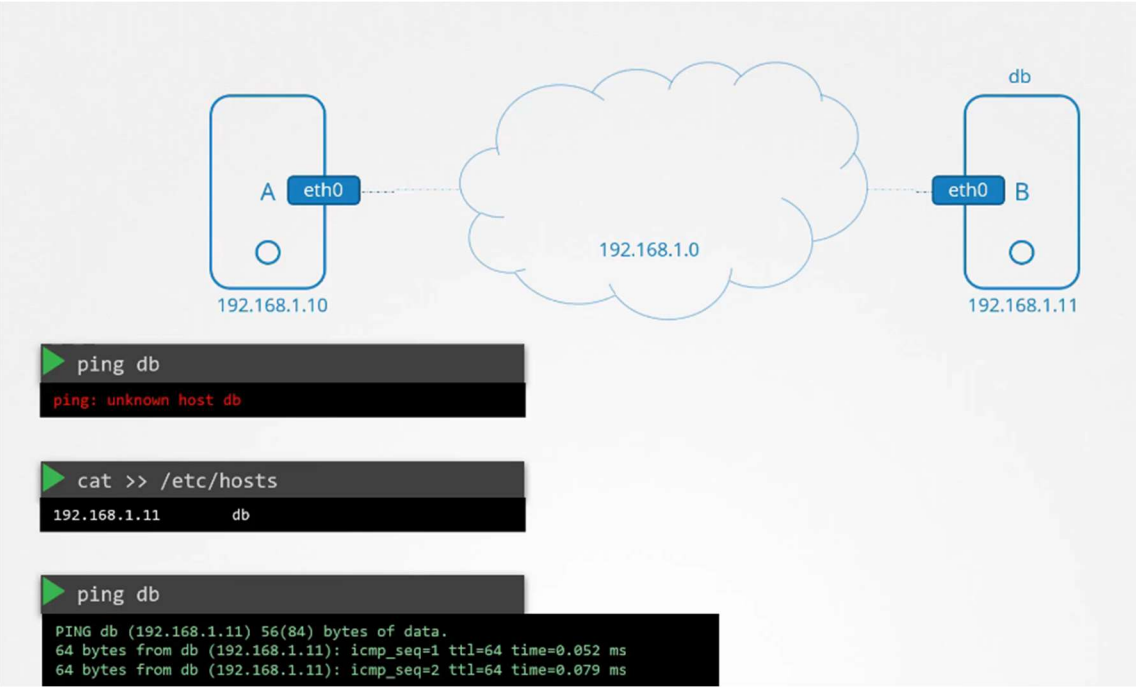
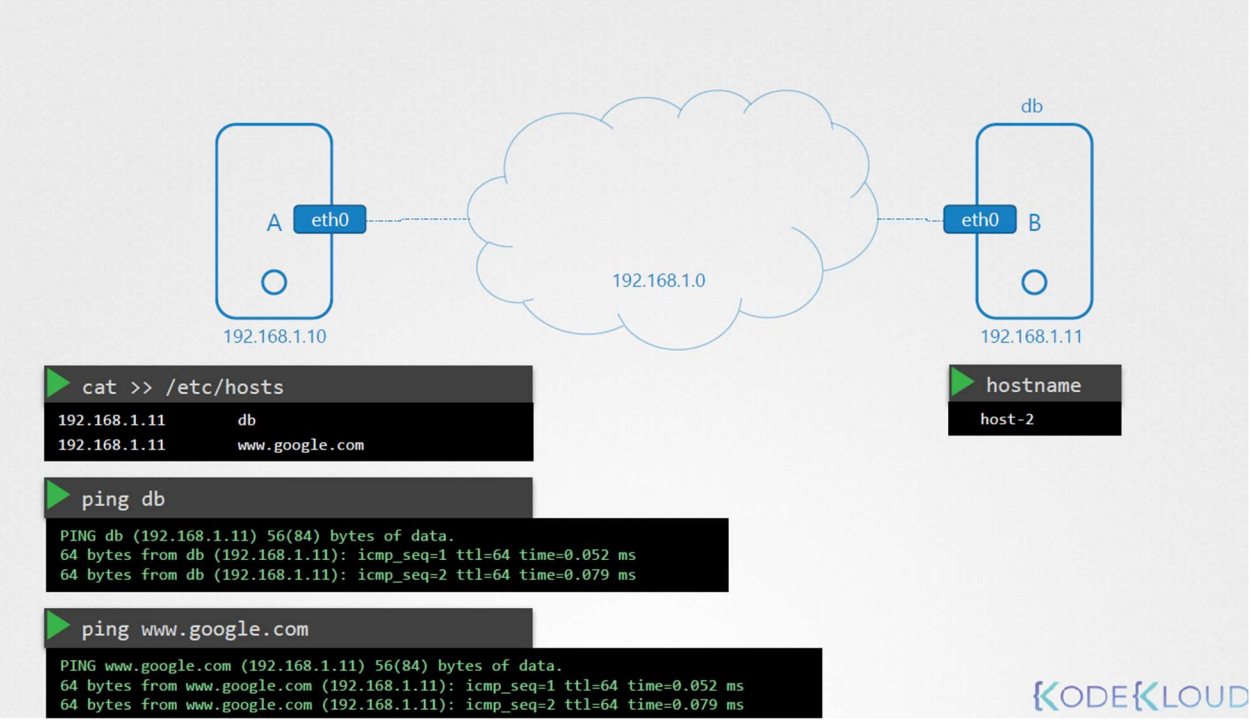
We have two computers, A and B, both part of the same network, and they've been assigned with IP addresses 192.168.1.10, and 1.11. You're able to ping one computer from the other using the other computer's IP address. You know that system B has database services on them. So instead of having to remember the IP address of system B, you decide to give it a name db. Going forward, you would like to ping system B using the name db instead of its IP address. If you try to ping db now, you would see that host A is unaware of a host named db.

