liatrio

CSCI 300 - Computer Science Seminar

Container Best Practices

Chris Schreiber (chriss@liatrio.com)

Agenda

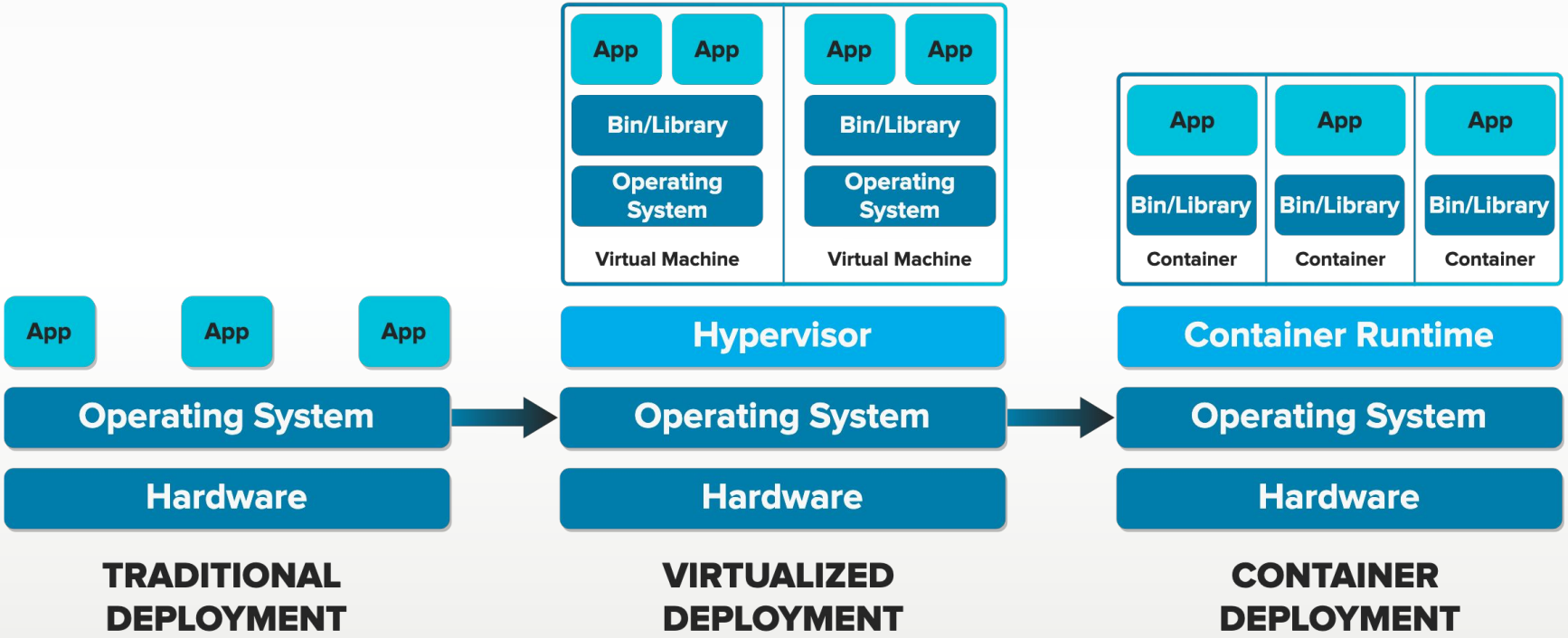
1. What is a container?
2. Why use containers?
3. Building containers (best practices)

WHAT IS A CONTAINER?

*A **container** is a standard unit of software that packages up **code** and all its **dependencies** so the application runs quickly and reliably from one computing environment to another.*

*A **container image** is a lightweight, standalone, **executable** package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.*

Containers vs Virtual Machines vs Bare Metal



Things you (don’t) do with a container

Using containers vs Virtual Machines / Bare Metal

Do these

- Build (docker build)
- Share (docker push/pull)
- Run (docker run)

Don't do these

- Install / configure an operating system
- Configure authentication (usernames and passwords)
- Update / patch software or configuration

