Containers & Orchestration

CSC 410/510

Richard Kelley

Last Time

- Overview of Containers
  - Container concept
  - Docker
  - Dockerfile
  - Mentioned idea of orchestration

Today

- Installing Docker on EC2
- Some web stuff
  - curl
  - reverse proxies
- Docker commands
- Assignment 2

Installing Docker on EC2

Update system and install prerequisites

```
$ sudo apt update
$ sudo apt install -y ca-certificates curl gnupg
```

Add Docker’s official GPG key

```
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
   sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Add the Docker apt repository

```
$ echo \
   "deb [arch=$(dpkg --print-architecture) \
   signed-by=/etc/apt/keyrings/docker.gpg] \
   https://download.docker.com/linux/ubuntu \
   $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

$ sudo apt update
```

Install Docker Engine + CLI

```
$ sudo apt install -y docker-ce docker-ce-cli containerd.io
```

Verify it works

```
$ sudo docker run hello-world
```

(Optional) Allow your user to run docker without sudo

```
$ sudo usermod -aG docker $USER
```

After this command, you have to log out and then log back in for the command to take effect!!

curl

HTTP

HTTP Request Methods

- **GET**
- PUT
- **POST**
- DELETE

HTTP GET

HTTP POST

Curl: Testing HTTP Interfaces

Curl for GET requests

```
curl -i http://localhost/api/health
```

Curl for POST requests

```
curl -i -X POST http://localhost/api/items \
  -H "Content-Type: application/json" \
  -d '{"name":"alpha"}'
```

Reverse Proxies

Reverse Proxy — Core Concept

- A **reverse proxy** is a server that sits **in front of one or more backend servers**
- It receives client requests on behalf of those servers
- It forwards the request to an internal server
- It returns the backend server’s response to the client
- To the client, it appears as though it *is* the origin server

Forward Proxy vs Reverse Proxy (Clarification)

- **Forward proxy**
  - Sits between client and internet
  - Clients explicitly connect to it
  - Often used for filtering, caching, anonymity
- **Reverse proxy**
  - Sits between internet and servers
  - Clients are unaware it exists
  - Used for infrastructure control and scaling

Basic Request Flow

- Client sends HTTP request to example.com
- DNS resolves to reverse proxy IP
- Reverse proxy receives request
- Reverse proxy forwards request to backend server
- Backend responds to reverse proxy
- Reverse proxy returns response to client

Why Use a Reverse Proxy?

- **1. Load Balancing**
- Distributes traffic across multiple backend servers
- Enables horizontal scaling
- Can use:
  - Round-robin
  - Least connections
  - Weighted routing
  - Health checks

Why Use a Reverse Proxy?

- **TLS Termination**
  - Reverse proxy handles HTTPS encryption/decryption
  - Backend servers can communicate over plain HTTP internally
  - Centralizes certificate management
  - Reduces cryptographic overhead on app servers

Why Use a Reverse Proxy?

- **Security Isolation**
    - Backend servers are not directly exposed to the internet
    - Internal IP addresses remain hidden
    - Can enforce:
        - IP allow/deny lists
        - Rate limiting
        - Web Application Firewall (WAF) rules

Why Use a Reverse Proxy?

- **Caching**
  - Frequently requested content can be cached at proxy
  - Reduces load on backend
  - Improves latency

Why Use a Reverse Proxy?

- **URL Routing / Path-Based Routing**
  - Route different paths to different services:
    - /api → API server
    - /app → Web frontend
    - /static → CDN or static server
  - Enables microservice architectures

- **Centralized Logging & Observability**
  - All traffic passes through one point
  - Easier to monitor:
    - Request rates
    - Errors
    - Latency
  - Supports distributed tracing headers

Common Reverse Proxy Software

- Nginx
- Apache HTTP Server
- HAProxy
- Traefik
- Envoy
- Caddy

Architectural Placement

- Cloud deployment

Internet
↓
Reverse Proxy / Load Balancer
↓
Application Servers (1..N)
↓
Database (internal only)

Conceptual Summary

- Reverse proxy = **traffic controller in front of your servers**
- Hides backend servers
- Improves scalability
- Improves security
- Centralizes TLS and routing
- Essential building block in modern cloud architectures

Dockerfile Commands

FROM

- **FROM specifies the base image for the build**
  - Every Docker image starts from an existing image layer.
- Example
  - FROM python:3.12-slim

ENV

- **ENV sets environment variables inside the container image**
  - These values apply at runtime to all processes in the container.
- example
  - ENV PYTHONDONTWRITEBYTECODE=1 \
    PYTHONUNBUFFERED=1
- PYTHONDONTWRITEBYTECODE=1
  - Prevents Python from creating .pyc bytecode files (keeps container filesystem cleaner).
- PYTHONUNBUFFERED=1
  - Forces stdout/stderr to flush immediately (logs appear in real time in docker logs).

RUN

- **RUN executes a command at build time**
  - Modifies the image filesystem and creates a new layer.
- example:
  - RUN useradd --create-home --shell /usr/sbin/nologin appuser
  -

WORKDIR

- **WORKDIR sets the working directory for subsequent instructions**
  - All following COPY, RUN, and CMD commands execute relative to this path.
- example:
  - WORKDIR /app
  - Creates (if necessary) and switches into /app inside the container filesystem.

COPY

- **COPY copies files from the build context into the image**
    - Source is on the host; destination is inside the container filesystem.

USER

- **USER sets the user for subsequent instructions and at container runtime**
  - The container will no longer run as root.
- example
  - USER appuser

EXPOSE

- **EXPOSE documents which port the container listens on**
    - Indicates the intended network port for the application.
- example
    - EXPOSE 8000
    - FastAPI (via uvicorn) listens on port 8000 inside the container.

Assignment 2

- https://github.com/RichardKelley-CUA/DA510_assignment_2