So how do you fix that? Basically, you want to tell system A that system B at IP address 192.168.1.11 has a name db. You want to tell system A that when I say db, I mean the IP 192.168.1.11. You can do that by adding an entry into the **/etc/hosts** file on system A. Mention the IP address and the name you want your host to see system B has. We told system A that the IP at 192.168.1.11 is a host named db. Pings to db now gets sent to the correct IP and are successful.

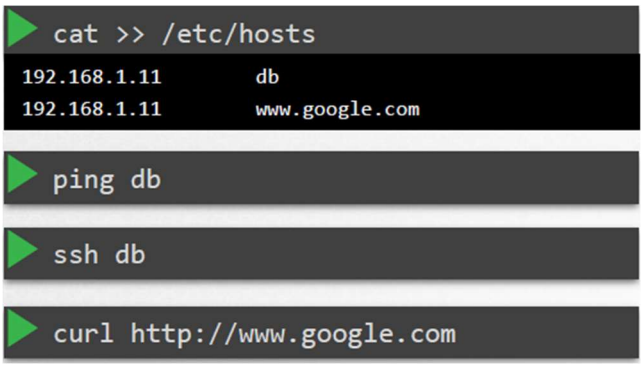


Now there is an important point to note here. We told system A that the IP at 192.168.1.11 is a host named db. Host A takes that for granted. Whatever we put in the /etc/hosts file is the source of truth for host A, but that may not be the truth. Host A does not check to make sure if system B's actual name is db. For instance, running a **hostname** command on system B reveals that it is named host-2, but host A doesn't care. It goes by what's in the host file.

