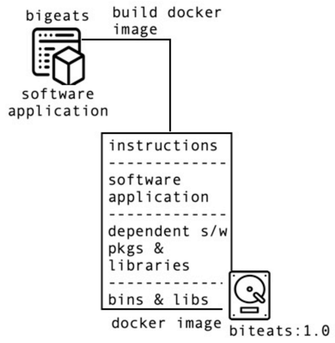
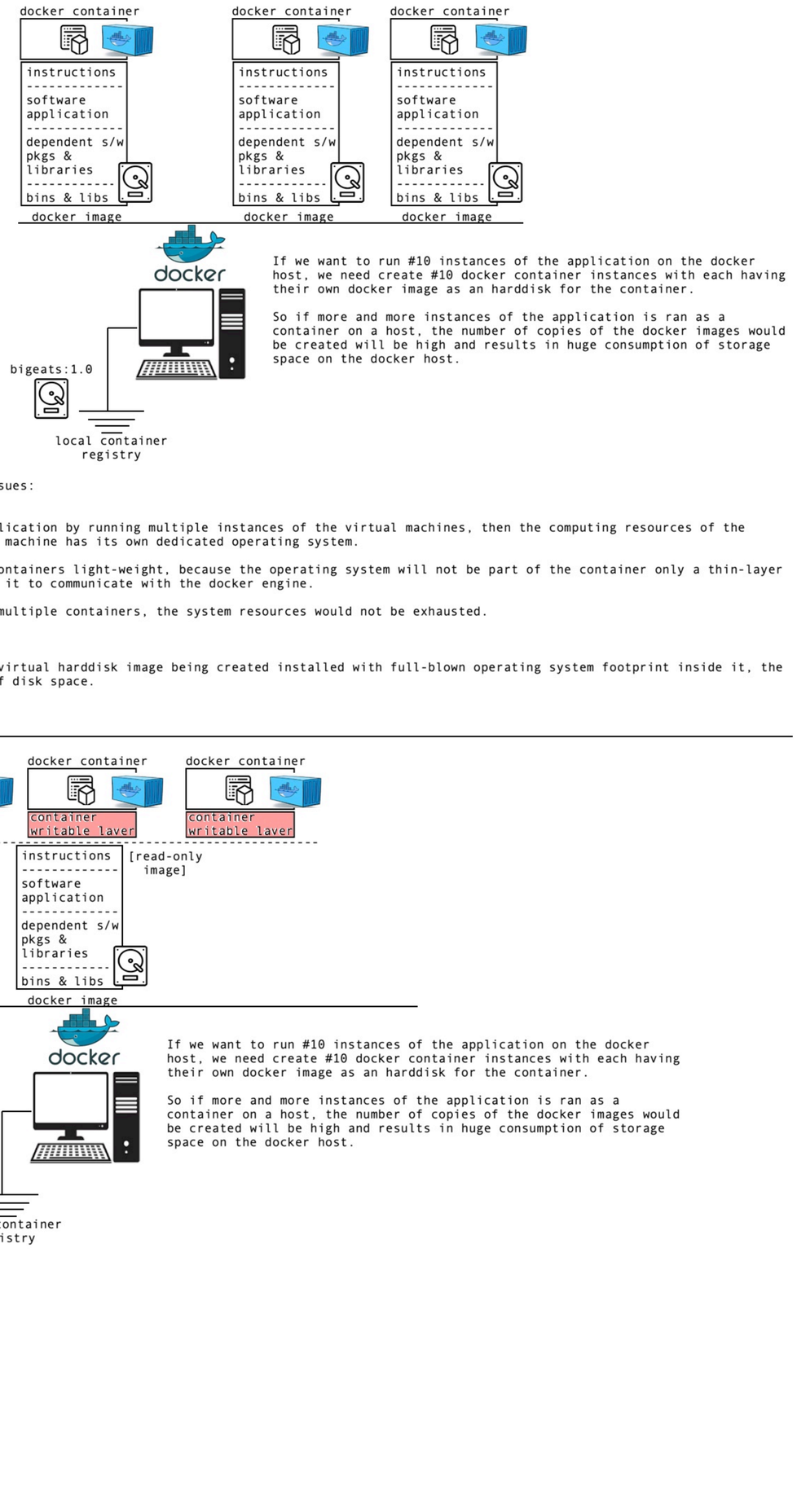
docker images are read-only



Let us assume if docker engine is created one image per one container, then we run into below problems:



If we want to run #10 instances of the application on the docker host, we need create #10 docker container instances with each having their own docker image as an harddisk for the container.

So if more and more instances of the application is ran as a container on a host, the number of copies of the docker images would be created will be high and results in huge consumption of storage space on the docker host.

When we scale an application we usually run into #2 issues:
1. computing resources (cpu, ram etc)

In case of virtual machines, if we try scaling the application by running multiple instances of the virtual machines, then the computing resources of the underlying machine would be exhausted, as each virtual machine has its own dedicated operating system.

whereas the docker addressed this problem by keeping containers light-weight, because the operating system will not be part of the container only a thin-layer which are bins/libs are part of the container enabling it to communicate with the docker engine.

In-short: if we scale the application by running into multiple containers, the system resources would not be exhausted.

2. Storage

Incase of virtual machines, as they have an dedicated virtual harddisk image being created installed with full-blown operating system footprint inside it, the more the virtual machines we run, we quickly run out of disk space.

How does this problem is solved in docker?
That is where docker images are read-only.

If we want to run #10 instances of the application on the docker host, we need create #10 docker container instances with each having their own docker image as an harddisk for the container.

So if more and more instances of the application is ran as a container on a host, the number of copies of the docker images would be created will be high and results in huge consumption of storage space on the docker host.