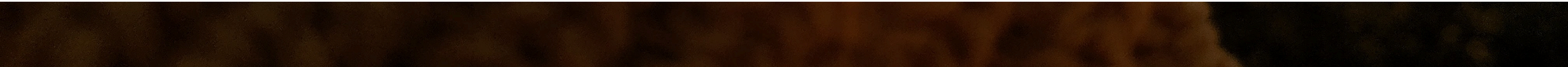
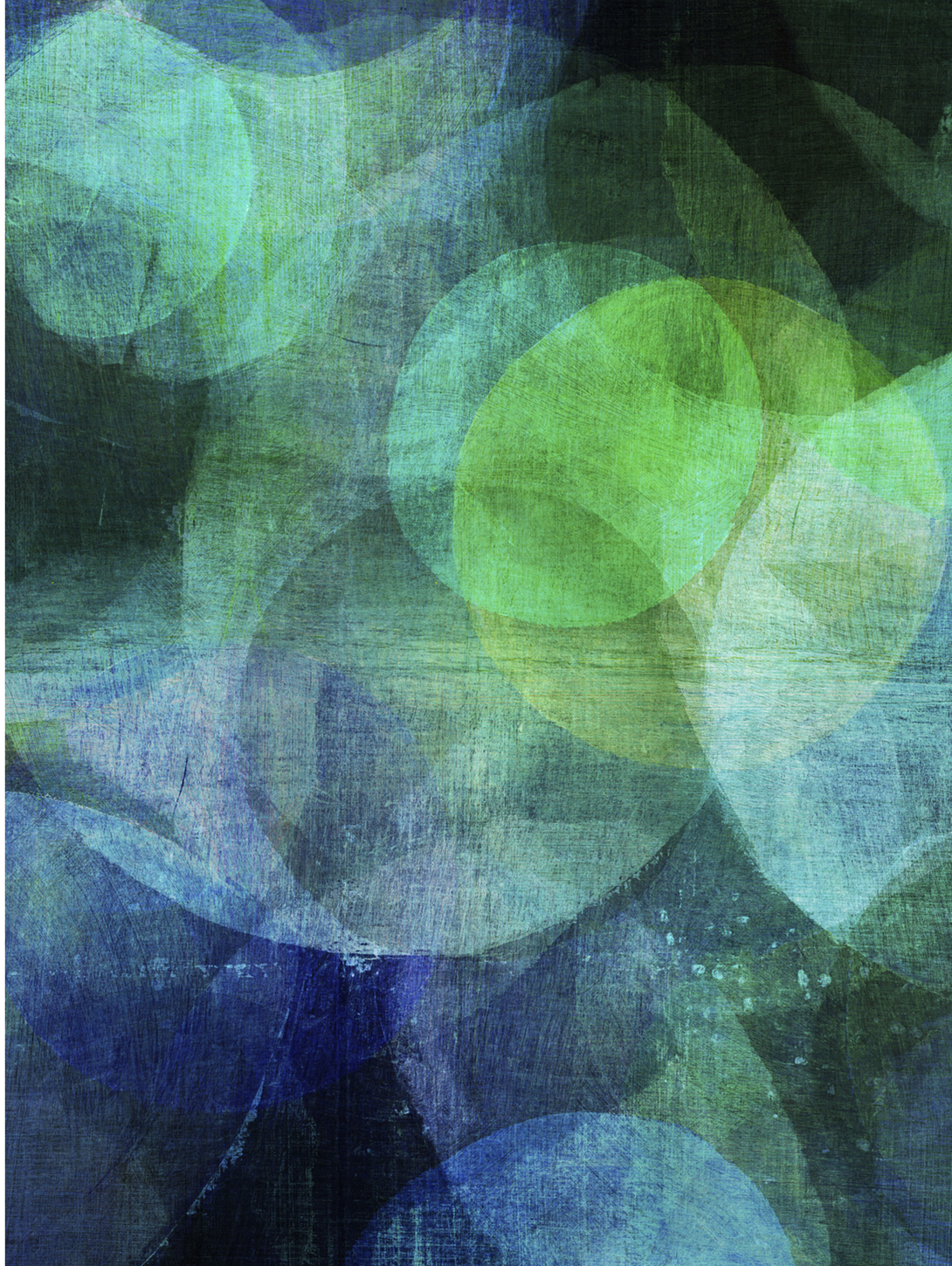
INTRODUCTION TO DOCKER

