CoGrammar

Welcome to this session:

Docker and Containerization

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Cloud Web Development

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Cloud Web Development

- For all **non-academic questions**, please submit a query:
  www.hyperiondev.com/support
- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***
- We would love your feedback on lectures: *Feedback on Lectures*
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Which one of the following is a benefit of a distributed system?

A. Faster communication between services
B. Easier to configure and setup
C. Isolated environments for each service
D. None of the above

What are the drawbacks of running services on their own dedicated servers?

A. There is no way for the services to communicate with one another
B. Having a lot of servers can be expensive
C. Dedicated servers can be slow
D. None of the above.

CoGrammar

Learning Objectives

- ❖ Overview
  - ➢ Define containerization and explain its purpose in web development.
  - ➢ Describe the role of the Dockerfile in building a React application container.
  - ➢ Write a Dockerfile to containerize a simple React application.

CoGrammar

Introduction to Containerization

- Different applications require a unique set of dependencies in order to run.
- When deploying different services, we want to avoid conflict in dependencies by running each service in it’s on isolated environment
- **Distributed services** involve having different servers for each service that we want to run, but the issue with that is:
  - **Cost -** Creating different VMs can be more expensive than having a single VM with more resources
  - **Maintenance -** To manage servers, you would need to SSH into each server to make updates
  - **Managing Private Services -** If a specific service is only being used internally by the application, it would still need to be exposed to the internet so it can be used in the correct places.

CoGrammar

Introduction to Containerization

- **Containerization** allows us to set-up lightweight Linux environments for our services to run on.
- Containerization tools offer virtualization on the OS layer:
  - VMs won't reserve a set amount of resources, they will share resources with the base OS
  - Each container environment is isolated from other containers and the host OS and can have it's own tools and services.
  - Services can be accessed through the host OS through port forwarding
  - Services can communicate with each other through virtual networks

CoGrammar

Introduction to Containerization

- **Docker** is the most popular containerization tool
- With **Docker** we're able to create lightweight Linux VMs to run our applications or any other services.
- Benefits of Docker:
  - **Reproducible environments -** As long as the Docker engine is installed on a system, the service can be run the same everywhere.
  - **Less servers** - All services can be deployed to a single server while maintaining the isolation of dependencies.

CoGrammar

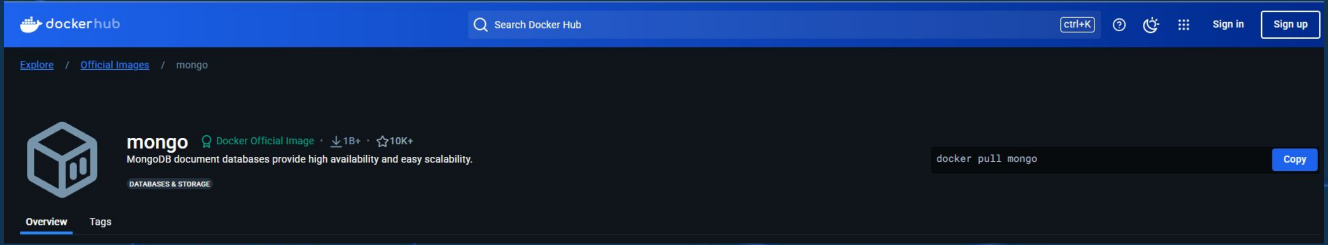
Introduction to Containerization

- **Core Concepts**
  - **Image -** A Docker image is a predefined environment containing all of the dependencies required to run a service
  - **Container -** A container is a running instance of an image. The container allows the service to perform it’s required operations.
  - **Dockerfile -** A file that is used to define the dependencies for a specific Docker image. The file is similar to a bash script, when it’s executed, it will install the dependencies and create a docker image.
  - **Docker Compose -** A service that allows you to run multiple docker containers at once. It’s mainly used for distributed services that run on a single server

CoGrammar

Running an image

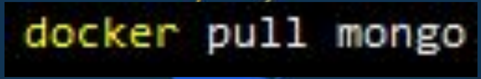
- ➢ Many services and applications have been containerized with Docker.
- ➢ To access pre-existing applications, you can run the docker pull <image-name> command to install the image on your local instance of docker.
  - ○ By default, the docker engine looks for images from Dockerhub when performing the pull command
- ➢ To bring the service up, you can use the docker run <image-name> command
  - ○ The command can also be run without the pull, if the image does not exist locally, the docker engine will automatically pull the image



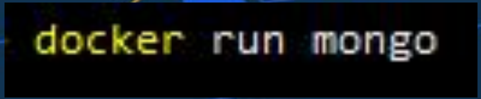
mongo

MongoDB document databases provide high availability and easy scalability.

```
docker pull mongo
```



```
docker pull mongo
```



```
docker run mongo
```

