

**Submitted by: Submitted to:**

**NAME: Rahul Gusain Dr. Hitesh Kumar Sharma**

**SAP: 500084143**

**ROLL NO.: R214220900**

**BATCH: 1 DevOps**

# Build and Release Management

**LAB EXERCISE 9**

**Aim: Create a Docker Volume and Complete the following Scenario on Katacoda.**

https://www.katacoda.com/courses/docker/persisting-data-using-volumes

In this scenario, we'll learn how to use Docker Volumes to persist data within Containers. Docker Volumes allow directories to be shared between containers and container versions.

Docker Volumes allows you to upgrade containers, restart machines and share data without data loss. This is essential when updating database or application versions.

**Step 1 - Data Volumes**

Docker Volumes are created and assigned when containers are started. Data Volumes allow you to map a host directory to a container for sharing data.

This mapping is bi-directional. It allows data stored on the host to be accessed from within the container. It also means data saved by the process inside the container is persisted on the host.

**Task**

This example will use Redis as a way to persist data. Start a Redis container below, and create a data volume using the -v parameter. This specifies that any data saved inside the container to the /data directory should be persisted on the host in the directory /docker/redis-data.

docker run -v /docker/redis-data:/data \ --name r1 -d redis \ redis-server -- We can pipe data into the Redis instance using the following command.

cat data | docker exec -i r1 redis-cli --pipe

Redis will save this data to disk. On the host we can investigate the mapped direct which should contain the Redis data file.

ls /docker/redis-data

This same directory can be mounted to a second container. One usage is to have a Docker Container performing backup operations on your data.

docker run -v /docker/redis-data:/backup ubuntu ls /backup

Persisting Data Using Volumes

**Step 2 - Shared Volumes**

Data Volumes mapped to the host are great for persisting data. However, to gain access to them from another container you need to know the exact path which can make it error prone.

An alternate approach is to use -volumes-from. The parameter maps the mapped volumes from the source container to the container being launched.

In this case, we're mapping our Redis container's volume to an Ubuntu container. The /data directory only exists within our Redis container, however, because of -volumes from our Ubuntu container can access the data.

docker run --volumes-from r1 -it ubuntu ls /data

This allows us to access volumes from other containers without having to be concerned how they're persisted on the host.

