

### 3. Tech Fundamentals

⌚ Created	@May 31, 2022 11:45 AM
✖ Class	
✖ Type	
✖ Materials	
✓ Reviewed	<input type="checkbox"/>
☰ Property	
📅 Date	

 Tech Fundamentals

 YAML- mostly used for cloud formation

## YAML Introduction (Lists)

Press Esc to exit full screen

<https://learn.cantrill.io>  adriancantrill  

YAML can also represent **sequences** or **lists**, in this case “adrianscats” is a **list**, comma-separated elements are enclosed within [ & ] - this format is known as ‘**inline**’

```
adrianscats: ["roffle", "truffles", "penny", "winkie"]
```

... or the same list can be represented like below ...

```
adrianscats:
  - "roffle"
  - "truffles"
  - 'penny'
  - winkie
```

The Same Thing

Indentation matters in YAML.  
In this case, it shows ..  
“roffle”, “truffles”, “penny”  
and “winkie” are part the  
value for adrianscats

The “-” means each item is a member of  
a list, same indentation = same list. You  
can nest lists in lists using indentation

Values can be  
enclosed in "", ', or  
not - all are valid but  
enclosing can be  
more precise

JSON: used for cloud formation plus other policy docs



# JSON Introduction

<https://learn.cantrill.io>

adria

The top level is a collection of unordered key:value pairs, where the value is a JSON object

Each Nested Object is a collection of key: value pairs where the value can be a scalar, a list or a JSON object

```
{<----  
  "roffle" : {  
    "color" : "mixed",  
    "numofeyes": "2"  
  },  
  "truffles" : {  
    "color": "mixed",  
    "numofeyes": "2"  
  },  
  "penny" : {  
    "color": "grey",  
    "numofeyes": "2"  
  },  
  "winkie": {  
    "color" : "white",  
    "numofeyes": "1"  
  }  
}<----  
Four  
Nested  
JSON  
Objects
```

{ JSON OBJECT }



# JSON Introduction (CloudFormation)

<https://learn.cantrill.io>

adriancan

This JSON template has a 'Resources' key, its value is a JSON OBJECT  
Within it, a key 's3bucket', its value is a JSON OBJECT  
..containing 'Type' and 'Properties' keys. Type has a string value  
....Properties is a JSON OBJECT containing 'BucketName'  
.....BucketName is a key with the value of "ac1337catpics"

```
{  
  "Resources": {  
    "s3bucket": {  
      "Type": "AWS::S3::Bucket",  
      "Properties": {  
        "BucketName": "ac1337catpics"  
      }  
    }  
  }  
}
```



OSI layers

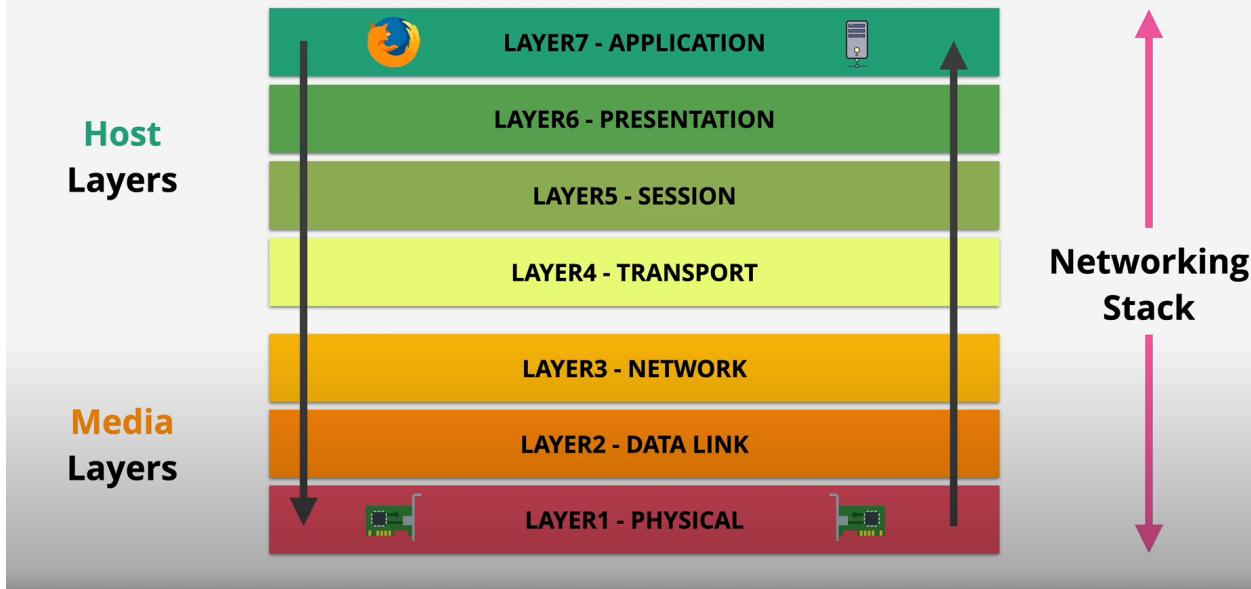
All People Seems To Need Data Processing



# OSI 7-Layer Model

<https://learn.cantrill.io>

@adriancantrill



L1: Physical: 01010 no intelligence all get the same info  
It can't detect the particular link so all data is broadcasted



