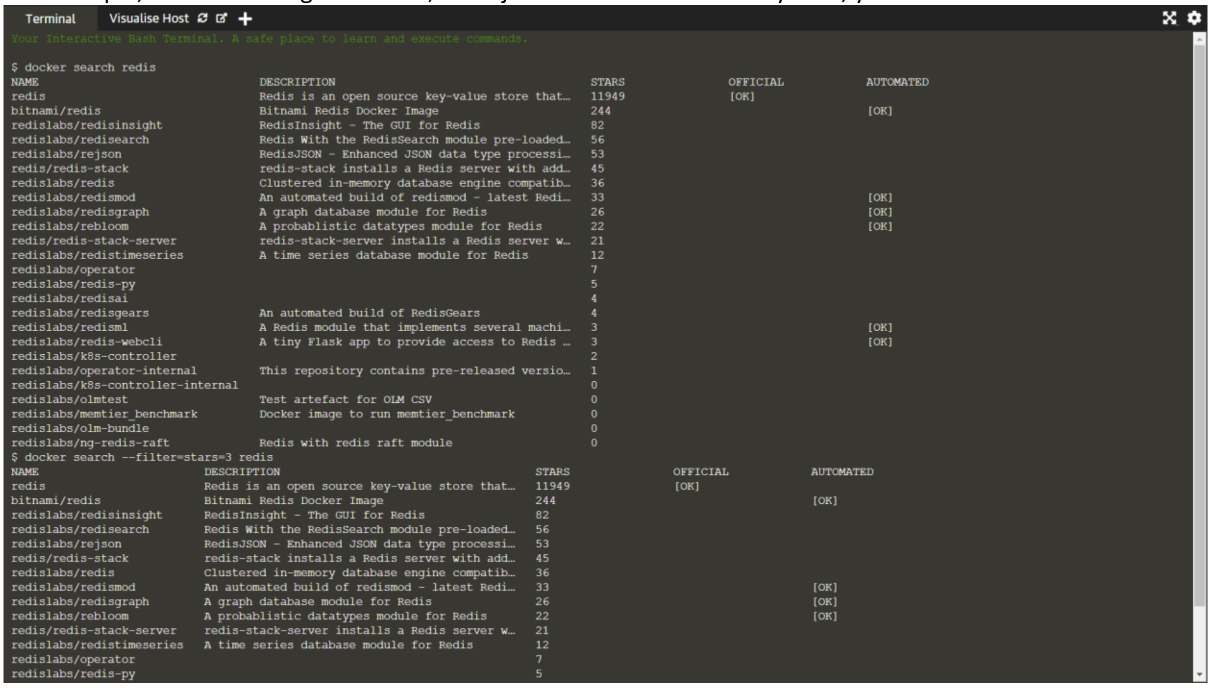
What Is Docker?
Docker describes themselves as "an open platform for developers and sysadmins to build, ship, and run distributed applications".

Docker allows you to run containers. A container is a sandboxed process running an application and its dependencies on the host operating system. The application inside the container considers itself to be the only process running on the machine while the machine can run multiple containers independently. As they're sandboxed, you avoid the possibility of conflicts between dependencies and simplify deployment as all installation and configuration are done ahead of time.

Docker has three key components. First is the Docker Engine, which provides a way to start containers on multiple different operating system platforms. Second is the Docker client, which allows you to communicate with the Engine. Third is the public Docker Registry that hosts Docker Images. These images can be launched or extended to match your requirements and application deployment.

Step 1 - Running A Container
With Docker, all containers are started based on a Docker Image. These images contain everything required to launch the process; the host doesn't require any configuration or dependencies.

