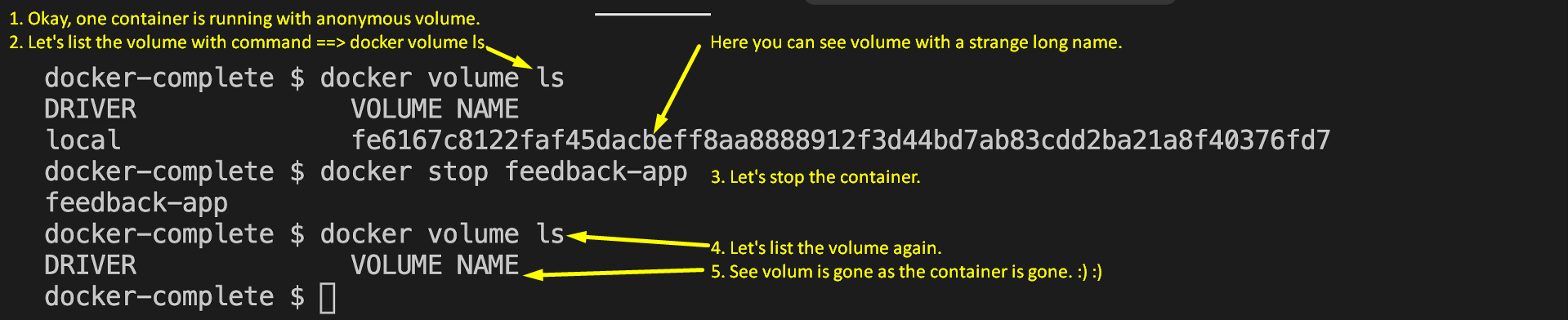
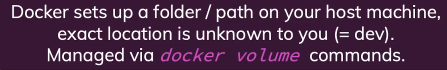
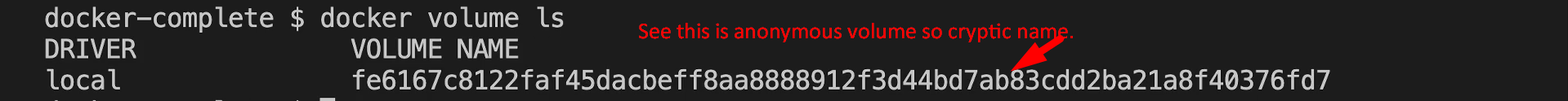
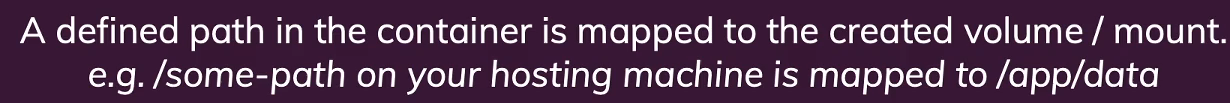
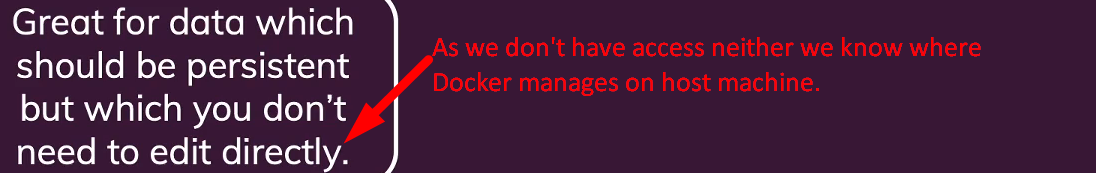
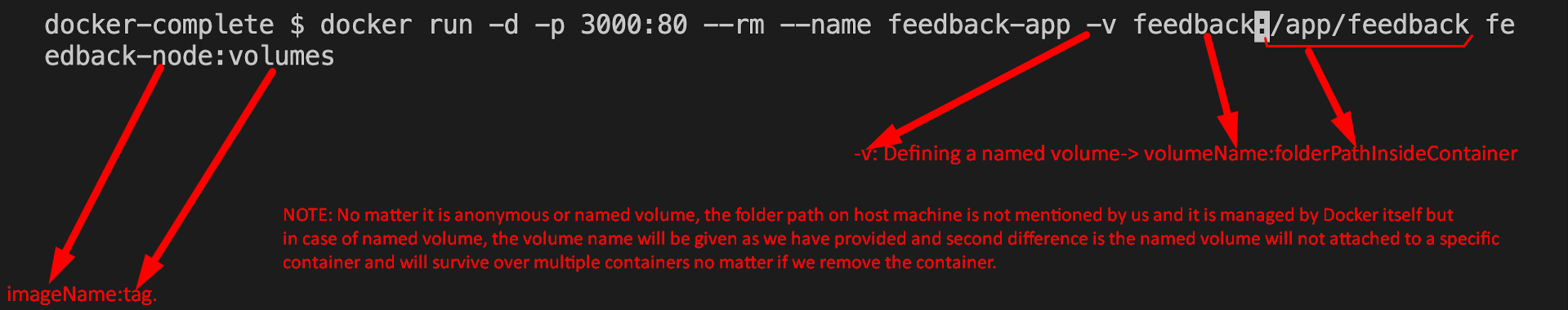
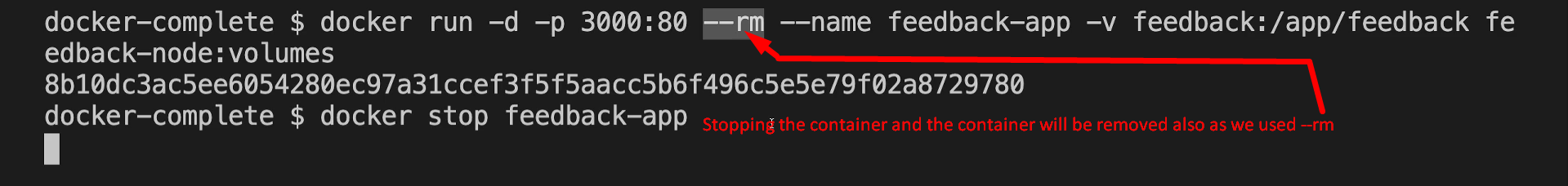
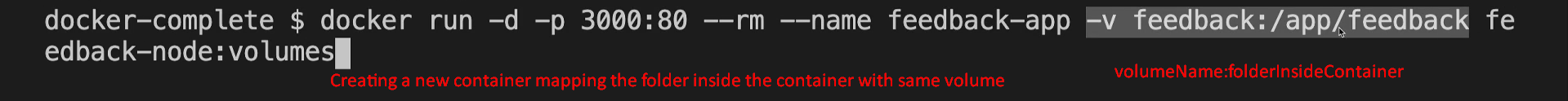
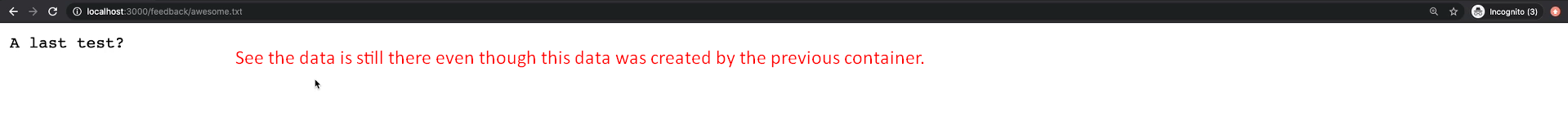
1. Docker provides **two external Data Storage Mechanisms**.
   1. **Volumes**.
   2. **Bind Mounts.**
2. Let’s focus on **Volumes**.
3. We saw **volume** in last lecture, didn’t work as we expected. This is because we have **two kinds of volumes** with their own **special purposes & use cases**.
   1. **Anonymous Volumes**:
      1. In Last lecture, we used **anonymous volume** in an image and therefore in the running container.  
           
