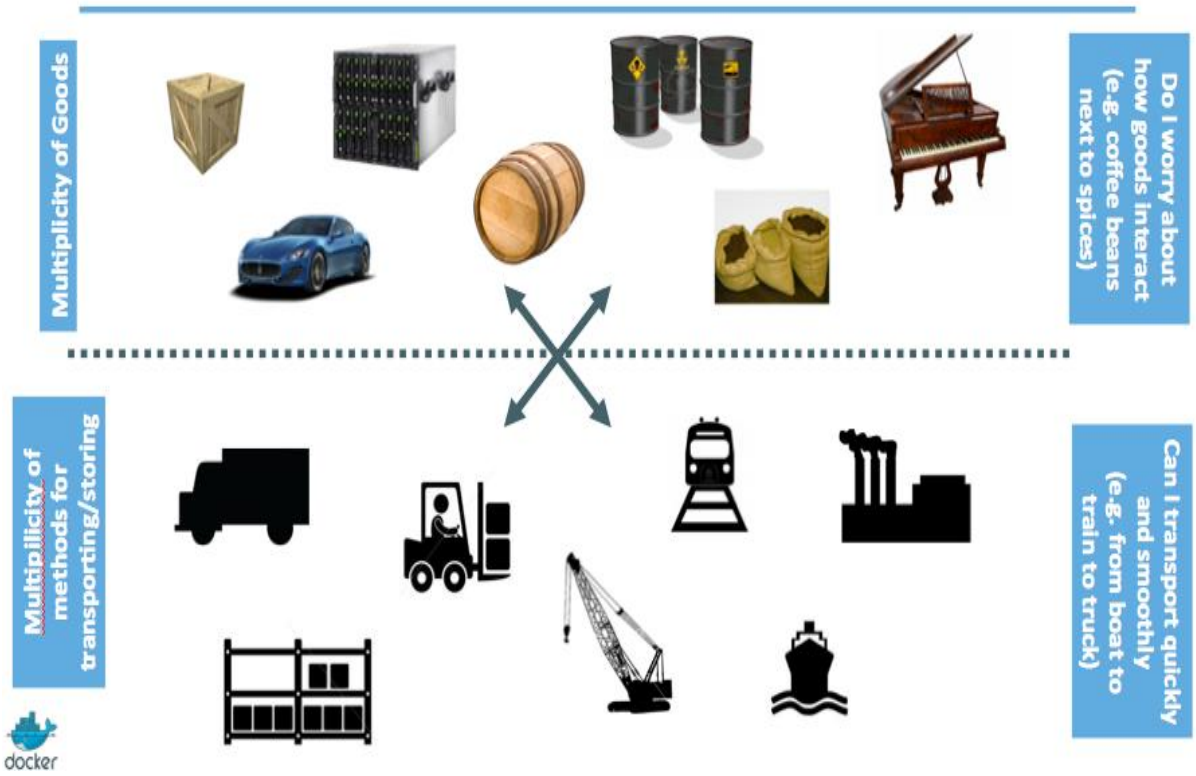
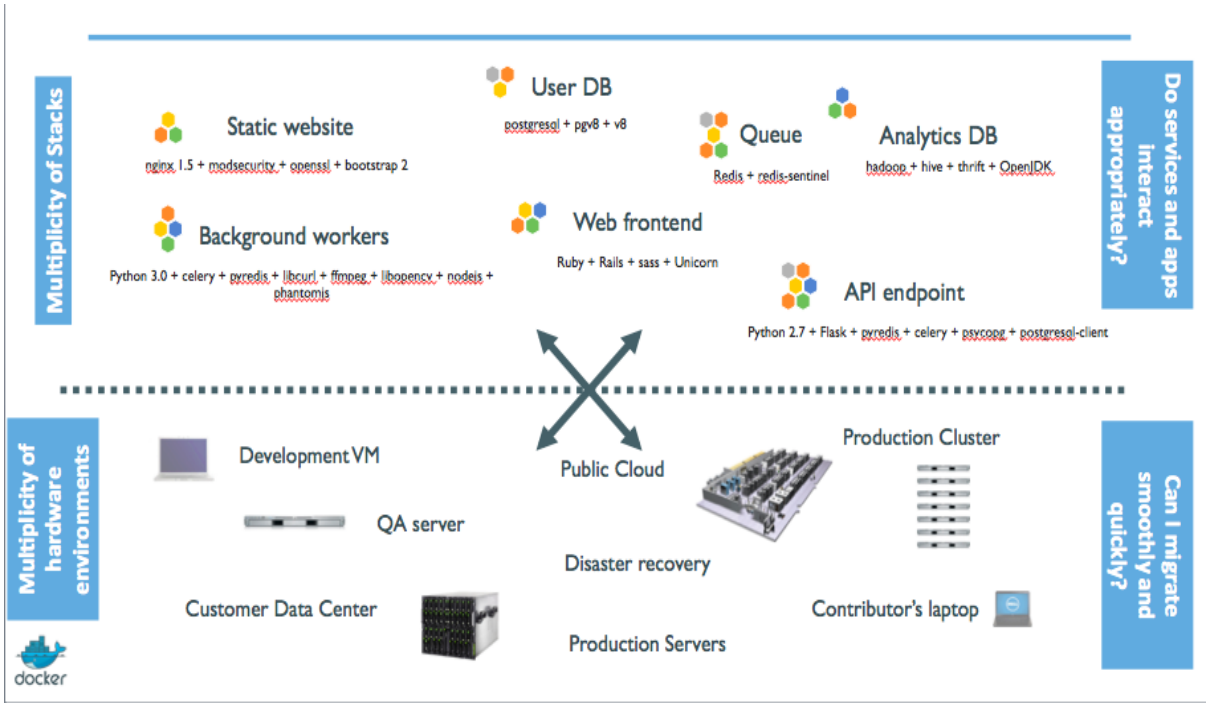
Docker Tutorial