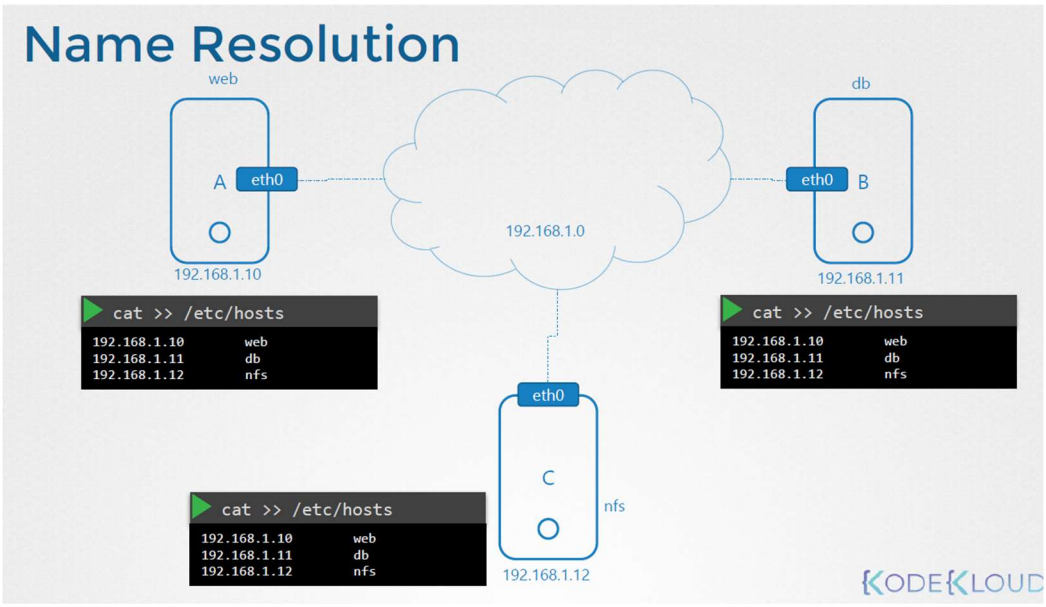
You can even fool system A to believing that system B is Google. Just add an entry into the host file with an IP mapping to www.google.com. Then ping Google and you will get a response from system B. So we have two names pointing to the same system, one as db and another as Google And we can use either names to reach system B.



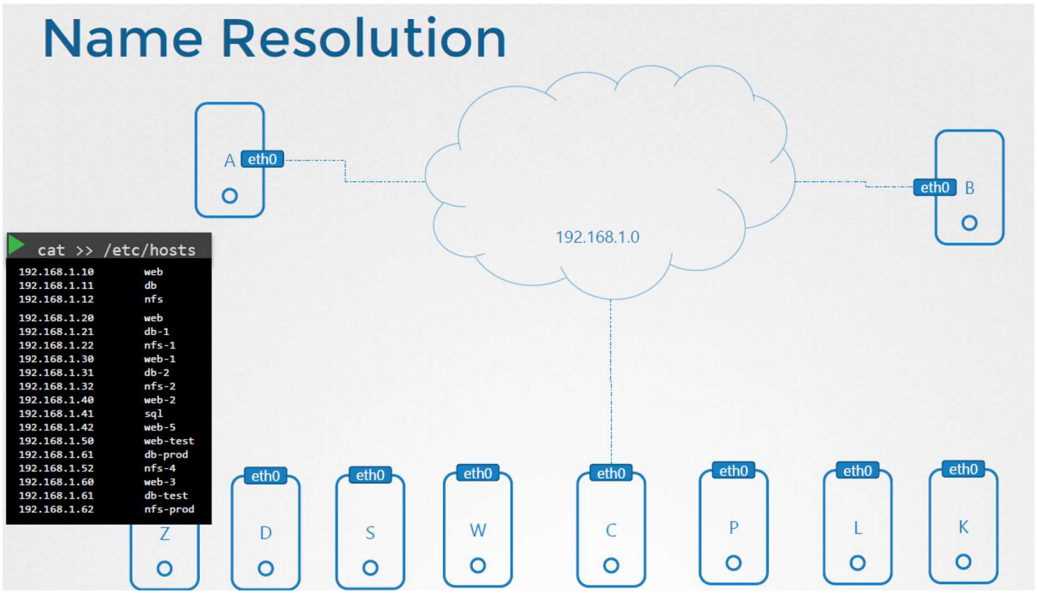
You can have as many names as you want for as many servers as you want in the /etc/hosts file. Every time we reference another host by its name from host A through a ping command or SSH command, or through any of the applications or tools within this system, it looks into its /etc/hosts file to find out the IP address of that host. Translating hostname to IP address this way is known as name resolution.