To start a container, you can either build your Docker image or, as in this scenario, use an existing image created by Docker and the community.
Existing images can be found at registry.hub.docker.com/ or by using the command
docker search <image-name>.
For example, to find an image for Redis, an object-relational database system, you would use



```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker search redis
NAME                                DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
redis                               Redis is an open source key-value store that…  11949               [OK]
bitnami/redis                       Bitnami Redis Docker Image                      244                                     [OK]
redislabs/redisinsight              RedisInsight - The GUI for Redis                82
redislabs/redisearch                Redis With the RedisSearch module pre-loaded…   56
redislabs/rejson                    RedisJSON - Enhanced JSON data type processi…   53
redis/redis-stack                   redis-stack installs a Redis server with add…   45
redislabs/redis                     Clustered in-memory database engine compatib…   36
redislabs/redismod                  An automated build of redismod - latest Redi…   33                                      [OK]
redislabs/redisgraph                A graph database module for Redis               26                                      [OK]
redislabs/rebloom                   A probablistic datatypes module for Redis       22                                      [OK]
redis/redis-stack-server            redis-stack-server installs a Redis server w…   21
redislabs/redistimeseries           A time series database module for Redis         12
redislabs/operator                                                                  7
redislabs/redis-py                                                                  5
redislabs/redisai                                                                   4
redislabs/redisgears                An automated build of RedisGears                4
redislabs/redisml                   A Redis module that implements several machi…   3                                       [OK]
redislabs/redis-webcli              A tiny Flask app to provide access to Redis …   3                                       [OK]
redislabs/k8s-controller                                                            2
redislabs/operator-internal         This repository contains pre-released versio…   1
redislabs/k8s-controller-internal                                                   0
redislabs/olmtest                   Test artefact for OLM CSV                       0
redislabs/memtier_benchmark         Docker image to run memtier_benchmark           0
redislabs/olm-bundle                                                                0
redislabs/ng-redis-raft             Redis with redis raft module                    0
$ docker search --filter=stars=3 redis
NAME                        DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
redis                       Redis is an open source key-value store that…  11949               [OK]
bitnami/redis               Bitnami Redis Docker Image                      244                                     [OK]
redislabs/redisinsight      RedisInsight - The GUI for Redis                82
redislabs/redisearch        Redis With the RedisSearch module pre-loaded…   56
redislabs/rejson            RedisJSON - Enhanced JSON data type processi…   53
redis/redis-stack           redis-stack installs a Redis server with add…   45
redislabs/redis             Clustered in-memory database engine compatib…   36
redislabs/redismod          An automated build of redismod - latest Redi…   33                                      [OK]
redislabs/redisgraph        A graph database module for Redis               26                                      [OK]
redislabs/rebloom           A probablistic datatypes module for Redis       22                                      [OK]
redis/redis-stack-server    redis-stack-server installs a Redis server w…   21
redislabs/redistimeseries   A time series database module for Redis         12
redislabs/operator                                                          7
redislabs/redis-py                                                          5
```

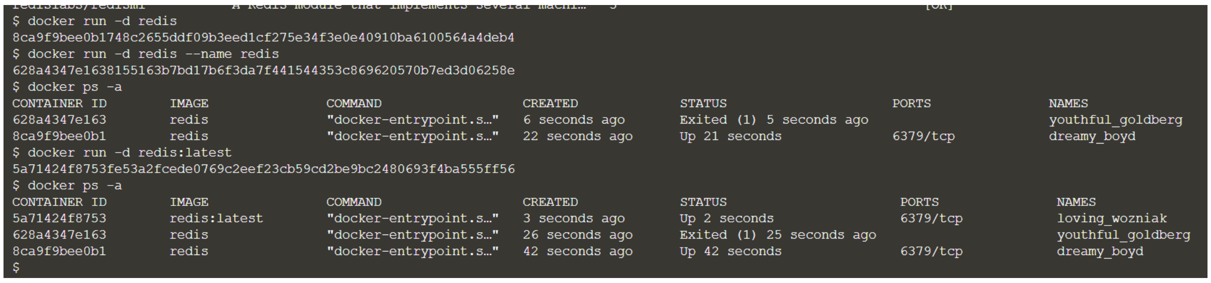
The --filter=stars=3 option indicates that only images with at least three stars should be displayed. People give projects they recommend a star rating on the Docker Registry. It's recommended you either go with the "Official" docker image which has been verified by Docker, or the one with the most stars.

Task
To complete this step, launch a container in the background running an instance of Redis based on the official image.

