Cloud Computing
Assignment 2

Deploy a simple service via Docker Compose

Outline

- Install Docker
- Docker Basics
- Dockerfile
- Docker Compose
- What to do?

Install Docker

- Environment
  - Linux, Windows or macOS
- Need root privilege
- Documentation

Docker Basics

- Image -> VM image file
  - Template to create a container
- Container -> VM
  - Runnable instance create from an image

Docker Basics

- Container states
  - Created
  - Running
  - Stopped
  - Paused

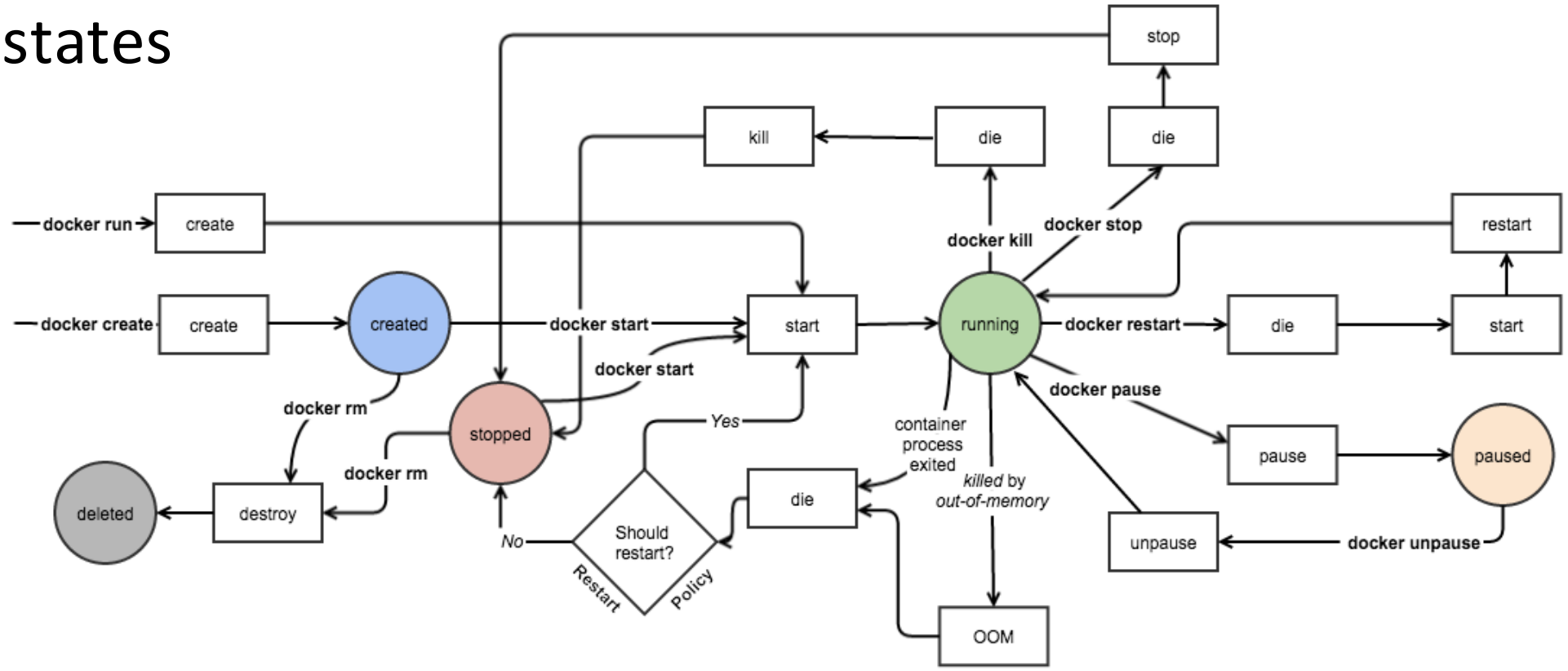


Image Source and Credits: http://docker-saigon.github.io/post/Docker-Internals/

Docker Basics

- Pull image
  - docker pull [options] <name[:tag|@digest]>
- Create container
  - docker create [options] <image> [cmd] [args…]
- Start container
  - docker start [options] <containers…>
- Stop container
  - docker stop [options] <container> [containers...]

Docker Basics

- Run container (same as create + start)
  - docker run [options] <image> [cmd] [args…]
- Some common flags of run
  - -i : Keep stdin open
  - -t : Allocate a terminal
  - -d : Run container in background
  - --name : Name the container
  - --rm : Remove the container when it stops
  - -v : Bind a volume

Docker Basics

- Stop container
  - docker stop [options] <container> [containers...]
- Pause container
  - docker pause <container> [containers...]
- Unpause container
  - docker unpause <container> [containers...]

