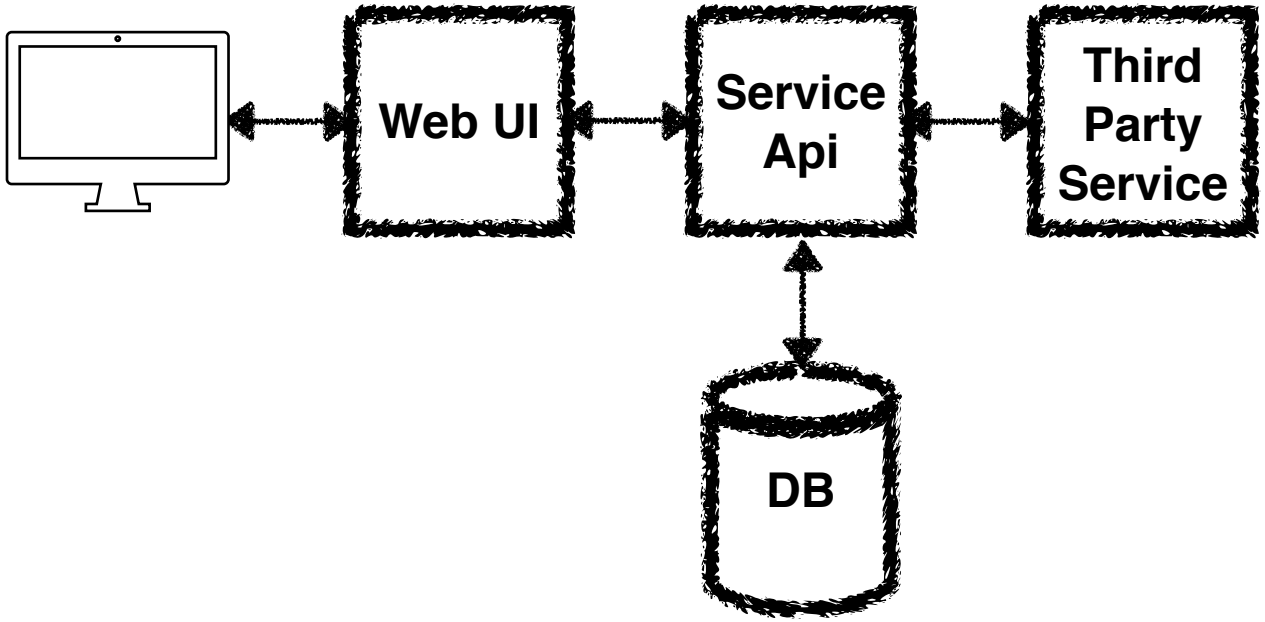
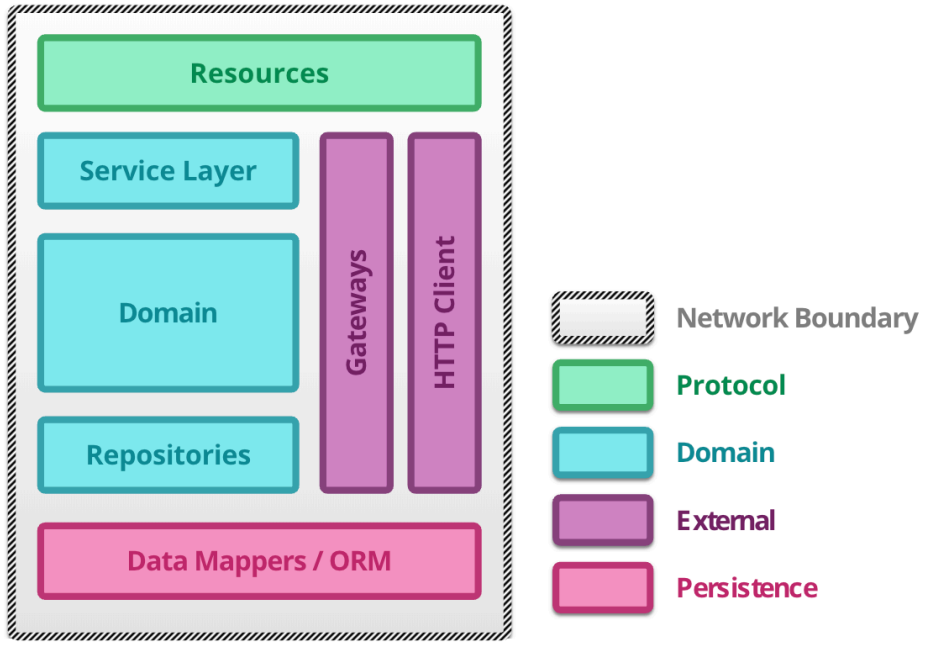
Docker For Testers