INTRODUCTION TO DOCKER

Docker Compose

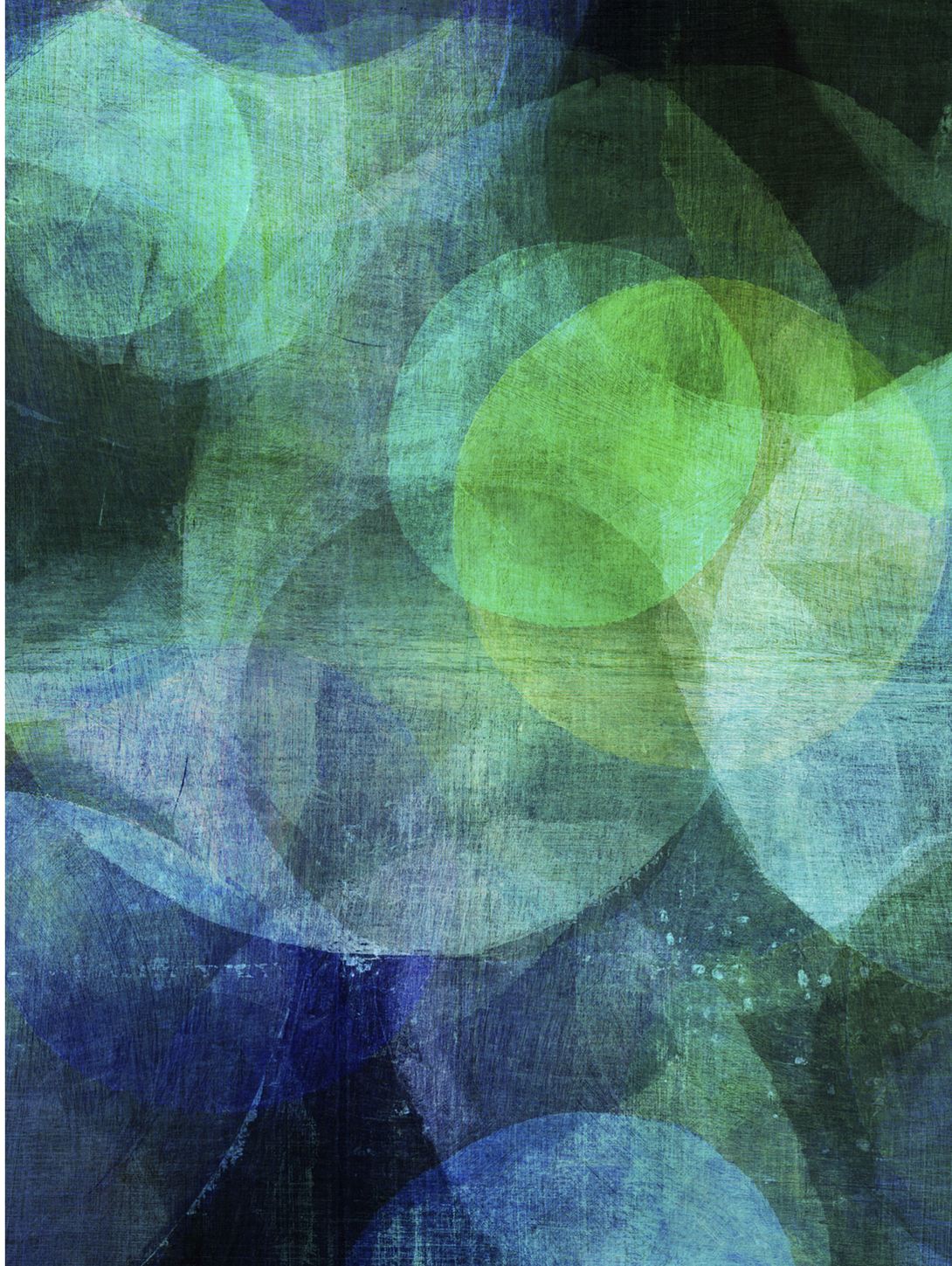


AGENDA



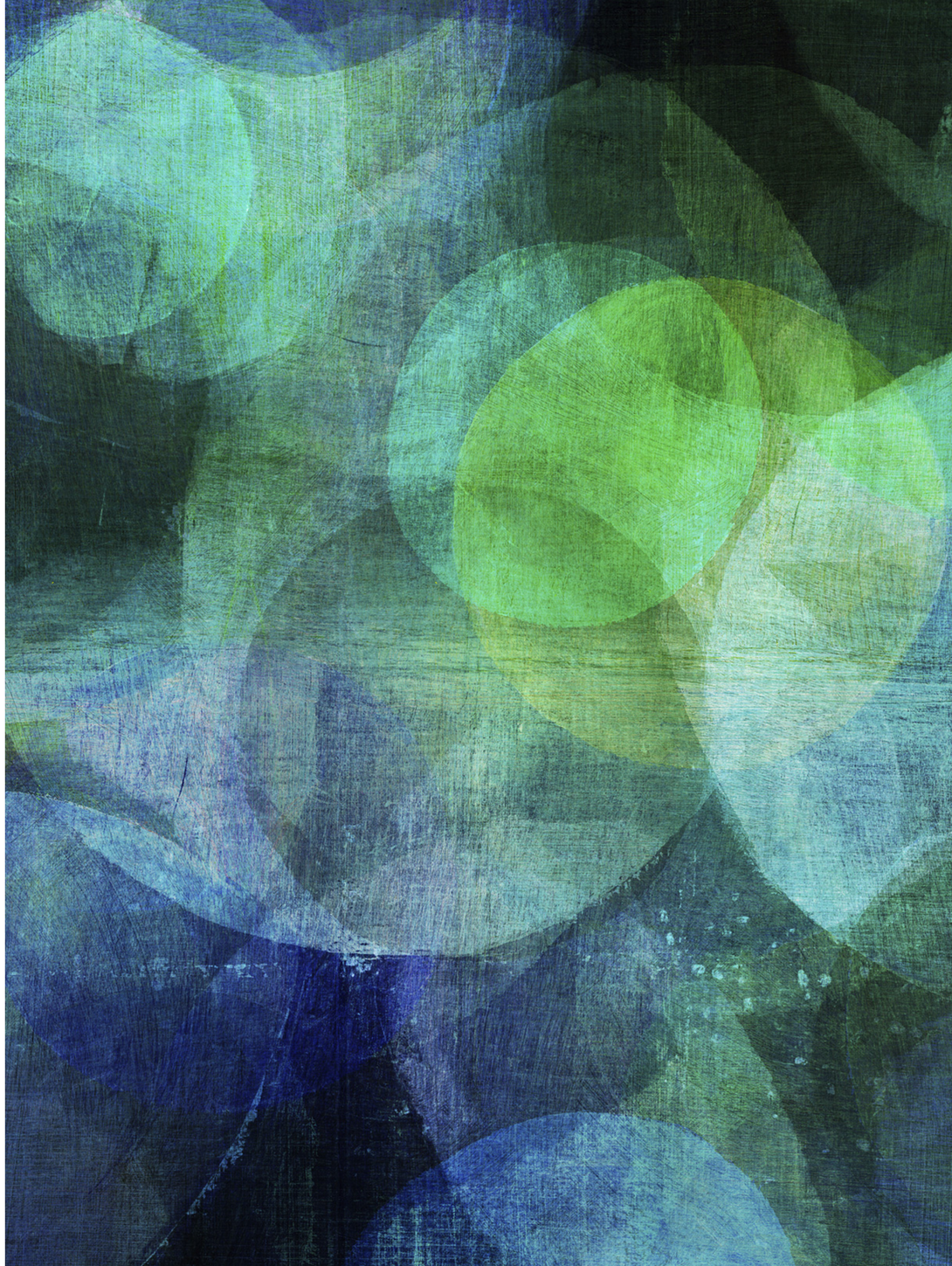
AGENDA

- Overview of Docker Compose



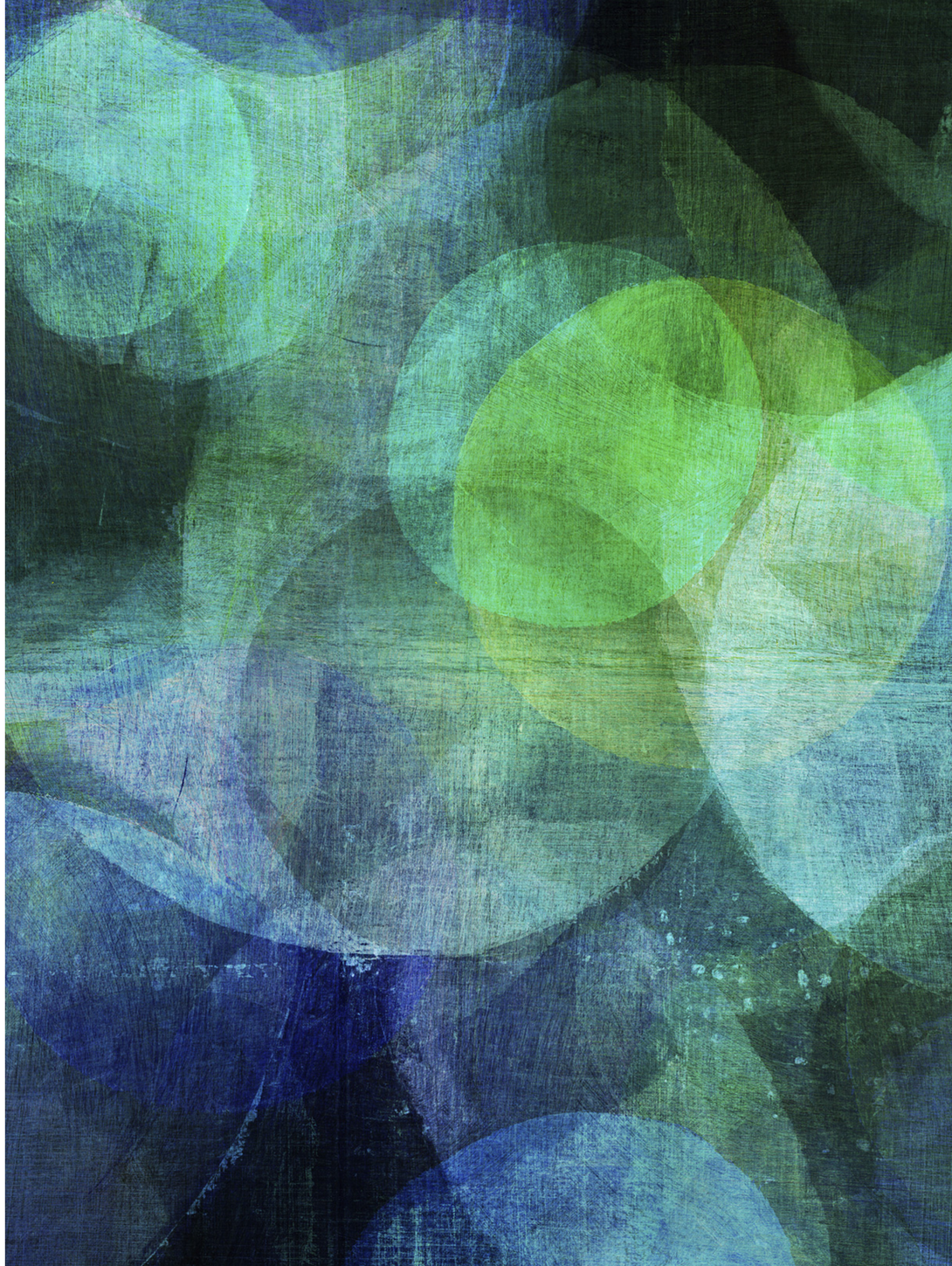
AGENDA

- Overview of Docker Compose
- Docker Compose File



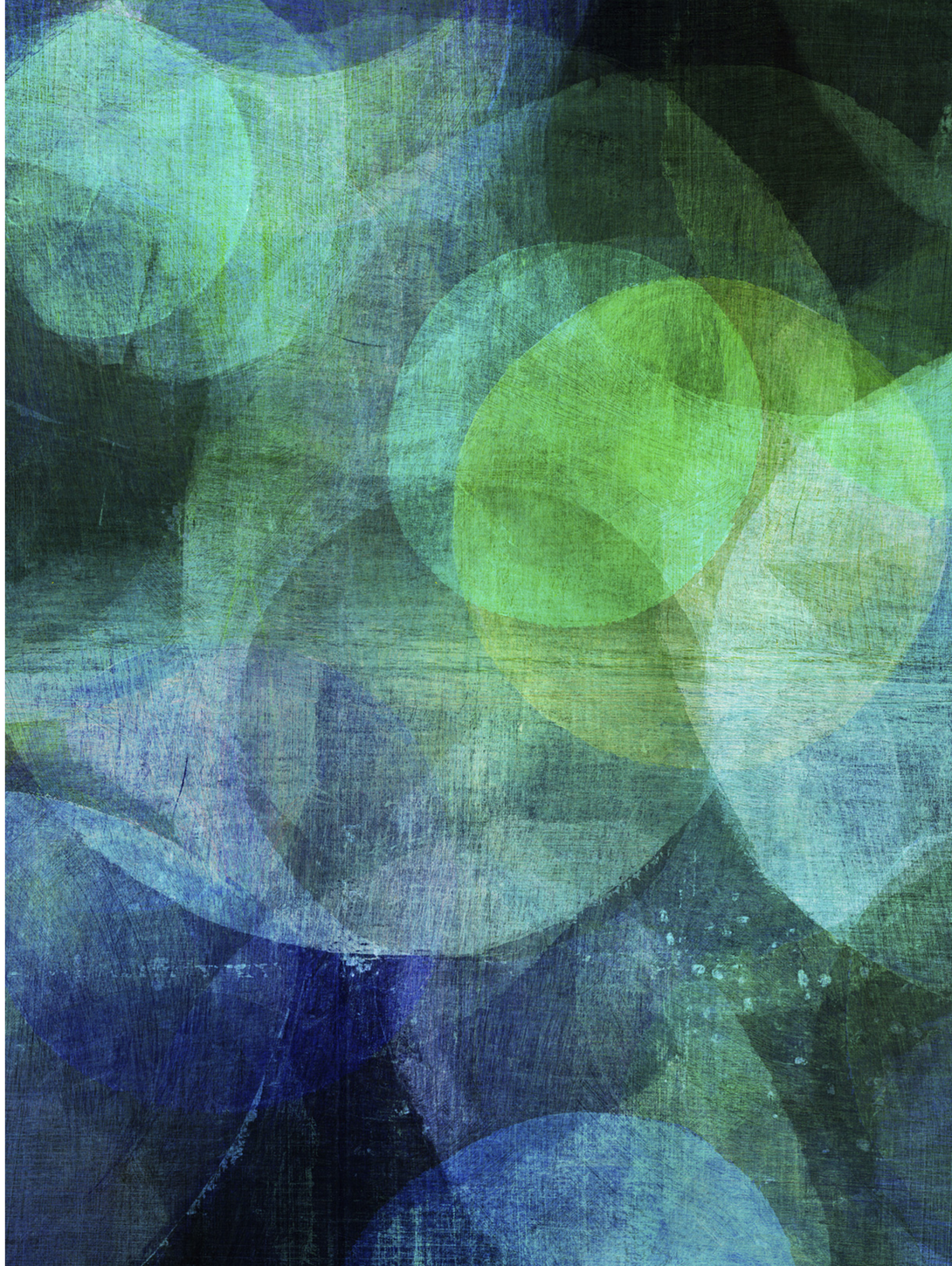
AGENDA

- Overview of Docker Compose
- Docker Compose File
- Docker Compose Commands



AGENDA

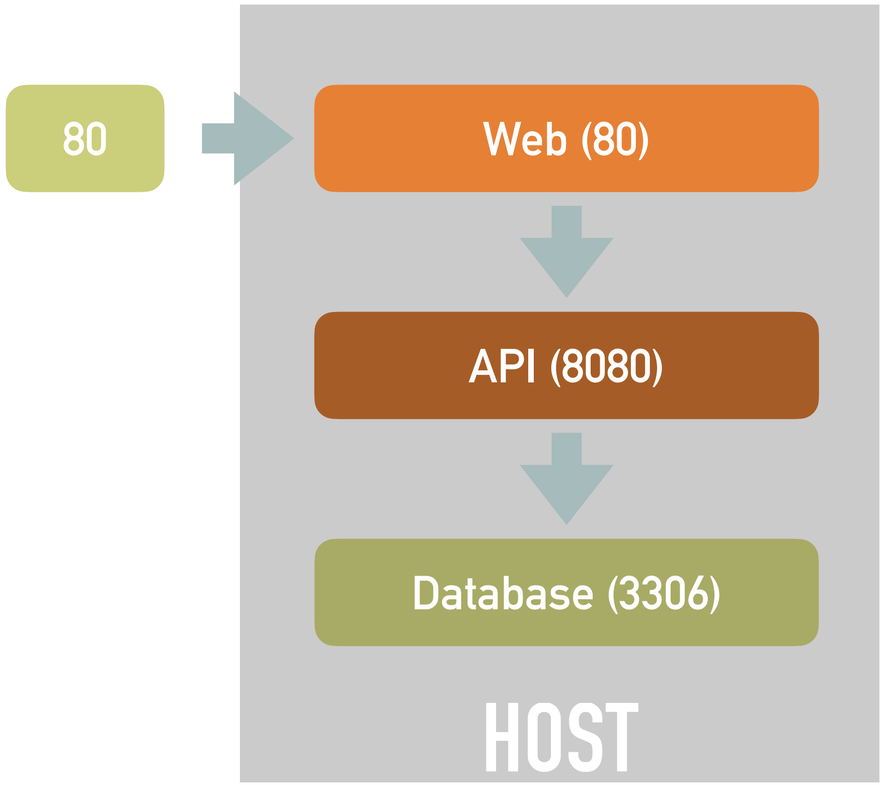
- Overview of Docker Compose
- Docker Compose File
- Docker Compose Commands
- Common use cases