Docker Basics

- List container
  - docker ps [options]
- Remove container
  - docker rm [options] <container> [container…]
- List image
  - docker images [options] [repoistory[:tag]]
- Remove image
  - docker rmi [opitons] <image> [image…]

Docker Basics

- Tag container
  - docker tag <source_image>[:tag] target_image[:tag]
- Push container
  - docker push [options] <name>[:tag]

Dockerfile

- A text file to build image automatically
- Reference

Docker Compose

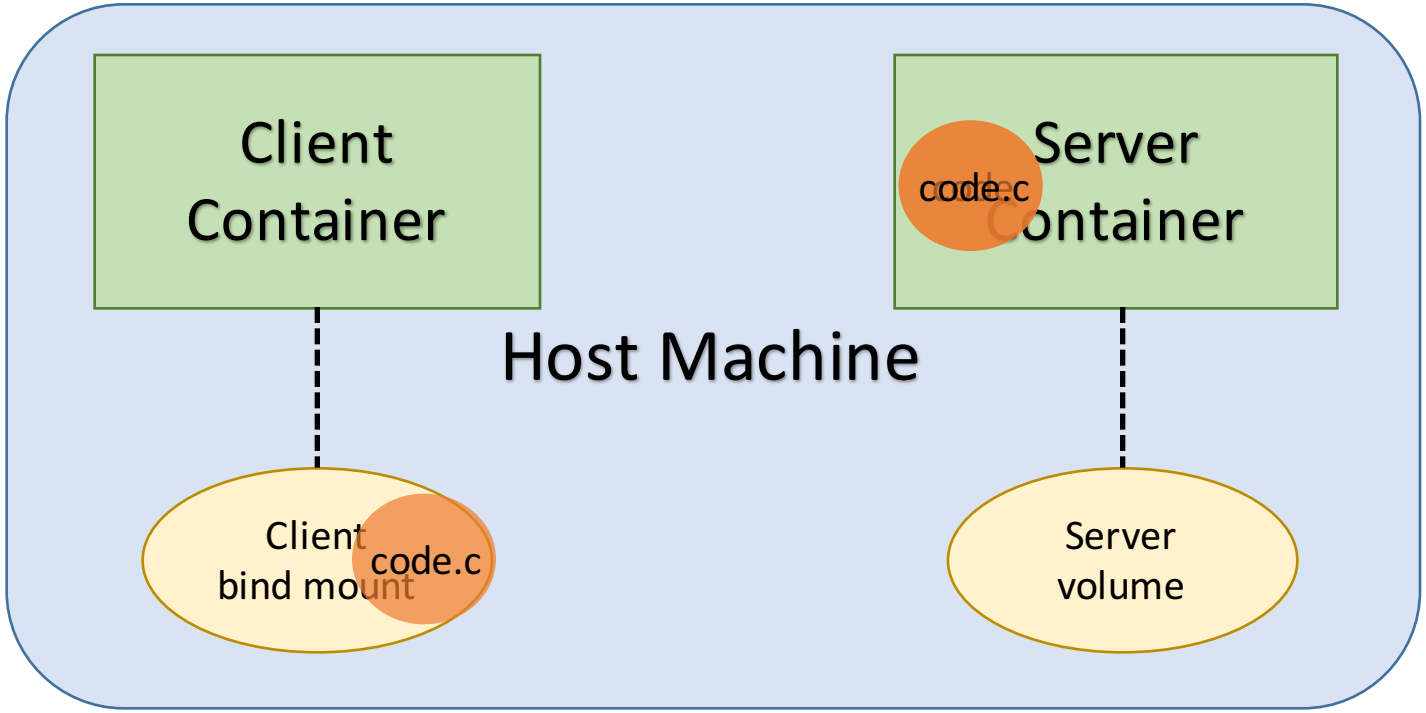
- Tool to deploy Docker applications in only one command
- Reference

What to do?

- Implement a simple compile server service using Docker
- The service is consists of two different container, one for client, the other for server
- The client keeps watching a certain directory, when a file close event occurs, the closed file is send to the server via network
- The server receives a file from client, compile it and send back to the client

What to do?

- Flow



What to do?

- Details for client
  - Create bind mount and mounts it to a directory on host machine for client. The specified directory should be relative inside the docker compose file.
  - The client have to watch file closed events on the mounted directory, the result executable is save at a directory named result under the mount point.

    For example, if your YAML file is under /home/user/workspace, then you can only mount under such directory like /home/user/workspace/path/to/some/mnt/point. And the result will save under /home/user/workspace/path/to/some/mnt/point /result. Any directory outside /home/user/workspace is invalid
  - You have to push your client image to DockerHub.

What to do?

- Details for server
  - The server needs to have the following package to compile c/cpp/java code.
    - gcc >= 7, JDK 9
  - Create a volume for server to store all the input code.
  - An input file ends with c/cpp/java is a valid input file. There may be some number of input files at the same time. (up to 5)
  - The name of the output file will base on the input file
    - Main.c > Main
    - Main.cpp > Main
    - Main.java > Main.class
  - You don’t need to worry about compile error once the input file is valid.

What to do?

- Other details
  - The base image must be formal linux distributions. (Ubuntu, CentOS, Fedora…)
  - The file must transfer through network. You can implement the code in any language, be sure your images contain packages to run your code.
  - Instead for using default network bridge(172.17.0.1), you need to define a new network bridge for the service. For simplicity, you can assign static ip to client and server.
  - You have to write dockerfile for both images.
  - In compose file, the server image will built from dockerfile while the client image will pull from registry.

Report

- You have to submit a report named HW2_{student_id}.pdf. In your report, you must include following items:
  - a. Simply explain your code. (10 points)
  - b. What is the main difference between container and VM? (10 points)
  - c. What are the three method for container to store data in host machine, what is the difference between them? (10 points)

Submission Details

- Please package your report, docker compose directory (consists of dockerfile, compose file, and any other files used for building) in a file named HW2_{student-ID}.zip and upload to iLMS

Demo

- You have to give a demo to TA. (70 points)
- You don’t need to bring your own machine.
- TA will download the code from iLMS, you should show that by simply running command ‘’docker-compose up’’, the service will automatically deploy on the host machine we prepared.
- Please make sure your service can deploy on linux system with docker 18-ce up.

Deadline

- 5/18 23:59