High-level Structure / Internal Structure





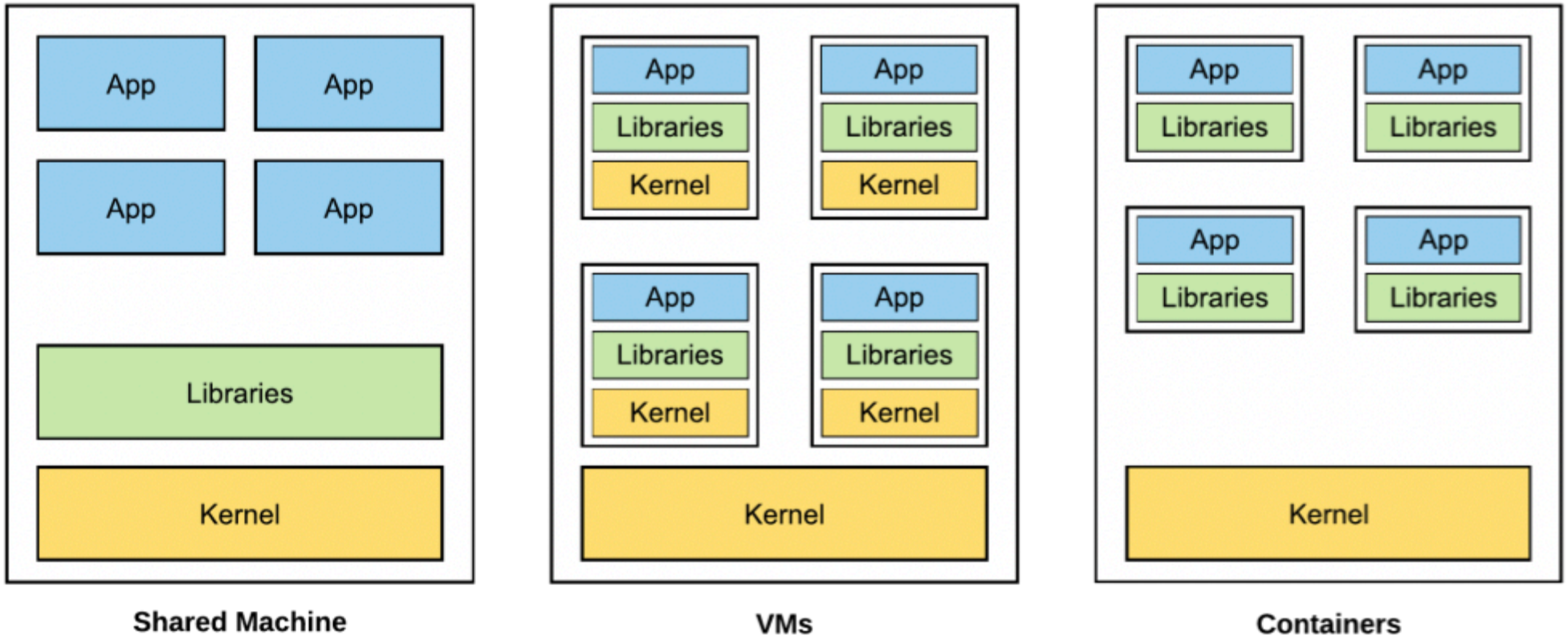
reference: https://martinfowler.com/articles/microservice-testing/

Practical Test Pyramid

- Unit
- Integration
- Contract
- *UI/API*
- *End-to-End*
- *Acceptance*
- Exploratory
- The Confusion About Testing Terminology
- Putting Tests Into Your Deployment Pipeline
- Avoid Test Duplication
- Writing Clean Test Code

reference: https://martinfowler.com/articles/practical-test-pyramid.html

Evolution of Containers



reference: Getting Started with Kubernetes

Several Specific Benefits of Containers

- Language Flexibility
- Isolation Without Overhead: light weight
- Developer Efficiency: Isolating Dependencies(libs, configuration)
- **Reproducibility**: Containers make it easier to reproduce your application environment.

Install Docker and Editor

1. get-docker
2. install windows home
   WSL2 must be installed before you can install and use Docker.
3. Install VSCode & Docker Plugin

Docker Desktop for Mac

A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.

Docker Desktop for Windows

A native Windows application which delivers all Docker tools to your Windows computer.

Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.

Hello, World Docker

```
docker run hello-world
```

Docker run: behind the scenes

- Docker looks for the image on this computer
- Is it installed
- Docker searches Docker Hub for the image.
- Is it on Docker Hub
- Docker downloads the image
- The image layers are installed on the computer
- Docker create a new container and starts the program.
- The container is running.

