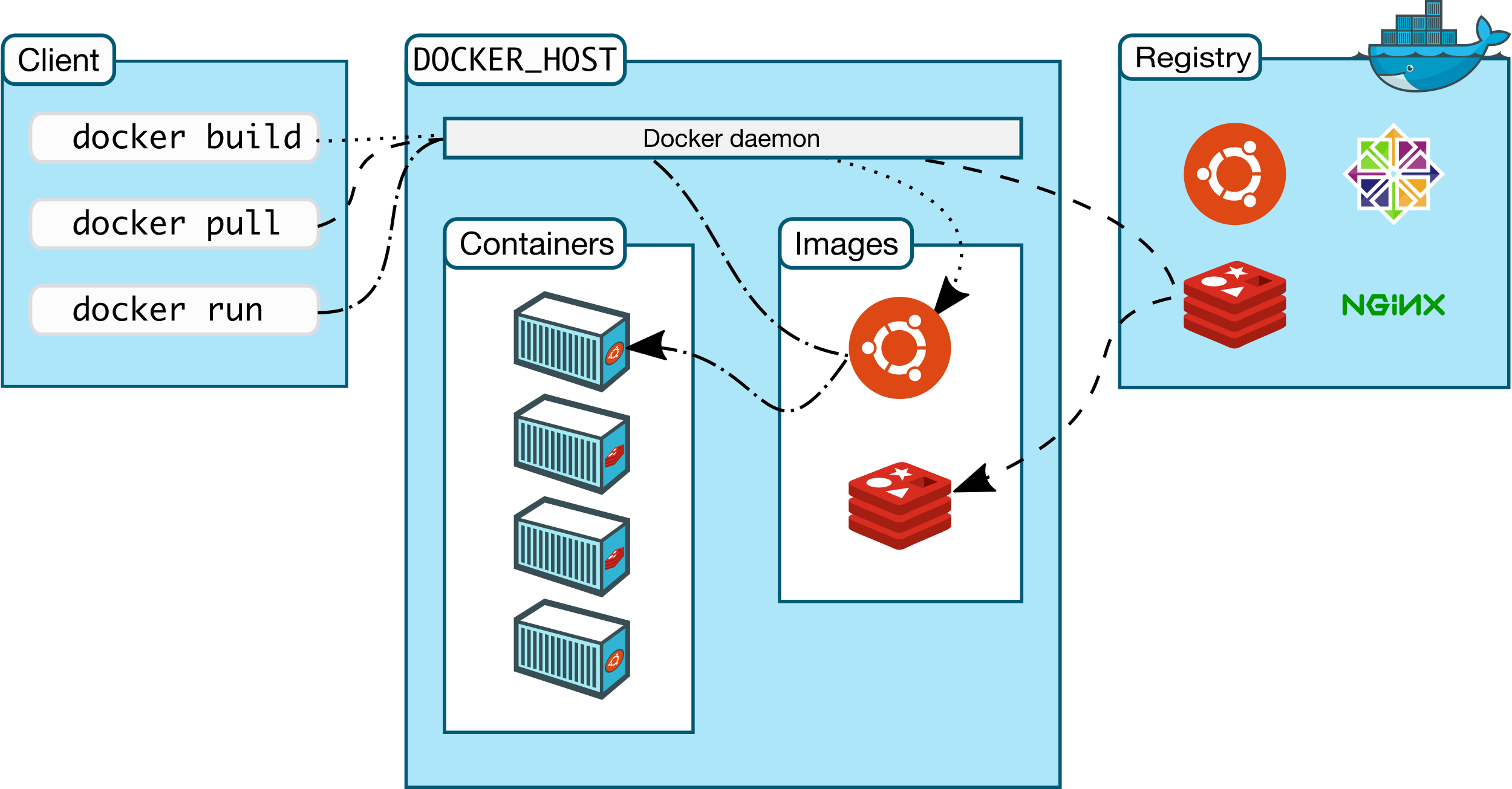
Docker  
  
As we have discussed, containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package  
Docker is the software to facilitate creating, deploying, and running containers

The starting point would be the source code of the Image “Docker file” from them you can build the “Docker Image” this image can be stored and distributed to any registry -most common Docker hub- and finally you use this image to run the containers.

Docker uses a client-server architecture. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon  
The Docker daemon, does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.



Containers doesn’t exists in a vacuum and in production you won’t have just one host with multiple container but multiples of hosts running 100s if not 1000s of containers,  
 which raise two important sets of questions   
  
1- How these containers communicate with each other on the same host or in different host as well with outside world? Basically, the networking part of containers   
2- Who determine which containers get lunch over which host? and based on what? Upgrade? Number of containers per application? Basically, who orchestrate that   
  
  
we will try to answer these two questions in detail for the rest of the book but if you want a quick answer just think of Contrail and Kubernetes !!  
let’s start first lay the basics foundation of contrail