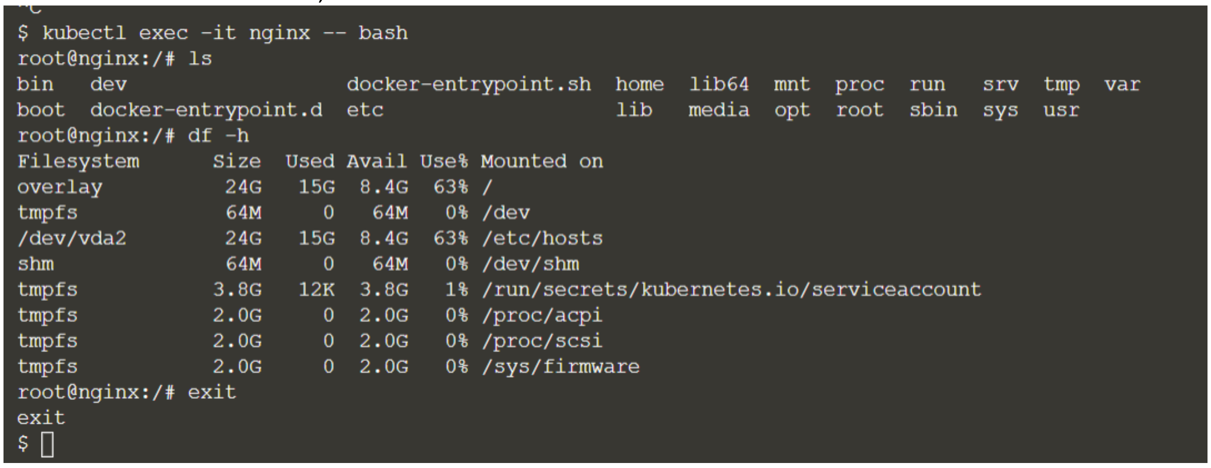
```
2023/04/09 13:52:00 [notice] 1#1: start worker process 31
$ kubectl logs nginx -f
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/04/09 13:52:00 [notice] 1#1: using the "epoll" event method
2023/04/09 13:52:00 [notice] 1#1: nginx/1.23.0
2023/04/09 13:52:00 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
```

Executing a Command in the Container
You can also use the exec command to execute a command in a running container:

kubectl exec -it nginx -- bash

This will provide you with an interactive shell inside the running container so that you can perform more debugging. To target a specific container, use the -c command-line option.

To leave the container's shell, run the exit command.



```
^C
$ kubectl exec -it nginx -- bash
root@nginx:/# ls
bin   dev                   docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   media  opt  root  sbin  sys  usr
root@nginx:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
overlay          24G   15G  8.4G  63% /
tmpfs            64M     0   64M   0% /dev
/dev/vda2        24G   15G  8.4G  63% /etc/hosts
shm              64M     0   64M   0% /dev/shm
tmpfs           3.8G   12K  3.8G   1% /run/secrets/kubernetes.io/serviceaccount
tmpfs           2.0G     0  2.0G   0% /proc/acpi
tmpfs           2.0G     0  2.0G   0% /proc/scsi
tmpfs           2.0G     0  2.0G   0% /sys/firmware
root@nginx:/# exit
exit
$ 
```

Attaching a Terminal to the Container
If you don’t have bash or some other terminal available within your container, you can always attach to the running process:

kubectl attach -it nginx

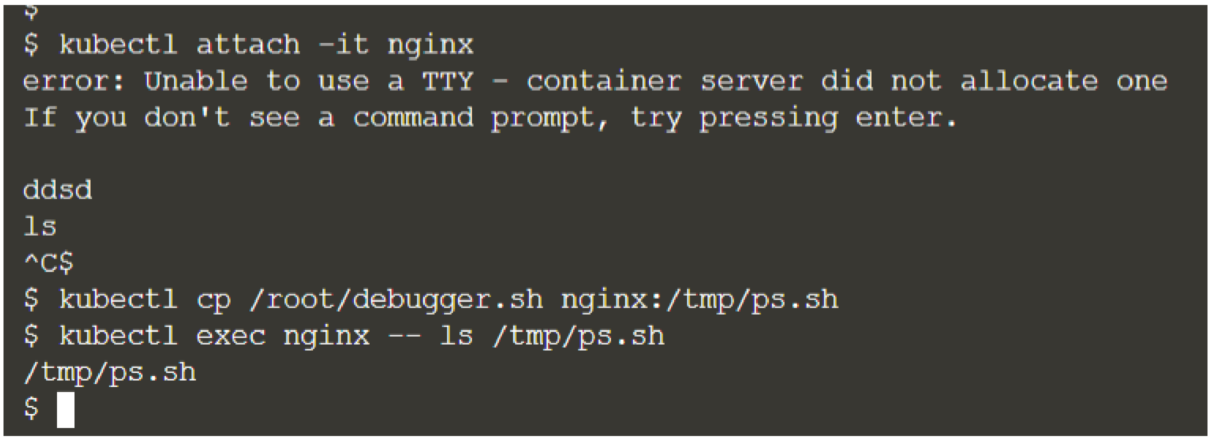
This will attach to the running process. It is similar to kubectl logs but will allow you to send input to the running process, assuming that process is set up to read from standard input.

