

Introduction to Docker

From Containers to Compose

Introduction to Docker

What are Containers?

Using Basic Docker Commands

Example: NGINX Hello World

Build Your Own Container

Example: Dockerizing a Flask App

Docker Compose

Introduction to Docker

- Open-source
- Easy creation, deployment, and debugging of containers
- Easy to move apps between dev, testing, and prod env

Compared to VMs:

- Faster startup times
- Lower resource usage
- Easier management
- Easier scaling



What are Containers?

- Lightweight virtual environments
- Allow packaging of apps and dependencies
- Isolated file system, network, and resources

Compared to VMs:

- Same benefits as Docker
- Uses OS kernel, reducing overhead



Docker Commands

--> docker pull

--> docker images

--> docker run

--> docker ps

--> docker stop

```
~ docker ps
CONTAINER ID   IMAGE                                COMMAND
9e1b82c3bf51   redis                              "docker-entryp
494bb3f1f20a   8c2c38aa676e                      "/kube-vpnkit-
e-system_f0111bce-ef8b-44e3-81ac-cf11abf4033f_2816
72098825b24d   99f89471f470                      "/storage-provi
_kube-system_a0948d23-c69c-4dc2-8ea2-d39f41959286_100
7354c213f440   4fa642720eea                      "kube-scheduler
sktop_kube-system_57b58b3eb5589cb745c50233392349fb_737
2f4a6caalee0   kubernetesui/dashboard            "/dashboard --i
rd-76577bd7bb-8h7ps_kubernetes-dashboard_69df8be8-af69-
5f0e27294605   bfe3a36ebd25                      "/coredns -conf
stem_51c1c960-e2ee-4095-8d83-819036f3dcb2_57
7e03d51dda99   bfe3a36ebd25                      "/coredns -conf
stem_52376493-c6bf-4607-9d18-cd905ed25229_56
685c6e9f7cbf   9d368f4517bb                      "/usr/local/bin
_2c15e0fd-5444-46d6-adf1-ba2e696a8992_56
b6e7ca157ae1   48d79e554db6                      "/metrics-sidec
rics-scraper-79c5968bdc-6krhx_kubernetes-dashboard_d76d
2a175d531b39   k8s.gcr.io/pause:3.2              "/pause"
bce-ef8b-44e3-81ac-cf11abf4033f_67
5620d7d8ddb7   k8s.gcr.io/pause:3.2              "/pause"
ps_kubernetes-dashboard_69df8be8-af69-4c04-bef9-e222e41
cd553511bae9   k8s.gcr.io/pause:3.2              "/pause"
c-6krhx_kubernetes-dashboard_d76d925b-87ea-4a97-8e41-44
29c901ba240f   k8s.gcr.io/pause:3.2              "/pause"
48d23-c69c-4dc2-8ea2-d39f41959286_61
880a434fca24   k8s.gcr.io/pause:3.2              "/pause"
_51c1c960-e2ee-4095-8d83-819036f3dcb2_58
575de7a279fd   k8s.gcr.io/pause:3.2              "/pause"
_52376493-c6bf-4607-9d18-cd905ed25229_58
f51e66a5f62c   k8s.gcr.io/pause:3.2              "/pause"
fd-5444-46d6-adf1-ba2e696a8992_56
ee6240189b8c   0369cf4303ff                      "etcd --adverti
7f1e78367a800caa891919cc4b583f_71
9e2e8475b023   67b3bca112d1                      "kube-controlle
-manager-docker-desktop_kube-system_77e9d7fdbb29bf4b560
93cc84a7745a   c15e4f843f01                      "kube-apiserver
sktop_kube-system_4ac4b5ee26e7058a1ed090c12123e3a6_112
ec93a2e1124c   k8s.gcr.io/pause:3.2              "/pause"
f1e78367a800caa891919cc4b583f_57
25f0886a9f6c   k8s.gcr.io/pause:3.2              "/pause"
system_4ac4b5ee26e7058a1ed090c12123e3a6_57
ac3767266e3b   k8s.gcr.io/pause:3.2              "/pause"
```

Example: NGINX Hello World

```
docker run -p 8080:80 nginx
```

Visit 127.0.0.1:8080 in browser

```
docker ps -a
```

(Look for ID)

```
docker rm <id>
```

```
docker image rm nginx
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Building Your Own Container

- Built from Dockerfiles and existing containers
- Several different instructions you can use
- Run using “docker run” or (later) “docker compose”

Dockerfile Instructions:

FROM	ENV	RUN	WORKDIR
COPY	LABEL	EXPOSE	CMD

```
FROM python:3.10.0-slim-buster
```

```
# Keeps Python from generating .pyc files in the container  
# Turns off buffering for easier container logging  
# Force UTF8 encoding for funky character handling  
# Needed so imports function properly
```

```
ENV PYTHONDONTWRITEBYTECODE=1  
ENV PYTHONUNBUFFERED=1  
ENV PYTHONIOENCODING=utf-8  
ENV PYTHONPATH=/app
```

```
# Install MySQL and Poetry
```

```
RUN apt-get update -y  
RUN apt-get install --no-install-recommends -y build-essential libmariadb-dev-com  
RUN pip install https://github.com/python-poetry/poetry/releases/download/1.2.0b2
```

```
# Add Poetry path to PATH
```

```
ENV PATH="${PATH}:/root/.local/bin"
```

```
# Install project dependencies with Poetry
```

```
COPY pyproject.toml .  
RUN poetry config virtualenvs.create false  
RUN poetry install --no-interaction --no-ansi --only main --all-extras --no-root
```

```
# Place where the app lives in the container
```

```
WORKDIR /app  
COPY . /app
```

```
# During debugging, this entry point will be overridden.
```

```
CMD ["python", "/app/src/bot.py"]
```

Example: Dockerizing a Flask App

Download boilerplate from the BUILDS Workshop GitHub Repo

Fill in the Dockerfile based on the comments

Run “docker build . -t docker-workshop”

Run “docker run -p 8080:8000 docker-workshop”

Visit 127.0.0.1:8080 in your browser

CMD/CTRL-C when finished

Docker Compose

- YML File Format
- Used to define container configurations
- Allows for automatic dependency management
- Easy scaling with Kubernetes

Compared to Docker Run:

- One file for multiple services
- Removes having to remember command-line args
- Lower likelihood to err in deployment

```
version: "3.9"
services:
  character-bot:
    container_name: 1up-call-bot
    image: braxton/1up-call-bot:latest
    restart: unless-stopped
    environment:
      # Discord
      DISCORD_GUILD_ID:
      DISCORD_BOT_TOKEN:
      # Logging
      LOG_LEVEL: INFO
      # Database
      DB_HOST: database
      DB_DATABASE: 1up-call-bot
      DB_USER: 1up-call-bot
      DB_PASSWORD:
    depends_on:
      mariadb:
        condition: service_healthy
    links:
      - mariadb:database
  mariadb:
    container_name: mariadb
    image: mariadb
    restart: unless-stopped
    volumes:
      - ./config/db:/var/lib/mysql
    environment:
      MYSQL_DATABASE: 1up-call-bot
      MYSQL_USER: 1up-call-bot
      MYSQL_PASSWORD:
      MYSQL_ROOT_PASSWORD:
    healthcheck:
      test: "/usr/bin/mysql --user=$$MYSQL_USER --password=$$MYSQL_PASSWORD --execute='SELECT 1'"
      # test: "/usr/local/mysql/bin/mysql --user=foo --password=f"
      interval: 3s
      timeout: 1s
      retries: 5
```

