Docker

Testing docker is running

```
# Get Docker version
$ docker version
```

Launching Containers

```
$ docker run [flags] < image > [args]
$ docker run docker/whalesay cowsay "Hello world! "
# See running containers
$ docker ps
# See all launched containers, both running and stopped
$ docker ps -a
```

Launching nginx in Docker

```
# Launch nginx
$ docker run -d -p 8080:80 nginx
```

Stopping and removing containers

```
# Stop the nginx container
$ docker stop 8f19e6efea49

# Remove the container (must be stopped first)
$ docker rm 8f19e6efea49

# Remove a running container (stop and remove)
$ docker rm -f 8f19e6efea49
```

Logging into our containers

```
 $ docker run --rm -it debian

 root@dba9683049bd:/# echo "You're now in Debian!"
```

Run commands with exec

```
# Run a command in an already running container
$ docker exec -it < container_id > < command >

# Launch a bash shell in our nginx container
$ docker exec -it 8f19e6efea49 bash
```

Running DB in container

```
$ docker run -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=pass mysql

# Connect to your containerised DB from your computer
$ mysql -h 127.0.0.1 -P3306 -uroot -p
```

Getting container logs

```
# Get latest logs from container
$ docker logs <container_id>

# Follow logs created by container
$ docker logs -f <container_id>
```

Dockerfile Example

```
FROM < a base image >

COPY < your application, probably >

RUN < commands required for the application to be able to run >

EXPOSE < a port for other containers to use, if needed... >

ENTRYPOINT < what you want to execute when starting the container >
```

Managing Docker Images

```
# List existing images
$ docker images

# Build a new image
## `cd` into the directory where the Dockerfile is located
$ docker build -t <name>:<tag> .
```

```
# Remove an existing image
$ docker rmi <image_id>

# Download an image
## Might require logging onto Docker Hub
$ docker pull <image_address>

# Log into Docker Hub
$ docker login

# Push an image (requires login)
docker push <image>:<tag>
```

Running containers (advanced)

```
# Run container in the background, map port 80 in the container to port
8080 in the host
$ docker run -d -p 8080:80 nginx

# Mount directory into container at launch
$ docker run -v /path/to/dir/in/your/machine:/path/in/container <image>
```