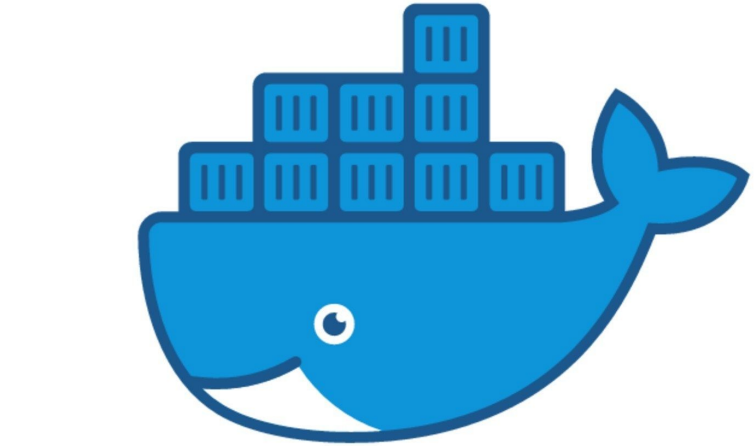
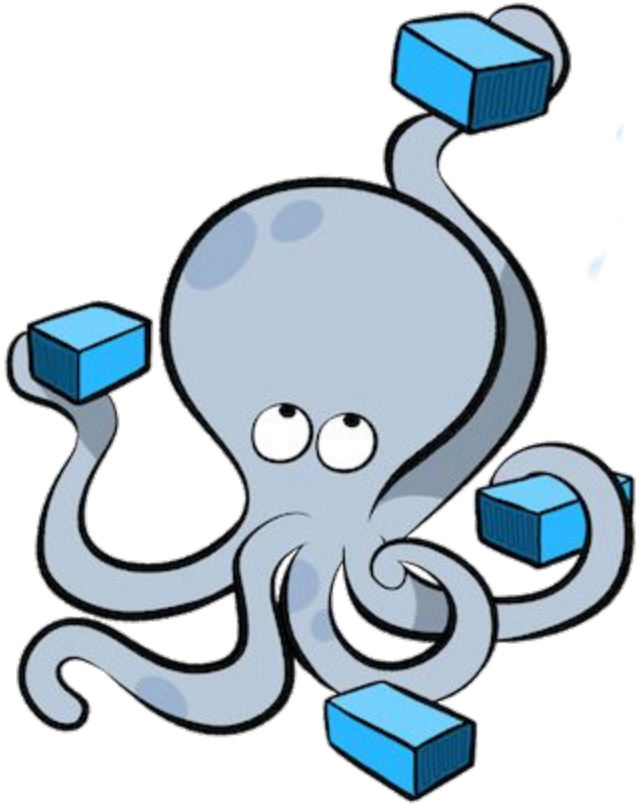


docker®



Why Docker Compose?

Capture your Docker configuration in a config file

Handles things like container linking, volumes, secrets

For production can also handle networking, distribution

It's the standard for configuring Docker containers

How does it work

Configuration lives in `docker-compose.yml`

You can set environment variables for your compose file with a `.env` file

That's pretty much it

A sample `docker-compose.yml`

```
version: '3.7'
services:
  python:
    image: python:3.7.2
    ports:
      - 8000:80
    volumes:
      - ./local/volume:/etc/container-volume
```

A sample `docker-compose.yml`

```
version: '3.7'
services:
  python:
    image: python:3.7.2
    ports:
      - 8000:80
    volumes:
      - ./local/volume:/etc/container-volume
```

There are various versions of Docker Compose.
(This is **not** like Ansible versions)

A sample `docker-compose.yml`

```
version: '3.7'
services:
  python:
    image: python:3.7.2
    ports:
      - 8000:80
    volumes:
      - ./local/volume:/etc/container-volume
```

Docker Compose refers to the containers it manages as **services**

When running `docker-compose` always use service names not container ids

A sample `docker-compose.yml`

```
version: '3.7'
services:
  python:
    image: python:3.7.2
    ports:
      - 8000:80
    volumes:
      - ./local/volume:/etc/container-volume
```

In this case we're using a pre-existing image from Docker Hub

You can also build an image from a local Dockerfile using the `build` directive

A sample `docker-compose.yml`

```
version: '3.7'
services:
  python:
    build .
    ports:
      - 8000:80
    volumes:
      - ./local/volume:/etc/container-volume
```

Like this!

A sample `docker-compose.yml`

```
version: '3.7'
services:
  python:
    image: python:3.7.2
    ports:
      - 8000:80
    volumes:
      - ./local/volume:/etc/container-volume
```

You can specify ports to be exposed on your local host. The one on the left is the port on the container; the one on the right is the local port.

A sample `docker-compose.yml`

```
version: '3.7'
services:
  python:
    image: python:3.7.2
    ports:
      - 8000:80
    volumes:
      - ./local/volume:/etc/container-volume
```

Likewise for volumes. You'll notice Docker Compose lets you specify this relatively on your local machine, which the Docker `--volume` flag does not.

Environment variables

```
version: '3.7'
services:
  python:
    image: python:3.7.2
    environment:
      my_api_key: '12g4f5ee5d6h'
```

You can also define environment variables on the service container.

However, this isn't particularly useful if you don't want to commit them...

Secrets

If necessary, you can add secrets from your local machine.
(e.g. for things you wouldn't want to commit)

These can be in files, or created using the `docker secret` command.

```
secrets:
  my_private_key:
    file: ~/.ssh/my-private-key
  some_api_key:
    external: true
```

Secrets

Once you have defined your secrets, you need to give services access to them.

```
services:
  python:
    image: python:3.7.2
    secrets:
      - my_private_key
      - some_api_key
```

See the documentation for more info how this works for defining secrets and giving services access to secrets.

.env files

Define some configuration for your Dockerfile.

The docker4drupal config uses this to let you easily switch versions of things.

.env files

```
# Sample .env file

PYTHON_VERSION=3.7.2

CONTAINER_NAME=my_container
```

```
version: '3.7'
services:
  python:
    image: python:${PYTHON_VERSION}
    container_name: $CONTAINER_NAME
    ports:
      - 8000:80
```

A note on hostnames

Remember how I said Docker Compose uses service names for everything?

If you want to address a container from within another one, the service name is the hostname.

E.g., if I wanted to connect to MySQL from my PHP container

```
mysql -u foo -p -h my_mysql_container
```

Where `my_mysql_container` is the key it has in the `services` block.

Docker Sync

Filesystem syncing for volumes can be very slow on OSX/Windows

Docker Sync can help make this faster

Check it out at http://docker-sync.io

(OSX also has some oddities with permissions for things like PHP)

Questions?