Waterford Institute of Technology

Cloud Technologies

The Automation of Infrastructure Orchestration and Application Deployment using Ansible and Docker Swarm

Stephen Coady

20064122

BSc in Applied Computing

October 23, 2016

Contents

1 Introduction

Modern applications are becoming increasingly complex, meaning it can also be complex to deploy the application. This research paper will examine application deployment, and will aim to show how modern tools and technologies can be used to simplify the process of building and deploying an application to the cloud.

It will compare these tools with "legacy" processes, evaluating the strengths and weaknesses of both. It will do this under the premise of a problem domain, discussed in Section 2.

Not only is application deployment a problem, but provisioning the servers which the application is hosted on is also something which needs to be considered. This paper will look at the automation of this process.

2 Problem Domain

A fitting definition of DevOps is "*DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support*" (Mueller et al., 2016).

Here the "service lifecycle" is the key term, as it means everything involved in creating a service and making it available for use in production. In older legacy systems the process would have involved separate teams performing each task along this timeline separately and without much overlap of personnel. In a simplified version of this model, the developers would write the code, the testers would test it, and the operations personnel would then deploy it along with provisioning the servers it would be deployed on.

This model however has shifted. In a survey conducted by (Logan, 2014), it is shown that, in general, DevOps orientated teams spend slightly more time on *all* tasks, whereas traditional IT roles will focus more on their primary tasks. This puts emphasis on the fact that developers now need to be more competent at multiple disciplines within IT. This, paired with the fact that the survey also shows DevOps orientated teams tend to automate more, shows that the importance of automated deployments has increased in recent times.

Also, with the rise in popularity of containers as a deployment vehicle, as can be seen for Docker specifically in Figure 1 it is more and more important to make the deployment of complicated applications more streamlined and reproducible.

To this end, this paper will aim to propose a solution to the problem of provisioning a production server and then deploying a fault-tolerant, scalable application to the server. It will try to show how the process of building infrastructure and deploying an application to that infrastructure can be automated, increasing productivity and also allowing for an easily reproducible environment.

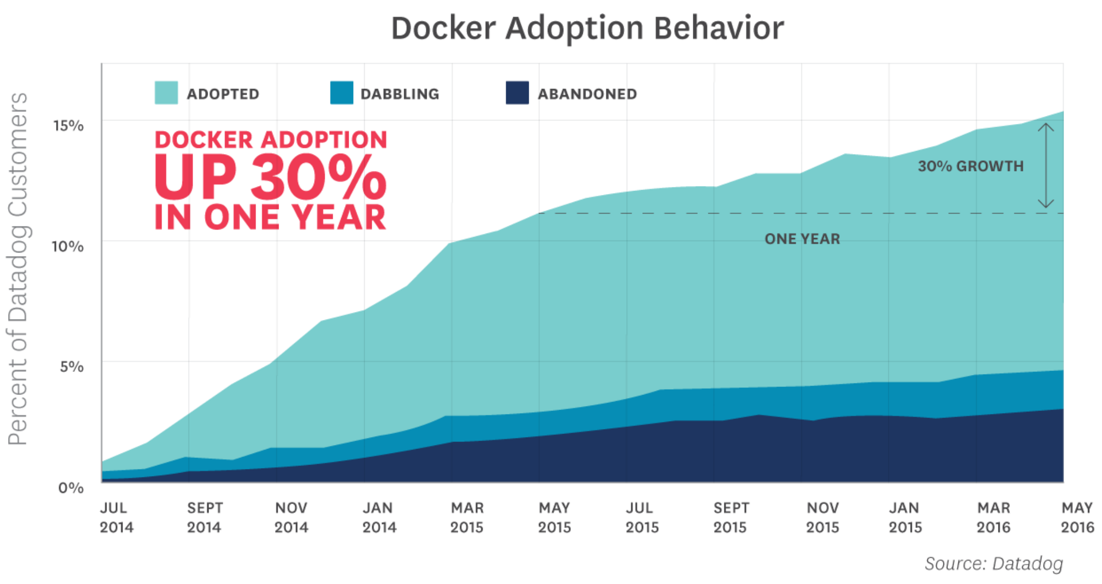

Figure 1: *Rise in Docker Usage. Credit: (DataDog, 2015)*

3 Technology Background

In this chapter we will examine a number of technologies that together will aim to solve the problem discussed in Section 2.

3.1 Containers

In software, a container is a process isolated from the host which generally runs in a very lightweight wrapper. It uses the underlying Linux kernel to do the work it needs to but ultimately it is a separate process which is self-contained (Matthias and Kane, 2015). They contain complete filesystems which can house everything the packaged application needs to run, including code, environment variables, libraries and dependencies.

Containers are often compared with virtual machines, however this view is a bit simplistic. Virtual machines are fully fledged operating systems running on a hypervisor which emulates dedicated hardware. It is made up of virtual devices which emulate the physical devices of a real host. Containers however, are not as full featured. Instead, they can be made so that they only have those resources that they need - and nothing else. So while virtual machines are emulating a real server, it could be said that a container is emulating a single *process* on a server - and only packaging exactly what that process needs to run.

Advantages
Some advantages of containers are:

- Lightweight - images can be as small as 15 MB
- Ephemeral - containers lifespan can be as small as is needed, they can be started to perform a single process and then stop as soon as they are done.
- Cheap - it is not CPU intensive to start a container
- Portable - containers can be built from a single file
- Secure - Using containers ensures applications are isolated from each other. An added benefit here is that multiple versions of the same application be running on the same host.

Disadvantages

There can also be drawbacks to containers, these include:

- Potential for added work - while containers can be useful they can also add to the work load and increase the lifecycle management of the application infrastructure.
- Orchestrating large applications can be complex
- The containers share the same kernel. Any issues with the kernel and the container engine running on it will affect all containers.

3.2 Docker

3.2.1 Swarm

3.2.2 Alternatives

3.3 Ansible

3.3.1 Alternatives

3.4 Amazon Web Services

3.4.1 Alternatives

4 Building an Application

5 Conclusion

6 Bibliography

DataDog (2015), ‘Docker adoption’, `https://www.datadoghq.com/docker-adoption/`. Accessed: 03-10-2016.

Logan, M. (2014), ‘Devops.com’, `https://devops.com/2014/01/23/fresh-stats-comparing-traditional-it-and-devops-oriented-productivity/`. Accessed: 03-10-2016.

Matthias, K. and Kane, S. (2015), *Docker: Up & Running*, O’Reilly.
URL: *https://goo.gl/YqyFMj*

Mueller, E., Wickett, J., Gaekwad, K. and Karayanev, P. (2016), ‘What is devops’, `https://theagileadmin.com/what-is-devops/`. Accessed: 03-10-2016.