

**Date:** 06/11/2019  
**Name:** David Goerig (djg53)  
**School:** School of Computing  
**Programme of Study:** Advanced Computer Science (Computational Intelligence)  
**Module name:** C0894 Development Frameworks  
**E-mail:** [djg53@kent.ac.uk](mailto:djg53@kent.ac.uk)  
**Academic Adviser:** DR P Kenny ([P.G.Kenny@kent.ac.uk](mailto:P.G.Kenny@kent.ac.uk))

## **Coursework 2:**

### **Tools Supporting the Software Development Process**

<b>1. Tool description</b>	<b>3</b>
1.1. Brief description	3
1.2. What's a container?	3
1.3. Docker main services	4
1.3.1. Docker Hub	4
1.3.2. Docker Compose	4
1.3.3. Docker Swarm and Kubernetes	4
<b>2. Performant development work: Dockerization of 3 application (Java, C++, Python)</b>	<b>5</b>
2.1. Task description	5
2.2. Dockerization of a Java project using the official image	6
2.3. Dockerization of a Python project with the creation of the Dockerfile	7
<b>3. Benefits</b>	<b>9</b>
3.1. The difference with the virtualization	9
3.2. Creating container is easy with YAML	9
3.3. Many docker official images	9
3.4. Development process easier	9
3.5. Turnkey application	10
3.6. Same behaviour on every systems	10
3.7. The monitoring	10
3.8. Scalable	10
<b>4. Limits and drawbacks</b>	<b>11</b>
4.1. Link Windows / Linux	11
4.2. Security problems	11
4.3. Only supports its own container format	11
<b>5. Sources</b>	<b>12</b>

## 1. Tool description

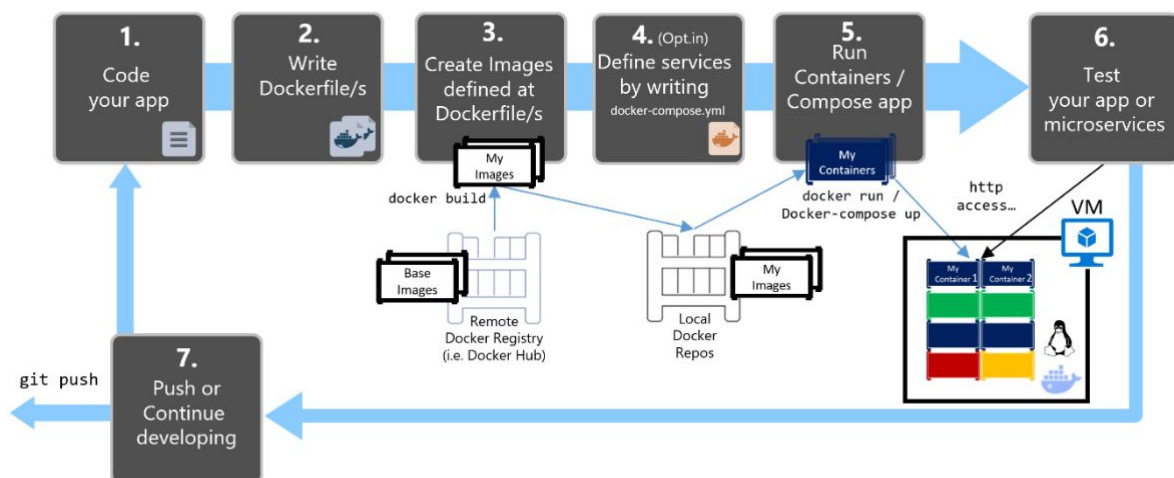


### 1.1. Brief description

Docker is an open source software that aims to make it easy to launch applications in containers. The tool can fully package an application, and its dependencies into an isolated container. This container can be then used on any server. The software is written in Go. From a more technical view, Docker extends the standard Linux container format with a high-level API that provides a convenient virtualization solution that runs processes in isolation.

So, docker is not a part of the development process, but it creates a new one. Instead of creating scripts, we need to create Dockerfiles. Instead of running the app, we run our container. And Docker defines the deployment too, offering turnkey applications. We can see, for example, that docker intervenes at all stages of the development workflow of a C # application:

### Inner-Loop development workflow for Docker apps



### 1.2. What's a container?

The purpose of a container is to host services on the same physical server while isolating them. It can be different types of services: web server, database, applications,... And can be run independently in their dedicated container, containing only the necessary dependencies. Each container is connected by virtual networks.

