Chapter 7: Docker Compose in the Real World

7.1. Introduction

Nothing important here.

7.2. Why is it worth to learn Docker Compose?

It's for development and production. It automates all the Docker commands that we saw earlier.

7.3. Adding docker compose support to our web app

Docker compose example:

```
version: '3'
services:
  localstack:
    image: localstack/localstack
    ports:
      - "4567-4584:4567-4584"
      - "${PORT_WEB_UI-8080}:${PORT_WEB_UI-8080}"
    environment:
      - SERVICES=${SERVICES- }
      - DEBUG=${DEBUG- }
      - DATA_DIR=${DATA_DIR- }
      - PORT_WEB_UI=${PORT_WEB_UI- }
      - LAMBDA_EXECUTOR=${LAMBDA_EXECUTOR- }
      - KINESIS_ERROR_PROBABILITY=${KINESIS_ERROR_PROBABILITY- }
      - DOCKER_HOST=unix:///var/run/docker.sock
    volumes:
      - "${TMPDIR:-/tmp/localstack}:/tmp/localstack"
  resolver:
    build: '.'
    depends_on:
      - 'localstack'
    ports:
      - '3000:3000'
    volumes:
      - '.:/app'
```

7.4. Managing our web app

Build images mentioned in docker-compose.yml:

```
docker-compose build
```

Pull images mentioned in docker-compose.yml:

```
docker-compose pull
```

Run docker compose project:

```
TMPDIR=/private$TMPDIR docker-compose up
```

Build and run docker compose project:

```
docker-compose up --build -d
```

Show docker-compose processes (working containers):

```
docker-compose ps
```

Stop or restart all docker-compose processes:

```
docker-compose stop|restart
```

Run commands on container:

```
docker-compose exec resolver ls -la
docker-compose run resolver ls -la
```

Remove stopped containers:

```
docker-compose rm
```

We can override `CMD` sections of a Dockerfile from docker-compose.yml with `command: foo` statement.

7.5. Docker Compose API V1/V2/V3

API V1 is the deprecated version. You can easily notice this, because in V1, there is no `version: N` statement. API V2 and V3 are still relevant and in use.