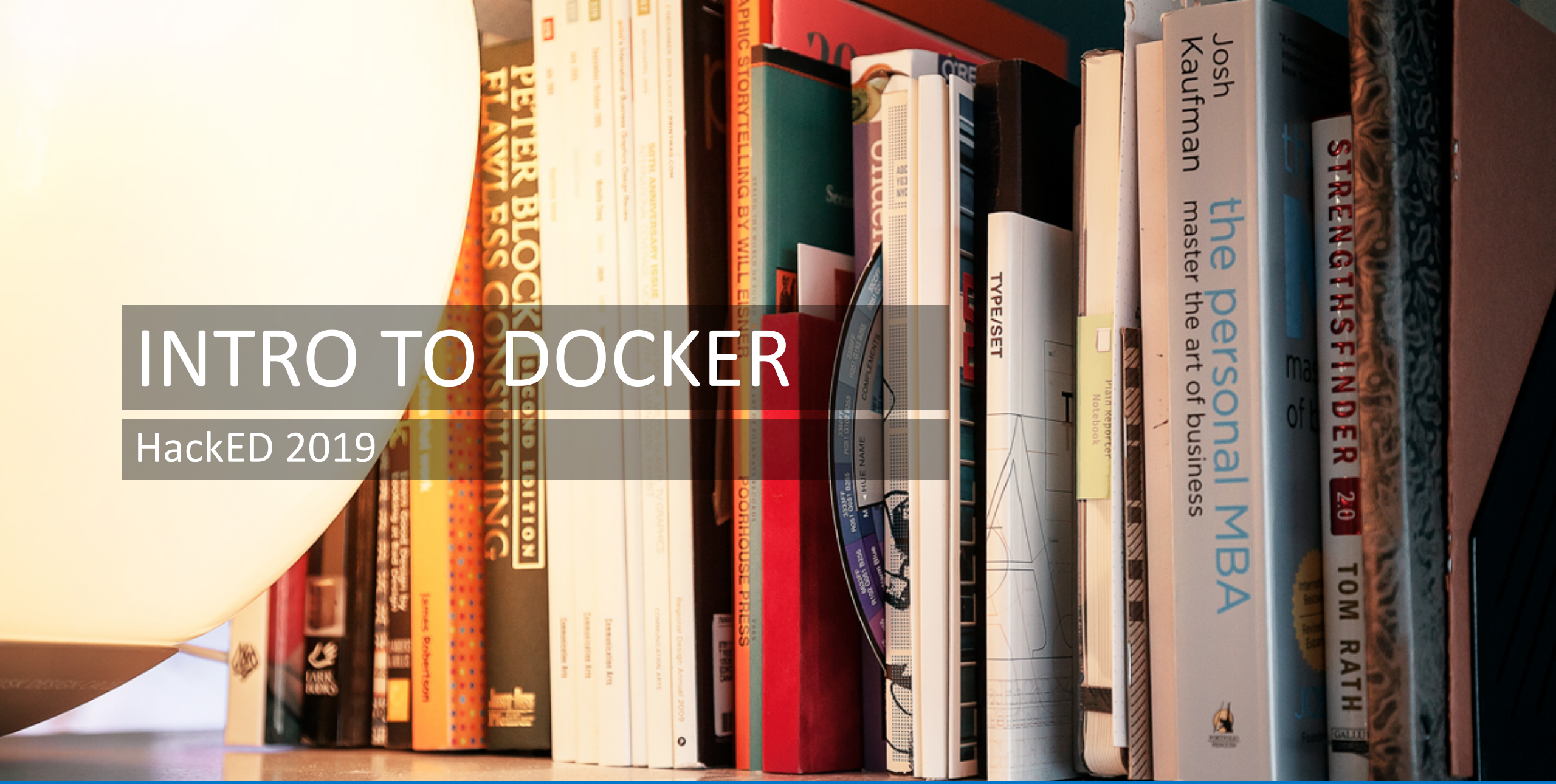
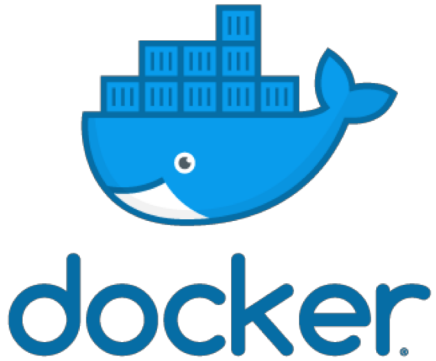