==========

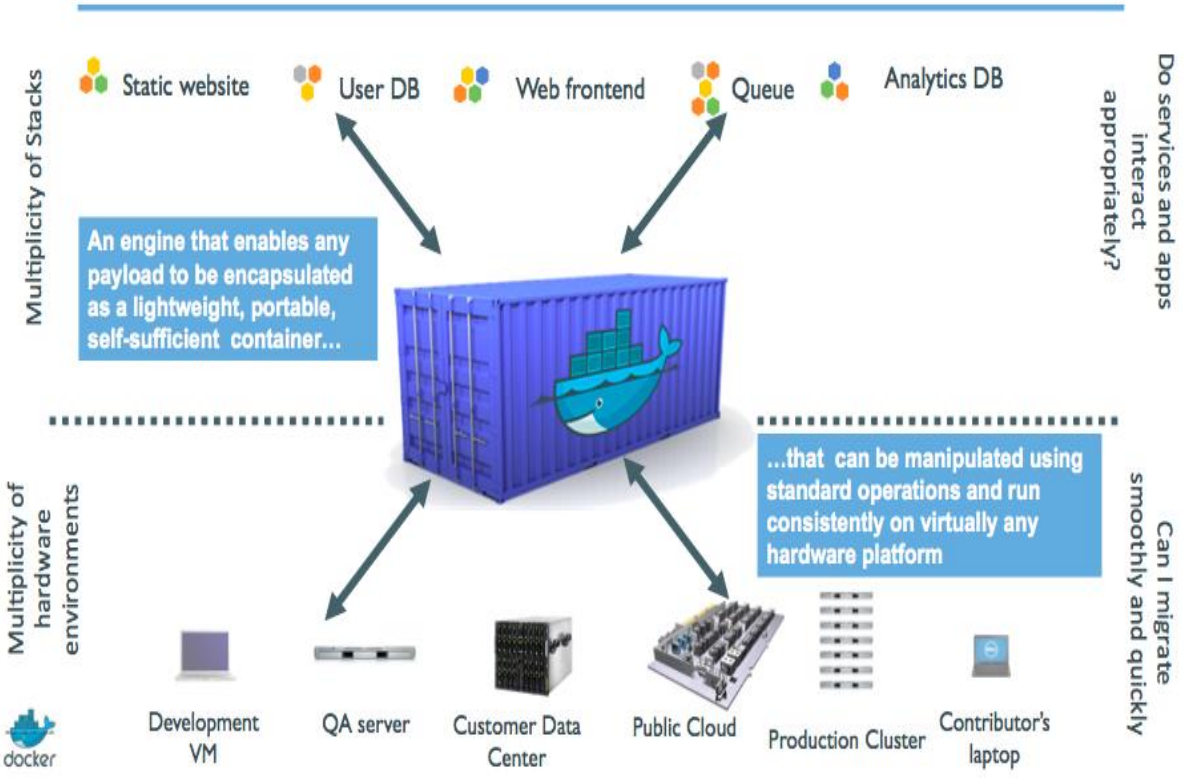
1. Why containers (non-technical elevator pitch).



