Docker Compose

Mohanraj Shanmugam

Overview of Docker Compose

- Compose is a tool for defining and running multi-container Docker applications.
- With Compose, you use a Compose file to configure your application’s services. Then, using a single command, you create and start all the services from your configuration.

Overview of Docker Compose

- Compose uses a project name to isolate environments from each other. You can make use of this project name in several different contexts:
- on a dev host, to create multiple copies of a single environment (e.g., you want to run a stable copy for each feature branch of a project)
- on a CI server, to keep builds from interfering with each other, you can set the project name to a unique build number
- on a shared host or dev host, to prevent different projects, which may use the same service names, from interfering with each other
- Compose is great for development, testing, and staging environments, as well as CI workflows

Overview of Docker Compose

- Using Compose is basically a three-step process.

1. Define your app's environment with a `Dockerfile` so it can be reproduced anywhere.
2. Define the services that make up your app in `docker-compose.yml` so they can be run together in an isolated environment.
3. Lastly, run `docker-compose up` and Compose will start and run your entire app.

A `docker-compose.yml` looks like this:

```
version: '2'
services: web:
        build: .
        ports:
        - "5000:5000"
        volumes:
        - .:/code
        - logvolume01:/var/log
        links:
        - redis
        redis:
                image: redis
        volumes:
                logvolume01: {}
```

A `docker-compose.yml` looks like this:

- 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

Compose Features

- Multiple isolated environments on a single host
  - Compose uses a project name to isolate environments from each other. You can make use of this project name in several different contexts:
  - on a dev host, to create multiple copies of a single environment (e.g., you want to run a stable copy for each feature branch of a project)
  - on a CI server, to keep builds from interfering with each other, you can set the project name to a unique build number
  - on a shared host or dev host, to prevent different projects, which may use the same service names, from interfering with each other
  - The default project name is the basename of the project directory. You can set a custom project name by using the -p

Compose Features

- Preserve volume data when containers are created
  - Compose preserves all volumes used by your services. When`docker-compose up` runs, if it finds any containers from previous runs, it copies the volumes from the old container to the new containe
- Only recreate containers that have changed
  - Compose caches the configuration used to create a container. When you restart a service that has not changed, Compose re-uses the existing containers. Re-using containers means that you can make changes to your environment very quickly.

Compose Features

- Variables and moving a composition between environments
  - Compose supports variables in the Compose file. You can use these variables to customize your composition for different environments, or different users.

Common Use Cases

- Development environments
  - When you're developing software, the ability to run an application in an isolated environment and interact with it is crucial. The Compose command line tool can be used to create the environment and interact with it.
  - The Compose file provides a way to document and configure all of the application's service dependencies (databases, queues, caches, web service APIs, etc). Using the Compose command line tool you can create and start one or more containers for each dependency with a single command (docker-compose up).

Automated testing environments

- An important part of any Continuous Deployment or Continuous Integration process is the automated test suite.
- . Automated end-to-end testing requires an environment in which to run tests.
- Compose provides a convenient way to create and destroy isolated testing environments for your test suite.
- By defining the full environment in a Compose file you can create and destroy these environments in just a few commands:

Single host deployments

- Compose has traditionally been focused on development and testing workflows, but with each release we're making progress on more production-oriented features.
- You can use Compose to deploy to a remote Docker Engine

Install Docker Compose

- Install Docker Engine version 1.7.1 or greater
- Download the Docker compose file
  - curl -L https://github.com/docker/compose/releases/download/1.6.2/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
- Apply executable permissions to the binary:
  - $ **chmod** +**x** /usr/**local**/bin/docker-compose
- Optionally, install command completion for the `bash`and `zsh` shell.
- Test the installation.
  - $ docker-compose *--version*
  - docker-compose version: 1.6.2