Big Book of MLOps

Jamie Ralph

2024-09-24

Table of contents

Preface

MLOps is a big field. I work in MLOps and spend a lot of time learning things. Unfortunately my brain can only store so much information, and it's easy to forget stuff.

This book is a repository of the knowledge I've learned since starting in MLOps. The content can roughly be divided into these categories:

- Short summaries of topics with links to articles that explain things nicely
- Notes I've taken from video courses (I find these the most time consuming and attention hungry. Taking notes mean I only sit through them once)
- Useful hints from technical books I've read

Networking in Kubernetes

Notes from Certified Kubernetes Administrator (CKA) with Practice Tests

Pre-requisites on networking

How can computers talk to each other on a network?

- In a simple network, two computers (let's call them A and B) can exchange information over a network
- This communication is transmitted via a **network switch**, a piece of equipment that can connect IT devies. Network siwtches can vary in speed (e.g gigabytes per second).
- A computer sends information to the switch via an **interface**. This interface can be a piece of hardware or software depending on the situation, but is essentially a point of connection between the device and the network
- On linux, running `ip link` will print a list of interfaces in the terminal
- If we assume the network has an ip address of 192.165.1.0, we could add ip addresses to A and B with `ip addr add`, e.g. `ip addr add 192.165.1.10/24 dev eth0` on A and `ip addr add 192.165.1.11/24 dev eth0` on B. Note: Here, **dev** stands for **device** and eth0 is the first ethernet interface on the system.
- This would mean that A and B can now exchange packets with each other. Packets are small segments of a larger piece of information being sent over the network, which are recombined by the device that receives them.

Routing

- A router helps connect different networks
- The router is visible to each network with a different ip address
- Networks are configured with **gateways** which connects two different networks
- The `route` command in Linux will print the routing table
- Routes can be added using `ip route add`
- Let's assume this network setup:
  - A and B exist on network 192.165.1.0 - we'll call this network 1
  - C and D exist on network 192.165.2.0 - we'll call this network 2

  - A router is connected to network 1 via the ip 192.165.1.1, and connected to network 2 via the ip 192.165.2.1

- We can connect device A to network 2 by running `ip route add 192.165.2.0/24 via 192.165.1.1`
- Running `route` shows that the router is now a gateway to network 2
- You can set default routes instead of adding an entry for every single network - `ip route add default via 192.165.1.1` - default is sometimes seen as '0.0.0.0'
- Linux servers can act as hosts themselves but packet forwarding between interfaces needs to be enabled. This can be a security threat if one interface connects to a public network and the other to a private network.

Domain Name Systems (DNS)

- In a small simple network (let's use A and B again, connected via a switch), we can give names to each device. I can add an entry in the `/etc/hosts` file of A:

```
192.165.1.11    myname
```

- I can now run `ping myname` to check connectivity to computer B. However, A will not actually check that B's host name is myname. And this tasks would quickly become impossible as the network grew
- An internal DNS server solves the problem - it is a server containing a single source of truth
- If we assume the DNS server's ip address is 192.165.1.100, an entry in A's `/etc/resolv.conf` file tells it where to resolve domain names:

```
nameserver  192.165.1.100
```

- A DNS functions like an internet phonebook. Domain names are linked to IP addresses - it means humans don't need to memorise long IP addresses
- Domain names are strings pointing to a specific web services
- A domain name is usually comprised of several elements - e.g. www.google.com can be broken down to:
  - .com - top level domain (other examples are .edu, .org, .io) - can be a sign of the intent of the server e.g. .edu is for educational institutions, .org for non-profit
  - google - the second level domain
  - www - a subdomain (other google examples could be mail or maps)
- There are DNS servers on the internet that are searched to find the ip address of the server that is hosting the web applications
- ip addresses can be cached by browsers to speed up subsequent requests

- DNS can contain records:
  - A records - map ip addresses to hostnames
  - AAAA (quad A) records - map IPv6 to hostnames
  - CNAME (canonical name) - map one name to another e.g. if flowers.example.com had a CNAME record with a value of example.com, a lookup of "flowers.example.com" provies the ip address of "example.com", which is the canonical name

Notes from online articles

Proxies and Reverse Proxies

Proxies (also called forward proxies) are intermediary servers that intercept client requests. A proxy might be implemented in an organisation to do things like block restricted content. Meanwhile, reverse proxies sit in front of servers and accept requests coming over the internet. They can provide functions such as load balancing and SSL encryption/decryption. The article What is a reverse proxy? by CloudFlare provides a good overview of these concepts. NGINX is a popular reverse proxy - this introduction to NGINX provides a nice practical demonstration of its abilities.

Resources

Some of the other resources I used to understand networking basics:

- What is a network switch? by Juniper Networks
- What is computer networking? by AWS
- What is a network gateway? by NordLayer
- What is a DNS CNAME record? by CloudFlare