Container Core Concepts

Big ideas that make containers important

Separation of Concerns

Containers execute only a single process

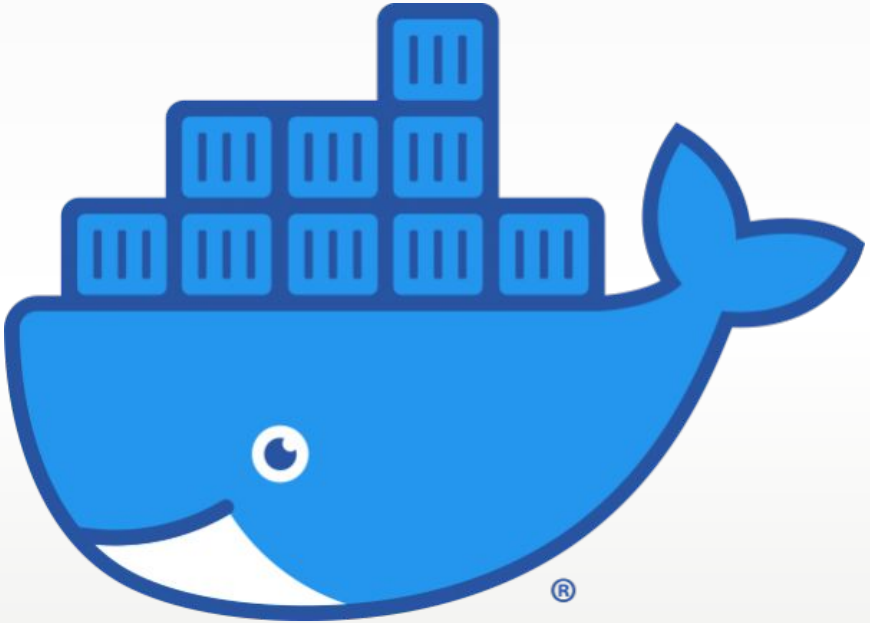
- Avoid dependency conflicts
- Increase scalability
- Manage resources precisely
- Monitor and recover from failures

Immutable

Container won't be modified during its life

- A specific container version represents a known state
- Replace containers with new image instead of modifying
- Deployments are consistent and safe
- Roll back to previous state by deploying previous image

What is Docker?



QUESTIONS

What are containers?

Why use containers?

1

Lightweight

- Low resource overhead
  - Easier to develop locally
  - Maximize hardware resources
- Fast startup
  - Reduces development feedback time
  - Recover from failures faster
  - Scale up on demand

2 Portable

- Same build and run process for local development as production deployment
- Include image build instructions with code / share with peers and community
- Open Source standard
- Decentralized registry

3

Scaleable

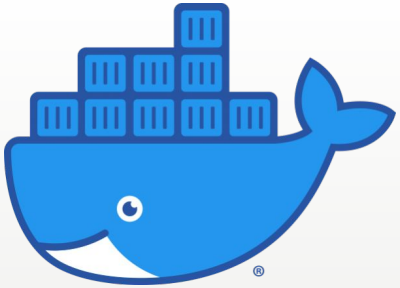
- Add more capacity on demand by adding more containers
  - Separation of concerns
  - Low resource overhead
- Immutable infrastructure make scaling & deploying updates safer

Why containers?

4

Community

kubernetes



CLOUD NATIVE
COMPUTING FOUNDATION

QUESTIONS

Why use containers?

Building Containers

Best Practices

Follow along at https://github.com/ChrisSchreiber/CSCI-300

How to build a container

Dockerfile

- Defines individual steps to create container
- Each instruction creates a layer which is cached and reused in subsequent builds
- **FROM** instruction defines base image
- **ENTRYPOINT** instruction defines process container is attached to
- **`docker build`** command creates image

Dockerfile.build01 > ...

```
1   FROM node:12-alpine
2
3   RUN apk add --no-cache python g++ make
4
5   WORKDIR /app
6
7   COPY app /app
8
9   RUN yarn install --production
10
11  ENTRYPOINT ["node", "src/index.js"]
```

Best Practices

Consistent builds

- Use official base images specific to you application's purpose (1)
- User version specific tags for base images (2)
- Build your application from source at container build time (3)