After identifying the image name using search, you can launch it using docker run <options> <image-name>. By default, Docker will run a command in the foreground. To run in the background, you need to specify the option -d.

At this stage, you will not have a local copy of the image so that it will be downloaded from the Docker registry. If you launch a second container, the local image will be used.

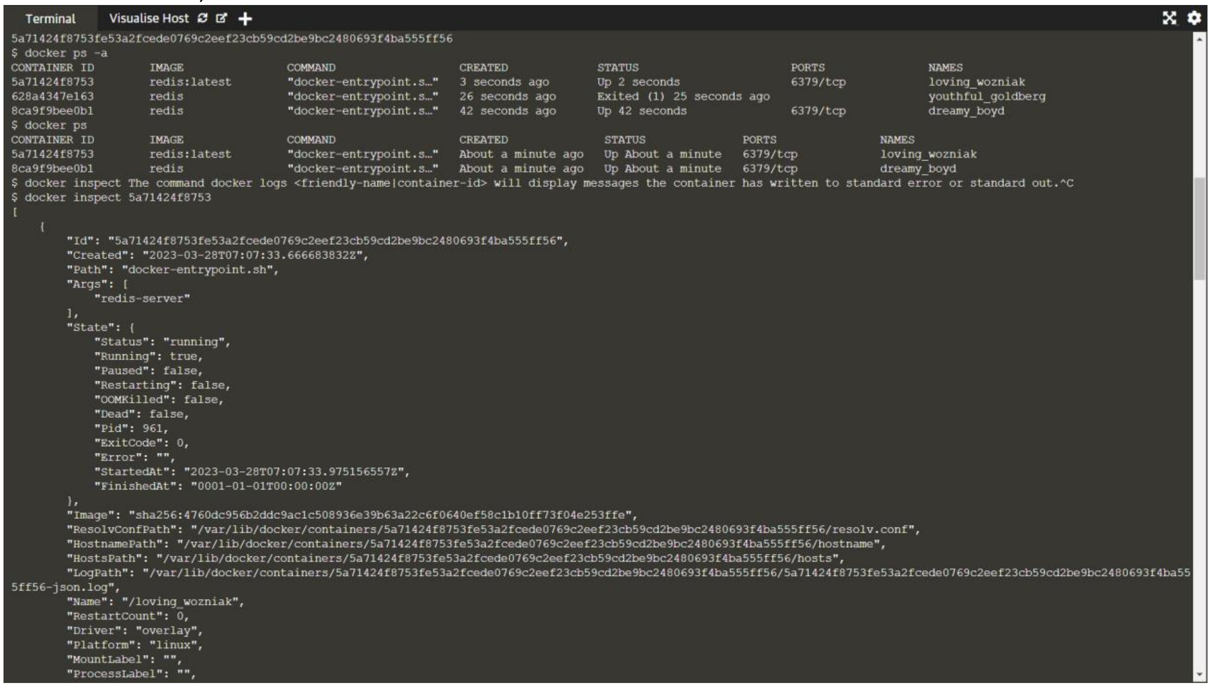
Protip
All containers are given a name and id for use in other Docker commands. You can set the friendly name by providing the option --name <new-name> when launching a container such as --name redis



```
$ docker run -d redis
8ca9f9bee0b1748c2655ddf09b3eed1cf275e34f3e0e40910ba6100564a4deb4
$ docker run -d redis --name redis
628a4347e1638155163b7bd17b6f3da7f441544353c869620570b7ed3d06258e
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
628a4347e163        redis               "docker-entrypoint.s…"   6 seconds ago       Exited (1) 5 seconds ago                        youthful_goldberg
8ca9f9bee0b1        redis               "docker-entrypoint.s…"   22 seconds ago      Up 21 seconds               6379/tcp            dreamy_boyd
$ docker run -d redis:latest
5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba555ff56
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                       PORTS               NAMES
5a71424f8753        redis:latest        "docker-entrypoint.s…"   3 seconds ago       Up 2 seconds                 6379/tcp            loving_wozniak
628a4347e163        redis               "docker-entrypoint.s…"   26 seconds ago      Exited (1) 25 seconds ago                        youthful_goldberg
8ca9f9bee0b1        redis               "docker-entrypoint.s…"   42 seconds ago      Up 42 seconds                6379/tcp            dreamy_boyd
$
```