CoGrammar

Understanding a container

CoGrammar

Containers

- ➢ Purpose
  - ○ Running services locally without having to install them on the main OS
  - ○ Allows us to isolate services that would otherwise have conflicts if they were run on the same host system
- ➢ How do they work
  - ○ A container starts with a lightweight VM
  - ○ The VM will have the service that needs to be run
  - ○ The service only exists within that VM and to access it, we need to change the networking rules.
  - ○ When we perform the `docker run` command, the VM and it’s services are spun up

CoGrammar

Building a Docker Image

CoGrammar

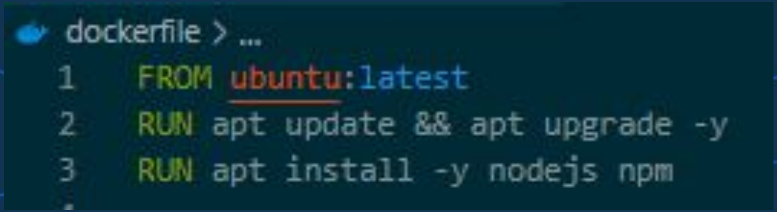
Core of a Docker Image

- As shown, we can run a lightweight VM using Docker.
- Once the VM is running, we can install tools like we would on a normal linux machine.
- This approach is not easily replicated as it requires the same manual process as deploying to different VMs
- Solution:
  - We can create our own Docker image that includes all of the dependencies to run a specific service.
  - The container can be easily setup by using the docker run command against the custom image and run within seconds without having to manually install dependencies or wait for dependencies to be installed.

CoGrammar

Core of a Docker image

- The **Dockerfile** specifies the services that we're using and some configurations that need to be performed
- The dockerfile can be treated like a bash script that is used to configure a server
- ***From*** - Specifies the base image
  - Every Docker image requires a base image
  - Anything we do in the Dockerfile will affect the instance of the base image
- ***Run*** - Specifies the terminal command that needs to be run
  - Runs UNIX commands in our base containers environment
  - Typically used for installing dependencies and configuring the environment
  - Any commands that are executed with the RUN command will only be executed in the process of creating the image



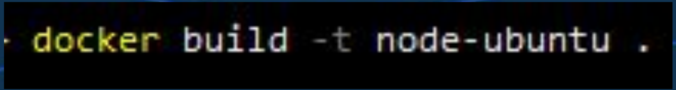
dockerfile > ...

```
FROM ubuntu:latest
RUN apt update && apt upgrade -y
RUN apt install -y nodejs npm
```

CoGrammar

Core of a Docker image

- To create a Docker image from a Dockerfile, we'll need to run the **docker build** command
- ***build*** -
  - Tells the docker engine the build command is being used
  - Allows us to create an image from a Dockerfile
- ***-t*** -
  - Used to give the image a name
  - This is important if you want to reference the image locally
- ***node-ubuntu***
  - This is the name provided for the image
  - Goes with the -t flag
  - Can be any name, if we build the image again using the same name, the old one will be overwritten
- **.**
  - Specifies the directory where the Dockerfile is located
  - Used when the Dockerfile is in the same directory as where the command is being run



```
docker build -t node-ubuntu .
```

CoGrammar

Core of a Docker image

- ➢ The Dockerfile shown will create a docker environment with Node and NPM installed
- ➢ This environment will allow us to run server side JavaScript applications.
- ➢ We can think of this more like a preconfigured Linux environment for building or hosting server side JavaScript applications.

