Using Conda and Docker for Bioinformatics Analysis

Ian Yi-Feng Chang Ph.D

Molecular Medicine Research Center

Chang Gung University

2019/5/10

Credit:

- https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b
- https://yeasy.gitbooks.io/docker_practice/ Docker — 从入门到实践
- Introduction to Docker Containers: https://www.youtube.com/watch?v=wUV9a6IimVc

Outline

- What is conda?
- Why using conda?
- What is docker?
- Why using docker?
- Lab

Conda

What is conda?



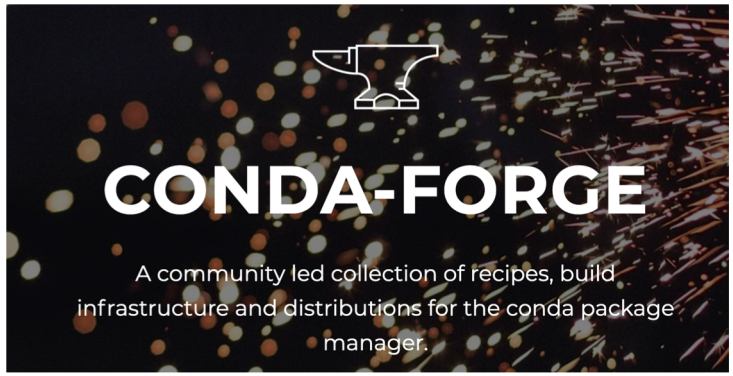
- *Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN*
- Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies.
- Conda easily creates, saves, loads and switches between environments on your local computer.
- It was created for Python programs, but it can package and distribute software for any language.

https://docs.conda.io/en/latest/

Update, search, install, and activate

- conda update conda
- conda search biopython
- conda create --name snowflakes biopython python=3.7
- conda activate snowflakes

Conda channels



- defaults
- conda-forge: https://conda-forge.org/
- Bioconda: https://bioconda.github.io/
  - **Bioconda** is a channel for the conda package manager specializing in bioinformatics software. Bioconda consists of:
  - a repository of recipes hosted on GitHub
  - a build system turning these recipes into conda packages
  - a repository of packages containing over 6000 bioinformatics packages ready to use with conda install
  - over 600 contributors and 500 members who add, modify, update and maintain the recipes

BIOCONDA®

How to use bioconda

- conda config --add channels defaults
- conda config --add channels bioconda
- conda config --add channels conda-forge
- Any packages on the bioconda channel can be installed into the current conda environment:
    - conda install bwa
- Use a new environment can be created
    - conda create -n aligners bwa bowtie hisat star

Conda env is portable

- If your computer broken, how to migrate your env to a new computer?
- conda env create -f environment.yml
- source activate {env_name}
- But installation will take a while

insight.yaml

```yaml
name: insight
channels:
- defaults
- conda-forge
- ericmjl
dependencies:
- python=3.6
- matplotlib
- numpy
- pandas
- scipy
- seaborn
- jupyter
- jupyterlab
- pymc3
- scikit-learn
- statsmodels
- ipykernel
- flask
- bokeh
- rise
- sqlalchemy
- psycopg2
- environment_kernels
- sqlalchemy-utils
```