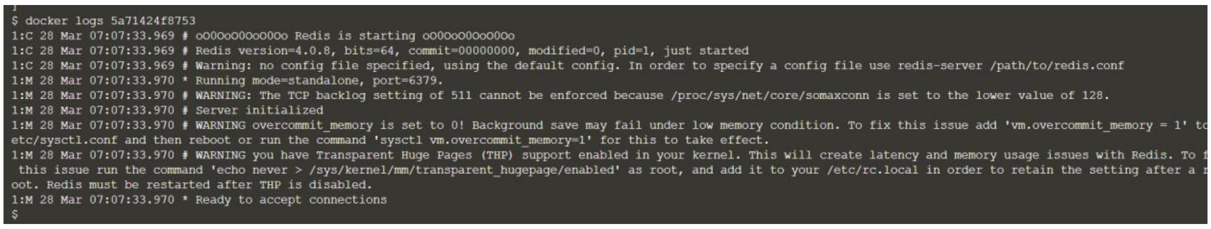
Step 2 - Listing Running Containers
While the launched container is running in the background, the docker ps command lists all running containers, the image used to start the container and uptime.

This command also displays the friendly name and ID that can be used to find out information about individual containers.
The command docker inspect <friendly-name|container-id> provides more details about a running container, such as IP address, volumes mounted and their locations and its current execution state.



```
5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba555ff56
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
5a71424f8753        redis:latest        "docker-entrypoint.s…"   3 seconds ago       Up 2 seconds                6379/tcp            loving_wozniak
628a4347e163        redis               "docker-entrypoint.s…"   26 seconds ago      Exited (1) 25 seconds ago                       youthful_goldberg
8ca9f9bee0b1        redis               "docker-entrypoint.s…"   42 seconds ago      Up 42 seconds               6379/tcp            dreamy_boyd
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS               NAMES
5a71424f8753        redis:latest        "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp            loving_wozniak
8ca9f9bee0b1        redis               "docker-entrypoint.s…"   About a minute ago   Up About a minute   6379/tcp            dreamy_boyd
$ docker inspect The command docker logs <friendly-name|container-id> will display messages the container has written to standard error or standard out.^C
$ docker inspect 5a71424f8753
[
    {
        "Id": "5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba555ff56",
        "Created": "2023-03-28T07:07:33.666683832Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "redis-server"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 961,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2023-03-28T07:07:33.975156557Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:4760dc956b2ddc9ac1c508936e39b63a22c6f0640ef58c1b10ff73f04e253ffe",
        "ResolvConfPath": "/var/lib/docker/containers/5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba555ff56/resolv.conf",
        "HostnamePath": "/var/lib/docker/containers/5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba555ff56/hostname",
        "HostsPath": "/var/lib/docker/containers/5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba555ff56/hosts",
        "LogPath": "/var/lib/docker/containers/5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba555ff56/5a71424f8753fe53a2fcede0769c2eef23cb59cd2be9bc2480693f4ba55
5ff56-json.log",
        "Name": "/loving_wozniak",
        "RestartCount": 0,
        "Driver": "overlay",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
```

The command docker logs <friendly-name|container-id> will display messages the container has written to standard error or standard out.