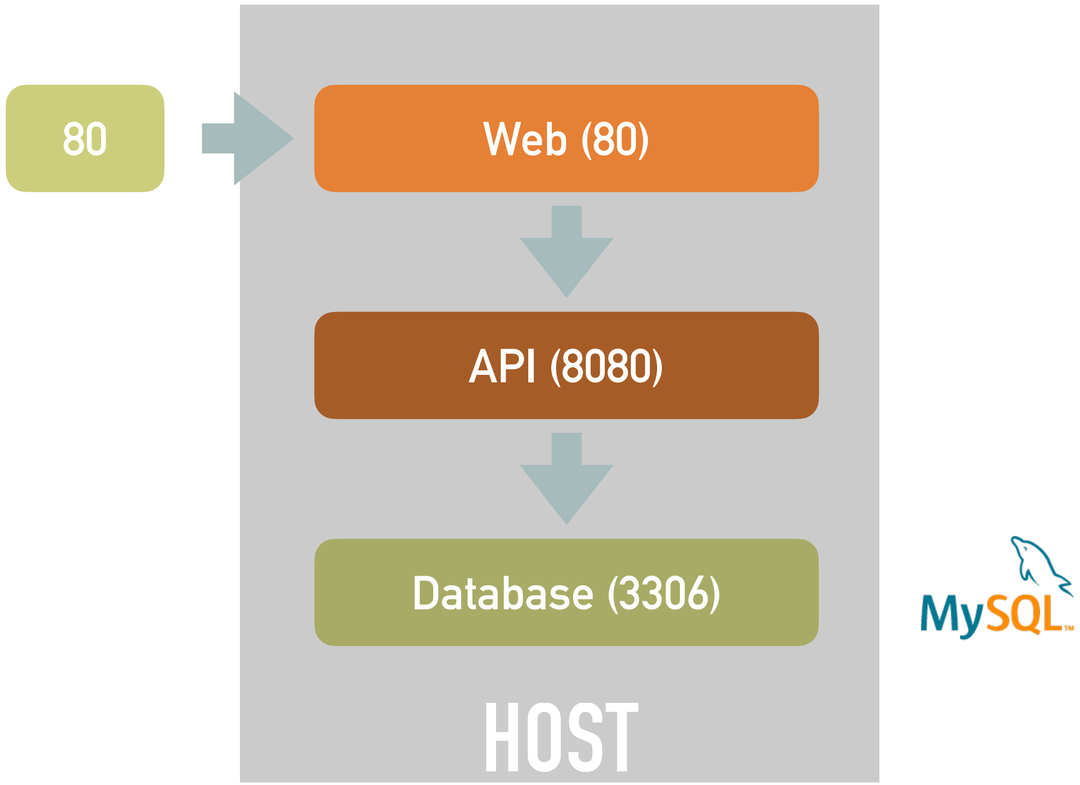
OVERVIEWS OF DOCKER COMPOSE

MULTIPLE CONTAINERS/COMPONENTS APPLICATION

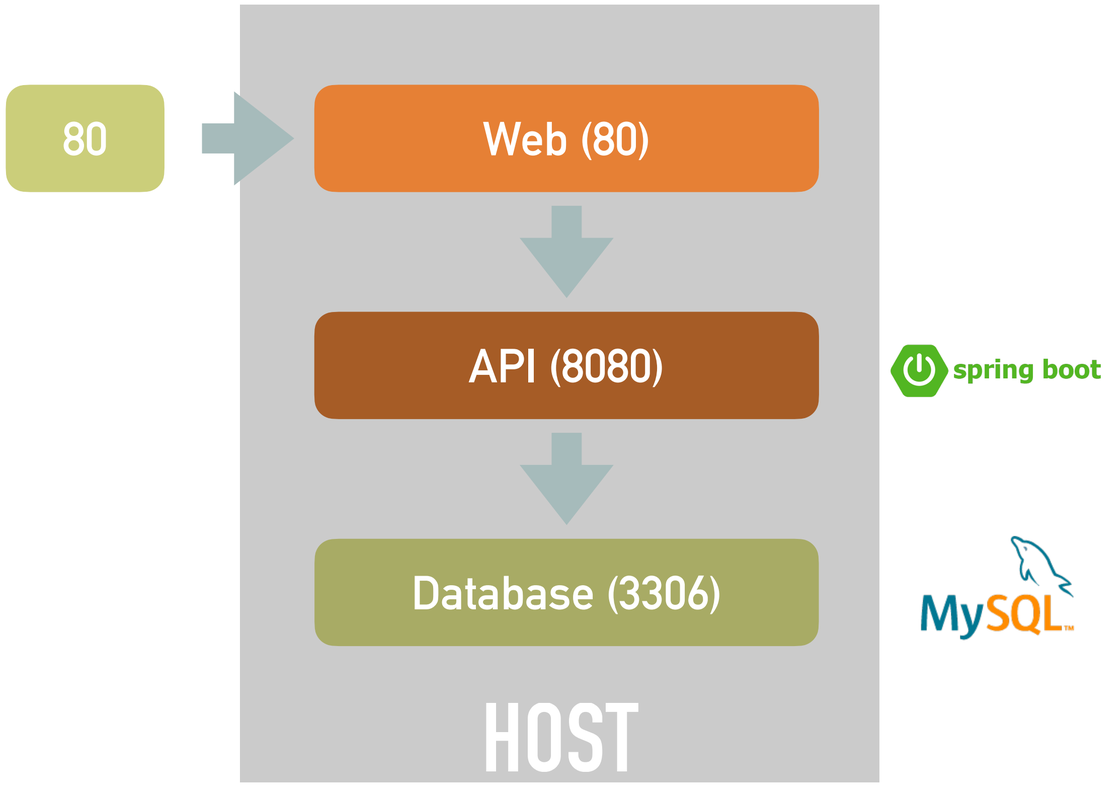
MULTIPLE CONTAINERS/COMPONENTS APPLICATION



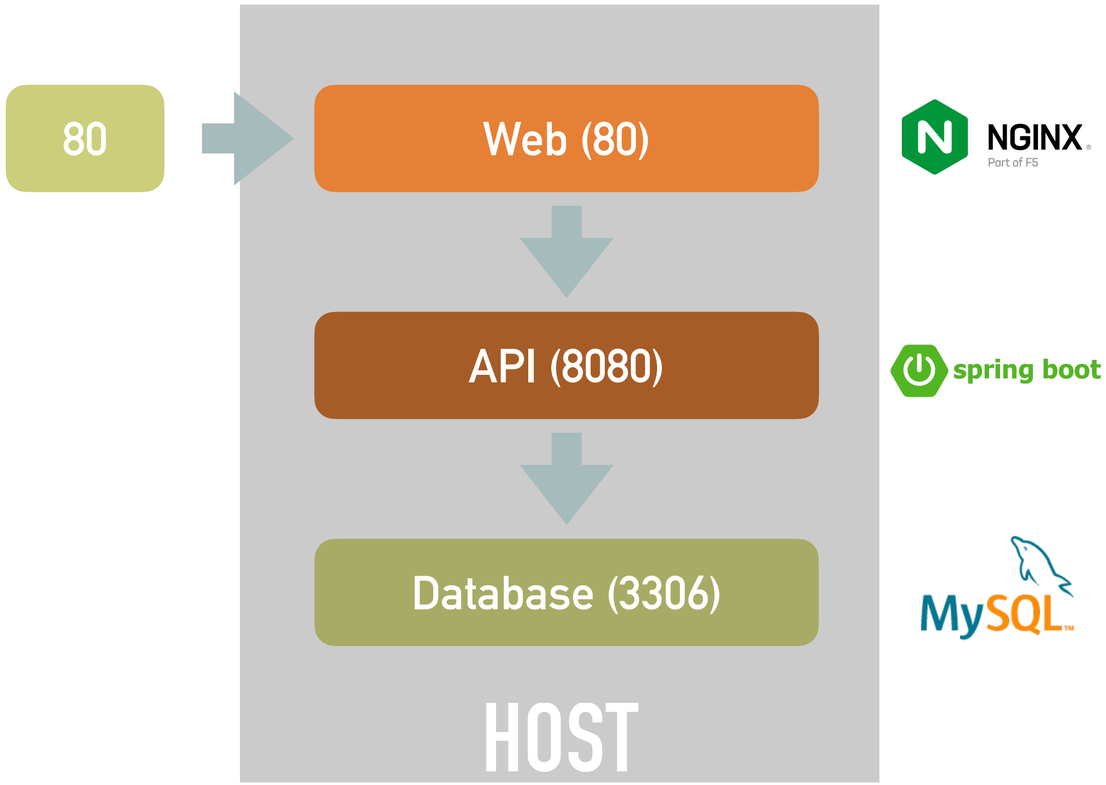
MULTIPLE CONTAINERS/COMPONENTS APPLICATION



MULTIPLE CONTAINERS/COMPONENTS APPLICATION

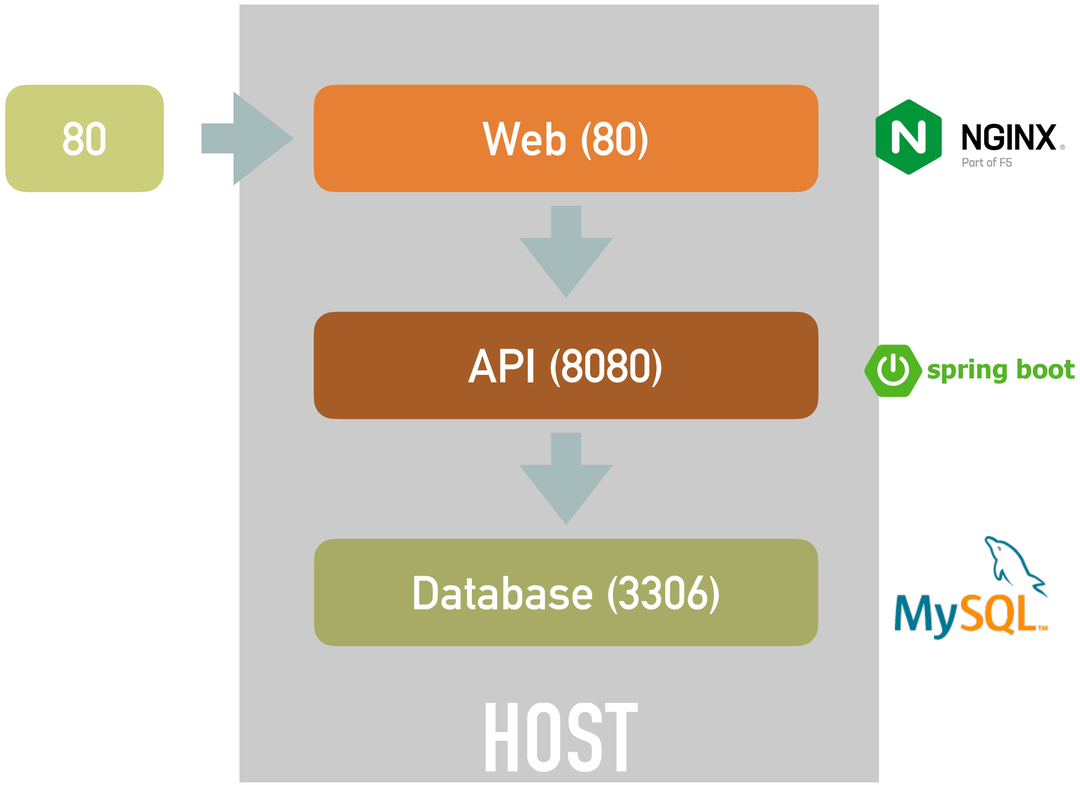


MULTIPLE CONTAINERS/COMPONENTS APPLICATION

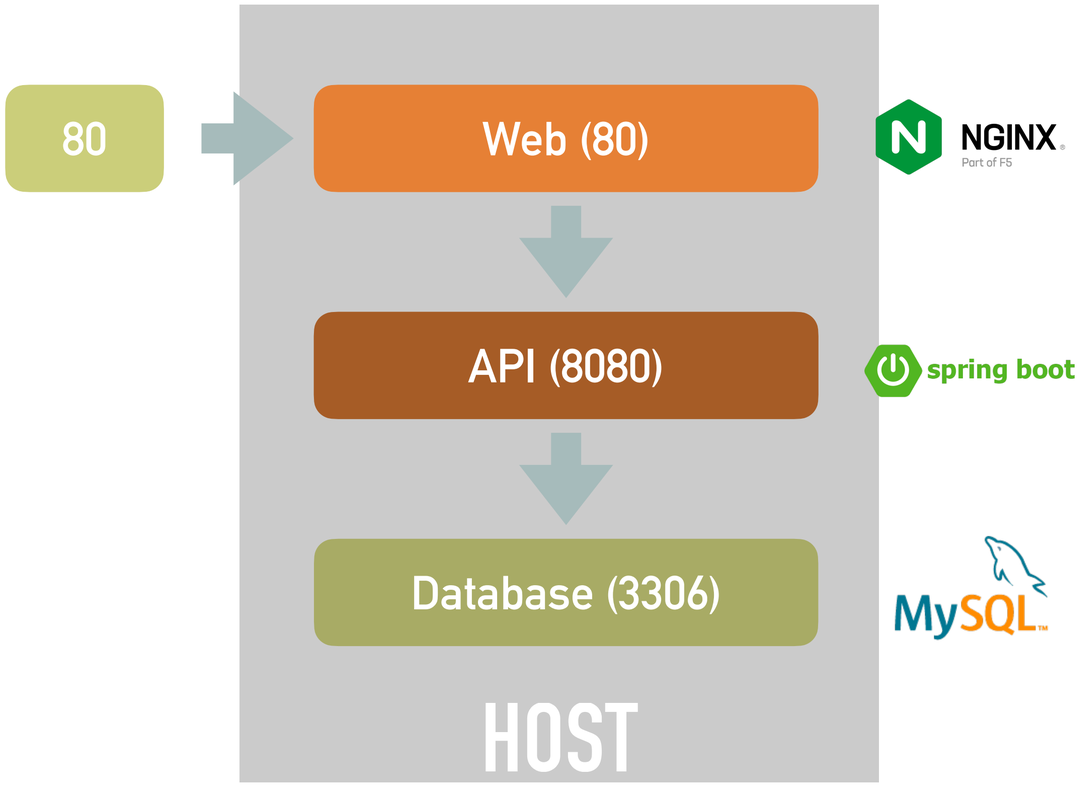


BUILD MULTIPLE IMAGES

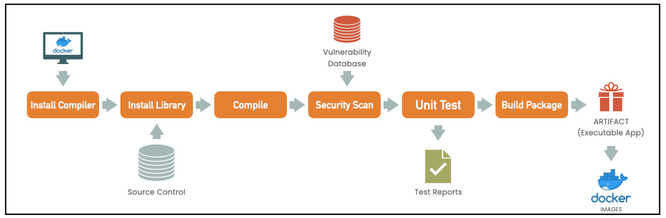
BUILD MULTIPLE IMAGES



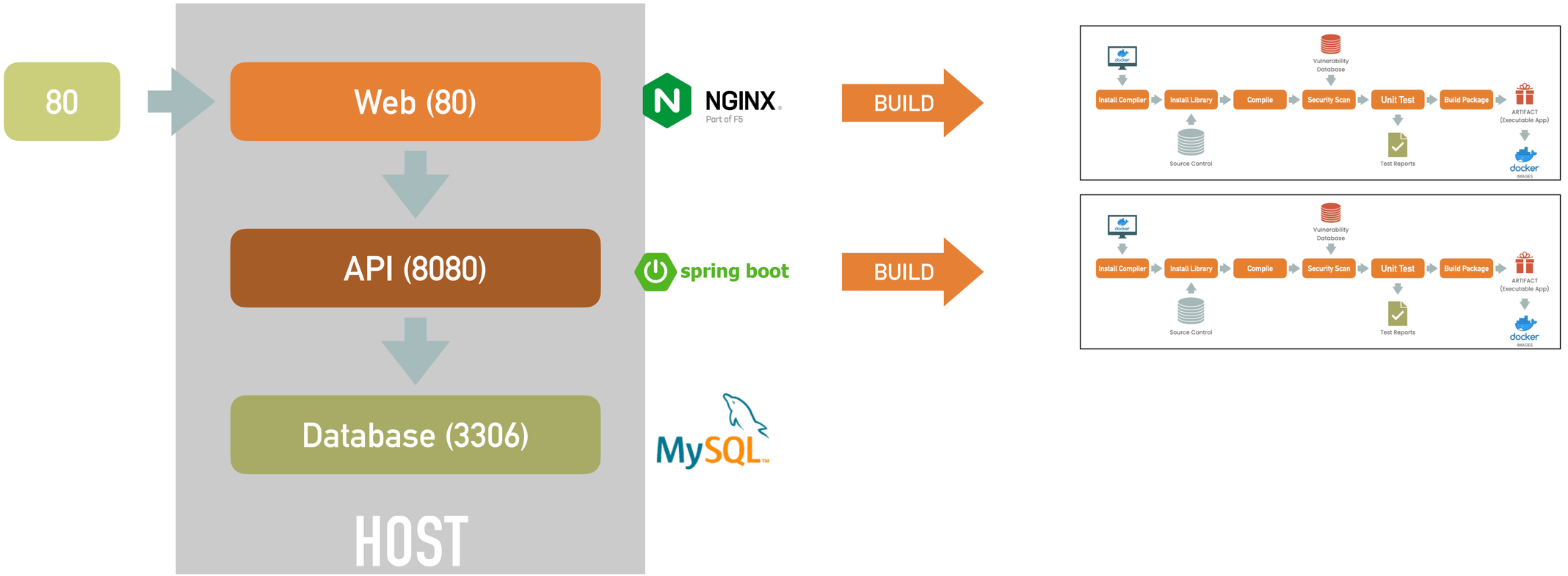
BUILD MULTIPLE IMAGES



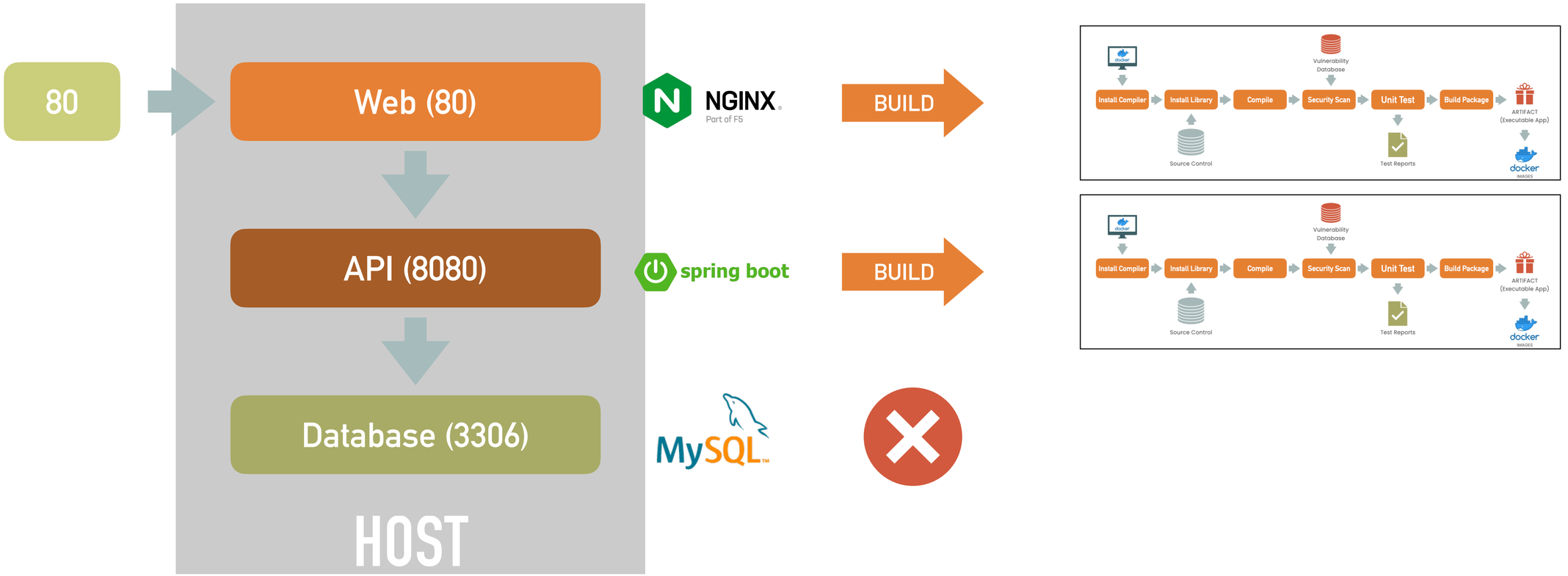
BUILD



BUILD MULTIPLE IMAGES

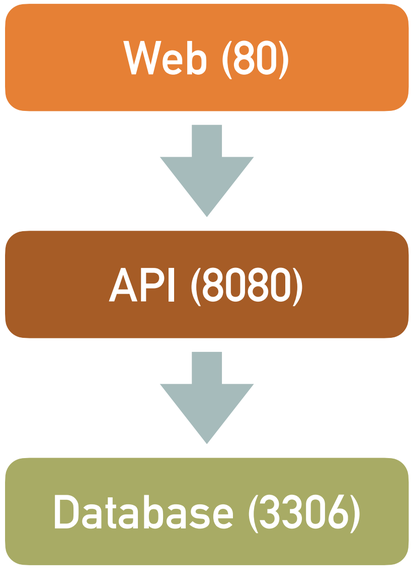


BUILD MULTIPLE IMAGES



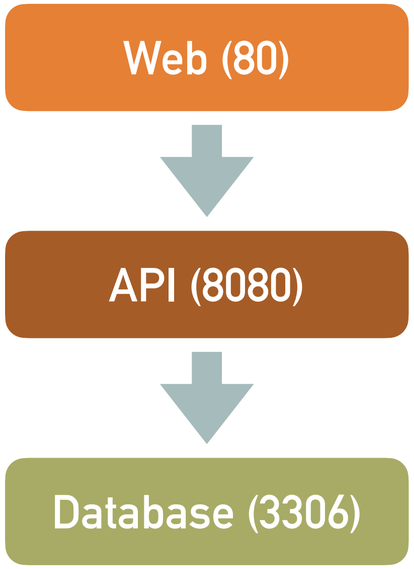
MULTIPLE CONTAINERS/COMPONENTS APPLICATION

MULTIPLE CONTAINERS/COMPONENTS APPLICATION

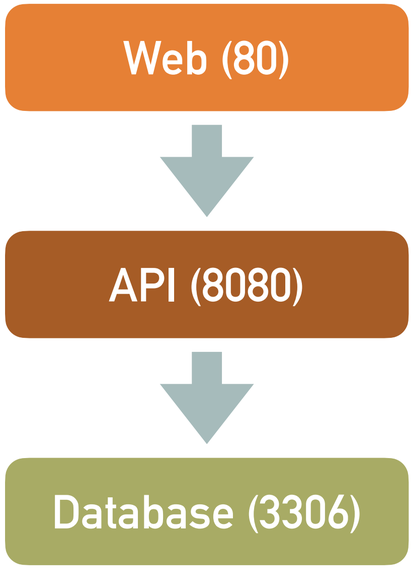


MULTIPLE CONTAINERS/COMPONENTS APPLICATION

