GETTING STARTED WITH DC/OS

21st Century Data Center Orchestration

Tom Barber

SPICULE
IT SYSTEMS AUTOMATION

Getting Started with DC/OS

Tom Barber

This book is for sale at http://leanpub.com/gettingstartedwithdcos

This version was published on 2017-02-09

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Tom Barber by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought Getting Started with @dcos by @spiculeim !

The suggested hashtag for this book is #getstarted-dcos.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#getstarted-dcos

To the guy at O'Reilly who didn't bother to respond to my email. Thanks for the inspiration!

Also many thanks to Colin Scotland (http://logicalfox.com) for getting me going, the folks at NASA for putting up with grumpy brit and the Mrs for letting me work endless hours in pursuit of something!

Contents

About me

I'm Tom, I'm a freelance systems designer and administrator based in the UK. I have worked with data systems for the past decade, deploying, managing and maintaining them. Over that time, we've moved from Unix and Windows being the mainstream data center operating systems, to Linux being the operating system of choice in data centers and the cloud. In 2016 I setup Spicule to offer those same services to clients, customers and friends who all suffer from problems caused by legacy or, in some cases, questionable data center design and deployment. At Spicule we offer on site or remote operations support for businesses looking to fix problems, increase stability or develop a roadmap for growth.

I enjoy working with large scale platforms and data systems, over this past year we have worked some remarkable and worthwhile projects including work with NASA JPL in designing and implementing a large genomics research platform and working with DARPA to help automatically detect human trafficking and the drugs trade on the dark web. It is truly amazing what computers can be used for in the 21st Century.

Anyway enough about me I hope you enjoy the book. You can find out more about me and Spicule at http://spicule.co.uk[1]

[1] http://spicule.co.uk

About this book

This book is a living ebook that I have written using the lovely Leanpub publishing platform. Their mantra is release early, release often, I'm sure the book has kinks, issues and missing topics, but the great thing is that I can edit it and add more as people request new content, or new features are added. So if you purchase it now, you could consider yourself the lucky few who get to see this book develop through the books lifecycle and also get new content when DC/OS gets upgrades and updates or I learn cool new things.

This book isn't supposed to be a wholly in depth guide to DC/OS and its operation, it may turn into that, but for now its a high level guide. If you are new to DC/OS or are considering deploying it on your infrastructure, this book is for you.

Introduction

DC/OS Background

The world of applications is changing. Not too long ago to deploy software, you used apt-get or yum to install your packages and then edited configuration files to make them behave how you wanted. Then along came server automation with the likes of Puppet, Chef and so on; that did the same but abstracted it away in a repeatable and scalable manner. In recent years containerisation and specifically, Docker has been the new kid on the block in the software deployment arena.

Containers allow you to deploy software inside very lightweight virtual machines that you can spin up and destroy on your hosts with relative ease. You can also build upon and extend existing containers very easily which makes them ideal to tweak and adjust. The problem was, until recently there wasn't a truly flexible and scalable multi container solution on the market. A few have now come to the fore, the most mainstream being, Docker Swarm, Core OS, Kubernetes and the Data Centre Operating System (DC/OS).

These new container orchestration systems bring around a whole different deployment and scaling method that wasn't available to administrators before. Instead of systems administrators commissioning new servers and then running their Puppet scripts over the top they can now use Cloud API's to automatically provision new servers register them with their container provisioning system of choice and have the service deploy software and workloads to those servers automatically. What's more when server loads get too high or host resources get stretched, these platforms can scale and in many cases auto-scale to allow for more capacity without intervention from administrators and because containers contain a single process instead of the whole application stack, it enables applications to scale individually instead of as a whole further optimising performance. Similarly, should host hardware fail, they can self heal and reprovision containers instantly to ensure up time is maintained.

DC/OS has been developed by Mesosphere[2] and is based on Apache Mesos and builds upon the simple premise that you treat your entire data centre and resource capacity as a single entity. Going back to my earlier discussion about Puppet, for example, when I deploy services using Puppet I decide which server to place these services on, how many units are deployed etc. When I deploy

[2] https://mesosphere.com/

services using DC/OS, I don't (most of the time) care what server my service has been deployed to, all I care is that it has enough CPU, RAM and Storage capacity to do the job. This greatly simplifies the deployment technique. Also because DC/OS is utilising container technology, as long as I design my containers in a way that allows them to be destroyed or torn down without loosing important data (as they should be) It really doesn't matter where they end up because if that server goes away they can instantly return somewhere else in my data centre.