Docker

Historic Timeline of Unix Containers

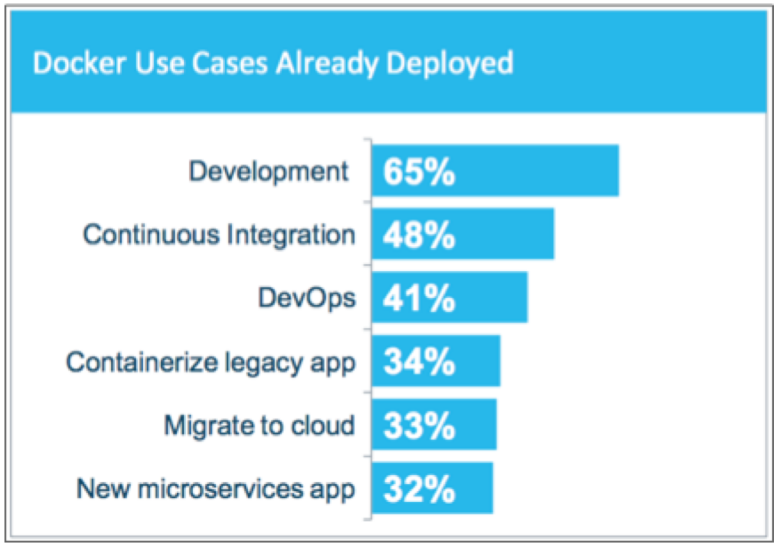
Docker is both a Company and Technology

While Docker has been playing a key role in adoption of the Linux container technology, they did not invent the concept of containers

However, they have made the technology consumable by mere humans



Excellent Use Cases for Containers



Ready to Run Application Stacks

- Excellent for Dev/Test setups
- Deployment in Seconds, not Hours/Days
- Start Up, Tear Down Quickly

New App Dev & Microservices

- Refactor all or part of legacy app
- Containers are great for Microservices

One-Time Run Jobs and Analytics

- Run the Job / Analysis and quit

Front-End App Servers

- Highly horizontally scalable
- Cattle Not Pets
- Fast A/B, Rolling Deployments
- Optimize CX
- Traditional Technologies - MW/Backend

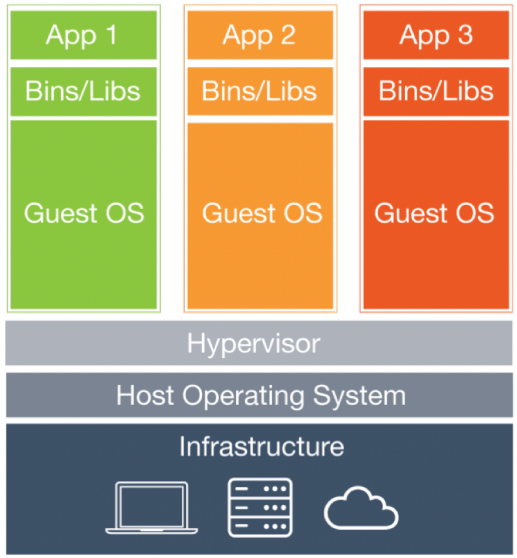
Server Density

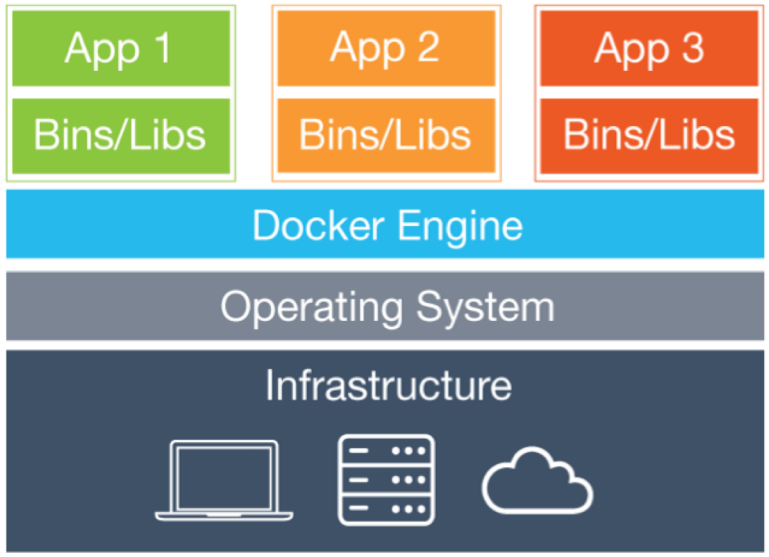
- Containers can use dynamic ports
- Run many of the same app on a server
  - instead of one per VM