$ docker pull mysql:8.0.28

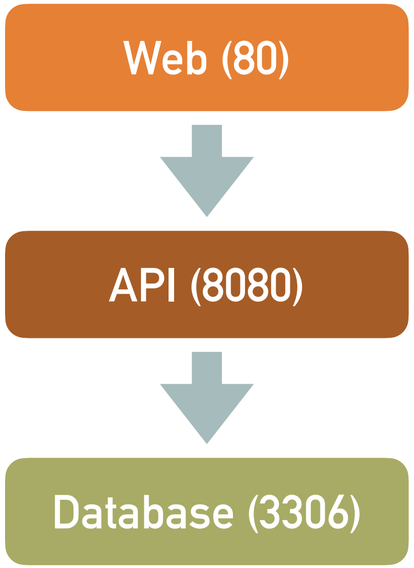


MULTIPLE CONTAINERS/COMPONENTS APPLICATION



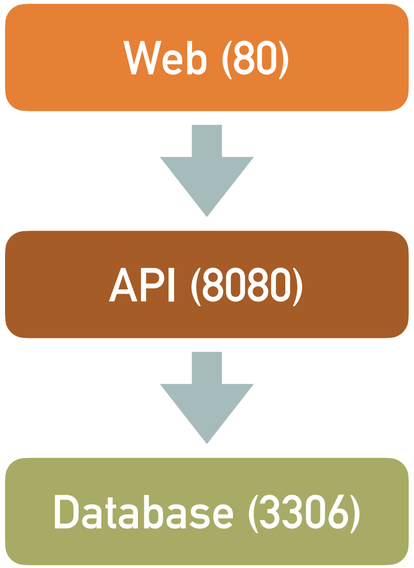

```
$ docker pull mysql:8.0.28
$ docker build -t api .
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



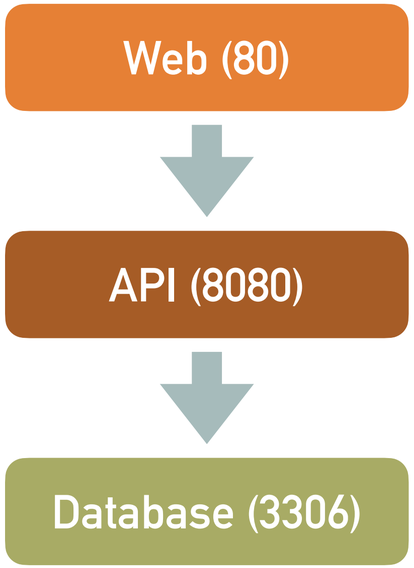

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



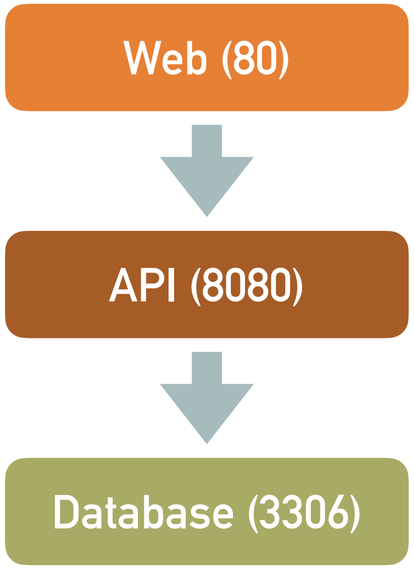

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



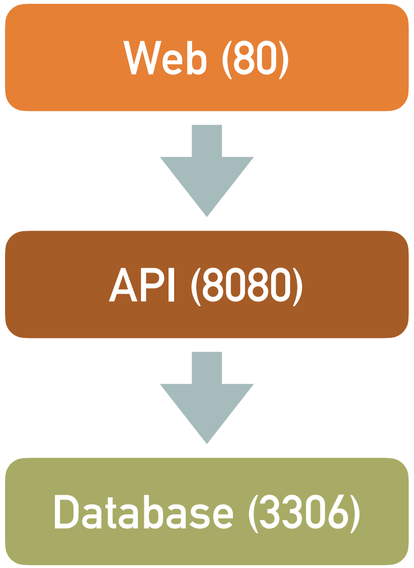

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



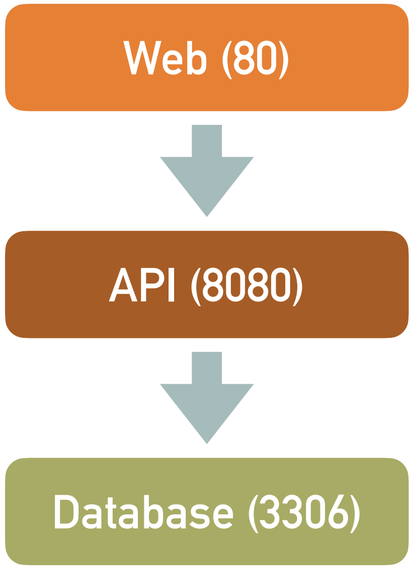

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



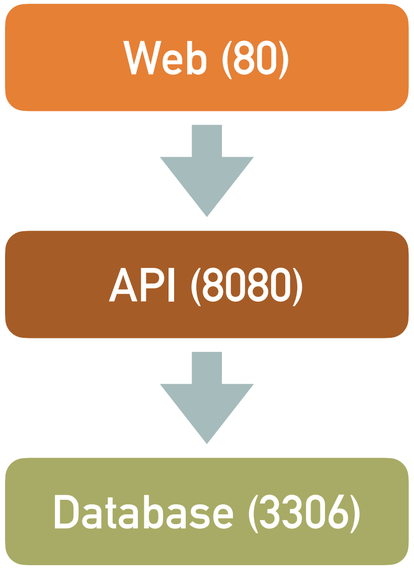

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



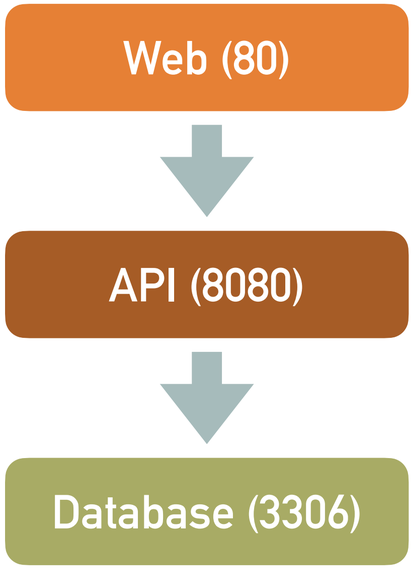

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
    -p 8080:3000
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



