Docker Crash Course

1 Key Concepts

1.1 Docker Image

A Docker image is a lightweight, standalone, and executable software package that includes everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and configuration files. Docker images are built from `Dockerfile` instructions.

1.2 Dockerfile

A Dockerfile is a text document that contains a series of instructions on how to build a Docker image. It is essentially a blueprint for creating Docker images. Each instruction in a Dockerfile creates a new layer in the image.

1.3 Docker Container

A Docker container is a runtime instance of a Docker image. Containers are isolated from each other and from the host system, ensuring a consistent runtime environment. They can be started, stopped, moved, and deleted using Docker commands.

1.4 Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications. With Docker Compose, you can use a YAML file to define and configure application services, making it easy to deploy and manage multi-container applications.

2 Useful Commands

2.1 Docker Image Commands

- `docker build -t <image_name> .` - Build an image from a Dockerfile in the current directory.
- `docker images` - List all Docker images on the system.

- `docker rmi <image_name>` - Remove a Docker image.

2.2 Docker Container Commands

- `docker run -d -p <host_port>:<container_port> <image_name>` - Run a container from an image in detached mode with port mapping.
- `docker ps` - List running containers.
- `docker stop <container_id>` - Stop a running container.
- `docker rm <container_id>` - Remove a stopped container.

2.3 Docker Compose Commands

- `docker-compose up` - Create and start containers as defined in the `docker-compose.yml` file.
- `docker-compose down` - Stop and remove containers, networks, images, and volumes defined in the `docker-compose.yml` file.
- `docker-compose ps` - List containers managed by Docker Compose.

3 Example Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

4 Example docker-compose.yml

```
version: '3'
services:
  web:
    build: .
    ports:
      - "5000:80"
    volumes:
      - .:/code
    environment:
      FLASK_ENV: development
  redis:
    image: "redis:alpine"
```