Basic Architecture and Nomenclature

Virtual Machines vs. Containers

VMs





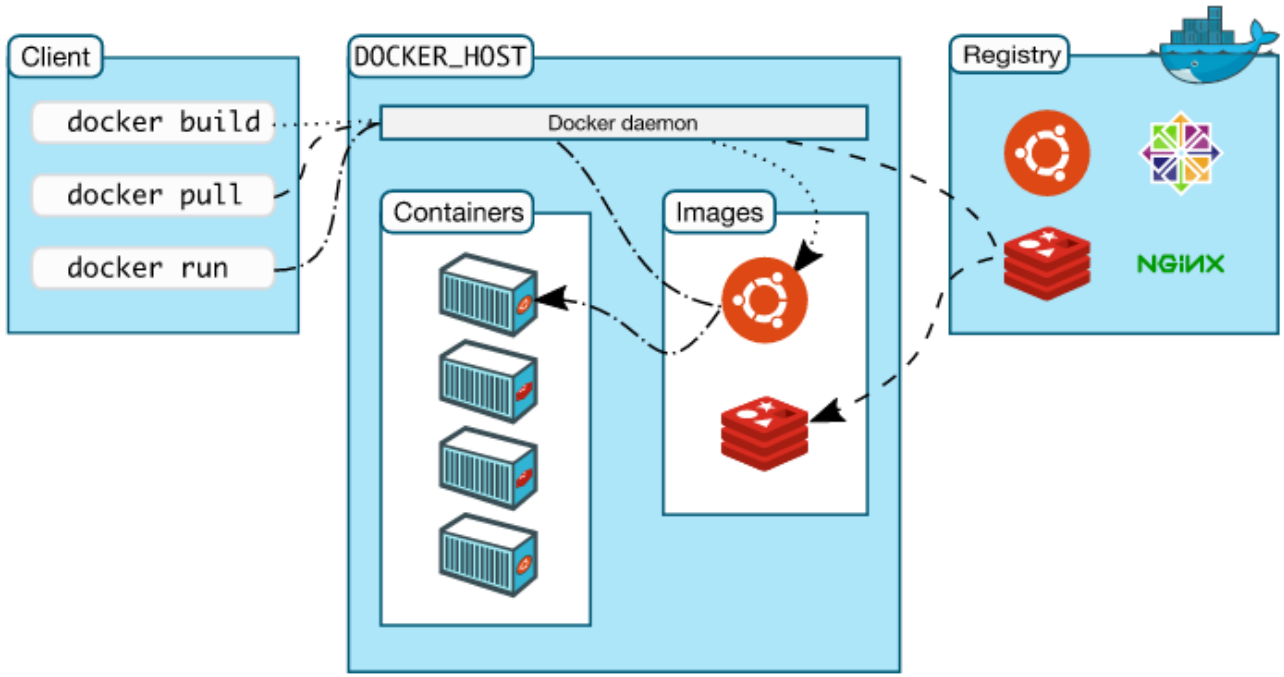
Containers

- **Virtual Machines**
  - Each virtual machine (VM) includes the app, the necessary binaries and libraries and an **entire guest operating system**

- **Containers**
  - Containers include the app & all of its dependencies, but **share the kernel** with other containers.
  - Run as an isolated process in userspace on the host OS
  - Not tied to any specific infrastructure – containers run
  - on any computer, infrastructure and cloud.

