Deploying Web Application

By Ivan Wong

Document Root

- The directory that Apache serves web pages from is known as the document root, with /var/www/html being the default.
- Inside that directory, you'll see an index.html file, which is actually the default page you see when you visit an unmodified Apache server.

Virtual Host

- You’re not limited to hosting just one website on a server, though. Apache supports the concept of a virtual host, which allows you to serve multiple websites from a single server.
- Each virtual host consists of an individual configuration file, which differentiates itself based on either name or IP address.
  - For example, you could have an Apache server with a single IP address that hosts two different websites, such as acmeconsulting.com and acmesales.com.

Workflow in Virtual Host

- The basic workflow for setting up a new site (virtual host) will typically be similar to the following:
  1. The web developer creates the website and related files
  2. These files are uploaded to Ubuntu Server, typically in a subdirectory of /var/www or another directory the administrator has chosen
  3. After adding the necessary files into the Document Root directory, the administrator will make sure that the www-data user owns all of the files within (in the case of Apache)
  4. The server administrator creates a configuration file for the site and copies it into the /etc/apache2/sites-available directory
  5. The administrator enables the site and reloads Apache

Hosting Single Website

- An additional virtual host is not required if you're only hosting a single site.
- The contents of /var/www/html are served by the default virtual host if you make no changes to Apache's configuration.
- This is where the example site that ships with Apache comes from.
- If you only need to host one site, you could remove the default index.html file stored in this directory and replace it with the files required by your website.

Hosting Single Website

- The 000-default.conf file is special, in that it's basically the configuration file that controls the default Apache sample website.
- If you look at the contents of the /etc/apache2/sites-available and /etc/apache2/sites-enabled directories, you'll see the 000-default.conf configuration file stored in sites-available and symlinked in sites-enabled.
- This shows you that, by default, this site was included with Apache, and its configuration file was enabled as soon as Apache was installed.
- For all intents and purposes, the 000-default.conf configuration file is all you need if you only plan on hosting a single website on your server.

Hosting Single Website

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Hosting Multiple Websites

- If you wish to host another site on the same server by creating an additional virtual host, you can use the same framework as the original file, with some additional customizations.
- Virtual host files are stored in the /etc/apache2/sites-available directory, with a filename ending in .conf.
- Here's an example of a hypothetical website, acmeconsulting.com.
- A virtual host file such as this might be saved as /etc/apache2/sites-available/acmeconsulting.com.conf:

Hosting Multiple Websites

```
<VirtualHost 192.168.1.104:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/acmeconsulting

    ErrorLog ${APACHE_LOG_DIR}/acmeconsulting.com-error.log
    CustomLog ${APACHE_LOG_DIR}/acmeconsulting.com-access.log com
</VirtualHost>
```

Name-based Virtual Hosts

- With a server that only has a single IP address, you can still set up multiple virtual hosts. Instead of differentiating virtual hosts by IP, you can instead differentiate them by name.
- This is common on Virtual Private Server (VPS) installations of Ubuntu, where you'll typically have a single IP address assigned to you by your VPS provider.
- For name-based virtual hosts, we would use the ServerName option in our configuration.

Name-based Virtual Hosts

```
<VirtualHost *:80>
    ServerName acmeconsulting.com
    DocumentRoot /var/www/acmeconsulting
</VirtualHost>

<VirtualHost *:80>
    ServerName acmesales.com
    DocumentRoot /var/www/acmesales
</VirtualHost>
```

What is containerization?

- Containers, unlike VMs, are not actual servers.
- What is a container, then? It’s probably best to think of a container as a filesystem rather than a VM.
- The container itself contains a file structure that matches that of the distribution it’s based on.
- A container based on Ubuntu Server, for example, will have the same filesystem layout as a real Ubuntu Server installation on a VM or physical hardware.

What is containerization?

- Portability is another strength of containerization.
- With a container, you can literally pass it around to various members of your development team, and then push the container into production when everyone agrees that it’s ready.
- The container itself will run exactly the same on each workstation, regardless of which operating system the workstation uses.

Docker

- Docker is probably the technology most of my readers have heard of.
- Docker is everywhere, and it runs on pretty much any platform.
- There’s lots of documentation available for Docker and various resources you can utilize to deploy it.
- Docker utilizes a layered approach to containerization.
- Every change you make to the container creates a new layer, and these layers can form the base of other containers, thus saving disk space.