         **Anonymous Volume** is defined from inside the **Dockerfile**.
      2. There are **named volumes** too. No matter what kind of **volume**, we use, we specify the **path** inside the **container** and **Docker** will **map** that **path** to some path on Host Machine (We don’t know where it is **mirrored**). But the only way for us to get access to these **volumes** is with the help of the **Docker VOUME Command (anonymous) and -v option (named volume) with “docker run” command**.
      3. An **anonymous volume** is dedicated to a **specific container** whereas named **volume** can be shared among other containers too (Even though named volume creation/initialization starts with a single container but later on the named volumes can be shared with other containers too).
      4. **Gotcha**:
         1. When the container is removed, the **volume** is also removed.
         2. 
      5. **Both kinds of volumes** are managed by **Docker** and we don’t have access.
      6. 
      7.   
         The above **anonymous volume** was created in the previous lecture.
      8. **Anonymous volume** definition is given in Dockerfile itself.
      9. 
         1. Actually, we know **container** has its own file system where our source code creates all files and folders if during app runtime it needs to create (remember image has its own file system but that is read-only and merging both file systems is the effective file system eventually).
         2. Now a particular folder which our app creates we want to map to some folder on our host machine.
         3. So, this mapping is done by **Docker** itself. We just provide the folder name inside our container which we want to map.   
            But don’t know the exact path on the host machine & this is neither concerned to me.
   2. **Named Volumes**:
      1. A **named volume** survives even after container is removed.
      2. 
      3. **Named volume** is not attached to a container unlike **anonymous volume** rather can be shared with other containers too.
      4. **Named Volume** is defined when creating a **container** (not inside Dockerfile as in case of **anonymous** **volume**).
      5. **Defining a volume**:
         1. **-v** : **Jatin**: Why we give name to a volume in case of **named volume**?  
            **Answer**: There are two cases:
            1. Suppose a **container** is running with a **volume** and we want to share the **volume** with other **containers**. So, you must have the **name of volume** of course for reference purpose.
            2. If a **container** is running with a **volume** and we want to replace the **running container** with a **new container** and we want to mount the same **volume** into this new **container**.
      6. Let’s create a **container**, save a file through the web site with name **feedback**, then **delete container** and then **create another container** and see if the previous file with name feedback persisted or not.
         1. Running a container with **named volume** and **--rm** auto remove option.A picture containing text

            Description automatically generated
         2. Accessing the dockerized web app and saving a file using name feedback.  
            Graphical user interface, application

            Description automatically generated
         3. Stopping and removing the container  
            
         4. Listing Volumes:  
            
         5. Creating a new container mapping with the same named volume  
              
            Mapping the outside path to the inside path is called mounting volume.
         6. See the previously saved data.  
            
4. Finally, we managed to persist data.
5. 