Persist a container's state and data

This part is WIP.

## Description

Unlike using virtual machines, containers require that data that must be stored (or persisted) outside of the container itself. A primary reason behind this is that `Docker` containers are not immediately portable between hosts. So in order to enable portability of the container between hosts, data must be persisted outside of the container.

## Persist Data Between Containers

\*\* small presentation on why persistence\*\*

A data volume is a specially-designated directory within one or more containers that bypasses the Union File System. Data volumes provide several useful features for persistent or shared data:

\* Volumes are initialized when a container is created. If the container’s base image contains data at the specified mount point, that existing data is copied into the new volume upon volume initialization.

\* Data volumes can be shared and reused among containers.

\* Changes to a data volume are made directly.

\* Changes to a data volume will not be included when you update an image.

\* Data volumes persist even if the container itself is deleted.

### Adding a data volume

You can add a data volume to a container using the -v flag with the docker create and docker run command. You can use the -v multiple times to mount multiple data volumes. Let’s mount a single volume now in our web application container.

```

docker run -d -P --name web -v /webapp training/webapp python app.py

```

This will create a new volume inside a container at /webapp.

## Creating and mounting a data volume container

If you have some persistent data that you want to share between containers, or want to use from non-persistent containers, it’s best to create a named Data Volume Container, and then to mount the data from it.

$ docker create -v /dbdata --name dbdata training/postgres /bin/true

```

You can then use the --volumes-from flag to mount the /dbdata volume in another container.

```

$ docker run -d --volumes-from dbdata --name db1 training/postgres

```

And another:

```

$ docker run -d --volumes-from dbdata --name db2 training/postgres

You can also extend the chain by mounting the volume that came from the dbdata container in yet another container via the db1 or db2 containers.

```

docker run -d --name db3 --volumes-from db1 training/postgres

```

```

## Docker 1.9 New and Enhanced Features for Data Persistence

Docker can now control the creation of volumes and specification of advanced volume options. Instead of creating volumes directly manually we can create them with Docker:

```

$ docker volume create --name=<yourVolumeName2> --opt=volumetype=io1 --opt=iops=100 --opt=size=10

```