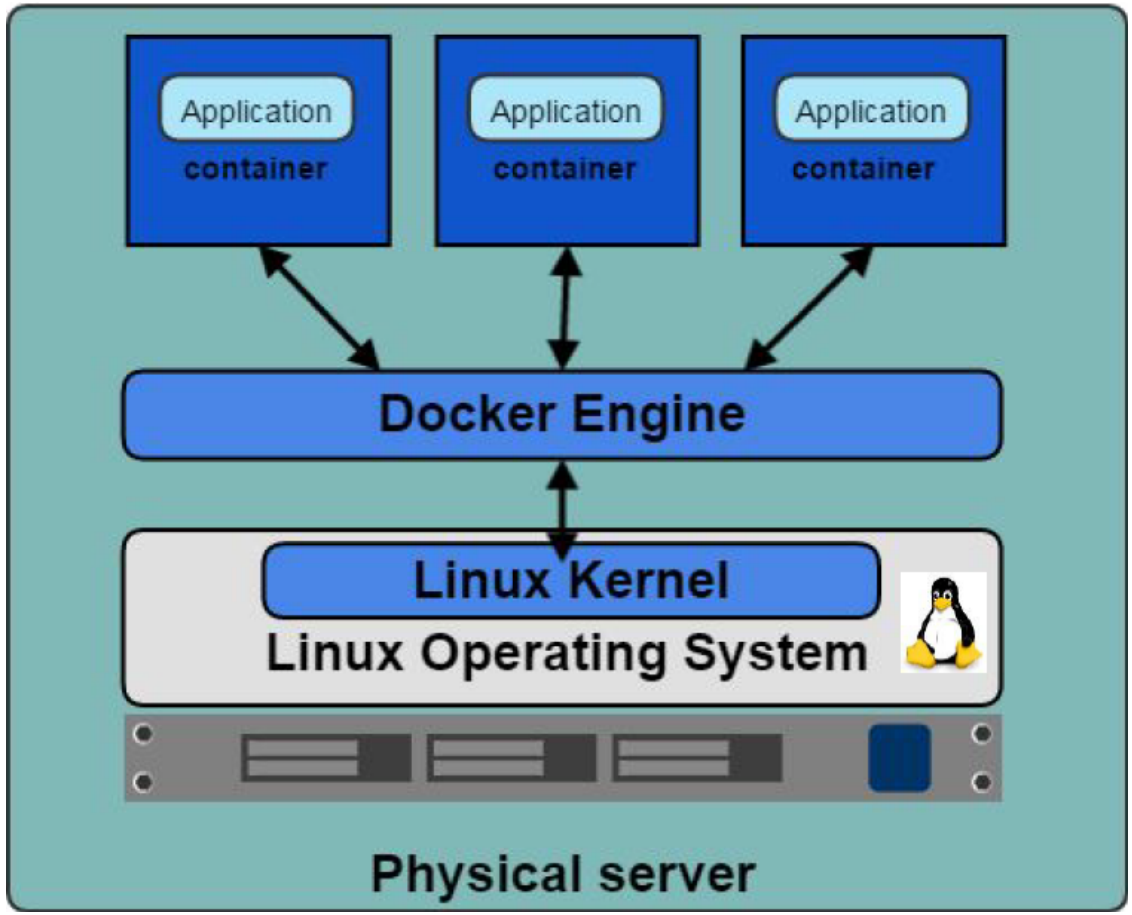
Docker Architecture

- Docker client – Command Line Interface (CLI) for interfacing with the Docker
- Dockerfile – Text file of Docker instructions used to assemble a Docker Image
- Image – Hierarchies of files built from a Dockerfile, the file used as input to the docker build command
- Container – Running instance of an Image using the docker run command
- Registry – Image repository