L2: Data Link Layer  
L2 has Mac add- 48 Bit

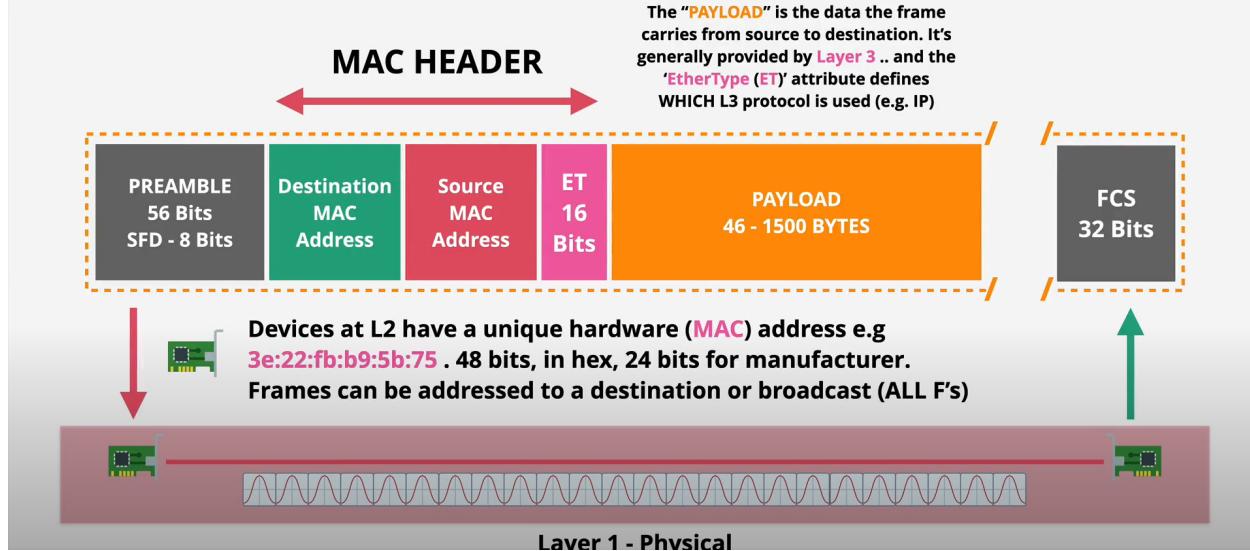
L2 provides frame and L1 handles the transmission without knowing it

ET: Ethertype is used to know which L3 protocol is original putting data into that frame

## FF:FF:FF Layer 2 - Data link - Frame

<https://learn.cantrill.io>

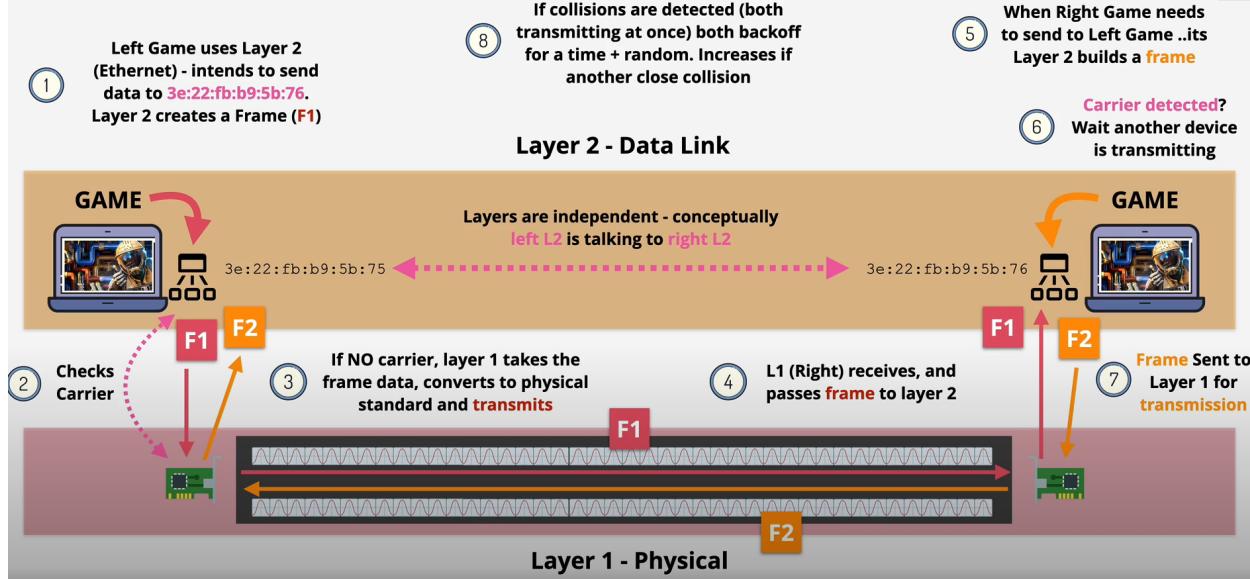
@adriancantrill



## FF:FF:FF Layer 2 - Data link - CSMA/CD

<https://learn.cantrill.io>

@adriancantrill

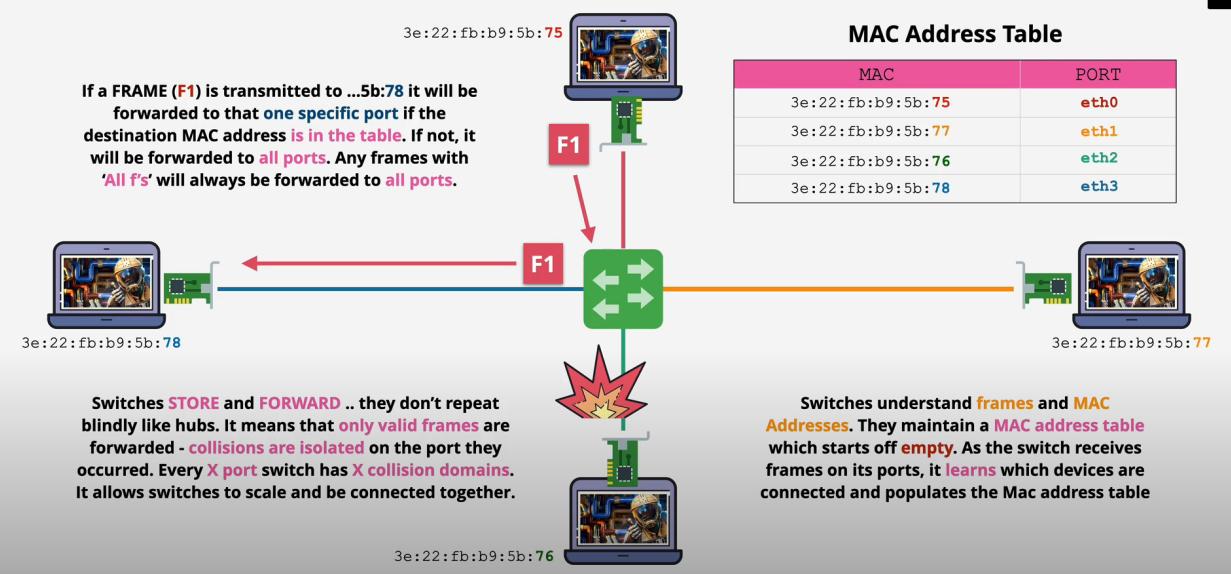


L2 with switch

## FF:FF:FF Layer 2 using a Switch

<https://learn.cantrill.io>

adriancantrill



💡 Use of L2:

## FF:FF:FF Layer 2 - Data link

<https://learn.cantrill.io>

adriancantrill

- Identifiable devices



- Media access control (sharing)



- Collision Detection



- Unicast .. 1:1



- Broadcast 1:ALL



- Switches - Like Hubs with Super powers (Layer 2)

## Decimal to Binary or visa versa

IP is 32 bit

to do this you have to understand the table of 8 bit which goes

1 2 4 8 16 32 64 128

1	2	3	4	5	6	7	8
128	64	32	16	8	4	2	1

 **Decimal => Binary**

Each "decimal number" between dots  
It has a value between 0-255  
Do each part 1 by 1 → **133.33.33.7**

"1" smaller than 2 (Rule #1)  
Add "0" and move on

"133" larger/equal than 128 (Rule #2)  
 $133 - 128 = 5$  (new decimal number)  
Add "1" in binary position

"5" smaller than 64, 32, 16, 8  
Add "0" and move on (Rule #1)

"1" larger/equal than 1 (Rule #2)  
 $1 - 1 = 0$  (finished)  
Add "1" in binary position

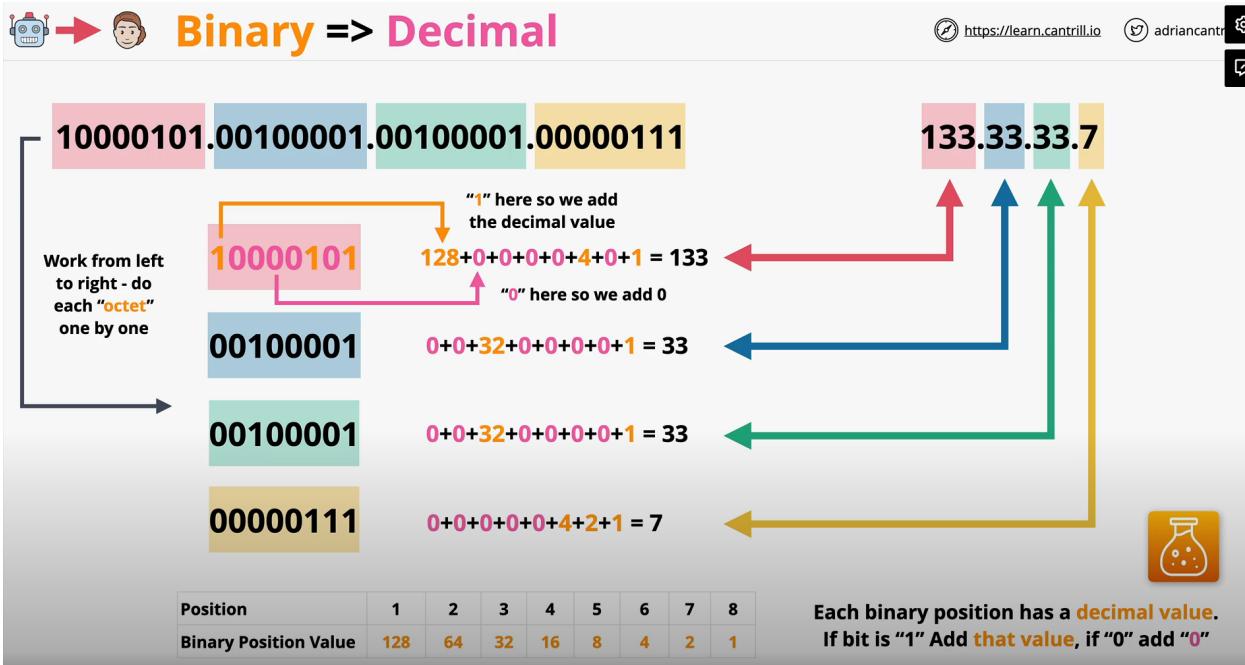
"5" larger/equal than 4 (Rule #2)  
 $5 - 4 = 1$  (new decimal number)  
Add "1" in binary position

Move through the binary table, left to right 

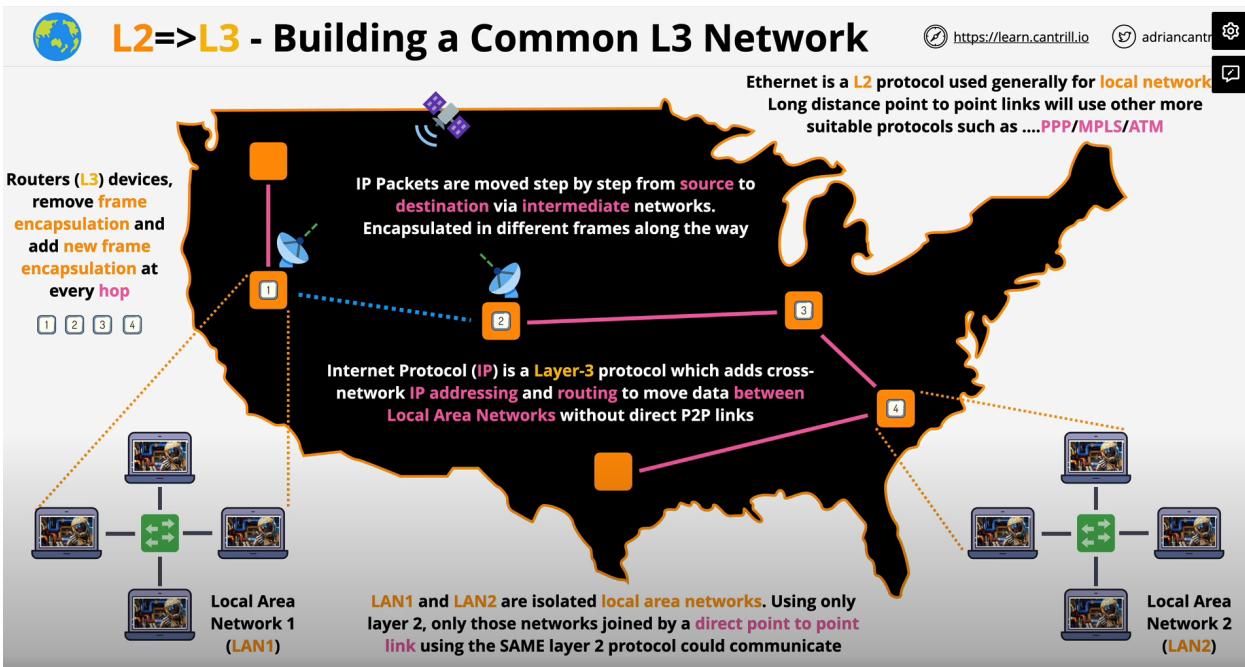
- 1 Compare decimal number to Binary position value, if SMALLER WRITE 0 => move on to next table position, goto #1
- 2 IF IT IS EQUAL OR LARGER - Minus the binary position value from your decimal number, add 1 in the binary value column
- 3 Move on to next position, goto #1 (with the new decimal value)



binary to dec same table



## Layer 3 : Network layer



## IP Addressing (v4) - IPv4

Dotted-decimal Notation  $4 \times 0\text{-}255$

All IP Addresses have a 'network' part

If the 'Network' part of two IP addresses match, it means they're on the same IP network. If not, they're on different networks.

