**Multistage Docker builds:**

**A whiteboard with text and words

Description automatically generated**

**A whiteboard with writing on it

Description automatically generated**

**A whiteboard with writing on it

Description automatically generated**

**A whiteboard with writing on it

Description automatically generated**

Points:

1) Divide the Docker file to two or multiple stages. Initial stages are the base and last stage will be the final which doesn’t have any packages and dependencies.

2) Base stage has only built activities and that is binary provided to final stage.

4) Final stage has only base image, copy of binary (runtime of java/python) and entry point.

5) If it is Golang, you don’t even need runtime in final stage.

6) Reduces the image size to very less.

7) It is secure**.**

**Distroless Images:**

A diagram of a computer program

Description automatically generated with medium confidence

Points:

1) Distroless image is very minimalistic in nature.

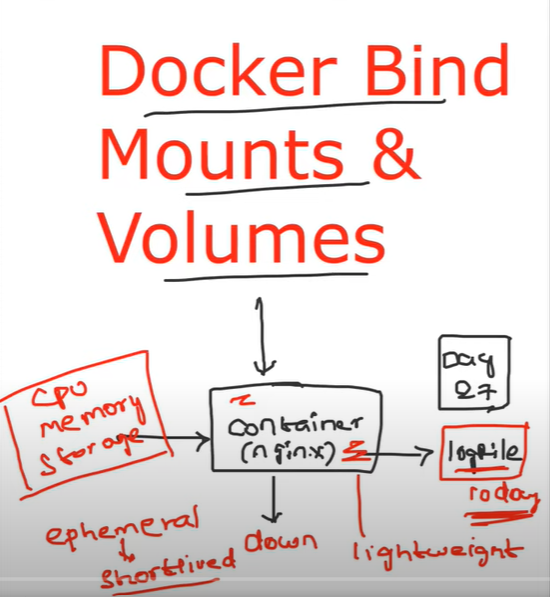
2) It is very secure and free from any OS vulnerability threats.

3) Scratch is the very minimal distroless image for Golang and for python/java get the respective minimal image from google.

**Mount Bind & Volumes:**

Issue 1:

Containers are down so no copy of logfiles of app because container filesystem is completely on hosting server. When it gets to know the container is down, host server free up its related space.



Issue 2:

There are backend writes the file and from front end reads it and display it on UI. What if the back end is down

A diagram of a software system

Description automatically generated with medium confidence

Issue 3:

There is application in container and cron job running at host and generate files. How can app read the file and delivers it to user.

A diagram of a software system

Description automatically generated with medium confidence

Docker has brought two ways to solve above issues.

Attaching the file system to the container from host.

A whiteboard with writing on it

Description automatically generated