Download/Clone: Material from github.com

https://github.com/boyone/docker-for-testers

Lab01 Create Docker Image

from Scratch

- Modify, Commit, and Tag
- Sign Up https://hub.docker.com
- Push Image to hub.docker.com
- Pull and Run

Run Docker

```
$ docker run hello-world
$ docker container run hello-world

$ docker image ls

$ docker container ls -a
```

Run
with -d

```
$ docker container run boyone/hello-world

$ docker container run -d boyone/hello-world

$ docker container ls

$ docker container logs <container id | container name>

$ docker container logs -f <container id | container name>
```

Run
with --name

```
$ docker container run boyone/hello-world

$ docker container run -d --name hello boyone/hello-world

$ docker container ls

$ docker container logs hello

$ docker container logs -f hello
```

Run
start, stop, and remove

```
$ docker container stop <container id| container
name>

$ docker container stop hello

$ docker container start hello

$ docker container logs -f hello

$ docker container start hello
$ docker container rm <container id| container name>
$ docker container rm hello
```

Run commit

```
$ docker container run -d --name hello boyone/hello-world

# copy new binary to container
$ docker container cp ./hello <container id| container name>:/root/

$ docker container cp ./hello hello:/root/

# commit container
$ docker container commit <container id| container name>
image_name:tag

$ docker container commit hello boyone/hello-world:0.0.2
```

Run
push

Sign Up https://hub.docker.com

```
# login to hub.docker.com
$ docker login

# push image to hub.docker.com
$ docker image push username/image_name:tag

$ docker image push boyone/hello-world:0.0.2

# tag latest
$ docker image tag boyone/hello-world:0.0.2
boyone/hello-world:latest
```

Run
remove image

```
# login to hub.docker.com
$ docker image rm image_name
$ docker image rm boyone/hello-world
$ docker image rm boyone/hello-world:0.0.2

$ docker image ls

$ docker container run boyone/hello-world
$ docker container run boyone/hello-world:0.0.2
```

Lab02 Create Docker Image

from Dockerfile

- Reference
  - FROM, COPY, RUN, CMD, ENTRYPOINT
- Dockerfile best practices
- Modify, Build + Tag, Push
- Pull and Run

File
Dockerfile

```
FROM golang:1.15.2-alpine3.12 AS builder
WORKDIR /go/src/app
COPY main.go .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o hello .

FROM alpine:3.12
WORKDIR /root/
COPY --from=builder /go/src/app/hello .
CMD ["./hello"]
```

Run
Build

```
# docker build -t <name>/hello-world:0.0.3 .
$ docker build -t boyone/hello-world:0.0.3 .

$ docker container run boyone/hello-world:0.0.3

$ docker image ls
```

Run
0.0.3

```
$ docker container run boyone/hello-world:0.0.3

$ docker image push boyone/hello-world:0.0.3
```

Modify main.go

```
> Replace this line

    fmt.Println("Hello, World!")

> with

    fmt.Println("Hello, World!", time.Now())
```

Run
Build

```
# docker build -t <name>/hello-world:0.0.4 .
$ docker build -t boyone/hello-world:0.0.4 .

$ docker container run boyone/hello-world:0.0.4

$ docker image ls

$ docker container run boyone/hello-world:0.0.4

$ docker image push boyone/hello-world:0.0.4
```

Run
tag

```
# docker image tag <name>/hello-world:0.0.4 <name>/hello-world:latest

docker image tag boyone/hello-world:0.0.4 boyone/hello-world:latest

docker image push boyone/hello-world:latest
```

Lab03 Working with API

- Pull docker images to run API testing
- Bind port
- Run
- Mount disk volume

File
Dockerfile

```
FROM andyrbell/mountebank:2.3.2
WORKDIR /imposters
COPY ./imposters .
EXPOSE 2525
EXPOSE 8000
CMD [ "mb", "start", "--configfile", "/imposters/store-service.json",
"--allowInjection" ]
```

Run
Store Service

```
$ docker container run boyone/store-service-stub
```

Run
with -p

```
$ docker container run -p 2525:2525 -p 8000:8000
boyone/store-service-stub
```

Run
plain image

```
$ docker container run -p 2525:2525 andyrbell/
mountebank:2.3.2

$ docker container run -p 2525:2525 -d --name mb andyrbell/
mountebank:2.3.2

$ docker container run -v `pwd`/imposters:/imposters -p 2525:2525
-p 8000:8000 -d --name mb andyrbell/mountebank:2.3.2 mb start --
configfile /imposters/store-service.json --allowInjection

$ docker container exec -it mb sh
```

Run
with docker-compose

```
$ docker-compose up
```

```
$ docker-compose -f docker-compose-build.yml up
```