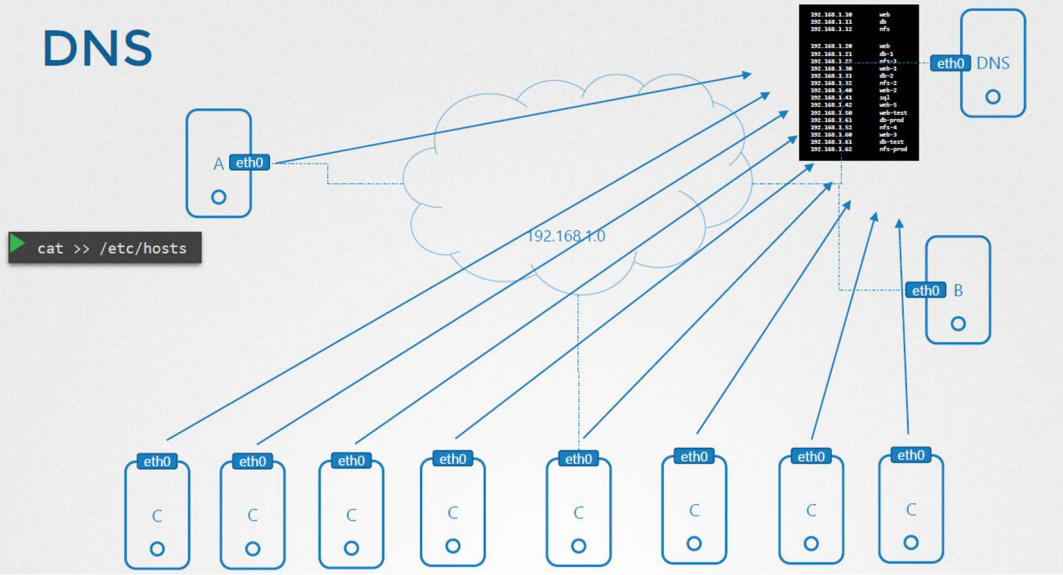
Within a small network of few systems, you can easily get away with the entries in the /etc/hosts file. On each system, I specify which are the other systems in the environment, and that's how it was done in the past.



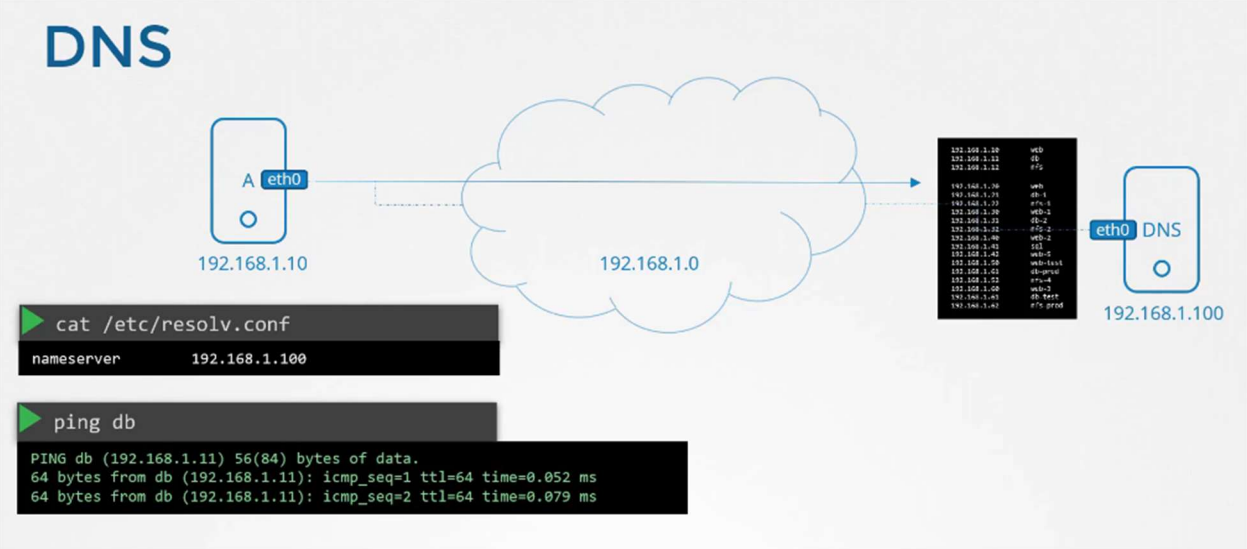
Until the environment grew and these files got filled with too many entries, and managing these became too hard. If one of the servers IP changed, you would need to modify the entries in all of these hosts, and that's where we decided to move all these entries into a single server who will manage it centrally. We call that our DNS server.