Docker Hands-on

Running Apache2 Web Server in Docker

- Install Docker
  - sudo apt update
  - sudo apt install -y docker.io
  - sudo systemctl enable docker –now
- Install Docker Buildx
  - sudo apt install docker-buildx
- Verify Docker Installation
  - docker –version
  - sudo docker run hello-world

Creating Image

- Create a Project Directory
  - mkdir ~/apache-docker
  - cd ~/apache-docker
- Create a Simple HTML Page
  - mkdir html
  - echo "<h1>Hello from Apache in Docker!</h1>" > html/index.html

Creating Image

- Create a Dockerfile

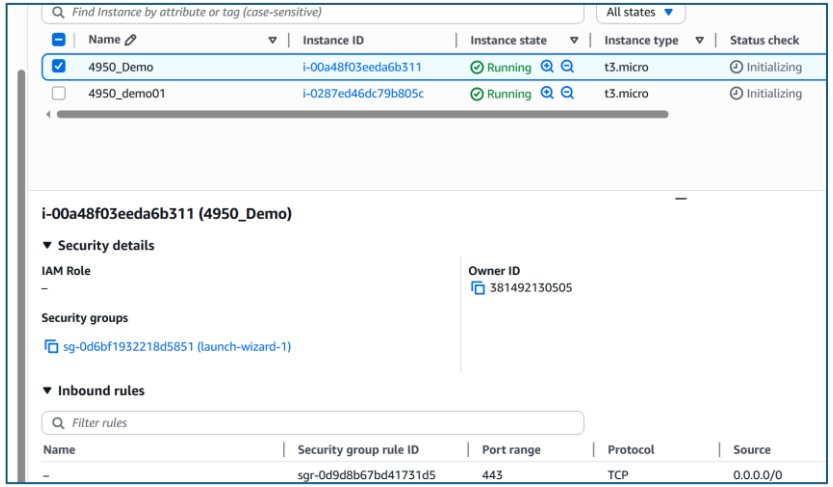
```
# Use official Apache image
FROM httpd:2.4

# Copy custom HTML into Apache web root
COPY ./html/ /usr/local/apache2/htdocs/
```

- Build the docker image
  - sudo docker build . -t my-apache2 .
- Check that the image exists
  - sudo docker images

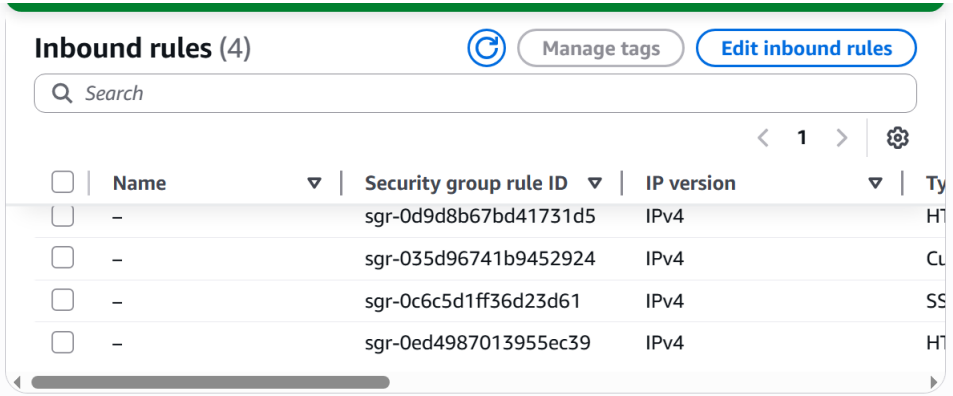
Run the Image

- Run the Apache Container
  - sudo docker run -dit --name apache-server -p 8080:80 my-apache2
  - Start a container and map port 8080 on host → port 80 in container:
- Add inbound rules to AWS instance
  - Go to AWS console and choose the instance
  - Choose the security group