What Is Apache Mesos and who uses it?

Apache Mesos started life as a research project at the University of California, Berkeley RAD lab by PhD students Benjamin Hindman, Andy Konwinski, and Matei Zaharia. Mesos entered the Apache Software Foundation incubator in late 2010 and graduated to a top level project in 2013. During that time, it was picked up by a number of high profile technology companies and extended to a solidified to underpin DC/OS today.

Notable Apache Mesos users include:

- AirBnB
- Apple
- CERN
- eBay
- Netflix
- Twitter
- NASA
- Uber

And many others[3]

Mesos is a distributed systems kernel but instead of running at a low level like the Linux kernel, it runs on each server in your data centre and abstracts away resources so that developers can program against the data centre as one, instead of deploying services to a single server.

It is also worth pointing out, Mesos isn't a dedicated container platform. It can certainly run containers, but it runs Tasks, tasks can be written in a number of different programming languages and instead of deploying static containers to units in your data centre, tasks can scale against your cluster, but can also be much easier to update over time.

Mesos offers Linear scalability, because of its relatively light weight nature and backed by Apache Zookeeper to keep the configuration in sync, Mesos can scale simply by adding new nodes and pointing them to the Zookeeper cluster to register their existence. Because of the Zookeeper backing they can provide High Availability for services as the agents will be fault tolerant and redeploy failed services automatically.

Why shouldn't I be using Kubernetes, Openstack or another platform?

The software landscape has changed drastically over the last decade, although you could probably say that about any decade. The way we install applications used to be apt-get or yum install. Of course in some ways it still is, but in many others it has changed mostly because of scale, availability and complexity. Not too long ago, Openstack was the next big thing. Being able to run your services on a private cloud offered, and still offers, businesses the ability to deploy services on their own hardware whilst retaining the flexibility that cloud providers like Amazon offer. Of course deploying services to infrastructure using Puppet or another automation platform is still very much the norm and it would be remiss to suggest otherwise, but the 2010's bring the concept of containers to the mainstream. Containers, being very lightweight virtual machines, lend themselves very well to being deployed and then torn down in their entirety once you've finished with them as the resource isn't being used, they are also very quick to stand up and don't usually require Puppet or similar to configure them.

So 2016 brought container orchestration requirements to the fore and as such businesses were experimenting with different orchestration services. Some of these included CoreOS, Docker Swarm, DC/OS and Kubernetes.

Swarm is written by Docker and whilst incorporates nicely with Docker Compose and will no doubt improve, the current consensus is that it struggles at scale and when the requirements get overly complex.

Kubernetes was developed by Google and donated to the Cloud Native Computing Foundation in 2015. Kubernetes has a number of primitives to help you design your service topology where you will

[3] http://mesos.apache.org/documentation/latest/powered-by-mesos/

end up deploying a number of 'pods' to your servers. Kubernetes will control your pods, or groups of pods and provide additional services like software defined networking to allow for inter-container communication.

CoreOS is a Linux distribution that builds on top of Kubernetes and a number of other services to provide container support that scales, for example they provide a self updating Kubernetes distribution, a docker repository service, support for alternative container formats and more.

DC/OS takes a slightly different approach installing itself within an existing CoreOS or Centos installation instead of providing its own operating system. It also doesn't currently support Kubernetes as a deployment method and instead uses Apache Mesos as its container controller. This also allows you to run Mesos tasks outside of a container across your cluster and means you can deploy services like Apache Hadoop much more flexibly across your data centre.

Container Orchestration is a rapidly developing and expanding technology so there is no right or wrong. In testing the various platforms, my personal preference when weighing up my requirements against the others on the market made me choose DC/OS because of its Mesos support (I deal a lot with Hadoop based services and other Apache Software Foundation projects), I also like the fact I can deploy tasks outside of a container context. DC/OS also provides a very useful web based administration console and is under heavy development. Of course there are a lot of people picking up and running with Kubernetes as it provides a very good set of container orchestration services. There is no right or wrong answer. Pick the one that fits your requirements the best and work with it. If that happens to be DC/OS then read on for more….