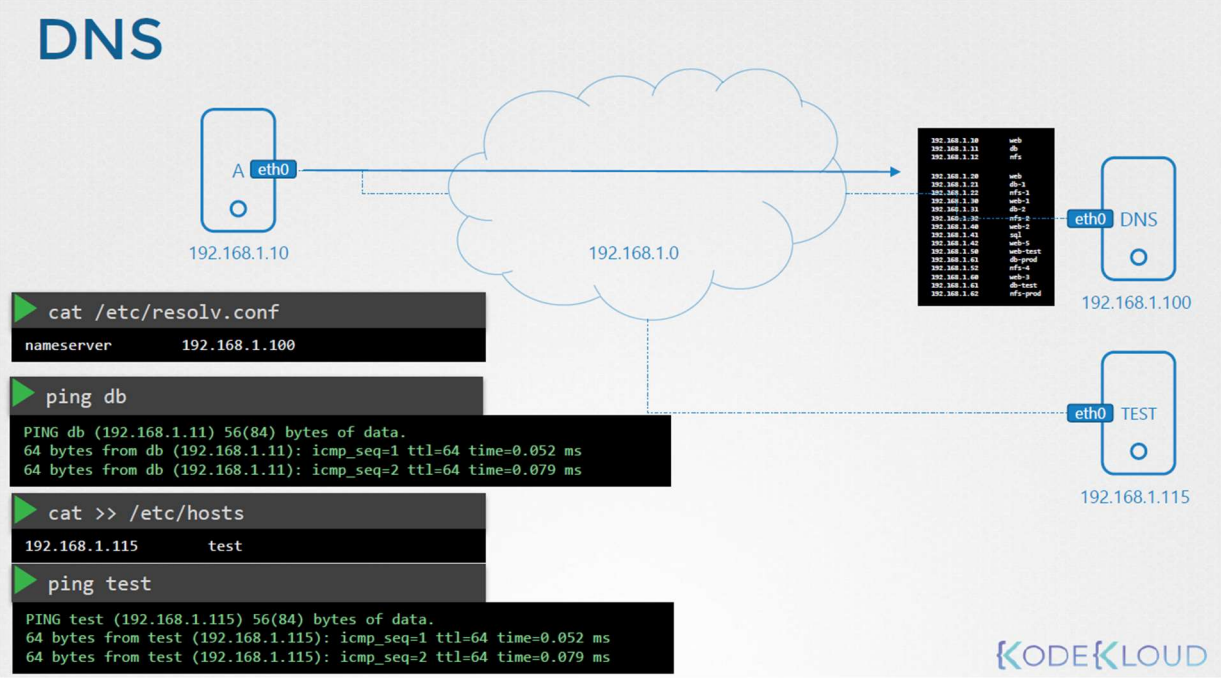
And then we point all hosts to look up that server if they need to resolve a host name to an IP address instead of its own /etc/hosts files.



So how do we do that? How do we point our host to a DNS server? Our DNS server has the IP 192.168.1.100. Every host has a DNS resolution configuration file at /etc/resolv.conf. You add an entry into it specifying the address of the DNS server. We say nameserver and point it to 192.168.1.100, and that should be it. Once this is configured on all of your hosts, every time a host comes up across a host name that it does not know about, it looks it up from the DNS server. If the IP of any of the host was to change, simply update the DNS server and all hosts should resolve the new IP address going forward.