2. Why containers (technical elevator pitch).



3. How Docker helps us to build, ship, and run.



4. The software industry has changed diff b/w Before and Now.

Before:

- monolithic applications
- long development cycles
- single environment
- slowly scaling up

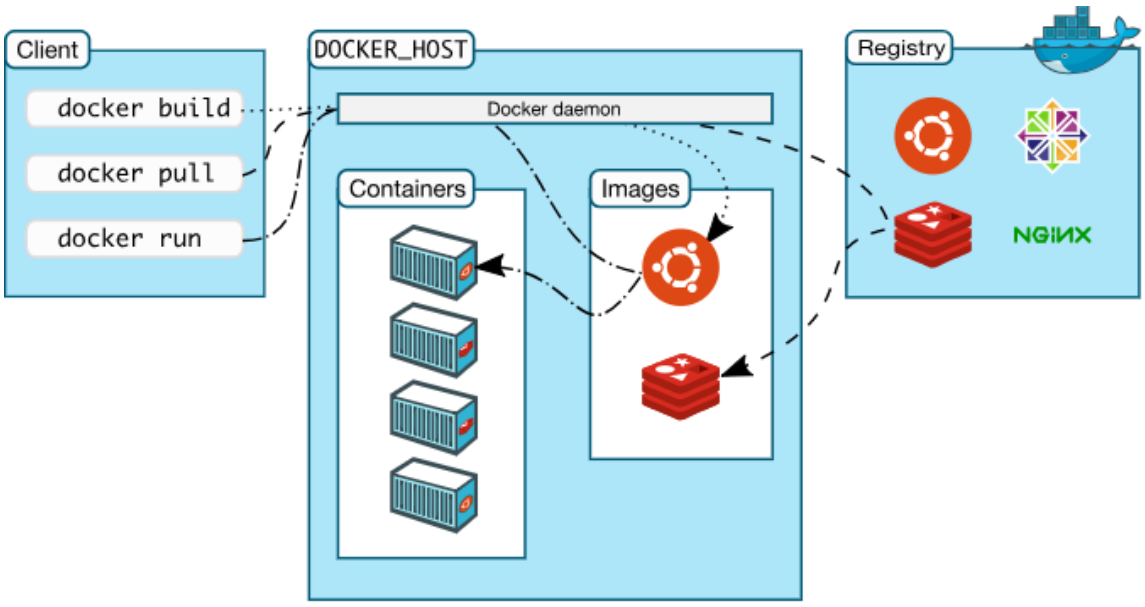
Now:

- decoupled services
- fast, iterative improvements
- multiple environments
- quickly scaling out

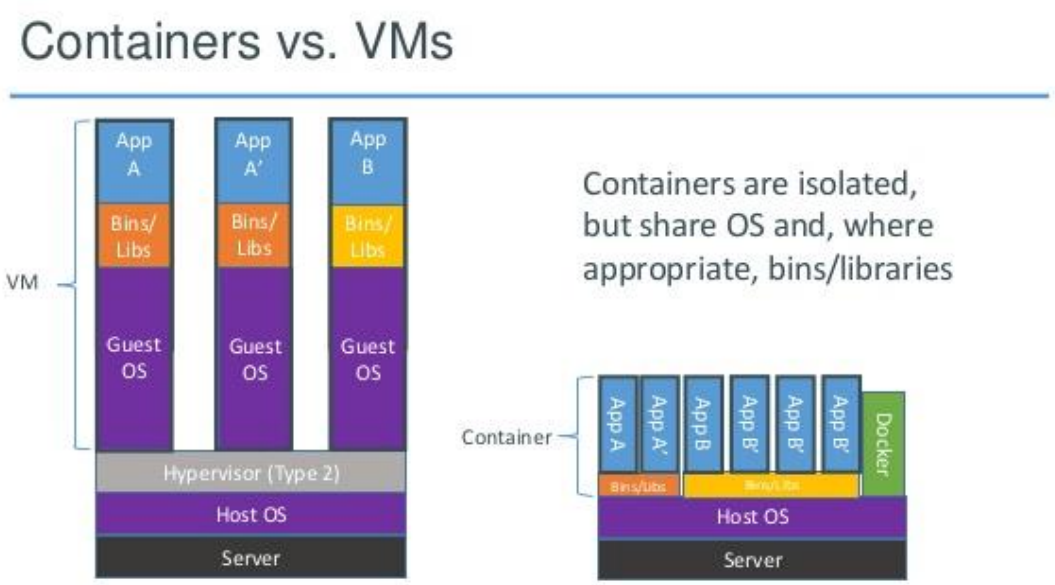
5. Results Of Docker.