This network has a /16 prefix. 16 bits of the IP are the network, and the remaining bits are for hosts

**133 . 33 . 3 . 7**

10000101      00100001      00000011      00000111  
8 bits            8 bits  
32 bits (4 bytes) -  $4 \times 8$  bits (Octets)

**133 . 33 . 33 . 37**

This matches Same Network      Different Host

And a 'Host' Part

IP Addresses are actually binary. 4 sets of 8 bits (octets) for a total of 32 bits

IP Addresses are assigned by machine (DHCP) or Humans

## Subnet Mask

A Subnet Mask is configured on a host device in addition to an IP address e.g. **255.255.0.0** & this is the same as a /16 prefix

A subnet mask is a dotted decimal version of a binary number which indicates which part of an IP address is NETWORK (1) and which part is HOST (0)

**133 . 33 . 3 . 7**

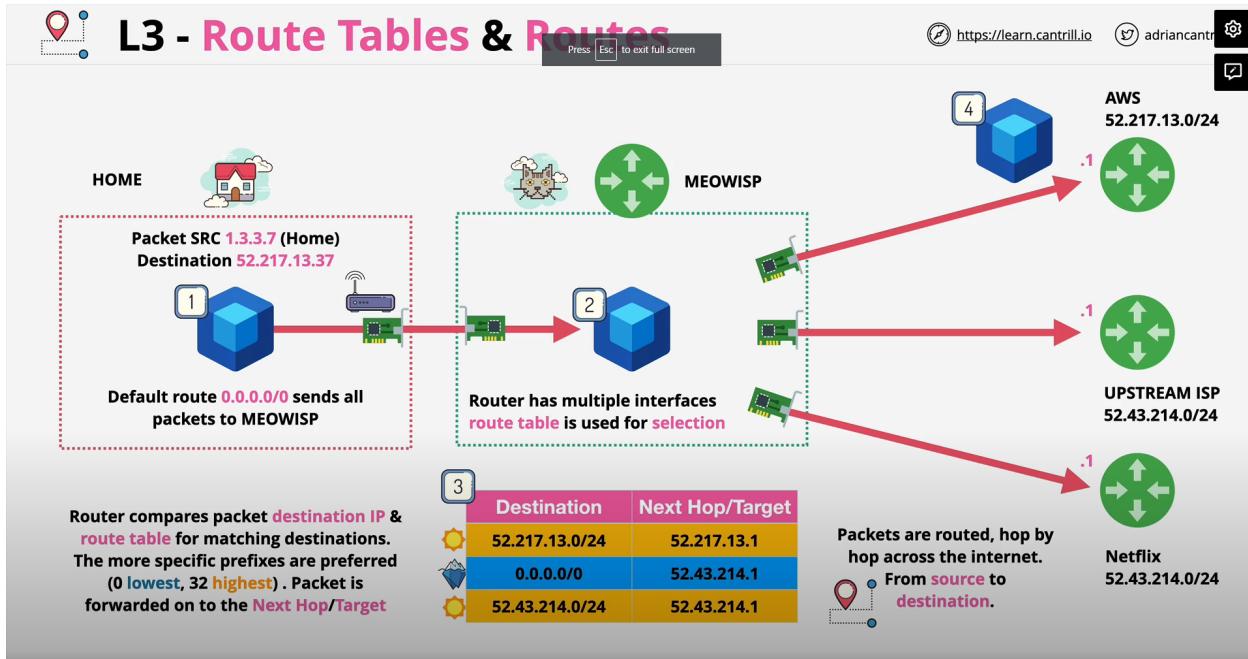
10000101      00100001      00000011      00000111  
11111111      11111111      00000000      00000000

The starting address of the local network ... is ALL-0's in the HOST part, the ending is ALL-1's in the HOST part

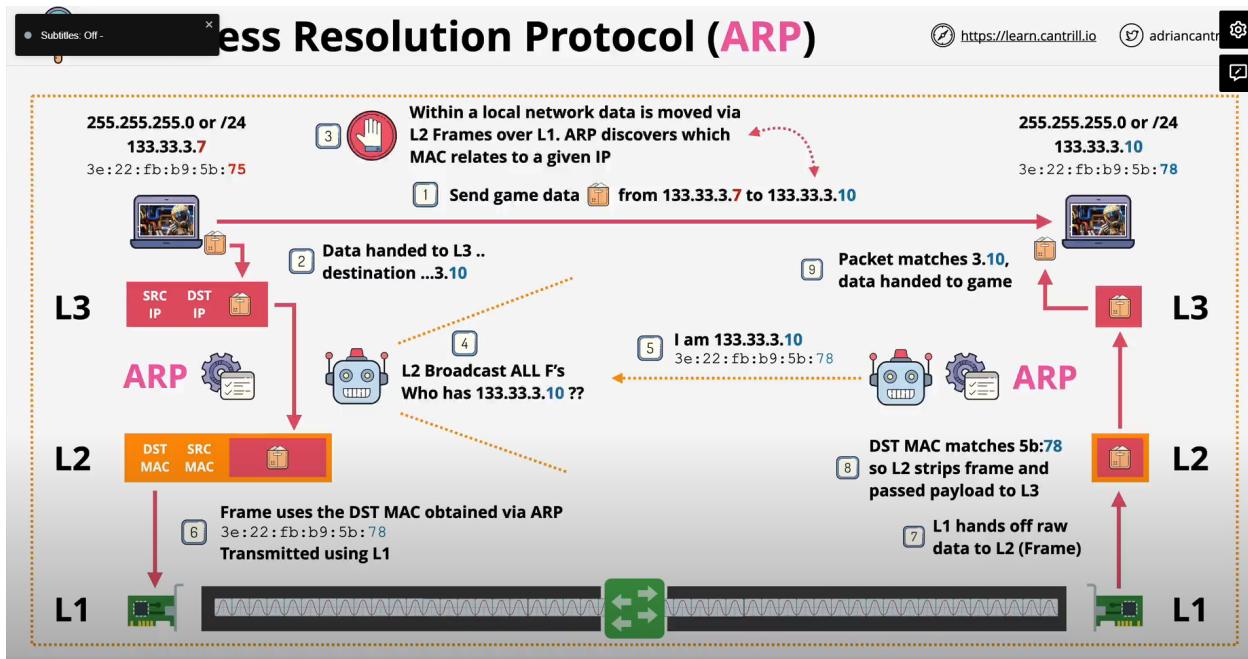
**NET START**      **133 . 33 . 0 . 0**  
 $\begin{array}{l} 10000101 \\ 10000101 \\ 00100001 \\ 00100001 \end{array}$   
**133 . 33 . 255 . 255**      **NET END**  
 $\begin{array}{l} 00000000 \\ 11111111 \\ 00000000 \\ 11111111 \end{array}$