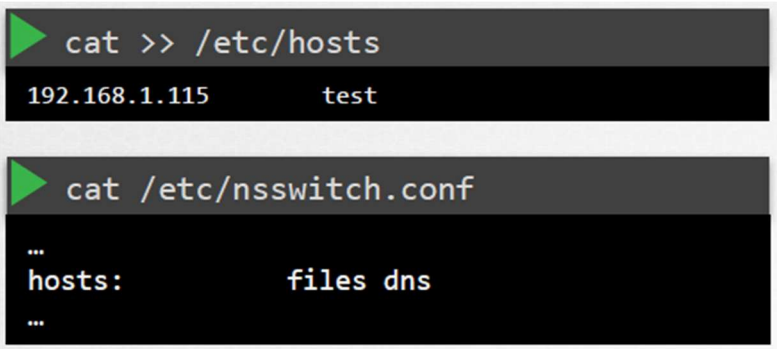
You no longer need any entries in the /etc/hosts file in any of the hosts, but that does not mean you can't have entries in host file. You still can, for example, say you were to provision a test server for your own needs. You don't think others would need to resolve the server by its name, so it may not be added to the DNS server. In that case, you can add an entry into your host /etc/hosts file to resolve this server. You can now resolve the server. However, no other system will be able to do that. So a system is able to use host name to IP mapping from the /etc/hosts file locally as well as from a remote DNS server.



What if you have an entry in both places, one in your /etc/hosts file and another in DNS? I have an entry in my local file set to 192.168.1.115, and someone added an entry for the same host to 192.168.1.116 on the server. In that case, the host first looks in the local /etc/hosts file and then looks at the name server.

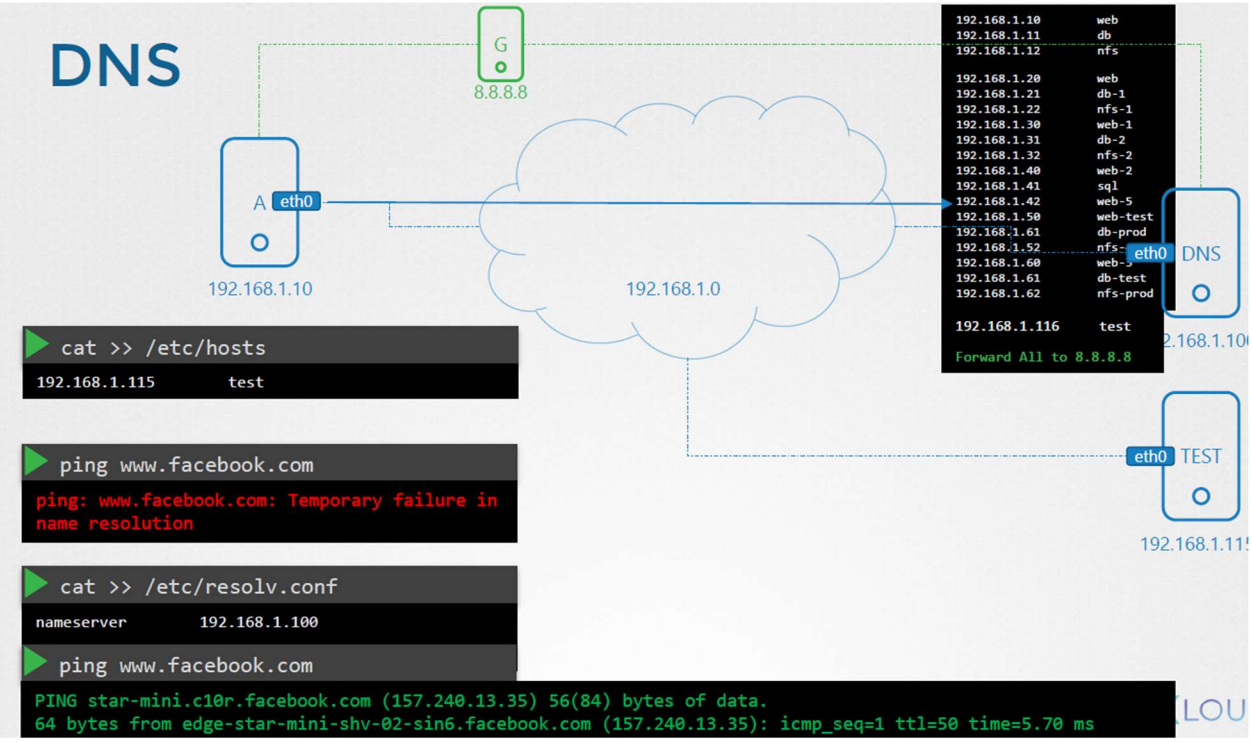
So if it finds the entry in the local /etc/hosts file, it uses that. If not, it looks for that host in the DNS server. But that order can be changed. The order is defined by an entry in the file **/etc/nsswitch.conf**. The line with the hosts entry, as you can see, the order is first files, and then followed by dns. Files refers to /etc/hosts file and dns refers to the DNS server.