Dockerfile.build02 > ...

```
FROM alpine AS BAD_EXAMPLE
# (1,2) our base image is not specific to our
RUN apk add --no-cache nodejs python3 g++ make

WORKDIR /app

COPY app /app

# (3) We have not installed our application dependencies
# Our application will only start if we have run npm
# install outside of our docker build

ENTRYPOINT ["node", "src/index.js"]

FROM node:12.22.6-alpine AS GOOD_EXAMPLE
# (1,2) We are using the node base image and using the
# version specific tag 12.22.6-alpine

RUN apk add --no-cache python g++ make

WORKDIR /app

COPY app /app

RUN yarn install --production
# (3) we are installing our application dependicnes are
# part of the build process

ENTRYPOINT ["node", "src/index.js"]
```

Best Practices

Optimize Build Time

- Order of operations
  - Copy operations (1)
  - Install dependencies (2)
- Be specific when copying files (3)

Dockerfile.build03 > ...

```
FROM node:12-alpine

RUN apk add --no-cache python g++ make

WORKDIR /app

COPY app/package.json app/yarn.lock /app/
# (1) only copy files needed to install dependencies

RUN yarn install --production
# (2) install depdencies in this layer

COPY app/src /app/src
# (1,3) copy only application files

ENTRYPOINT ["node", "src/index.js"]
```

Best Practices

Optimize Size

- Clean up files in layer
- Multi-stage builds

Dockerfile.build04 > ...

```
FROM alpine as GROOVY

RUN wget https://groovy.jfrog.io/artifactory/dist-release-local/groovy-zips/apache-groovy-binary-3.0.9.zip && \
    unzip  apache-groovy-binary-3.0.9.zip

FROM node:12-alpine as NODE

COPY --from=GROOVY groovy-3.0.9/bin/* .
```

QUESTIONS

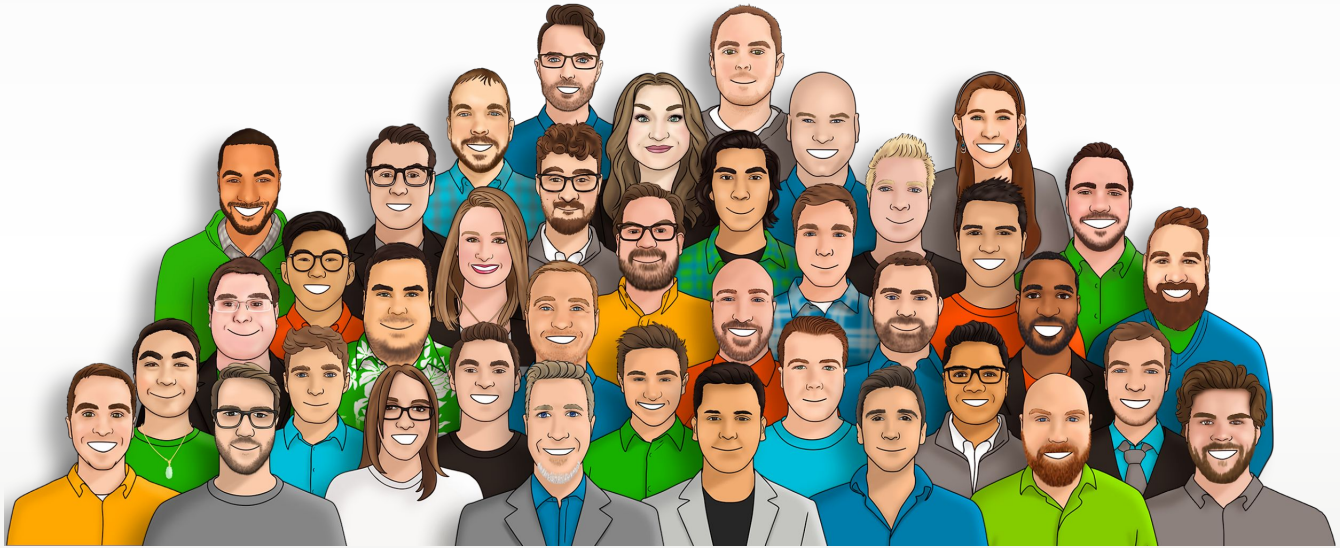
Building containers best practices

Liatrio

We are hiring apprentices for spring 2022

Paid apprenticeship focused on hands on learning about DevOps practices.

Learn more at our open house Thursday October 21st



Reach out at chriss@liatrio.com, get updates https://bit.ly/Liatrio20 or stop by our office 240 Main St #280 (above Woodstocks Pizza)