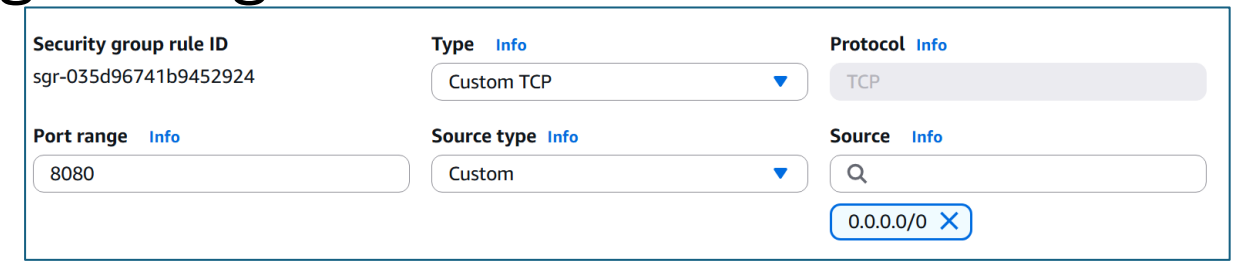


Run the Image

- Edit Inbound Rules

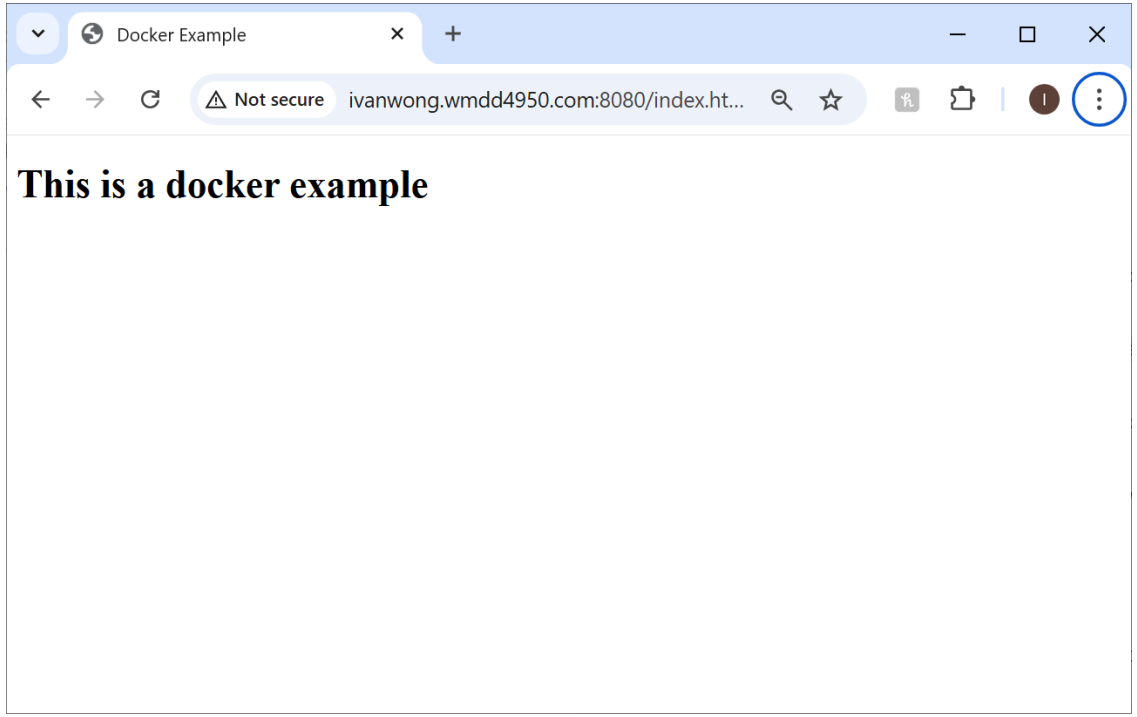


- Add a new rule with the following setting:
  - Type: Custom TCP
  - Port range: 8080
  - Source: 0.0.0.0/0
  - This means any IP can connect to it



Run the Image

- Test the Server
  - Open a browser and go to http://<your_ip>:8080/



Manage Container

- Stop the server:
  - sudo docker stop my-ubuntu-apache
- Start it again:
  - sudo docker start my-ubuntu-apache
- Remove it:
  - sudo docker rm -f my-ubuntu-apache
  - sudo docker rmi ubuntu-apache:latest

```
REPOSITORY      TAG       IMAGE ID       CREATED             SIZE
ubuntu-apache   latest    95f52a60d237   8 minutes ago       186MB
my-apache2      latest    7e48acf0dfb2   17 minutes ago      117MB
hello-world     latest    1b44b5a3e06a   6 weeks ago         10.1kB
```

Migrate the Image to another Server

- Save the Docker Image to a File
  - sudo docker save -o ubuntu-apache.tar ubuntu-apache
- Copy the Image File to Another Server
  - scp ubuntu-apache.tar user@other-server:/home/user/
  - Or use any SFTP client
- Load the Image on the New Server
  - sudo docker load -i ubuntu-apache.tar
  - sudo docker images
- Run the container as before

Using Ubuntu Base Image + Install Apache2

- Create a project directory:
  - mkdir ~/ubuntu-apache
  - cd ~/ubuntu-apache
- Create a Sample Web Page
  - mkdir html
  - echo "<h1>Hello from Apache inside Ubuntu container!</h1>" > html/index.html

Using Ubuntu Base Image + Install Apache2

- Create the Dockerfile

```
# Start from the official Ubuntu base image
FROM ubuntu:22.04

# Prevent interactive prompts during install
ENV DEBIAN_FRONTEND=noninteractive

# Update & install Apache2
RUN apt-get update && \
    apt-get install -y apache2 && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Copy our custom HTML into the web root
COPY ./html/ /var/www/html/

# Expose port 80 (Apache default)
EXPOSE 80

# Run Apache in foreground (so container stays alive)
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Run the Image

- Run the Apache Container
  - sudo docker run -dit --name my-ubuntu-apache -p 8080:80 ubuntu-apache
  - Start a container and map port 8080 on host → port 80 in container:

Summary

- Deploying Web application as Apache Virtual Host
- Reverse Proxy
- Creating Docker Image
- Migrating Docker Image

References

- LaCroix, J. (2022). *Mastering Ubuntu Server, Fourth Edition*. Packt Publishing Ltd.