So for every host name, the host first looks into the /etc/hosts file, and if it cannot find it there, it then looks at the DNS server. This order can be modified by editing this entry in the file. As per this order, our host would resolve the test server to 192.168.1.115.



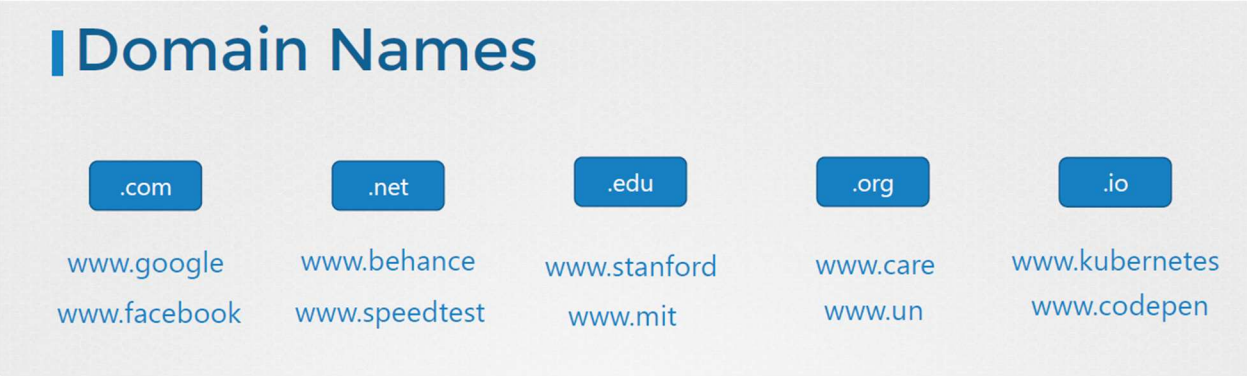
What if you try to ping a server that is not in either list? For example, I try and ping www.facebook.com. I don't have facebook.com in my etc host file and I don't have it in my DNS server, either. So in that case, it

will fail. You can add another entry into your (indistinct) resolv.conf file to point to a name server that knows Facebook. For example, 8.8.8.8 is a common well-known public name server available on the internet hosted by Google that knows about all websites on the internet. You can have multiple name servers like this configured on your host, but then you'll have to configure that on all your hosts in the network. You already have a name server within your network configured on all the hosts. So in that case, you can configure the DNS server itself to forward any unknown host names to the public name server on the internet, you should now able to ping external sites such as facebook.com.



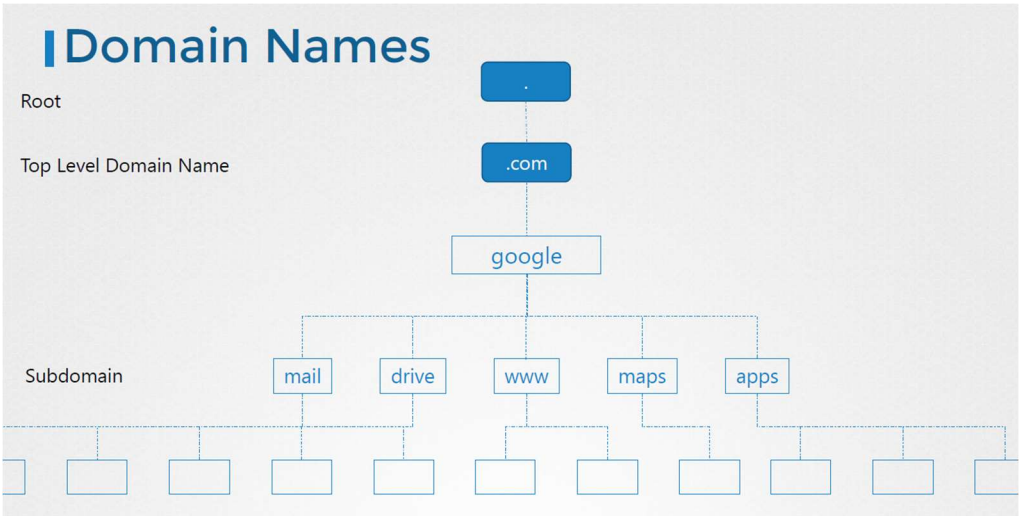
Until now, we have been just trying to read systems with their names like web, db, nfs, et cetera. But we just tried to ping Facebook at **www.facebook.com**. What is this name? With a www and .com at the end, it's called a domain name.