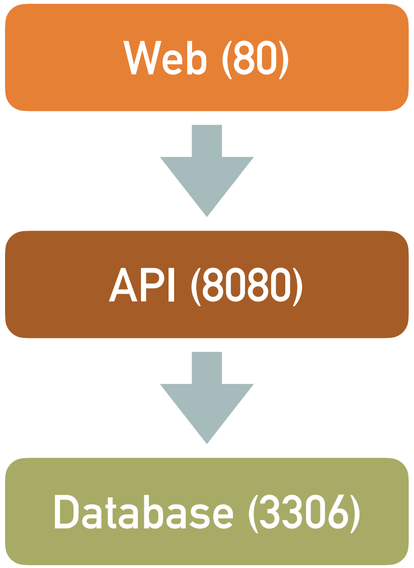

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
    -p 8080:3000
    -e DB_HOST=db
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



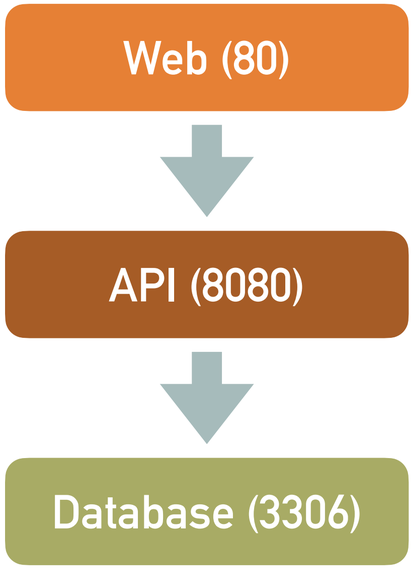

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
    -p 8080:3000
    -e DB_HOST=db
    -v $(pwd):/code myself/api
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



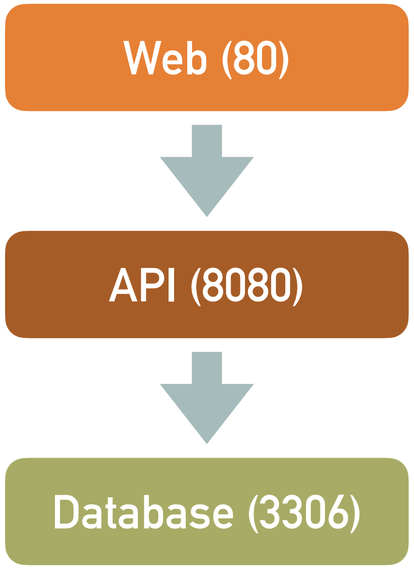

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
    -p 8080:3000
    -e DB_HOST=db
    -v $(pwd):/code myself/api
$docker run -d --name web --net=my_app_net
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



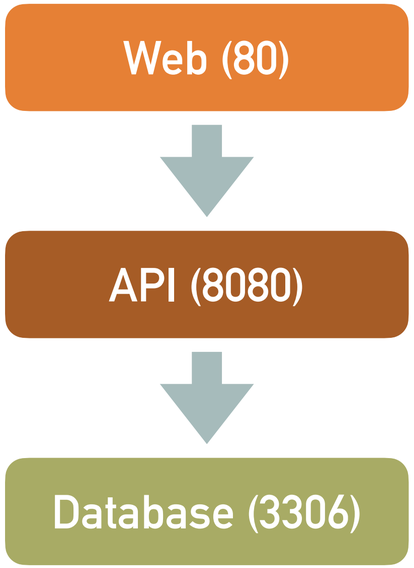

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
    -p 8080:3000
    -e DB_HOST=db
    -v $(pwd):/code myself/api
$docker run -d --name web --net=my_app_net
    -p 80:80
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION




```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
    -p 8080:3000
    -e DB_HOST=db
    -v $(pwd):/code myself/api
$docker run -d --name web --net=my_app_net
    -p 80:80
    -e API_HOST=api
```

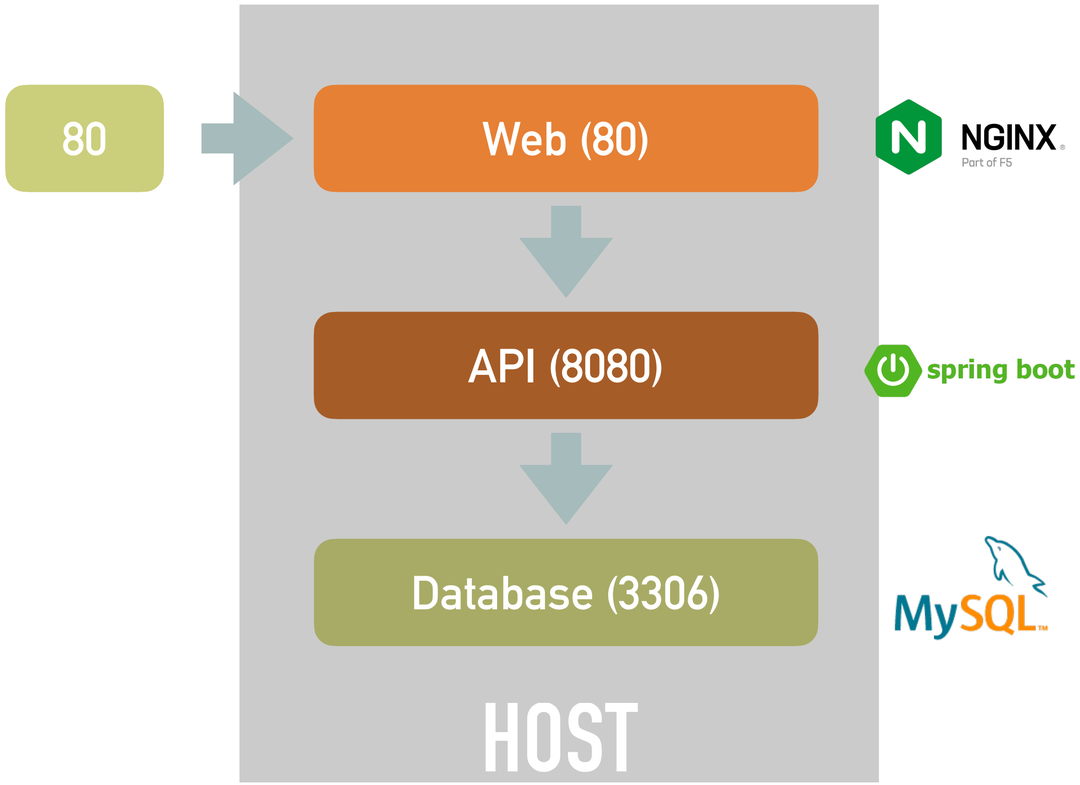
MULTIPLE CONTAINERS/COMPONENTS APPLICATION



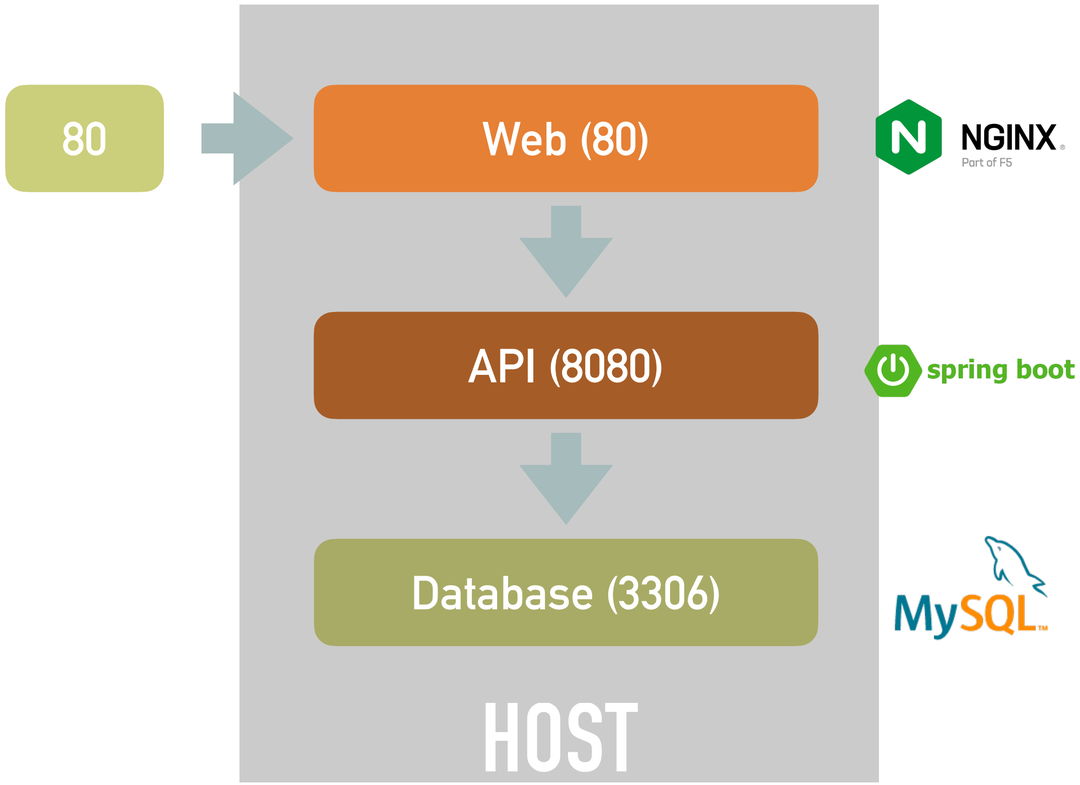

```
$ docker pull mysql:8.0.28
$ docker build -t api .
$ docker build -t web .

$ docker network create -d bridge my_app_net
$ docker run -d --name db --net=my_app_net mysql
$ docker run -d --name api --net=my_app_net
    -p 8080:3000
    -e DB_HOST=db
    -v $(pwd):/code myself/api
$docker run -d --name web --net=my_app_net
    -p 80:80
    -e API_HOST=api
    -v $(pwd):/cdn myself/web
```

MULTIPLE CONTAINERS/COMPONENTS APPLICATION

MULTIPLE CONTAINERS/COMPONENTS APPLICATION



MULTIPLE CONTAINERS/COMPONENTS APPLICATION

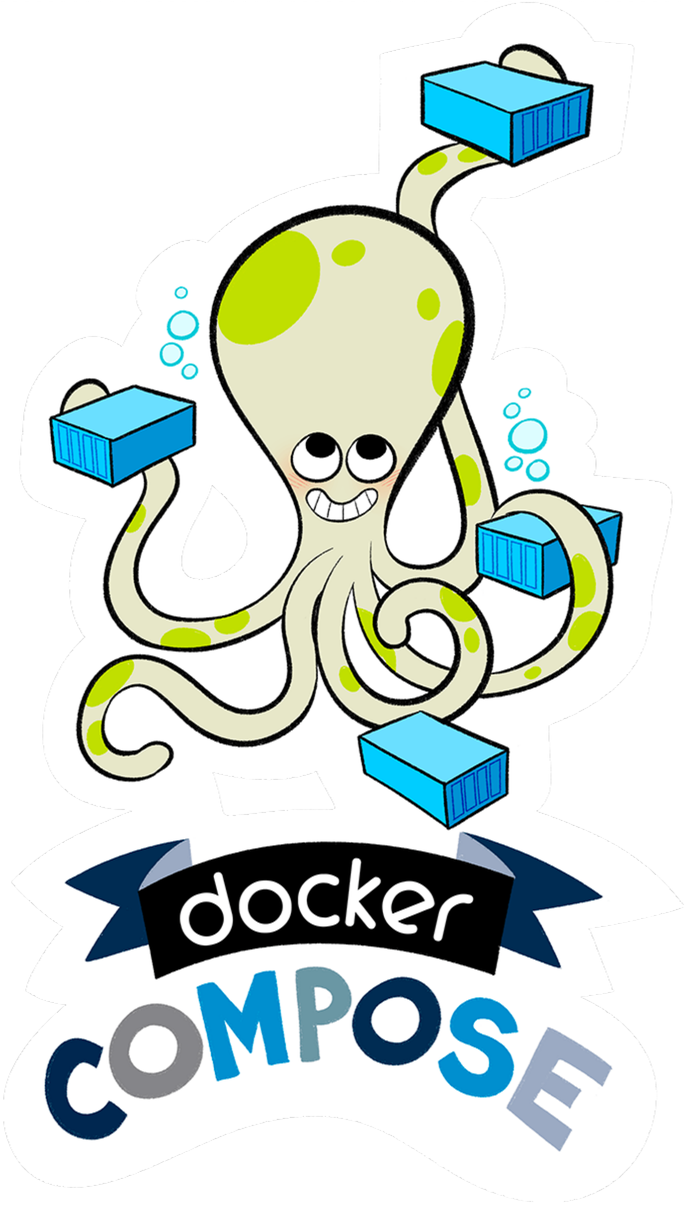


Difficult to CREATE/MANAGE

- Build images from Dockerfile
- Pull images from Hub/private/cache
- Configuration and create containers
- Start and stop containers
- Network and Volume
- Stream their logs

CAN WE RUN ALL OF THIS IN ONLY ONE COMMAND ?

DOCKER COMPOSE



Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

STEP PROCESS FOR USING DOCKER COMPOSE

Using Compose is basically a three-step process:

1. Define your app's environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Run docker compose up and the Docker compose command starts and runs your entire app. You can alternatively run docker-compose up using the docker-compose binary.

RUNNING APP IN ONE COMMAND-LINE

RUNNING APP IN ONE COMMAND-LINE


```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  Db:
    container_name: db
    image: redis
```

RUNNING APP IN ONE COMMAND-LINE

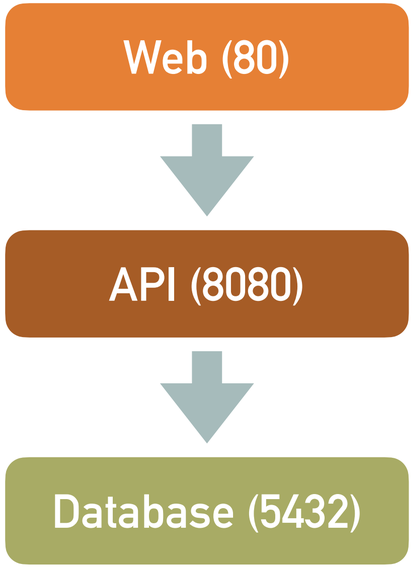

```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  Db:
    container_name: db
    image: redis
```

$ docker-compose up

RUNNING APP IN ONE COMMAND-LINE


```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  Db:
    container_name: db
    image: redis
```

$ docker-compose up



DOCKER COMPOSE COMMANDS

DOCKER COMPOSE COMMAND

```
$ docker-compose --help
```

DOCKER COMPOSE COMMAND

```
$ docker-compose --help
```

DOCKER COMPOSE COMMAND

```
$ docker-compose --help
```

```
Define and run multi-container applications with Docker.