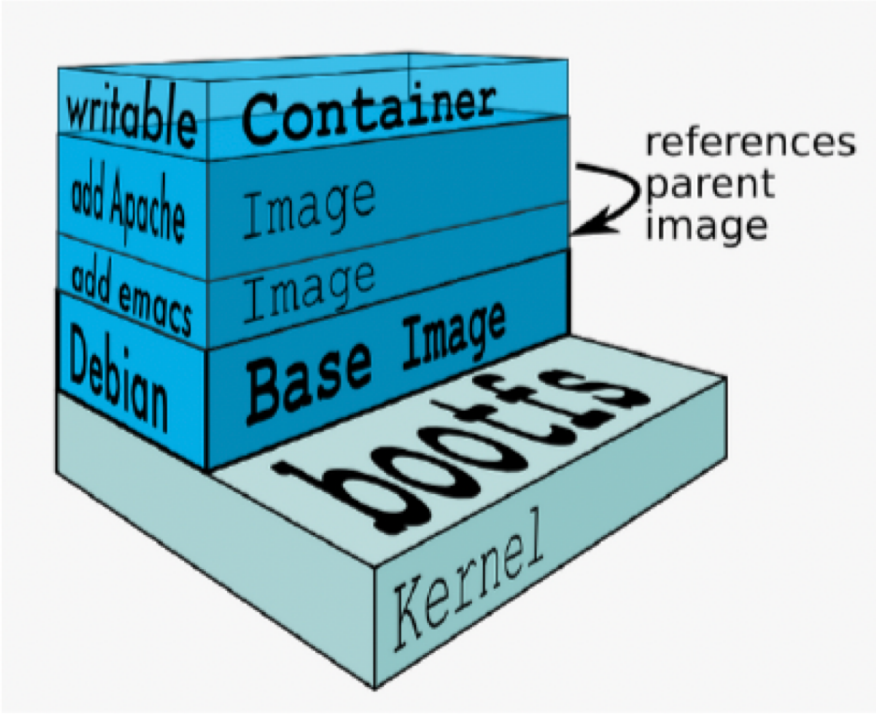
Docker Engine

- Container execution and admin
- Uses Linux Kernel namespaces and control groups
- Namespaces provide for isolated workspace



Docker Images

- An image is a collection of files and some meta data
- Images are comprised of multiple layers, multiple layers referencing/based on another image (Union File System)
- Each image contains software you want to run
- Every image contains a base layer
- Layers are read only



Dockerfile – Text file (recipe) used to create Docker images

- **Example Hello World Dockerfile**
- `$ vim Dockerfile`
-
```
FROM nginx:1.10.1-alpine
Add index.html /usr/share/nginx/html/index.html
# Override the nginx start from the base container
COPY start.sh /start.sh
RUN chmod +x /start.sh
ENTRYPOINT ["/start.sh"]
```
- **Docker build image CLI example**
- `$ docker build -t helloworld:1.0 .`
  NOTE: The "." references Dockerfile in local directory

Docker Hub

- Docker Inc.
  – Repository
  – public and private images
- Enables images to be shared and moved off the laptop
- Example usage:
  - `$ docker login # type username and password`
  - `$ docker tag docker-whale:latest username/docker-whale:latest`
  - `$ docker push username/docker-whale:latest`
  - `$ docker pull username/docker-whale:latest`

Docker CLI – Common / useful commands

```
docker build : build docker image from Dockerfile
docker run : run docker image
docker logs : show log data for a running or stopped container
docker ps : list running docker containers (analogous to ps)
docker ps –a : list all containers including not running
docker images : list all images on the local volume
docker rm : remove/delete a container
docker rmi : remove/delete an image docker tag : name a docker image
docker login : login to registry
docker push/pull : push or pull volumes to/from Docker Registries
docker exec: execute a command in a running container
docker inspect : return container run time configuration parameter
metadata
```

See the docs here: https://docs.docker.com/engine/reference/commandline/docker/

Docker Run

- Pulls the image and runs it as a container
- Examples:
  - Simple:
    - `$ docker run hello-world`
  - Complex:
    - ```
      $ docker run -d \
      --restart=always \
      -p=443:5000/tcp \
      -u=1000:1000 \
      -e="REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
      -e="REGISTRY_HTTP_TLS_KEY=/certs/registry.example.com.key" \
      -e="REGISTRY_AUTH=htpasswd" \
      -e="REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" \
      -e="REGISTRY_AUTH_HTPASSWD_REALM=Our Test Registry" \
      -v=/home/opc/certs:/certs \
      -v=/home/opc/auth:/auth \
      -v=/home/opc/registry:/var/lib/registry "registry:2"
      ```
      ```

Docker Compose

- Docker Compose
  - Docker Tool for defining and running multi-container Docker applications
  - Reference file defined in YAML
    - docker-compose.yml
- $ docker-compose up -d

```
version: '2'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: wordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "80:80"
     restart: always
     volumes:
       - /var/www/html:/var/www/html:rw
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
     db_data:
```

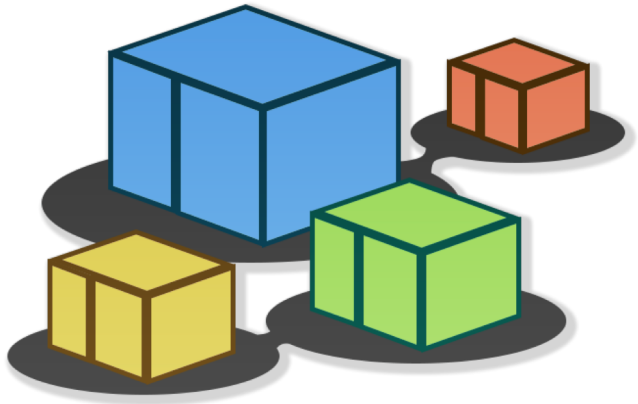
Why Containers?