### 1.3. Docker main services

#### 1.3.1. Docker Hub

Docker Hub aims to facilitate the exchange of containerized applications. Hosting thousands images of containers, this space is also integrated with GitHub. It covers many fields (analytics, databases, frameworks, monitoring, DevOps tools, security, storage, operating systems, etc.). Qualified as official, some images are directly maintained by Docker. Others are offered by contributors.

#### 1.3.2. Docker Compose

Docker Compose is a tool developed by Docker to create containerized software architectures. In this logic, each brick of the application (code, database, web server ...) will be hosted by a container. This tool is based on the YAML language to describe the architecture. Once it is coded in a YAML file, all application services will be generated via a single command.

#### 1.3.3. Docker Swarm and Kubernetes

Docker Enterprise includes the main tools needed to manage the deployment, management, security, and monitoring of such environments. On cluster<sup>1</sup> management, Kubernetes is already a reference.

---

<sup>1</sup> “A computer cluster is a set of loosely or tightly connected computers that work together so that, they can be viewed as a single system.”  
[https://en.wikipedia.org/wiki/Computer\\_cluster](https://en.wikipedia.org/wiki/Computer_cluster)

## 2. Performant development work: Dockerization of 3 application (Java, C++, Python)

### 2.1. Task description

First, we aim to dockerize a Java project (with Gradle). For that, we will use official images from Docker Hub.

But to go deeper in docker, we will then create our proper container on a Ubuntu system, using the YAML language in order to dockerize a python application (using FLASK). We will then run our docker on another environment (VM Debian) to analyze the behaviour.



The first idea was to create a docker on Windows, and then run it on Linux. But Windows containers cannot run on Linux (and vice-versa).

## 2.2. Dockerization of a Java project using the official image

Our application (<https://github.com/bobbylight/SpellChecker>) use Gradle so we need to find an official image on the Hub:

[https://hub.docker.com/\\_/gradle](https://hub.docker.com/_/gradle).

Using the documentation, and after installing docker, we can see that we need to use this command:

```
docker run --rm -u gradle -v "$PWD":/home/gradle/project -w /home/gradle/project gradle gradle <gradle-task>
```

This command will first try to find the image locally, and after not finding it, clones it from docker servers. We can then easily compute gradle command (we need to build and then run). Our application will be available on the localhost.

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
cd /home/david/Programation/Kent/AssessDevelopmentFramework2/github-commit-msg-lint; ./gradlew --configure-on-demand -x check run
Configuration on demand is an incubating feature.
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :configureRun

> Task :run
[main] INFO ratpack.server.RatpackServer - Starting server...
[main] INFO ratpack.server.RatpackServer - Building registry...
[main] INFO ratpack.server.RatpackServer - Ratpack started (development) for http://localhost:5050
```

### 2.3. Dockerization of a Python project with the creation of the Dockerfile

We aim to dockerize this application, that uses the Flask library:  
<https://github.com/anfederico/Flaskex>.

To create an image the first step is to create the Dockerfile in YAML:

```
FROM python:alpine3.7
RUN pip install -r requirements.txt
COPY ./index.py /app/
WORKDIR /app/
EXPOSE 5000
ENTRYPOINT ["python"]
CMD ["index.py"]
```

The commands in this dockerfile are read for building a Docker. All the supported commands are in the Dockerfile reference documentation<sup>2</sup>.

Here is the description of the used commands:

- FROM: make our dockerfile inherit from another one. Here, we choose python but can choose another compiler like java, etc.
- RUN: compute the given command. Here we use pip to install all the python libraries needed. For a Java project we can do the same running the command “gradle --refresh-dependencies”
- COPY: copy the source code to app folder
- WORKDIR: change the working directory
- ENTRYPOINT: assign “python” as entrypoint for the CMD. (for the java project we can here assign javac)
- CMD: set the script name

Then, we need to build the image (on the VM):

`docker build --tag image_name .`

---

<sup>2</sup> <https://docs.docker.com/engine/reference/builder/>

Finally, we can run the image with the run command. The parameters in the command can be found in the documentation. **-name** is used to name the container, and the **-p** parameter links the host's port to the container's port.