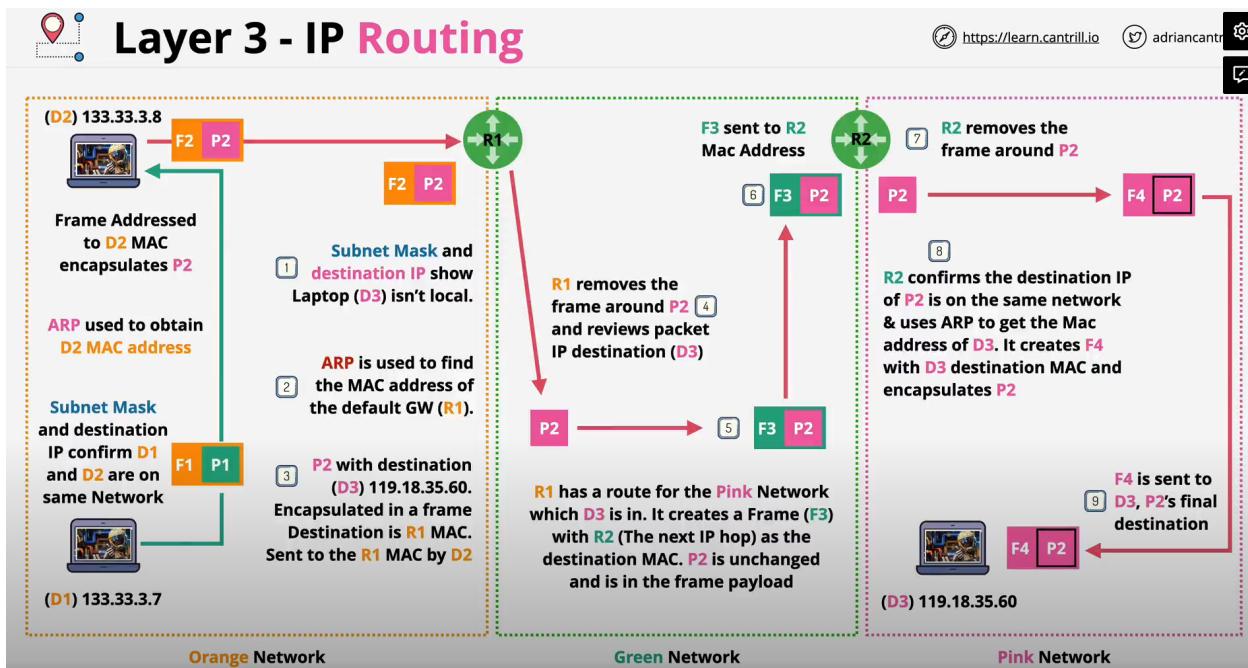
It's the Subnet Mask which allows a HOST to determine if an IP address it needs to communicate with is local or remote - which influences if it needs to use a gateway or can communicate locally