Usage:
  docker-compose [-f <arg>...] [--profile <name>...] [options] [--] [COMMAND] [ARGS...]
  docker-compose -h|--help

Options:
  -f, --file FILE             Specify an alternate compose file
                              (default: docker-compose.yml)
  -p, --project-name NAME     Specify an alternate project name
                              (default: directory name)
  --profile NAME              Specify a profile to enable
```

DOCKER COMPOSE COMMAND

```
Commands:
  build              Build or rebuild services
  config             Validate and view the Compose file
  create             Create services
  down               Stop and remove resources
  events             Receive real time events from containers
  exec               Execute a command in a running container
  help               Get help on a command
  images             List images
  kill               Kill containers
  logs               View output from containers
  pause              Pause services
  port               Print the public port for a port binding
  ps                 List containers
  pull               Pull service images
  push               Push service images
  restart            Restart services
```

EASY TO USE

Compose has commands for managing the whole lifecycle of your application:

- Start, stop, and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

EASY TO USE

Compose has commands for managing the whole lifecycle of your application:

- Start, stop, and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

```
$ docker build .
```

```
$ docker-compose build
```

EASY TO USE

Compose has commands for managing the whole lifecycle of your application:

- Start, stop, and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

```
$ docker build .
```

```
$ docker-compose build
```

```
$ docker run -d <image>
```

```
$ docker-compose up -d
```

EASY TO USE

Compose has commands for managing the whole lifecycle of your application:

- Start, stop, and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

```
$ docker build .
```

```
$ docker-compose build
```

```
$ docker run -d <image>
```

```
$ docker-compose up -d
```

-d = run in background

EASY TO USE

Compose has commands for managing the whole lifecycle of your application:

- Start, stop, and rebuild services
- View the status of running services
- Stream the log output of running services
- Run a one-off command on a service

```
$ docker build .
```

```
$ docker-compose build
```

```
$ docker run -d <image>
```

```
$ docker-compose up -d
```

-d = run in background

```
$ docker stop <container>
+ docker rm <container>
```

```
$ docker-compose down
```

LIST ALL CONTAINERS IN COMPOSE

LIST ALL CONTAINERS IN COMPOSE

```
$ docker container ls -a
```

```
CONTAINER ID   IMAGE                                              COMMAND                  CREATED         STATUS         PORTS                                       NAMES
cf3e8b82fcd9   siamchamnankit/toy-store-nginx:0.0.1               "nginx -g 'daemon of…"   4 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp                          store-nginx
0698304dad53   siamchamnankit/toy-store-service:0.0.1             "./app"                  4 minutes ago   Up 4 minutes   0.0.0.0:8000->8000/tcp                      store-service
a6d0f66fe5f9   mysql:8.0.22                                       "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp           store-database-mysql
5da9a8c22a18   siamchamnankit/toy-store-shippinggateway:0.0.1     "node bin/mb start -…"   4 minutes ago   Up 4 minutes   2525/tcp, 0.0.0.0:8883->8882/tcp            shipping-gateway
36dbe81d1e8a   siamchamnankit/toy-store-web:0.0.1                 "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:3000->3000/tcp                      store-web
2b8495cd62bf   redis:latest                                       "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:6379->6379/tcp                      store-cache
4807d79bbc10   siamchamnankit/toy-store-bankgateway:0.0.1         "node bin/mb start -…"   4 minutes ago   Up 4 minutes   2525/tcp, 0.0.0.0:8882->8882/tcp            bank-gateway
```

LIST ALL CONTAINERS IN COMPOSE

```
$ docker container ls -a
```

```
CONTAINER ID   IMAGE                                              COMMAND                  CREATED         STATUS         PORTS                                       NAMES
cf3e8b82fcd9   siamchamnankit/toy-store-nginx:0.0.1               "nginx -g 'daemon of…"   4 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp                          store-nginx
0698304dad53   siamchamnankit/toy-store-service:0.0.1             "./app"                  4 minutes ago   Up 4 minutes   0.0.0.0:8000->8000/tcp                      store-service
a6d0f66fe5f9   mysql:8.0.22                                       "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp           store-database-mysql
5da9a8c22a18   siamchamnankit/toy-store-shippinggateway:0.0.1     "node bin/mb start -…"   4 minutes ago   Up 4 minutes   2525/tcp, 0.0.0.0:8883->8882/tcp            shipping-gateway
36dbe81d1e8a   siamchamnankit/toy-store-web:0.0.1                 "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:3000->3000/tcp                      store-web
2b8495cd62bf   redis:latest                                       "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:6379->6379/tcp                      store-cache
4807d79bbc10   siamchamnankit/toy-store-bankgateway:0.0.1         "node bin/mb start -…"   4 minutes ago   Up 4 minutes   2525/tcp, 0.0.0.0:8882->8882/tcp            bank-gateway
```

LIST ALL CONTAINERS IN COMPOSE

```
$ docker container ls -a
```

```
CONTAINER ID   IMAGE                                          COMMAND                  CREATED         STATUS         PORTS                                       NAMES
cf3e8b82fcd9   siamchamnankit/toy-store-nginx:0.0.1           "nginx -g 'daemon of…"   4 minutes ago   Up 4 minutes   0.0.0.0:80->80/tcp                          store-nginx
0698304dad53   siamchamnankit/toy-store-service:0.0.1         "./app"                  4 minutes ago   Up 4 minutes   0.0.0.0:8000->8000/tcp                      store-service
a6d0f66fe5f9   mysql:8.0.22                                   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:3306->3306/tcp, 33060/tcp           store-database-mysql
5da9a8c22a18   siamchamnankit/toy-store-shippinggateway:0.0.1 "node bin/mb start -…"  4 minutes ago   Up 4 minutes   2525/tcp, 0.0.0.0:8883->8882/tcp            shipping-gateway
36dbe81d1e8a   siamchamnankit/toy-store-web:0.0.1             "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:3000->3000/tcp                      store-web
2b8495cd62bf   redis:latest                                   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:6379->6379/tcp                      store-cache
4807d79bbc10   siamchamnankit/toy-store-bankgateway:0.0.1     "node bin/mb start -…"   4 minutes ago   Up 4 minutes   2525/tcp, 0.0.0.0:8882->8882/tcp            bank-gateway
```

```
$ docker-compose ps
```

```
        Name                      Command                  State                Ports
-------------------------------------------------------------------------------------------------
bank-gateway            node bin/mb start --config ...   Up      2525/tcp, 0.0.0.0:8882->8882/tcp
shipping-gateway        node bin/mb start --config ...   Up      2525/tcp, 0.0.0.0:8883->8882/tcp
store-cache             docker-entrypoint.sh redis ...   Up      0.0.0.0:6379->6379/tcp
store-database-mysql    docker-entrypoint.sh mysqld      Up      0.0.0.0:3306->3306/tcp, 33060/tcp
store-nginx             nginx -g daemon off;             Up      0.0.0.0:80->80/tcp
store-service           ./app                            Up      0.0.0.0:8000->8000/tcp
store-web               docker-entrypoint.sh /bin/ ...   Up      0.0.0.0:3000->3000/tcp
```

KILL AND REMOVE CONTAINERS

KILL AND REMOVE CONTAINERS

```
$ docker kill <container>
```

```
$ docker-compose kill
```

KILL AND REMOVE CONTAINERS

```
$ docker kill <container>
```

```
$ docker-compose kill
```

```
$ docker rm <container>
```

```
$ docker-compose rm
```

KILL AND REMOVE CONTAINERS

```
$ docker kill <container>
```

```
$ docker-compose kill
```

```
$ docker rm <container>
```

```
$ docker-compose rm
```

```
$ docker stop <container>
```

```
$ docker-compose stop
```

SPECIFYING MULTIPLE COMPOSE FILE (-F)

SPECIFYING MULTIPLE COMPOSE FILE (-F)

```
$ docker-compose <command>
```

SPECIFYING MULTIPLE COMPOSE FILE (-F)

```
$ docker-compose <command>
```

default → docker-compose.yml

```
$ docker-compose -f docker-compose.deploy.yml <command>
```

specific → docker-compose.deploy.yml

SPECIFYING MULTIPLE COMPOSE FILE (-F)

```
$ docker-compose <command>
```

default → docker-compose.yml

```
$ docker-compose -f docker-compose.deploy.yml <command>
```

specific → docker-compose.deploy.yml

```
$ docker-compose -f docker-compose.yml
                 -f docker-compose.deploy.yml <command>
```

SPECIFYING MULTIPLE COMPOSE FILE (-F)

```
$ docker-compose <command>
```

default → docker-compose.yml

```
$ docker-compose -f docker-compose.deploy.yml <command>
```

specific → docker-compose.deploy.yml

```
$ docker-compose -f docker-compose.yml
                 -f docker-compose.deploy.yml <command>
```

merge

SPECIFYING MULTIPLE COMPOSE FILE (-F)

```
$ docker-compose <command>
```

default → docker-compose.yml

```
$ docker-compose -f docker-compose.deploy.yml <command>
```

specific → docker-compose.deploy.yml

```
$ docker-compose -f docker-compose.yml
                 -f docker-compose.deploy.yml <command>
```

merge → docker-compose.yml

USE

SPECIFYING MULTIPLE COMPOSE FILE (-F)

```
$ docker-compose <command>
```

default → docker-compose.yml

```
$ docker-compose -f docker-compose.deploy.yml <command>
```

specific → docker-compose.deploy.yml

```
$ docker-compose -f docker-compose.yml
                 -f docker-compose.deploy.yml <command>
```

merge → docker-compose.yml

USE

docker-compose.deploy.yml

Combine

SPECIFYING MULTIPLE COMPOSE FILE (-F)

```
$ docker-compose <command>
```

default → docker-compose.yml

```
$ docker-compose -f docker-compose.deploy.yml <command>
```

specific → docker-compose.deploy.yml

```
$ docker-compose -f docker-compose.yml
                 -f docker-compose.deploy.yml <command>
```

merge → docker-compose.yml

USE → docker-compose.deploy.yml

Combine → *combines them into a single configuration.*

in the order you supply the files

DOCKER COMPOSE FILE

DOCKER-COMPOSE.YML - VERSION AND SERVICES

```yaml
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  db:
    container_name: db
    image: redis
    links:
      - web
```

DOCKER-COMPOSE.YML - VERSION AND SERVICES

Version of Docker-Compose

```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  db:
    container_name: db
    image: redis
    links:
      - web
```

DOCKER-COMPOSE.YML - VERSION AND SERVICES

Version of Docker-Compose

Services

```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  db:
    container_name: db
    image: redis
    links:
      - web
```

DOCKER-COMPOSE.YML - VERSION AND SERVICES

Version of Docker-Compose

Services

```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  db:
    container_name: db
    image: redis
    links:
      - web
```

DOCKER-COMPOSE.YML - VERSION AND SERVICES

Version of Docker-Compose

Services

```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  db:
    container_name: db
    image: redis
    links:
      - web
```

DOCKER-COMPOSE.YML - VERSION AND SERVICES

Version of Docker-Compose

Services

```
version: '3'
services:
  web:
    container_name: web
    build: .
    ports:
      - 80:5000
    environment:
      - API_HOST=api
  api:
    container_name: redis
    context: api
  db:
    container_name: db
    image: redis
    links:
      - web
```

DOCKER-COMPOSE.YML - BUILDING SERVICE

DOCKER-COMPOSE.YML - BUILDING SERVICE

```
$ docker build .
```

DOCKER-COMPOSE.YML - BUILDING SERVICE

```
$ docker build .
```

```
version: '3'
services:
  web:
    build: .
    container_name: web
    ports:
      - 80:5000
    environment:
      - API_HOST=api
```

DOCKER-COMPOSE.YML - BUILDING SERVICE

```
$ docker build .
```

```
version: '3'
services:
  web:
    build: .
    container_name: web
    ports:
      - 80:5000
    environment:
      - API_HOST=api
```

```
$ docker build -t webapp:1.0.0
               -f web/Dockerfile.web
               web
```

DOCKER-COMPOSE.YML - BUILDING SERVICE

```
$ docker build .
```

```
version: '3'
services:
  web:
    build: .
    container_name: web
    ports:
      - 80:5000
    environment:
      - API_HOST=api
```

```
$ docker build -t webapp:1.0.0
               -f web/Dockerfile.web
               web
```

```
version: '3'
services:
  web:
    build:
      context: ./web
      dockerfile: Dockerfile.web
    image: webapp:1.0.0
    container_name: web
```

DOCKER-COMPOSE.YML - RUNNING SERVICE

DOCKER-COMPOSE.YML - RUNNING SERVICE

