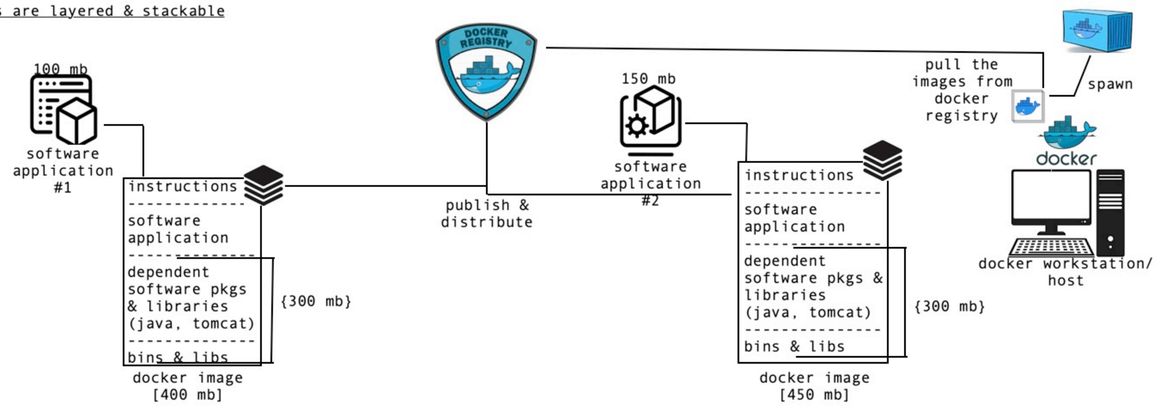
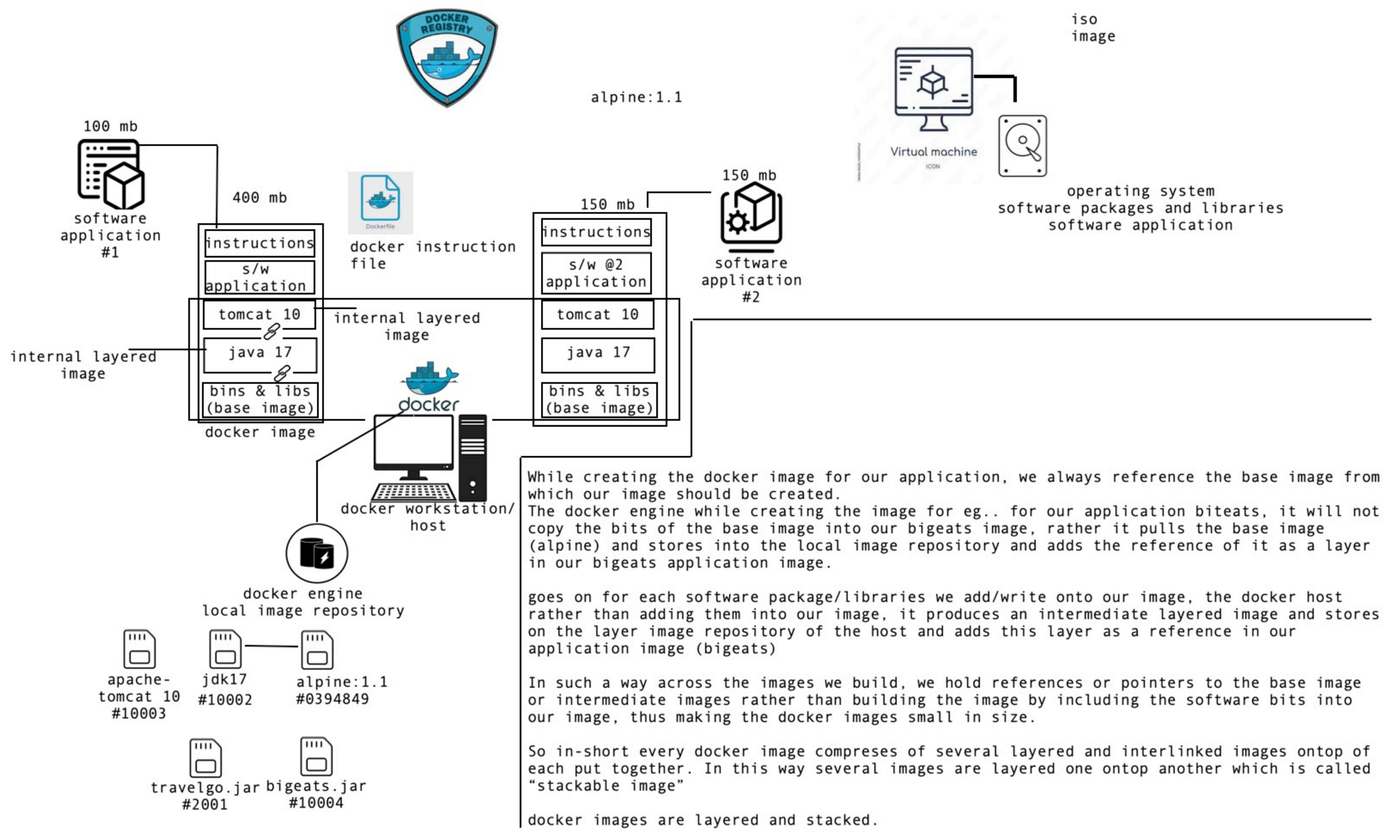
docker images are layered & stackable