## Route Table & routes

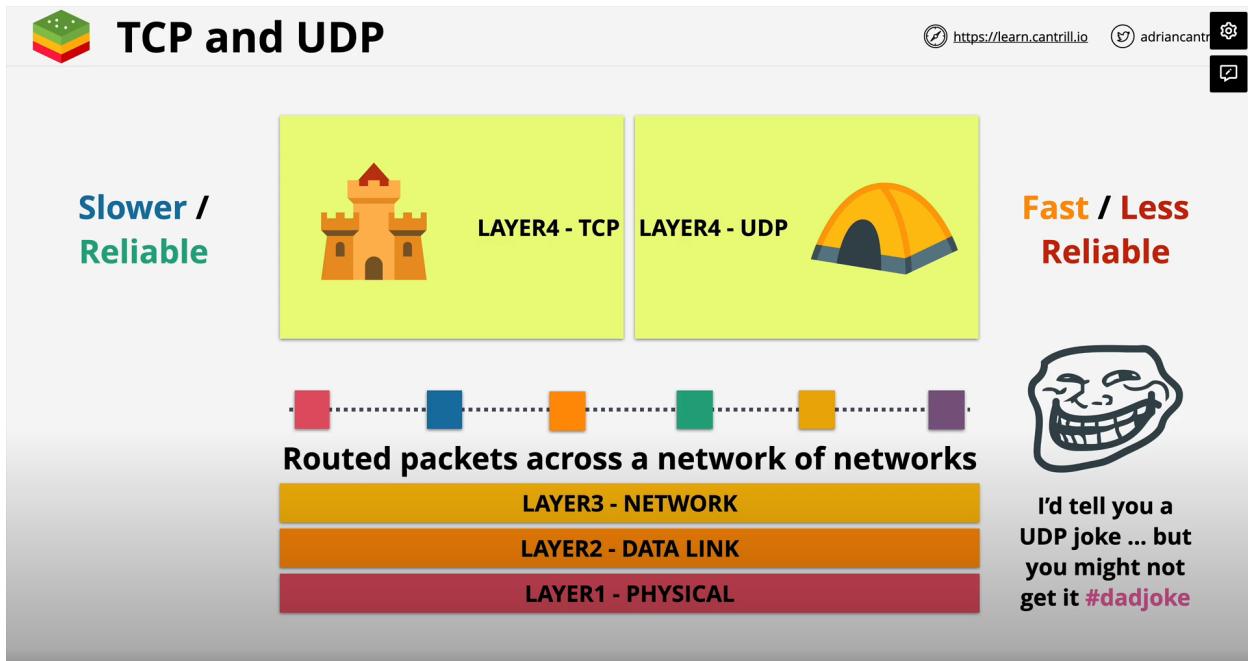


## ARP: Address Resolution Protocol





## Layer 4: Transport layer



## TCP header

segments are only for TCP

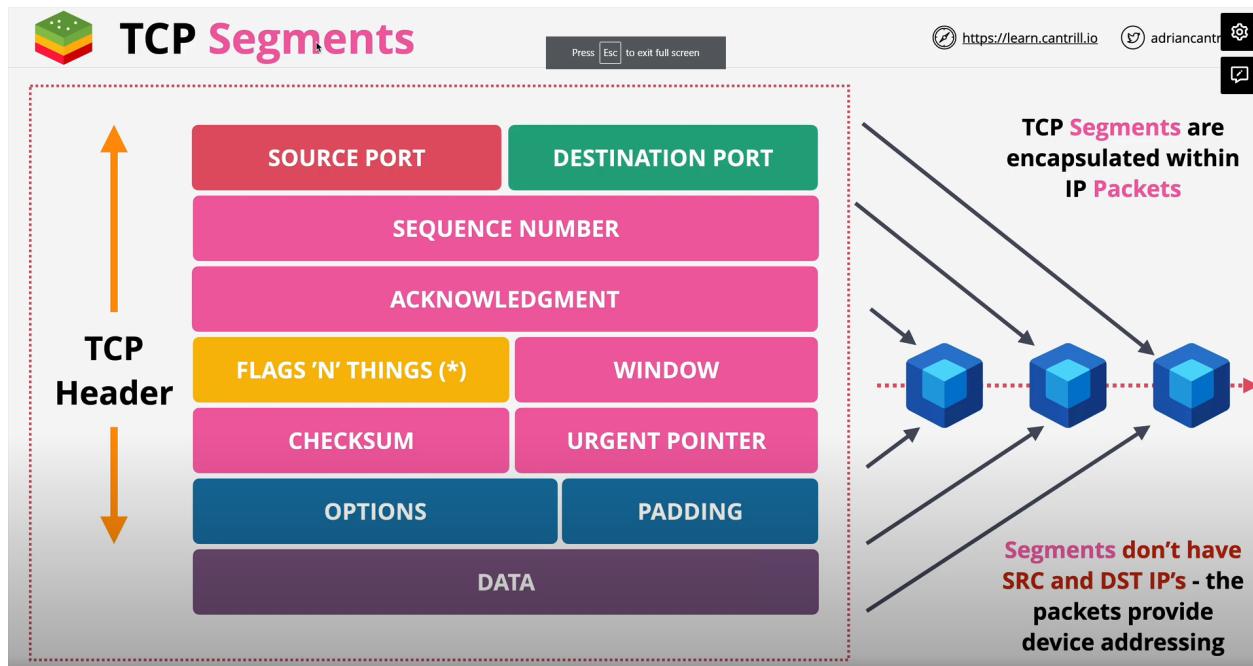
src port n destination port for sending and receiving data on the port

**sequence no:** is used to keep track of the sequence which is incremental and used to know if we are missing any packages for example lets say we sent seq 1-80 first and the destination has to acknowledge that he has received package 1-80

**flag n things** consist of flags n things which is used to close the connection

**window** is used to flow control based on receivers speed to avoid the traffic

**checksum:** error checking



3 way handshake for TCP



## TCP Connection 3-way Handshake

<https://learn.cantrill.io>

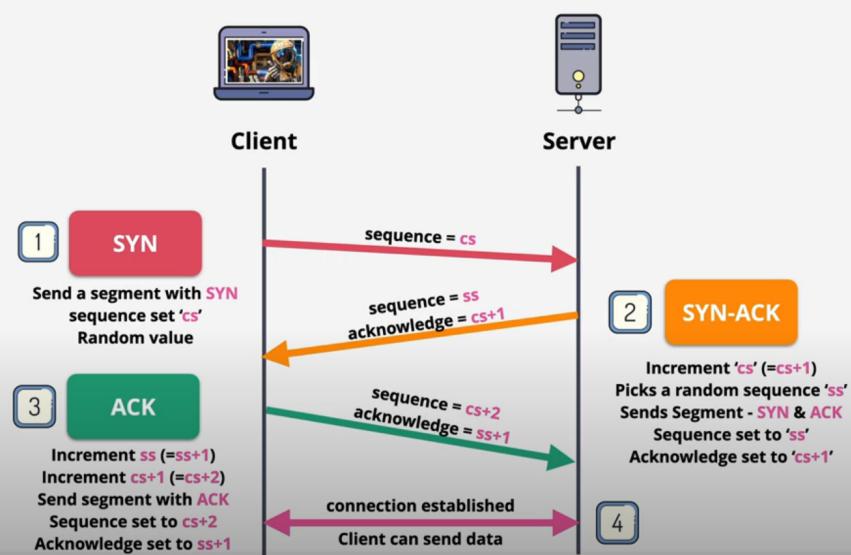
adriancantrill



### FLAGS 'N' THINGS (\*)

U	A	P	R	S	F
R	C	S	S	Y	I
G	K	H	T	N	N

Flags which can be set to alter the connection. e.g. **FIN** can be used to close, **ACK** for acknowledgments, **SYN** to synchronise sequence numbers