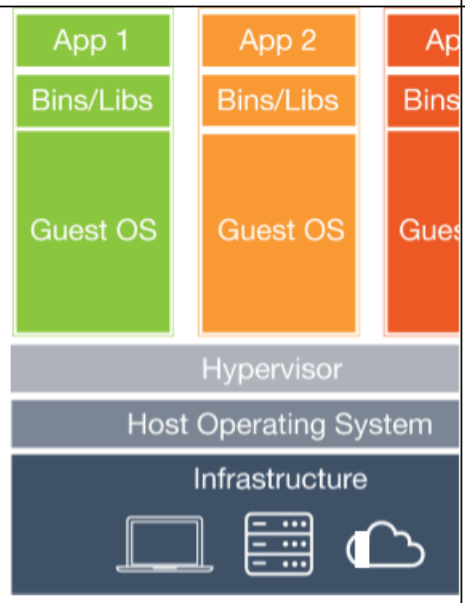
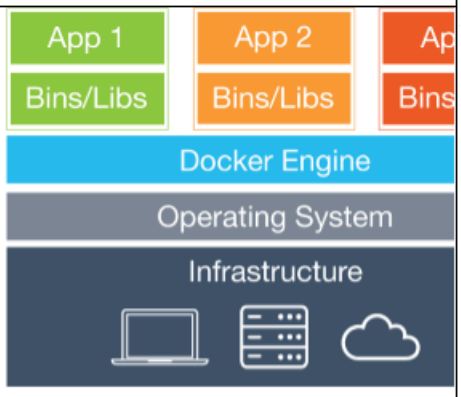
- Dev-to-prod reduced from 9 months to 15 minutes (ING)
- Continuous integration job time reduced by more than 60% (BBC)
- Dev-to-prod reduced from weeks to minutes (GILT)

Docker Architecture.



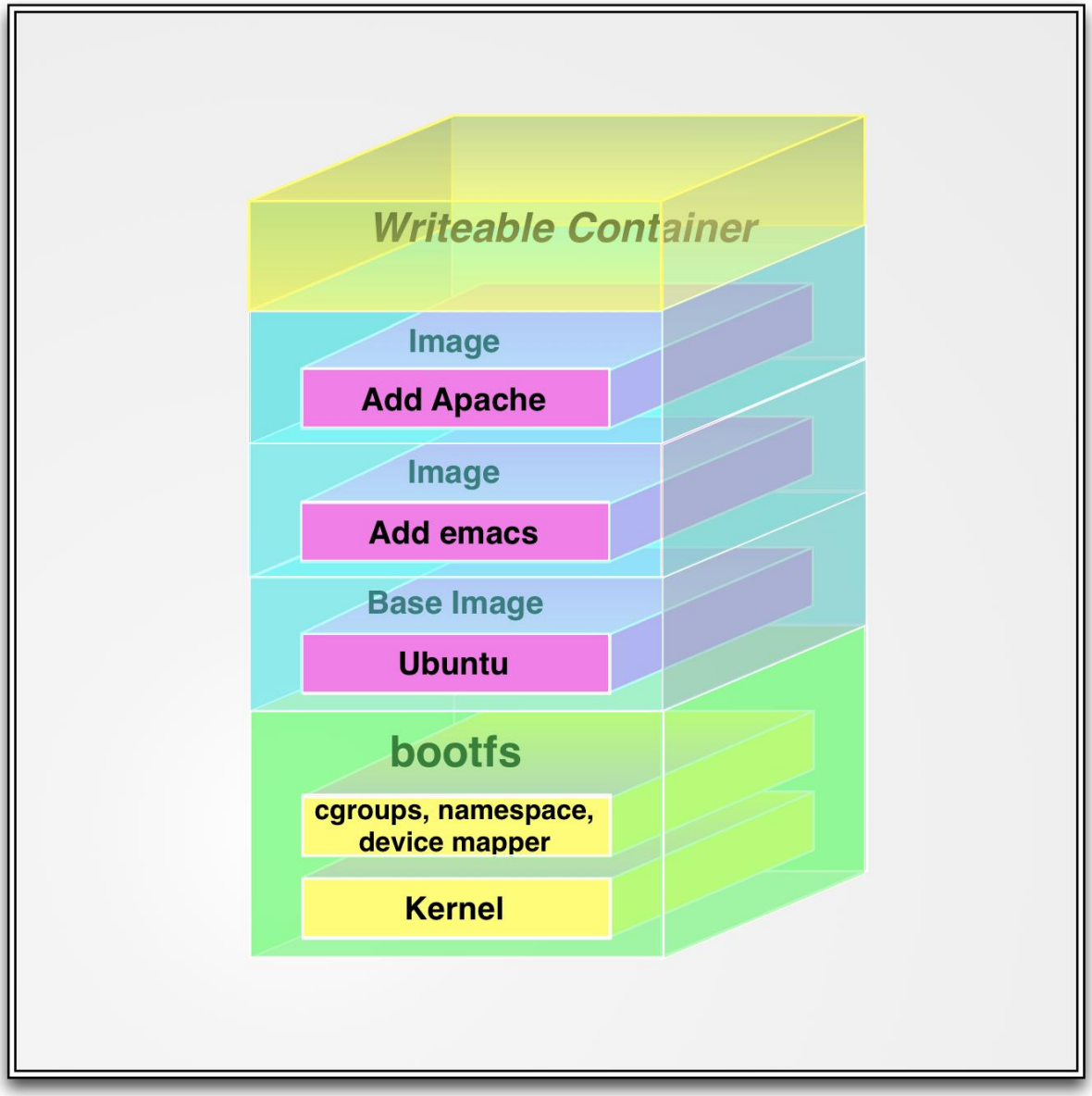
6. Difference between Virtual Machine and Dockers.