If we want to run #2 software applications out of an docker container, then we need to package both the applications into docker images and distribute them through docker registry.
When we are packaging the software applications into docker images, there are bins & libs and few dependent software packages/libraries (given the technology of the applications are similar) seems to be common across the applications are being baked into their respective docker images due to which we run into several problems:
1. By repeatedly packaging the bins & libs and few software packages and libraries which are in common across the image, the size of these images would be bloted up, so that when we are running the packaged applications on the docker host, we endup in holding several images of different applications on the machine where across these images most of the bins & libs and few s/w pkgs would be same, which results in huge consumption of storage in holding these images on the docker host.
2. since the images are huge in size the amount of time it takes in pulling the images or publishing them into the docker container registry will be more (network latency)
3. bandwidth consumption in pulling/publishing the images is very high

To avoid the above problems, and to keep the images light weight and easily distributable docker has comeup with layered and stackable images through linux union filesystem.

When we are creating our own image in packaging and distributing our application, we need to create our image by extending it from the docker base images that are provided by docker team, which are official images of docker. These base images contains bins/libs which are common across all the images that we bake or build.
In-short: we build our own docker images extending from base images that are provided by docker team (most of the time)



While creating the docker image for our application, we always reference the base image from which our image should be created.
The docker engine while creating the image for eg.. for our application biteats, it will not copy the bits of the base image into our bigeats image, rather it pulls the base image (alpine) and stores into the local image repository and adds the reference of it as a layer in our bigeats application image.

goes on for each software package/libraries we add/write onto our image, the docker host rather than adding them into our image, it produces an intermediate layered image and stores on the layer image repository of the host and adds this layer as a reference in our application image (bigeats)

In such a way across the images we build, we hold references or pointers to the base image or intermediate images rather than building the image by including the software bits into our image, thus making the docker images small in size.

So in-short every docker image compreses of several layered and interlinked images ontop of each put together. In this way several images are layered one ontop another which is called “stackable image”

docker images are layered and stacked.

all the earlier problems we have discussed in pulling, distributing and storing the images on the docker host are resolved by using the layered/stackable images:
1. across the images, since the layers might be same, only one copy of the layer will be kept on the docker host, thus reducing the storage foot print in holding the images on the docker host.
2. pulling and publishing the images onto the container registry will be fast, as many of these intermediate layers may be already part of the registry and doesnt require to pull/push again
3. bandwidth consumption on pulling/distributing the images will be very less.