## NAT : Network address translation



## Network Address Translation (NAT)

<https://learn.cantrill.io>

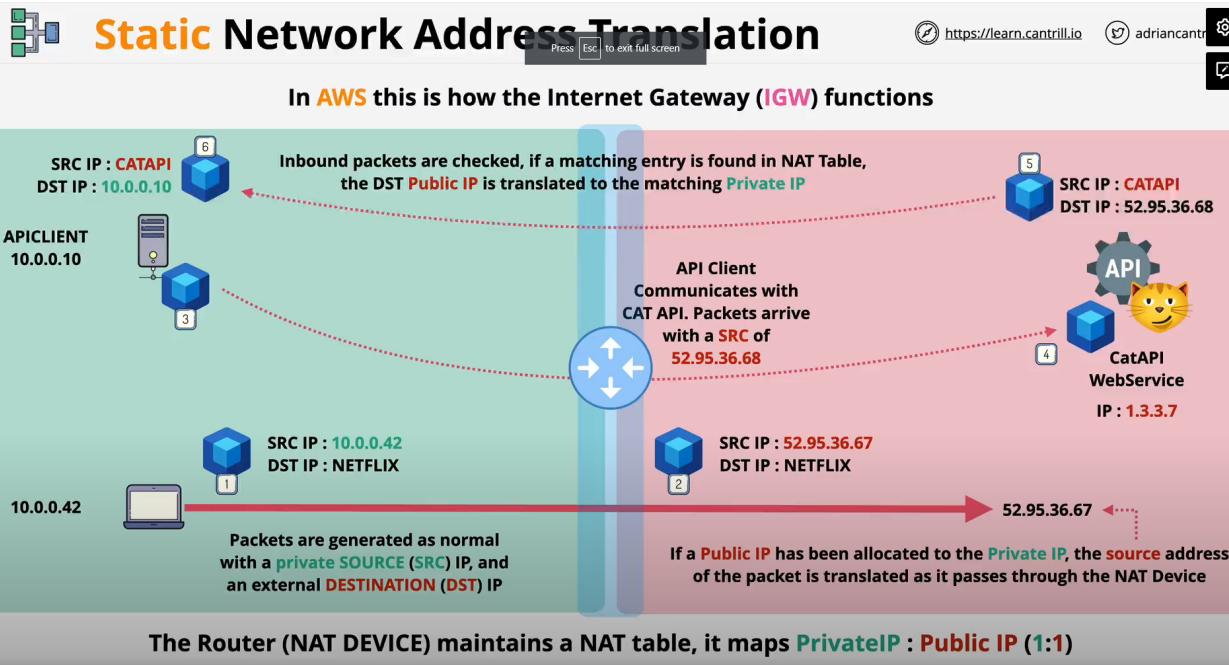
adriancantrill

- NAT is designed to overcome **IPv4 shortages**
- ... also provides some **security benefits**
- Translates **Private** IPv4 addresses to **Public**
- Static NAT - 1 private to 1 (fixed) public address (IGW)**
- Dynamic NAT - 1 private to 1st available Public**
- Port Address Translation (**PAT**) - **many private to 1 public** (NATGW)
- IPv4 only** ... makes no sense with IPv6

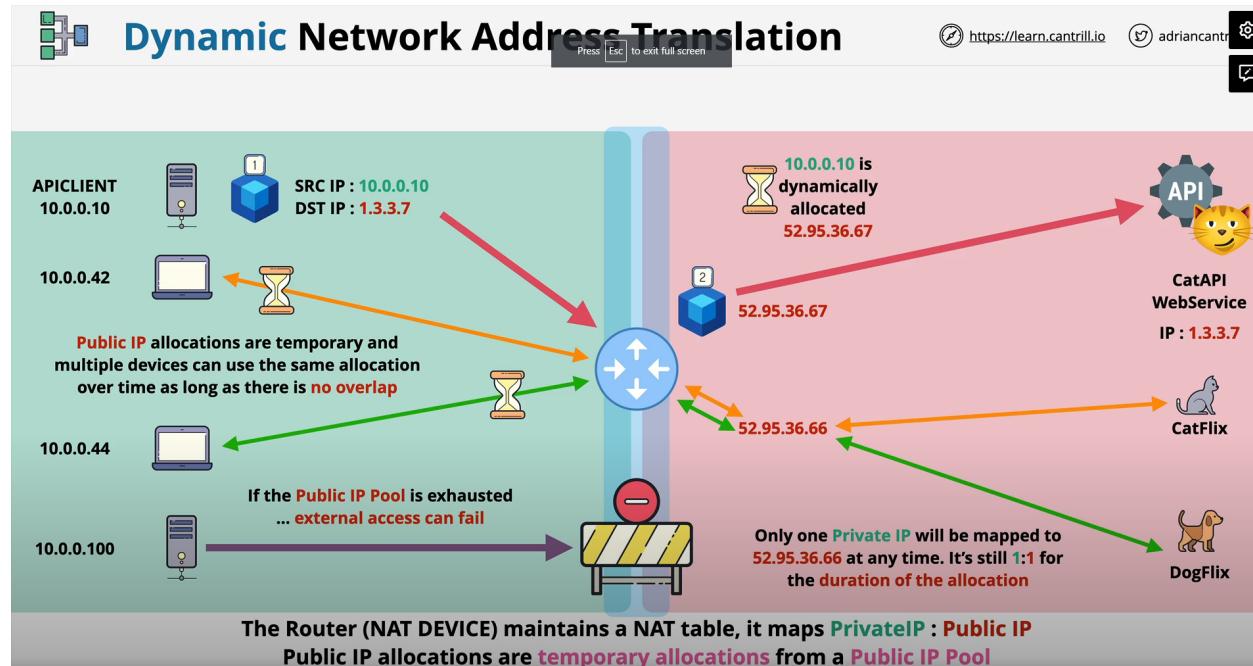
static NAT:

static NAT uses different public ip for different different private IP

and NAT table keeps the log



dynamic NAT: Uses number of Public address for multiple private IP  
this can be used when we have limited amount of public IP available