- Quickly create ready-to-run packaged applications, low cost deployment and replay
- Automate testing, integration, packaging
- Reduce / eliminate platform compatibility issues (“It works in dev!”)
- Support next gen applications (microservices)
- Improve **speed** and frequency of releases, reliability of deployments
- Makes app lifecycle efficient, consistent and repeatable – configure once, run many times
- Eliminate environment inconsistencies between development, test, production
- Improve production application resiliency and scale out / in on demand

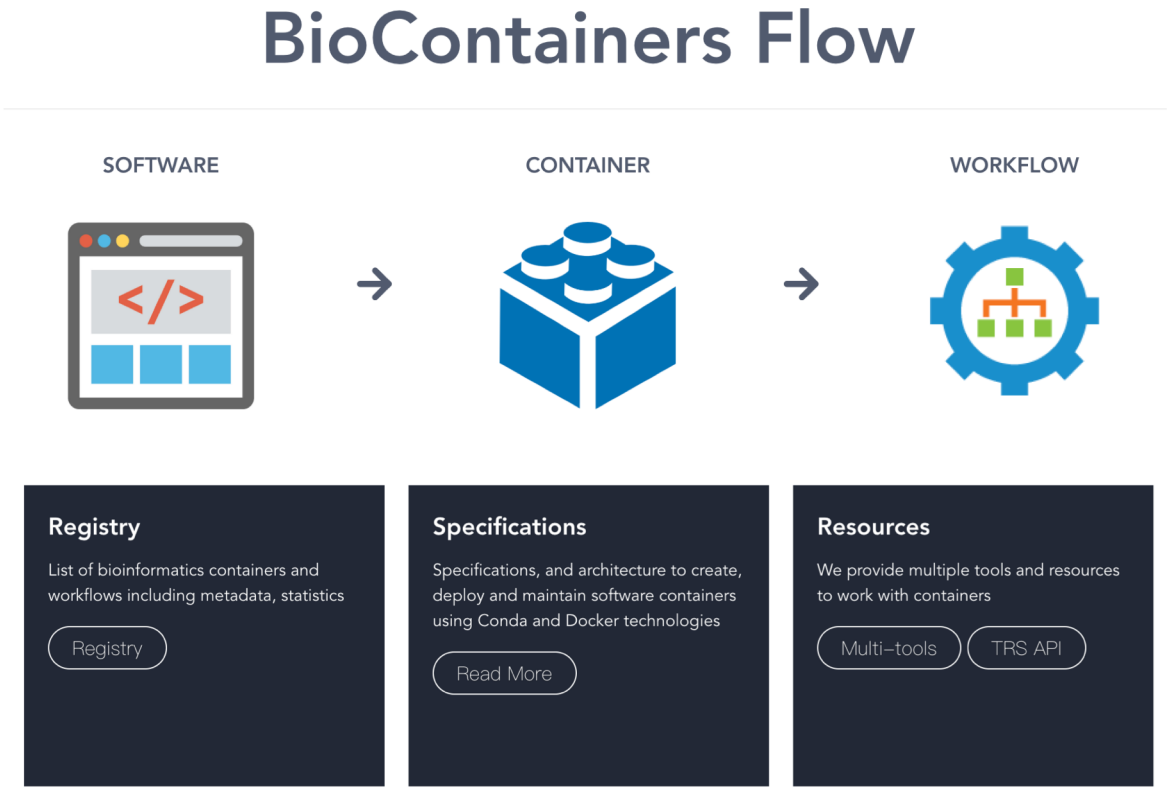
Docker can be only run by root by default

- Dangerous!!!!
- For non-root user, add users to docker group
  - This non-root user will have root privilege
  - Add trusted user to docker group

Biocontainers



- **7.5Ktools, 16.6Kversions, 53.5Kcontainers and packages**



https://biocontainers.pro/