```
$ docker run mysql
```

DOCKER-COMPOSE.YML - RUNNING SERVICE

```
$ docker run mysql
```

```
version: '3'
services:
  db:
    image: mysql
    ports:
      - 3306:3306
    environment:
        - MYSQL_ROOT_PASSWORD=root
        - MYSQL_USER=sealteam
        - MYSQL_PASSWORD=sckshuhari
        - MYSQL_DATABASE=toy
```

DOCKER-COMPOSE.YML - RUNNING SERVICE

```
$ docker run mysql
```

```
version: '3'
services:
  db:
    image: mysql
    ports:
      - 3306:3306
    environment:
        - MYSQL_ROOT_PASSWORD=root
        - MYSQL_USER=sealteam
        - MYSQL_PASSWORD=sckshuhari
        - MYSQL_DATABASE=toy
```

```
$ docker run --name database sql:8.0.22
```

DOCKER-COMPOSE.YML - RUNNING SERVICE

```
$ docker run mysql
```

```
version: '3'
services:
  db:
    image: mysql
    ports:
      - 3306:3306
    environment:
        - MYSQL_ROOT_PASSWORD=root
        - MYSQL_USER=sealteam
        - MYSQL_PASSWORD=sckshuhari
        - MYSQL_DATABASE=toy
```

```
$ docker run --name database sql:8.0.22
```

```
version: '3'
services:
  db:
    image: mysql:8.0.22
    container_name: database
    ports:
      - 3306:3306
    environment:
        - MYSQL_ROOT_PASSWORD=root
        - MYSQL_USER=sealteam
        - MYSQL_PASSWORD=sckshuhari
        - MYSQL_DATABASE=toy
```

DOCKER-COMPOSE.YML - ENVIRONMENT VARIABLES

DOCKER-COMPOSE.YML - ENVIRONMENT VARIABLES

```
$ docker run -e MYSQL_ROOT_PASSWORD=root
             -e MYSQL_USER=sealteam
             -e MYSQL_PASSWORD=sckshuhari
             -e MYSQL_DATABASE=toy
             mysql:8.0.22
```

DOCKER-COMPOSE.YML - ENVIRONMENT VARIABLES

```
$ docker run -e MYSQL_ROOT_PASSWORD=root
             -e MYSQL_USER=sealteam
             -e MYSQL_PASSWORD=sckshuhari
             -e MYSQL_DATABASE=toy
             mysql:8.0.22
```

```
version: '3'
services:
  db:
    image: mysql:8.0.22
    container_name: database
    environment:
        - MYSQL_ROOT_PASSWORD=root
        - MYSQL_USER=sealteam
        - MYSQL_PASSWORD=sckshuhari
        - MYSQL_DATABASE=toy
    ports:
      - 3306:3306
```

DOCKER-COMPOSE.YML - PUBLISH LIST (PORT/PROTOCAL)

DOCKER-COMPOSE.YML - PUBLISH LIST (PORT/PROTOCAL)

```
$ docker run -p 3306:3306 mysql:8.0.22
```

DOCKER-COMPOSE.YML - PUBLISH LIST (PORT/PROTOCAL)

```
$ docker run -p 3306:3306 mysql:8.0.22
```

```
version: '3'
services:
  db:
    image: mysql:8.0.22
    container_name: database
    environment:
        - MYSQL_ROOT_PASSWORD=root
        - MYSQL_USER=sealteam
        - MYSQL_PASSWORD=sckshuhari
        - MYSQL_DATABASE=toy
    ports:
      - 3306:3306
```

DOCKER-COMPOSE.YML - VOLUMES

DOCKER-COMPOSE.YML - VOLUMES

```
$ docker container run --name hello-nginx -d -p 80:80
                       -v /user/web:/usr/share/nginx/html
                       nginx:latest
```

DOCKER-COMPOSE.YML - VOLUMES

```
$ docker container run --name hello-nginx -d -p 80:80
                       -v /user/web:/usr/share/nginx/html
                       nginx:latest
```

```
version: '3'
services:
  nginx:
    image: nginx:latest
    cotainer_name: hello-nginx
    ports:
      - 80:80
    volumes:
      - /user/web:/usr/share/nginx/html
```

DOCKER-COMPOSE.YML - DEPEND-ON

DOCKER-COMPOSE.YML - DEPEND-ON

```
$ docker run -p 3306:3306 mysql:8.0.22
$ docker run -p 8080:8080 my/api-server:0.0.1
```

DOCKER-COMPOSE.YML - DEPEND-ON

```
$ docker run -p 3306:3306 mysql:8.0.22
$ docker run -p 8080:8080 my/api-server:0.0.1
```

```
version: '3'
services:
  db:
    image: mysql:8.0.22
    ports:
      - 3306:3306
  api:
    image: my/api-server:0.0.1
    ports:
      - 8080:8080
    depends_on:
      - db
```

DOCKER-COMPOSE.YML - CREATE NETWORK

```
$ docker network create -d bridge my_bridge
```

DOCKER-COMPOSE.YML - CREATE NETWORK

```
$ docker network create -d bridge my_bridge
```

```yaml
version: '3'
services:
  db:
    image: mysql:8.0.22
    ports:
      - 3306:3306
  api:
    image: my/api-server:0.0.1
    ports:
      - 8080:8080
networks:
  my_bridge:
```

DOCKER-COMPOSE.YML - CONFIG CONTAINER'S NETWORK

DOCKER-COMPOSE.YML - CONFIG CONTAINER'S NETWORK

```
$ docker container run --net=my_bridge --name db -d -p 3306:3306 mysql:8.0.22
```

DOCKER-COMPOSE.YML - CONFIG CONTAINER'S NETWORK

```
$ docker container run --net=my_bridge --name db -d -p 3306:3306 mysql:8.0.22
```

```
$ docker container run --net=my_bridge --name api -d -p 8080:8080 my/api:0.0.1
```

DOCKER-COMPOSE.YML - CONFIG CONTAINER'S NETWORK

```
$ docker container run --net=my_bridge --name db -d -p 3306:3306 mysql:8.0.22
```

```
$ docker container run --net=my_bridge --name api -d -p 8080:8080 my/api:0.0.1
```

```
version: '3'
services:
  db:
    image: mysql:8.0.22
    ports:
      - 3306:3306
    networks:
      - my_bridge
  api:
    image: my/api:0.0.1
    ports:
      - 8080:8080
    networks:
      - my_bridge
networks:
  my_bridge:
```

DOCKER-COMPOSE FILE EXAMPLE

docker-compose.yml

```yaml
version: "3.5"

services:
  store-service:
    image: sckseal/toy-store-service:0.0.3
    container_name: store-service
    build:
      context: store-service
    ports:
      - "8000:8000"
    depends_on:
      - store-database
    restart: always
    environment:
      - TEST_MODE=true
      - CACHE_ON=true
      - ELASTIC_APM_SERVER_URL=http://host.docker.internal:8200
      - ELASTIC_APM_SERVICE_NAME=shopping-cart

  store-cache:
    image: redis:latest
    container_name: store-cache
    ports:
      - "6379:6379"

  store-database:
    image: mysql:8.0.22
    container_name: store-database
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_USER=sealteam
      - MYSQL_PASSWORD=sckshuhari
      - MYSQL_DATABASE=toy
    volumes:
      - ./tearup/:/docker-entrypoint-initdb.d/
    ports:
      - "3306:3306"
```

store-service > Dockerfile

```dockerfile
FROM golang:1.12 as builder
WORKDIR /module
COPY ./go.mod /module/go.mod
COPY ./go.sum /module/go.sum
RUN go mod download
COPY . /module
RUN CGO_ENABLED=0 GOOS=linux go build -o ./bin/app ./cmd/main.go

FROM alpine:3
RUN apk add tzdata && \
    cp /usr/share/zoneinfo/Asia/Bangkok /etc/localtime && \
    echo "Asia/Bangkok" >  /etc/timezone && \
    apk del tzdata
WORKDIR /root/
COPY --from=builder /module/bin .
ENV GIN_MODE release
EXPOSE 8000
CMD ./app
```

DOCKER-COMPOSE FILE EXAMPLE

```yaml
store-web:
  #image: sckseal/toy-store-web:0.0.1
  image: toy-store-web:0.0.1
  container_name: store-web
  build:
    context: front-end
  ports:
    - "3000:3000"

store-nginx:
  image: sckseal/toy-store-nginx:0.0.2
  container_name: store-nginx
  restart: always
  build:
    context: nginx
  depends_on:
    - store-service
    - store-web
  ports:
    - "80:80"

bank-gateway:
  image: sckseal/toy-store-bankgateway:0.0.1
  build:
    context: thirdparty/bank-gateway
  container_name: bank-gateway
  restart: always
  ports:
    - "8882:8882"

shipping-gateway:
  image: sckseal/toy-store-shippinggateway:0.0.1
  build:
    context: thirdparty/shipping-gateway
  container_name: shipping-gateway
  restart: always
```

front-end > Dockerfile

```dockerfile
FROM node:12-alpine
WORKDIR /app
COPY package.json .
COPY package-lock.json .
RUN npm install
COPY . .
RUN npm run build
EXPOSE 3000
CMD npm start
```

COMMON USE CASES

USE CASE : DEVELOPMENT ENVIRONMENTS

docker-compose.yml

```yaml
version: "3.5"

services:
  store-service:
    image: sckseal/toy-store-service:0.0.3
    container_name: store-service
    build:
      context: store-service
    ports:
      - "8000:8000"
    depends_on:
      - store-database
    restart: always
    environment:
      - TEST_MODE=true
      - CACHE_ON=true
      - ELASTIC_APM_SERVER_URL=http://host.docker.internal:8200
      - ELASTIC_APM_SERVICE_NAME=shopping-cart

  store-cache:
    image: redis:latest
    container_name: store-cache
    ports:
      - "6379:6379"

  store-database:
    image: mysql:8.0.22
    container_name: store-database
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_USER=sealteam
      - MYSQL_PASSWORD=sckshuhari
      - MYSQL_DATABASE=toy
    volumes:
      - ./tearup/:/docker-entrypoint-initdb.d/
    ports:
      - "3306:3306"

  store-web:
    #image: sckseal/toy-store-web:0.0.1
    image: toy-store-web:0.0.1
    container_name: store-web
    build:
      context: front-end
    ports:
      - "3000:3000"

  store-nginx:
    image: sckseal/toy-store-nginx:0.0.2
    container_name: store-nginx
    restart: always
    build:
      context: nginx
    depends_on:
      - store-service
      - store-web
    ports:
      - "80:80"

  bank-gateway:
    image: sckseal/toy-store-bankgateway:0.0.1
    build:
      context: thirdparty/bank-gateway
    container_name: bank-gateway
    restart: always
    ports:
      - "8882:8882"

  shipping-gateway:
    image: sckseal/toy-store-shippinggateway:0.0.1
    build:
      context: thirdparty/shipping-gateway
    container_name: shipping-gateway
    restart: always
```

USE CASE : AUTOMATED TESTING ENVIRONMENTS

An important part of any Continuous Deployment or Continuous Integration process is the automated test suite. Automated end-to-end testing requires an environment in which to run tests. Compose provides a convenient way to create and destroy isolated testing environments for your test suite. By defining the full environment in a Compose file, you can create and destroy these environments in just a few commands:

```
$ docker-compose up -d
$ ./run_tests
$ docker-compose down
```

USE CASE : SINGLE HOST DEPLOYMENTS

Compose has traditionally been focused on development and testing workflows, but with each release we're making progress on more production-oriented features.

For details on using production-oriented features, see compose in production

THANK YOU