Q2:
Container  has become a standard tool for software developers and system administrators. It’s a neat way to quickly launch applications without impacting the rest of your system. You can spin up a new service with a single `docker run` command.

Container does the OS level virtualization.

Containers encapsulate everything needed to run an application, from OS package dependencies to your own source code. You define a container’s creation steps as instructions in a `Dockerfile`. Docker uses the Dockerfile to construct an *image*.

Working
Containers utilize operating system kernel features to provide partially virtualized environments. It’s possible to create containers from scratch with commands like `chroot`. This starts a process with a specified root directory instead of the system root. But using kernel features directly is fiddly, insecure, and error-prone.

Containers hold the components necessary to run desired software. These components include files, environment variables, dependencies and libraries. The host OS constrains the container's access to physical resources, such as CPU, storage and memory, so a single container cannot consume all of a host's physical resources.