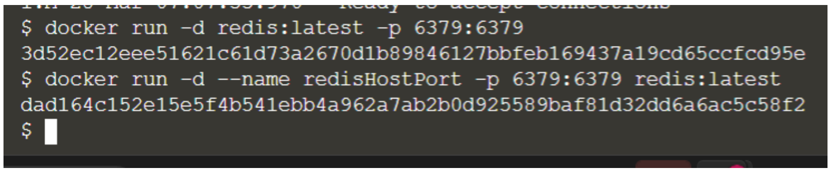
```
$ docker logs 5a71424f8753
1:C 28 Mar 07:07:33.969 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 28 Mar 07:07:33.969 # Redis version=4.0.8, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 28 Mar 07:07:33.969 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 28 Mar 07:07:33.970 * Running mode=standalone, port=6379.
1:M 28 Mar 07:07:33.970 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
1:M 28 Mar 07:07:33.970 # Server initialized
1:M 28 Mar 07:07:33.970 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to
etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 28 Mar 07:07:33.970 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To f
 this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a r
oot. Redis must be restarted after THP is disabled.
1:M 28 Mar 07:07:33.970 * Ready to accept connections
$
```

Step 3 - Binding Ports
As described in the introduction, each container is sandboxed from other containers. If a service needs to be accessible externally, then you need to expose a port to be mapped to the host. Once mapped, you will then be able to access the service as if the process was running on the host OS itself instead of in a container.

When starting the container, you define which ports you want to bind using the -p <host-port>:<container-port> option. The Redis container exposes the service on port 6379. If you wanted to map this port directly on the host, we'd use the option -p 6379:6379.

Task
Start a new Redis container in the background with the name redis and bind the host port 6379 to the container port 6379.

Protip
By default, the port on the host is mapped to 0.0.0.0, which means all IP addresses. You can specify a particular IP address when you define the port mapping, for example, -p 127.0.0.1:6379:6379



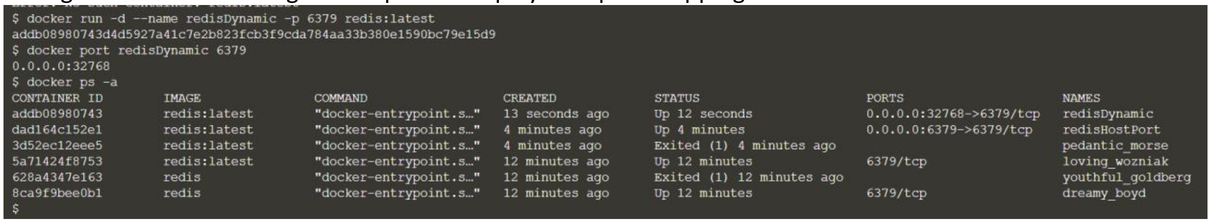
```
$ docker run -d redis:latest -p 6379:6379
3d52ec12eee51621c61d73a2670d1b89846127bbfeb169437a19cd65ccfcd95e
$ docker run -d --name redisHostPort -p 6379:6379 redis:latest
dad164c152e15e5f4b541ebb4a962a7ab2b0d925589baf81d32dd6a6ac5c58f2
$ 
```

Step 4 - Binding Ports
We've seen how to bind a port to a known port on the host. This makes it easier to access the service from other applications but has the disadvantage of only allowing a single instance of the service to be running. One of the advantages of running containers is that you can separate the application configuration (for example, which port to run on) from the deployment configuration (for example, which port to bind on the host).

Like binding a known port on the host, if you simply use -p 6379 to expose the port, then Docker will assign a random available port. This allows you to run multiple instances of the same service on the same host without changing any of the application configuration.

Task
Start an instance of Redis as in the previous task but allow Docker to assign an available port.

You can use the command docker port redis 6379 to find out the mapped port.
Listing the containers using docker ps also displays the port mapping information.



```
$ docker run -d --name redisDynamic -p 6379 redis:latest
addb08980743d4d5927a41c7e2b823fcb3f9cda784aa33b380e1590bc79e15d9
$ docker port redisDynamic 6379
0.0.0.0:32768
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS                     NAMES
addb08980743        redis:latest        "docker-entrypoint.s…"   13 seconds ago      Up 12 seconds               0.0.0.0:32768->6379/tcp   redisDynamic
dad164c152e1        redis:latest        "docker-entrypoint.s…"   4 minutes ago       Up 4 minutes                0.0.0.0:6379->6379/tcp    redisHostPort
3d52ec12eee5        redis:latest        "docker-entrypoint.s…"   4 minutes ago       Exited (1) 4 minutes ago                              pedantic_morse
5a71424f8753        redis:latest        "docker-entrypoint.s…"   12 minutes ago      Up 12 minutes               6379/tcp                  loving_wozniak
628a4347e163        redis               "docker-entrypoint.s…"   12 minutes ago      Exited (1) 12 minutes ago                             youthful_goldberg
8ca9f9bee0b1        redis               "docker-entrypoint.s…"   12 minutes ago      Up 12 minutes               6379/tcp                  dreamy_boyd
$
```