```
→ flaskapp sudo docker run --name flask-app -p 5000:5000 flask-app
* Serving Flask app "index" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 928-635-871
```

### 3. Benefits

#### 3.1. The difference with the virtualization

Unlike traditional virtual machines, a Docker container does not include an operating system, but rather relies on the operating system features provided by the host machine, and this makes containers lighter than virtual machines.

This results in a much faster launch. To achieve the same result, virtualization environments need an inactive VM pool provisioned beforehand.

	<p>IBM wrote an interesting article on the comparison of virtual machines and Linux containers:</p> <p><a href="https://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/\$File/rc25482.pdf">https://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/\$File/rc25482.pdf</a></p> <p>The conclusion is that Docker equals or exceeds the performance of the hypervisor - and this in all cases tested.</p>
---	---

#### 3.2. Creating container is easy with YAML

As we saw in the dockerization of a Python application, creating our own container is easy with the YAML. Once understood the YAML language, and the available commands, a container can be deployed easily.

#### 3.3. Many docker official images

That's one of the biggest strengths of the application, all the official images in Docker Hub, updated and maintained, permit us to adapt to most of the usual languages and frameworks (as Java Gradle, C++ QT, etc.).

#### 3.4. Development process easier

It is no more needed to install 3rd party application, we can run applications directly in containers. We can also run different versions of the same application at the same time. What allows us to update services from older to a newer version. It is difficult to maintain two different versions of the same application on the same environment without Docker.



### 3.5. Turnkey application

Most of the frameworks and programming languages have their own packaging managers. In order to make applications work on almost every host systems and cloud services, Docker offers an unified image format. So, it makes possible to deliver ready to run applications, including the required dependencies, and that in one piece.


### 3.6. Same behaviour on every systems

Docker behaves the same on local machine as on dev, staging, and production servers. The human factor still there, but errors caused by different versions of operating systems are near zero. Dependencies errors are avoided too.

As we saw before, we created our container on a Ubuntu OS, we then tested it on Debian, and it still worked.


### 3.7. The monitoring

Docker permits an easy and clear monitoring. For example, you can have access to unified log files from all running containers, and of course, custom them.

	<a href="https://docs.docker.com/config/containers/logging/configure/#supported-logging-drivers">https://docs.docker.com/config/containers/logging/configure/#supported-logging-drivers</a>  Documentation on how to monitor your app log files.
---	--

### 3.8. Scalable

By structure, Docker constraints you pursue its standards, for example, over environment variables, TCP and UDP communication, and so on. Also, on the off chance that you've done your application right, it will be prepared for scaling not just in Docker.

	Docker covers most of the point of the Twelve-Factor App methodology. This methodology regroups all the best practices to design software-as-service applications, and improve the portability of it.  <a href="https://12factor.net/">https://12factor.net/</a>
---	--

## 4. Limits and drawbacks

### 4.1. Link Windows / Linux

Originally limited to Linux, Docker has since been ported to Windows Server. The fact remains that containers created on Linux, cannot be portable "natively" on a Microsoft server, and vice versa. This is Docker's major limitation, and its main difference from traditional virtualization (example of well known virtual machine that can do both: Parallels Desktop, Oracle VM Virtualbox, QEMU, etc.).

### 4.2. Security problems

Docker knows security problems, due to the large range of use of it. Most of the problem was with Docker daemon running as a privileged user on the host and the root privilege in the container, that are the same as the root privilege on the host. What could give access to the container.

For example, rkt, is a security-minded container engine who try to solve these security problems.

### 4.3. Only supports its own container format

There are many other container engine as Docker (Rkt, LXC / LXD, containerd, etc.), and the Docker engine does only support its own container format.

## 5. Sources

<https://docs.microsoft.com/en-us/dotnet/architecture/microservices/docker-application-development-process/docker-app-development-workflow>

<https://stackify.com/docker-image-vs-container-everything-you-need-to-know/>

<https://stackoverflow.com/questions/23735149/what-is-the-difference-between-a-docker-image-and-a-container>

<https://www.docker.com/>

<https://docs.docker.com/config/containers/logging/configure/#supported-logging-drivers>

<https://coreos.com/rkt/>

[https://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/\\$File/rc25482.pdf](https://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf)

<https://docs.docker.com/engine/reference/builder/>

<https://stackoverflow.com/questions/42158596/can-windows-containers-be-hosted-on-linux>