CoGrammar

Command and Tips

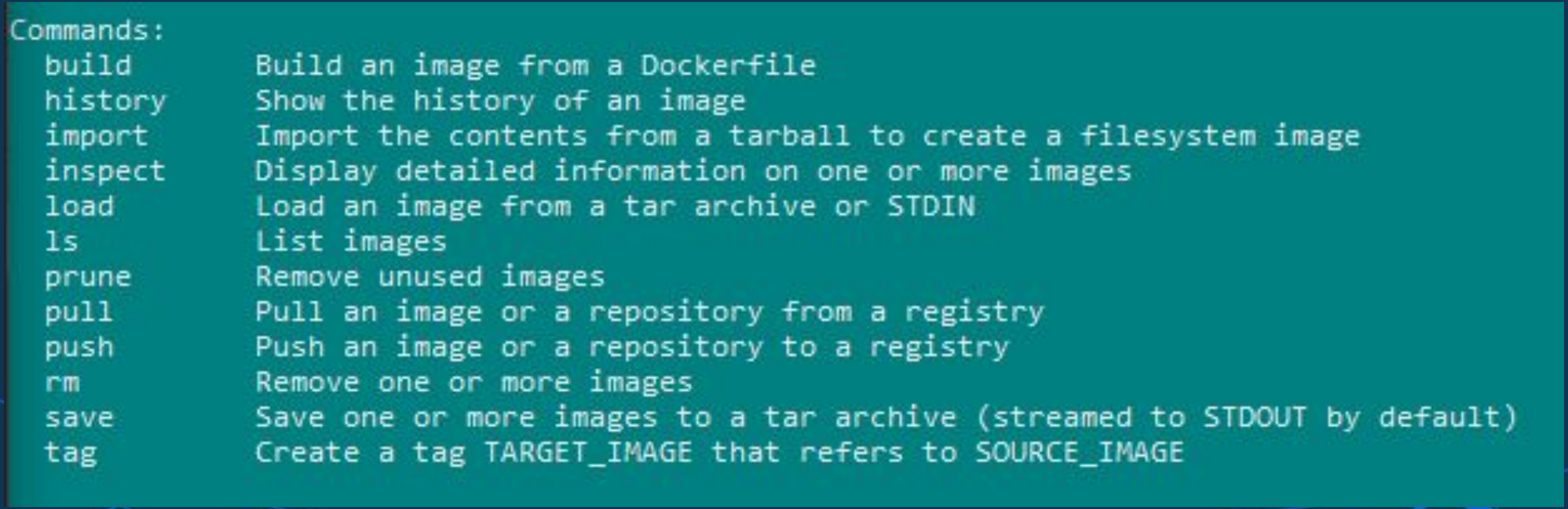
Shared commands

- ls - (lower case L, s)
  - Lists the created/available resources
  - Docker <resource-type> ls
- rm
  - Removes a specific resource
  - docker <resource-type> rm <resource-name/id>
- inspect
  - Shows details about the resource
  - docker <resource-type> inspect <resource-name/id>
- prune
  - Removes all unused resources
  - docker <resource-type> prune

CoGrammar

Image Commands

➢ Run **docker image**

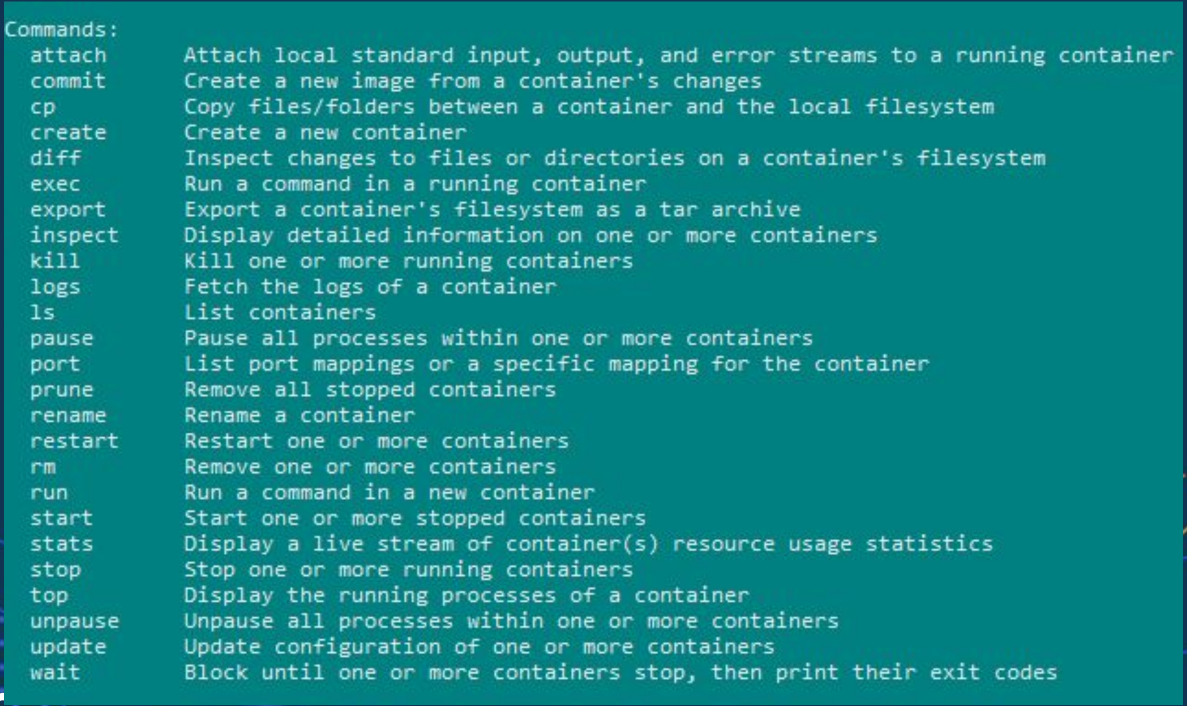


```
Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
```

CoGrammar

Container Commands

➢ Run **docker container**



```
Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  run         Run a command in a new container
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes
```

CoGrammar

Important Resources

CoGrammar

Helpful Resources

- ➢ https://docker-curriculum.com/#introduction
- ➢ https://docs.docker.com/get-started/
- ➢ https://cognitiveclass.ai/courses/docker-essentials
- ➢ https://labs.play-with-docker.com/
- ➢ https://docs.docker.com/desktop/setup/install/mac-install/#install-from-the-command-line
- ➢ https://www.docker.com/blog/how-to-dockerize-react-app/

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education