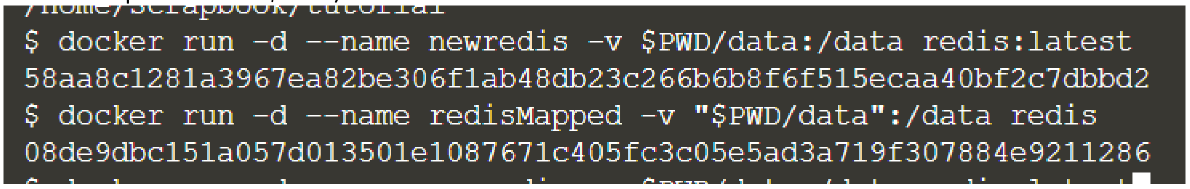
Step 5 - Binding Directories
So far, we've started containers and made them accessible by mounting ports. The next step is handling data.

Containers are designed to be stateless. Any data we want to be persisted after a container is stopped should be saved to the host machine. This is done by mounting/binding host directories into the container.

Binding directories (also known as volumes) in Docker is similar to binding ports using the option -v <host-dir>:<container-dir>. When a directory is mounted, the files which exist in that directory on the host can be accessed by the container and any data changed/written to the directory inside the container will be stored on the host. This allows you to upgrade or change containers without losing your data.

Task
The official Redis image stores logs and data into a /data directory. Start the container and mount the container's /data directory to the host /home/scrapbook/tutorial/data.

Protip
Docker allows you to use $PWD as a placeholder for the current directory. For example, the directory above could be replaced with "$PWD/data".



```
/home/scrapbook/tutorial
$ docker run -d --name newredis -v $PWD/data:/data redis:latest
58aa8c1281a3967ea82be306f1ab48db23c266b6b8f6f515ecaa40bf2c7dbbd2
$ docker run -d --name redisMapped -v "$PWD/data":/data redis
08de9dbc151a057d013501e1087671c405fc3c05e5ad3a719f307884e9211286
```

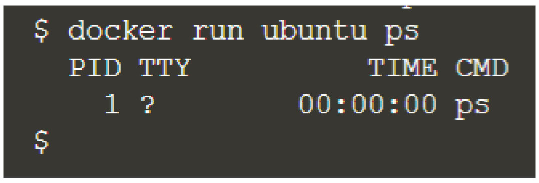
Double quotes are used in this command to ensure that the full path represented by $PWD/data is correctly interpreted even if it contains spaces or special characters.

Step 6 - Running A Container In The Foreground
Certain containers, such as databases, are best run in the background. However, Docker is not limited to just running background services. Containers can run any application in the same way they would run on a regular host. Previously, we used the -d to execute the container in a detached, background, state. Without specifying this, the container would run in the foreground. If we wanted to interact with the container (for example, to access a command shell) instead of just seeing the output, we'd include the options -ti.

As well as defining whether the container runs in the background or foreground, certain images allow you to override the command used to launch the image. Being able to replace the default command makes it possible to have a single image that can be re-purposed in multiple ways. For example, the Ubuntu image can either run OS commands or run an interactive bash prompt using /bin/bash

Example
The command docker run ubuntu ps launches an Ubuntu container and executes the command ps to view all the processes running in a container.

As we described at the start, from the container's point of view, the only process running is the one we launched.



```
$ docker run ubuntu ps
  PID TTY          TIME CMD
    1 ?        00:00:00 ps
$
```

Here we've demonstrated how to start containers in the foreground and background, bind ports and map directories. These commands are the cornerstone of running Docker in both development and production environments.