Copying a File to the Container
You can also copy files to and from a container using the cp command. The first command copies the local file named debugger.sh to the container while renaming it at the same time:

kubectl cp /root/debugger.sh nginx:/tmp/ps.sh

You should now find the file in the correct directory inside of the container:

kubectl exec nginx -- ls /tmp/ps.sh



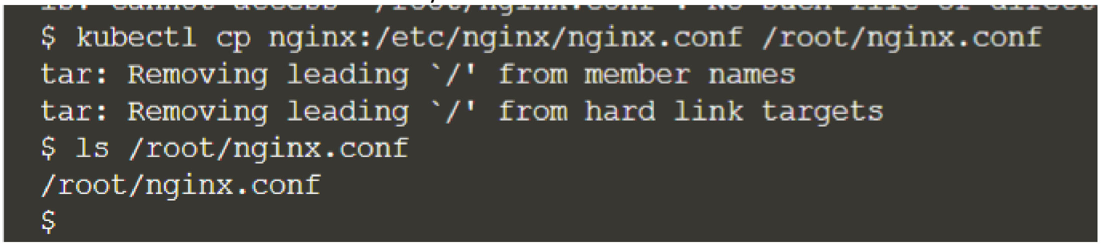
```
$ kubectl attach -it nginx
error: Unable to use a TTY - container server did not allocate one
If you don't see a command prompt, try pressing enter.

ddsd
ls
^C$
$ kubectl cp /root/debugger.sh nginx:/tmp/ps.sh
$ kubectl exec nginx -- ls /tmp/ps.sh
/tmp/ps.sh
$
```

The same command works if you need to download a file from the container to your local file system. Simply reverse the source and target file location:

kubectl cp nginx:/etc/nginx/nginx.conf /root/nginx.conf

The dowloaded file is now available on your local machine:

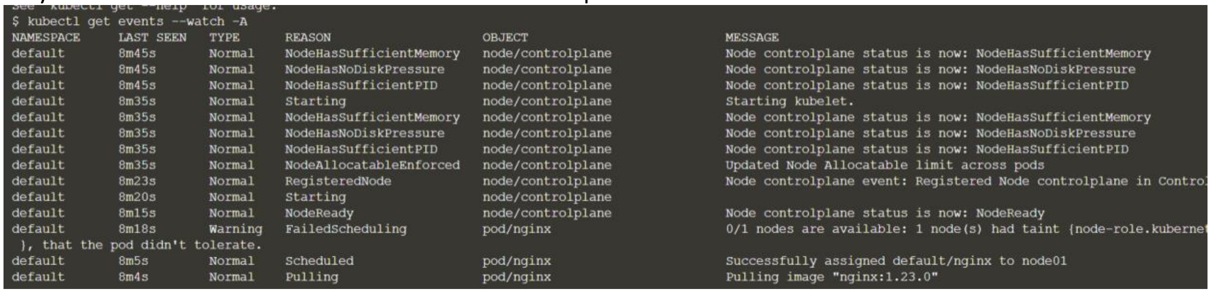


```
$ kubectl cp nginx:/etc/nginx/nginx.conf /root/nginx.conf
tar: Removing leading `/' from member names
tar: Removing leading `/' from hard link targets
$ ls /root/nginx.conf
/root/nginx.conf
$
```

Retrieving Container Events
If you want to view Kubernetes events, you can use the kubectl get events command to see a list of the latest 10 events on all objects in a given namespace:

kubectl get events

Additionally, you can stream events as they happen by adding --watch to the kubectl get events command. You may also wish to include -A to see events in all namespaces.



```
$ kubectl get events --watch -A
NAMESPACE     LAST SEEN   TYPE      REASON                      OBJECT                         MESSAGE
default       8m45s       Normal    NodeHasSufficientMemory     node/controlplane              Node controlplane status is now: NodeHasSufficientMemory
default       8m45s       Normal    NodeHasNoDiskPressure       node/controlplane              Node controlplane status is now: NodeHasNoDiskPressure
default       8m45s       Normal    NodeHasSufficientPID        node/controlplane              Node controlplane status is now: NodeHasSufficientPID
default       8m35s       Normal    Starting                    node/controlplane              Starting kubelet.
default       8m35s       Normal    NodeHasSufficientMemory     node/controlplane              Node controlplane status is now: NodeHasSufficientMemory
default       8m35s       Normal    NodeHasNoDiskPressure       node/controlplane              Node controlplane status is now: NodeHasNoDiskPressure
default       8m35s       Normal    NodeHasSufficientPID        node/controlplane              Node controlplane status is now: NodeHasSufficientPID
default       8m35s       Normal    NodeAllocatableEnforced     node/controlplane              Updated Node Allocatable limit across pods
default       8m23s       Normal    RegisteredNode              node/controlplane              Node controlplane event: Registered Node controlplane in Contro
default       8m20s       Normal    Starting                    node/controlplane
default       8m15s       Normal    NodeReady                   node/controlplane              Node controlplane status is now: NodeReady
default       8m18s       Warning   FailedScheduling            pod/nginx                      0/1 nodes are available: 1 node(s) had taint {node-role.kubernet
 }, that the pod didn't tolerate.
default       8m5s        Normal    Scheduled                   pod/nginx                      Successfully assigned default/nginx to node01
default       8m4s        Normal    Pulling                     pod/nginx                      Pulling image "nginx:1.23.0"
```