INTRO TO DOCKER

HackED 2019

Outline

- What is Docker?
- Why should I use ~~Docker~~ containers?
- How do I use Docker?
  - Installation
  - Core concepts
- Docker @ DevFacto
- Further Learning



What is Docker?

- Platform for developing, deploying and running applications in containers
- Containers comprise of app code + dependencies and nothing more

Why should I use ~~Docker~~ containers?

- Portable
  - The same stack can be run locally and in production
- Interchangeable
  - Breaking up your solutions into separate containers lets you make changes/upgrades quicker
- Lightweight
  - Unlike a VM, containers only take up space needed for app code + dependencies, and nothing else
  - Images, which are defined using small Dockerfiles, create containers

How do I use Docker?

- Installation
- Core concepts
  - Images
  - Containers
  - Volumes & bind mounts
  - Networking

Installation

- For Mac/Windows, use their stable Docker Desktop installer
  - https://www.docker.com/products/docker-desktop
  - You will need to sign up for a Docker Store account
- On all OSes, the default Docker daemon supports Linux containers
  - On Windows, there is the option to switch to a daemon that supports Windows containers

- Note: Docker daemon must run directly on the host OS (i.e. won't work in a VM)

Side-note: there is a repository

- https://github.com/ajyong/docker-by-example
- I will be alternating between these slides and the code above

Core Concepts: Images

- Docker builds images into containers by reading a Dockerfile
- Dockerfiles are just text files with instructions on each line
  - Each line creates a read-only, cacheable layer
  - This means subsequent builds only perform instructions that are new/changed

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Core Concepts: Containers

- Running an image creates a container
  - Internally, this is a writable layer written on top of your read-only image layers
- Can be started, stopped, destroyed
  - Stopped containers persist file changes made during runtime
  - Destroyed containers effectively means that the writable layer is deleted

Core Concepts: Volumes & bind mounts

- Both data persistence options help decouple data from containers
- Volumes
  - Data persistence that is managed by Docker
  - Can be shared amongst multiple containers
  - OS-agnostic
- Bind mounts
  - Mount a host file/directory into a container's
  - Host OS filesystem should be the same as the container's

Core Concepts: Networking

- Usually only a concern if you are managing multiple containers and/or Docker daemons
- Network drivers:
  - **Bridge**: default driver, isolates containers from the host network; ports can be mapped
  - **Overlay**: allows multiple Docker daemons on different host networks to coordinate as a Docker Swarm
  - There are others, but they're seldom used
    - https://docs.docker.com/network/

Docker @ DevFacto

- I am currently using Docker Compose!
  - PHP, ASP.NET Core, MySQL, nginx, certbot
  - Docker Swarm and Kubernetes are happening soon™
- Other colleagues have been helping their clients roll out Kubernetes and Swarm stacks

Further Learning

- Docker
  - Get started with Docker
- ASP.NET Core:
  - Develop ASP.NET Core Applications in a Container
  - dotnet-watch within a container
- Windows Containers
  - On Windows Containers
  - Modernizing .NET Framework apps with Docker


QUESTIONS?
Comments?
PETER BLOCK
SECOND EDITION
FLAWLESS CONSULTING
STORYTELLING BY WILL EISNER
POORHOUSE PRESS
TYPE/SET
Josh Kaufman
the personal MBA
master the art of business
STRENGTHSFINDER 2.0
TOM RATH