Virtual machines	Docker containers
 http://www.docker.com/	 http://www.docker.com/
Virtual machines depend on traditional virtualization. They can be considered hardware-level virtualization.	Containers depends on the containerization technique at the kernel level. They can be considered OS-level virtualization.
Each virtual machine contains a guest OS, binaries, library files, and the application itself.	Each container include application, binaries, and library files, but the major difference compared to virtual machine is the shared kernel: each container runs as an isolated process in the user-space on the host OS.
The size of a virtual machine is in the order of GBs, as each one runs on its own.	As each container runs as an isolated process in the user-space on the host OS and a separate OS is not required for each container, the size of each container is much smaller.
Each virtual machine has its own set of resources, which results in better isolation and less sharing of resources.	Each container shares the kernel, and hence, there's more scope of sharing resources.
A virtual machine cannot run on a container	A container can run on a virtual machine.

7.What is Docker Image?

A Docker image is made up of filesystems layered over each other. At the base is a boot filesystem, bootfs, which resembles the typical Linux/Unix boot filesystem. A Docker user will probably never interact with the boot filesystem. Indeed, when a container has booted, it is moved into memory, and the boot filesystem is unmounted to free up the RAM used by the initrd disk image.image.



layers are by default stored at `/var/lib/docker/graph`.

https://www.projectatomic.io/blog/2015/07/what-are-docker-none-none-images/

8. Listing Docker images

$ docker images

9. Pulling images

Docker pull <images name>

10. Searching for images.

Docker search <image name>

11. Building our own images.

Docker build –t <imagename> .

12. Create Docker Hub account.

13. Login to Docker Hub trough Docker CLI.

Docker login

14. Push to custom image to Docker Hub.

Docker push <imagename>

15. Run Docker Image as a Docker Container.

Docker run –it –p 8181:8080 –name <container name> <image name>

16. Remove Docker images.

Docker rmi <imagename >/<image_id>

17. Remove Docker Containers.

Docker rm <container name>/<container_id>

18. Remove Docker Multiple Docker images at a Time.

Docker rmi <image_id> <image_Id>

19. Remove <none> Docker images (dangling images) .

docker rmi -f “dangling=true” –q
docker rmi -f $(docker images -f "dangling=true" -q)

20. Stop Docker Container.

Docker stop <container_id>

21. Kill Docker Container.

Docker kill <container_id>

22. Difference between Docker kill and stop.

23. Listing Running Containers.

Docker ps

24. Listing Running and stopped containers.

Docker ps -a

Docker ps –f “status=exited”

25. Check the logs of running containers.

Docker logs <container_Id>

26. Follow up the logs of running containers for troubleshoot.

Docker

27. Login inside running container and troubleshoot.
28. Exit from the container.
29.