Docker Compose Intro

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.
Compose provides an easy way of creating and destroying isolated testing environments for your test suite.

Using Compose is basically a three-step process.

- Define your app's environment with a Dockerfile so it can be reproduced anywhere.
- Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
- 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
  redis:
    image: redis
```

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

Features
The features of Compose that make it effective are:

- Multiple isolated environments on a single host
- Preserve volume data when containers are created
- Only recreate containers that have changed
- Variables and moving a composition between environments

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 command line option or the COMPOSE_PROJECT_NAME environment variable.

Use Cases:
1. **Development Environment:**
   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).
   Together, these features provide a convenient way for developers to get started on a project. Compose can reduce a multi-page "developer getting started guide" to a single machine readable Compose file and a few commands.
2. **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:

```
$ docker-compose up -d
$ ./run_tests
$ docker-compose down
```

3. 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. The Docker Engine may be a single instance provisioned with Docker Machine or an entire Docker Swarm cluster.

For details on using production-oriented features, see compose in production in this documentation.

Docker Compose

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

Docker compose manages services

- Start, stop and rebuild services
- View status, logs of running services
    - e.g. docker-compose logs
    - Docker-compose <service name>
    - Docker-compose stop   to stop all the services

Compose provides a convenient way to create and destroy isolated testing environments for your test suite

What is a compose file?

- A configuration file where we can define all the different steps we generally perform on command line (powershell) into a file.
- The file is a Yaml file with extension .yml

Docker-compose Commands:

build	Build or rebuild services
bundle	Generate a Docker bundle from the Compose file
config	Validate and view the compose file
create	Create services
down	Stop and remove containers, networks, images, and volumes
events	Receive real time events from containers
exec	Execute a command in a running container
help	Get help on a command
kill	Kill containers
logs	View output from containers
pause	Pause services
port	**Print the public port for a port binding**
ps	**ist containers**
pull	Pulls service images
push	Push service images
restart	Restart services
rm	Remove stopped containers
run	Run a one-off command
scale	**Set number of containers for a service**
start	**Start services**
stop	**Stop services**

unpause	Unpause services
up	**Create and start containers**
version	Show the Docker-Compose version information