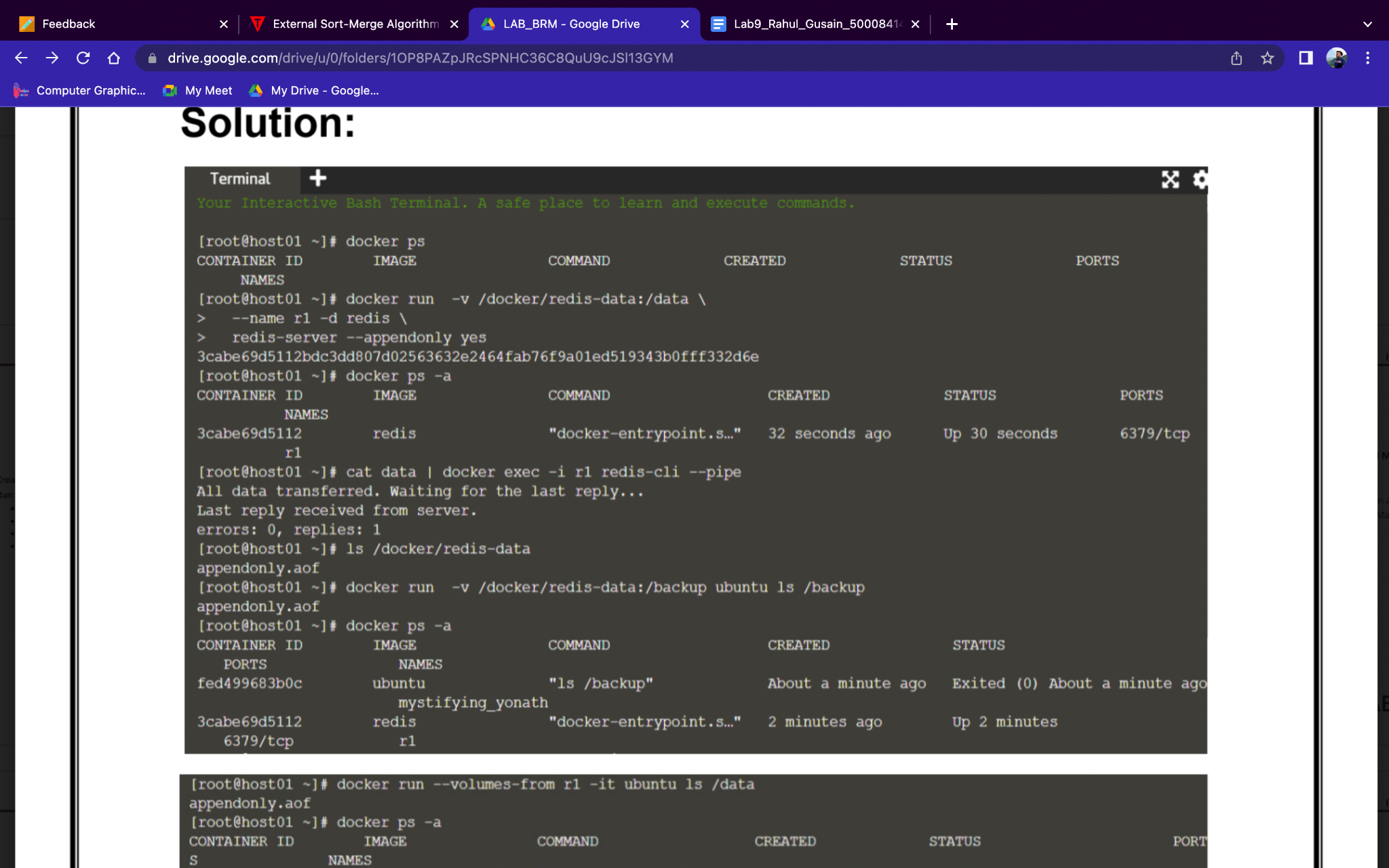
**Step 3 - Read-only Volumes**

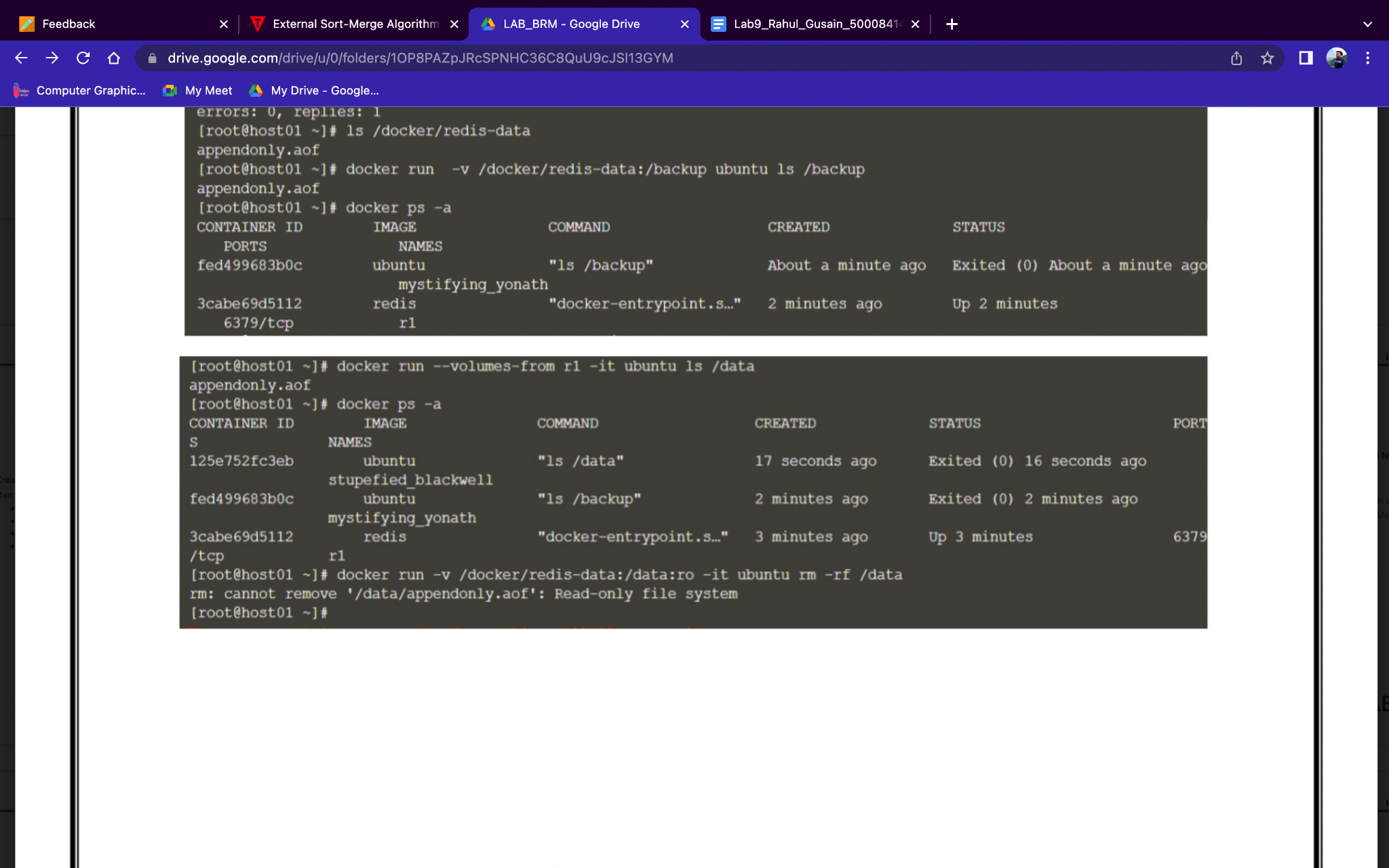
Mounting Volumes gives the container full read and write access to the directory. You can specify read-only permissions on the directory by adding the permissions :ro to the mount.

If the container attempts to modify data within the directory it will error.

docker run -v /docker/redis-data:/data:ro -it ubuntu rm -rf /data

**Solution :**

****

****