Docker-Compose

Docker Limerick Meetup - let’s ship together

http://about.me

- My name is Dave Cremins - https://www.linkedin.com/in/dave-cremins-08780a27/
- Senior Software Engineer at WP Engine
- Developing software for ~12 years
- Worked across a variety of languages - .NET/C#, JS/Node, Python, Ruby, PHP & tincy bit of Golang
- Working with Docker for the last year :)

What is Docker-Compose?

- Tool for defining and running multi-container Docker applications
  - E.g. 2 containers, wep app & db (How do we manage these??)
  - Simple YAML file + Single Compose command = Web & DB Services Created and Started
- Compose works on a variety of environments: staging, development, testing, as well as CI workflows
  - Production envs are supported but require extra files with production-appropriate configuration
- Compose has commands for managing the whole lifecycle of your application
  - Start, stop, and rebuild services
  - View the status of running services
  - Stream the log output of running services
  - Run a one-off command on a service

Using Docker-Compose

- 3-step process:
  - App environment defined via a ***Dockerfile*** so it is reproducible anywhere
  - Define the services that make up your app in a ***docker-compose.yml*** file so they can be run together in an isolated environment
  - Lastly, run ***docker-compose up*** and Compose will start and run your entire app

Example:

```
version: '3'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

What benefits does it offer?

1. Multiple isolated environments on a single host
    a. Compose uses a project name to isolate environments from each other, defaults to basename of the project directory or optionally pass the **-p** flag specifying a project name
2. Preserve volume data when containers are created
    a. If it finds any containers from previous runs, it copies the volumes ensuring no data loss
3. Only recreate containers that have changed
    a. Compose caches the configuration used to create a container
4. Variables and moving a composition between environments
    a. Compose supports variables in the Compose file e.g. **image: "postgres:${POSTGRES_VERSION}"**

Docker-Compose use cases

- Development
  - The Compose file provides a way to document and configure all of the application's service dependencies (databases, queues, caches, web service APIs, etc).
- Automated testing environments
  - The automated test suite is what underpins CI/CD which in turn require ephemeral testing environments, the Compose file defines these environments and allows them to be created/destroyed in only a few commands
    - $ docker-compose up -d
      $ ./run_tests
      $ docker-compose down
- Single Host deployments
  - Traditionally focused on development and testing workflows, but with each release more progress is made on more production-oriented features. You can use Compose to deploy to a remote Docker Engine that may be a single instance provisioned with Docker Machine or an entire Docker Swarm cluster

Docker-Compose FAQ's

- Service startup order
  - You can control the order of service startup with the **depends_on** option. Compose always starts containers in dependency order
- What's the difference between up, run, and start?
  - Use **up** to start or restart all the services defined in a docker-compose.yml.
  - The **run** command is for running "one-off" tasks. It requires the service name you want to run and only starts containers for services that the running service depends on. Use run to run tests or perform an administrative task such as removing or adding data to a data volume container. The run command acts like **docker run -ti** in that it opens an interactive terminal to the container and returns an exit status matching the exit status of the process in the container.
  - The docker-compose start command is useful only to restart containers that were previously created, but were stopped. It never creates new containers.

Enough! Let’s Demo DC

DEMO TIME