Dec 14th, 2018

Docker

Docker is a containerization tool which helps with the packaging and deployment of apps. By packing up the app and the environment it depends on into a container you can help avoid dependency hell as well as inconsistencies between states of environments.

<https://www.youtube.com/watch?v=aLipr7tTuA4>

Linux containers have been around awhile, but have been rapidly increasing in popularity lately. Why? Docker provides a well-rounded workflow for DevOps, plus Docker containers are lightweight, efficient, consistent, repeatable, isolated, portable, etc.

### Summary

* Separates apps from underlying OS
  + Similar to how VMs separate the OS from the bare metal
* Allows for you to use any Language, using any Stack, which are run on anything
* Prevents conflict between app development and sys admin
* Take a snapshot of your environment, upload it to a registry, then download that image and start making containers off it
* Deals with isolated containers, you can put multiple on a machine, make more efficient use of the resources, reduces machine requirements and licensing costs
* Helping to usher in the micro-services era
  + Traditionally would build monolithic tier based apps where they're all interdependent.
  + Now apps are more broken up into smaller, more independent parts
  + Ergo, more efficient to develop & deploy because you can isolate which parts are being replaced

## Components

* Core Components
  + Docker Daemon
    - Docker engine, runs on host machine
    - On windows/macos, there's an extra layer around the daemon: **boot2docker**, pree much have to install VirtualBox and spin up a Linux VM
  + Docker Client
    - CLI used to interact with the daemon
* Docker Workflow Components
  + Docker Image
    - Template that contains the environment base, the OS, the application, and the stack it runs on + all of the dependencies
    - Layered: if you start with two core components as v1, you can deploy that, continue development (adding layers) and if you haven't modified the inner layers, when you redeploy, they can just download the new layer. **#lightweight**
  + Docker Container
    - Created from image. { start, stop, move, delete }
    - ie, spin up a container, specify and image.
  + Docker Registry
    - Public/private repo use to store image
    - Docker hub is to docker images as github is to code repos
  + Docker File
    - Automates image construction
* Example process:
  + Pull image from registry into local docker host
  + `docker run` on image => container
    - w/in the container, install our software, stack, application, etc
  + commit the container => new image / new layer => version 1
  + push back up to repo so others could then spin up containers based on it
  + …
  + keep working with the container,
  + commit again => version 2
    - handy thing is that any existing containers running the images can just download that new v2 layer, rather than having to redownload the entire thing
  + \*dockerfile can hold the steps for installing software, application, etc, so we don't need to manually do that process of generating the image
* Lots of add-on tools for improvements of docker workflows
  + Ex/ Google's Kubernetes
    - Schedule & manage deployment of containers automatically