## PAT: Port Address Translation

multiple private IP uses same public ip



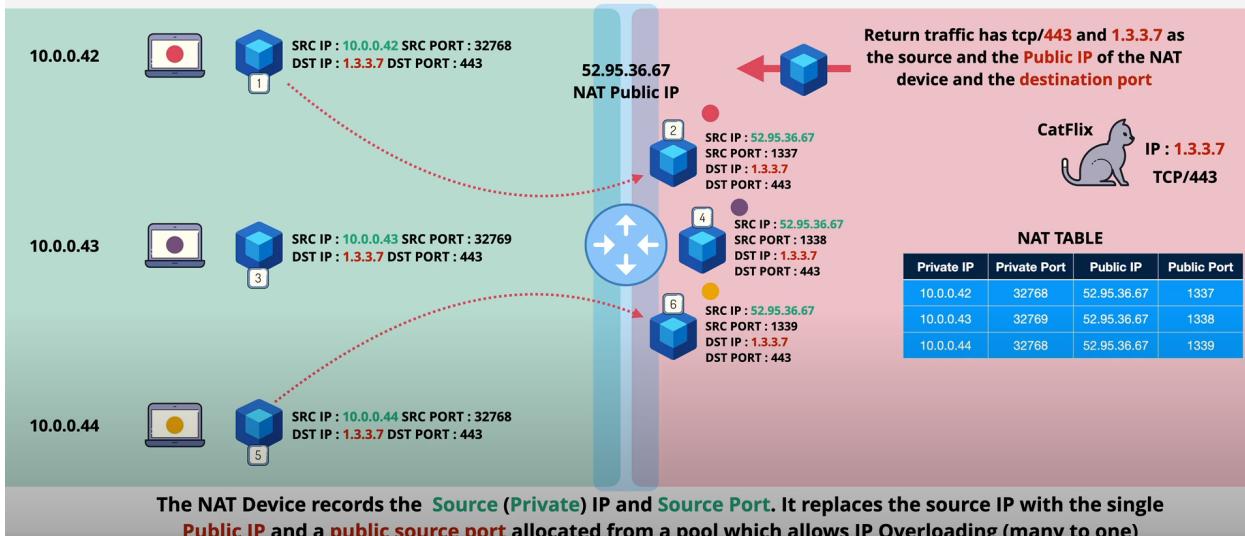
## Port Address Translation (PAT)

<https://learn.cantrill.io>

adriancantr



In AWS this is how the NATGateway (NATGW) functions - a (MANY:1) (PrivateIP:PublicIP) Architecture



## IP n Subnetting

public network



## IPv4 Address Space

<https://learn.cantrill.io>

adriancantr



START 0.0.0.0

A

128 NETWORKS  
16,777,216 IPs Each

- 0. (Reserved)
- 1.
- 2.
- 127.



Huge Businesses  
Early Internet



END 127.255.255.255

B

START 128.0.0.0

16,384 Networks  
65,536 IPs Each

128.0, 128.1, 128.2 ...  
... 191.253, 191.254, 191.255

END 191.255.255.255

C

START 192.0.0.0

2,097,152 NETWORKS  
256 IPs Each

192.0.1, 192.0.2 ....

END 223.255.255.255

CLASS D

CLASS E

Total of 4,294,967,296 IPv4 Addresses

## Private network

### IPv4 Addressing & Subnetting

https://learn.cantrill.io adrian

- Defined by a standard **RFC1918**
- 10.0.0.0 - 10.255.255.255** (**1 x Class A Network**)
  - .. **16,777,216** IPv4 addresses
- 172.16.0.0 - 172.31.255.255** (**16 x Class B Networks**)
  - .. **16 x 65,536** IPv4 Addresses
- 192.168.0.0 - 192.168.255.255** (**256 x Class C networks**)
  - .. **256 x 256** IPv4 Addresses

## subnet

### IP Subnetting

https://learn.cantrill.io adrian cantrill

**10.16.0.0/16**

**/16**

**10.16.0.0/17**

**/17**

**10.16.128.0/17**

**/17**

**10.16.0.0/17**

**/18**

**/18**

**10.16.128.0 /18**

**10.16.192.0 /18**

**10.16.0.0 /18**

**10.16.64.0 /18**

**10.16.128.0 /18**

**10.16.192.0 /18**

**/16 => 2 \* /17**

**Split network in 2**

**/17 => 2 \* /18**

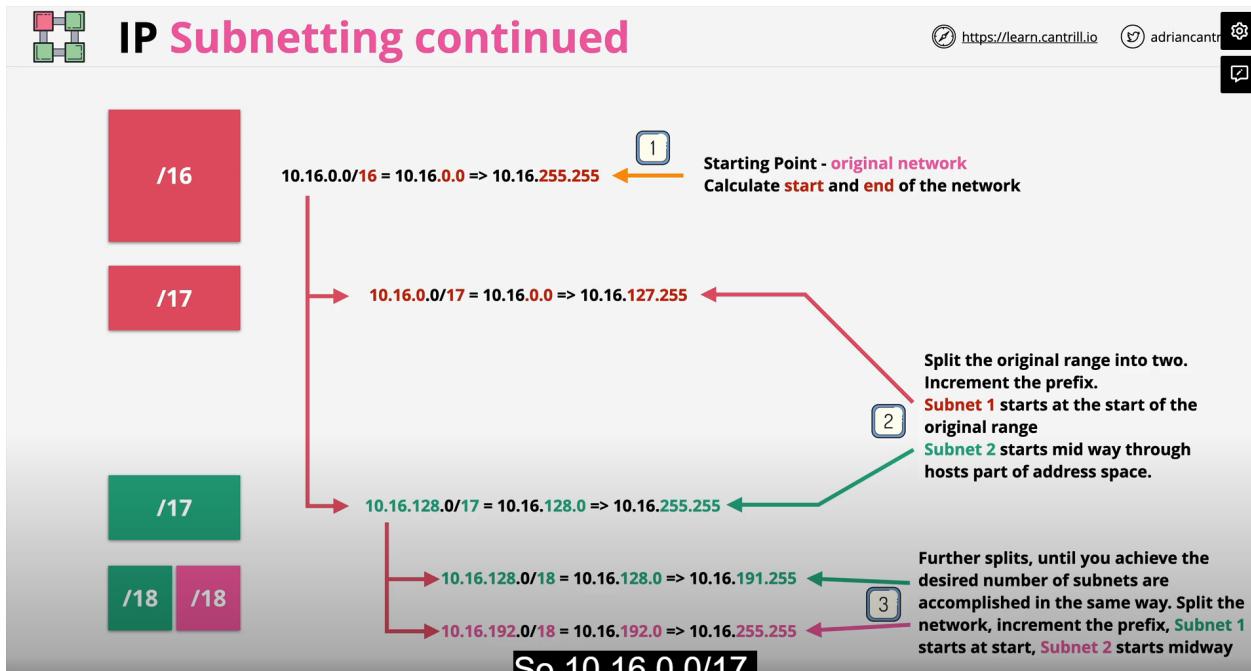
**2 smaller nets**

**Networks are usually split into 2, 4, 8 ... While unusual, odd number splits are valid**

**Subnetting is the process of taking a larger network, and breaking it into more smaller networks (higher prefix)**

**I've gone through it graphically,**

example



## SSL AND TLS

SSL and TLS

SSL: Secure socket layer

TLS: Transport Layer Security



# SSL and TLS

https://learn.cantrill.io

@adriancantrill