And it is how IPs translate to names that we can remember on the public internet, just like how we did for our hosts. Now, the reason they're in this format separated by dots is to group like things together. The last portion of the domain name, the .coms, the .nets, .edu, .org, et cetera, are the top level domains. They represent the intent of the website, .com for commercial or general purpose, .net for network, .edu for educational organizations, and .org for non-profit organizations.



Let's look at one in particular. In Google's case, the . is the root. That's where everything starts. .com is a top level domain. google is the domain name assigned to Google, and www is a subdomain. The subdomains help in further grouping things together under Google. For example, Google's map service is available at maps.google.com. So maps is a subdomain. Google's storage service is available at drive.google.com. Mobile apps are available at apps.google.com. Google's email service are available at mail.google.com. You can further divide each of these into as many subdomains based on your needs.

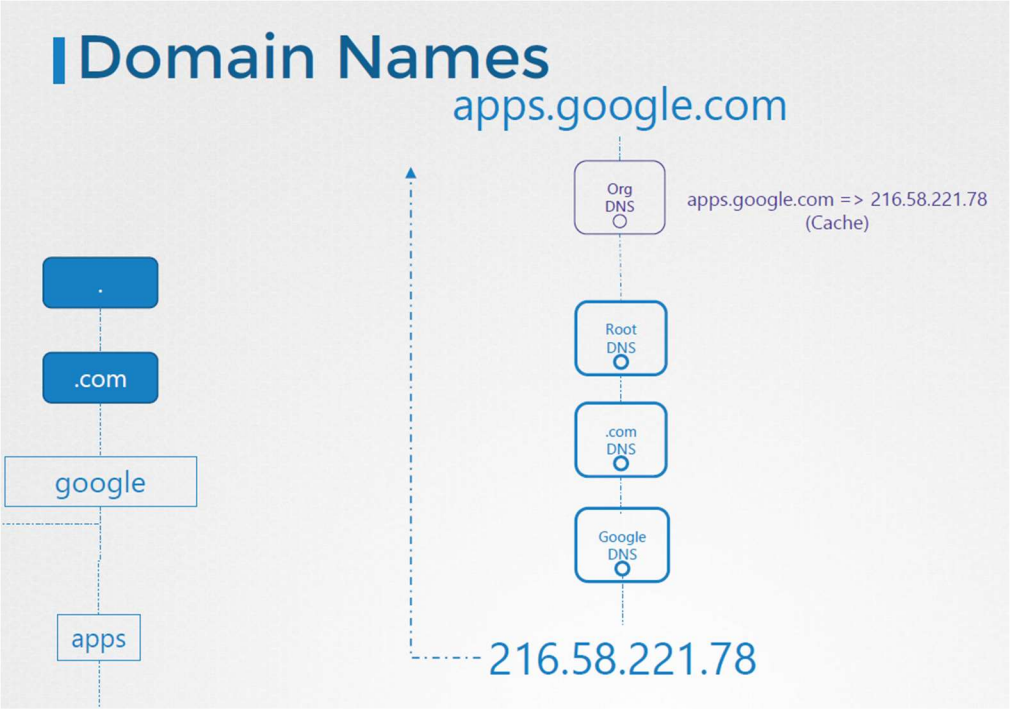
So you begin to see a tree structure forming.



When you try to reach any of these domain names, say **apps.google.com**, from within your organization, your request first hits your organization's internal DNS server. It doesn't know who apps or google is, so it forwards your request to the internet.

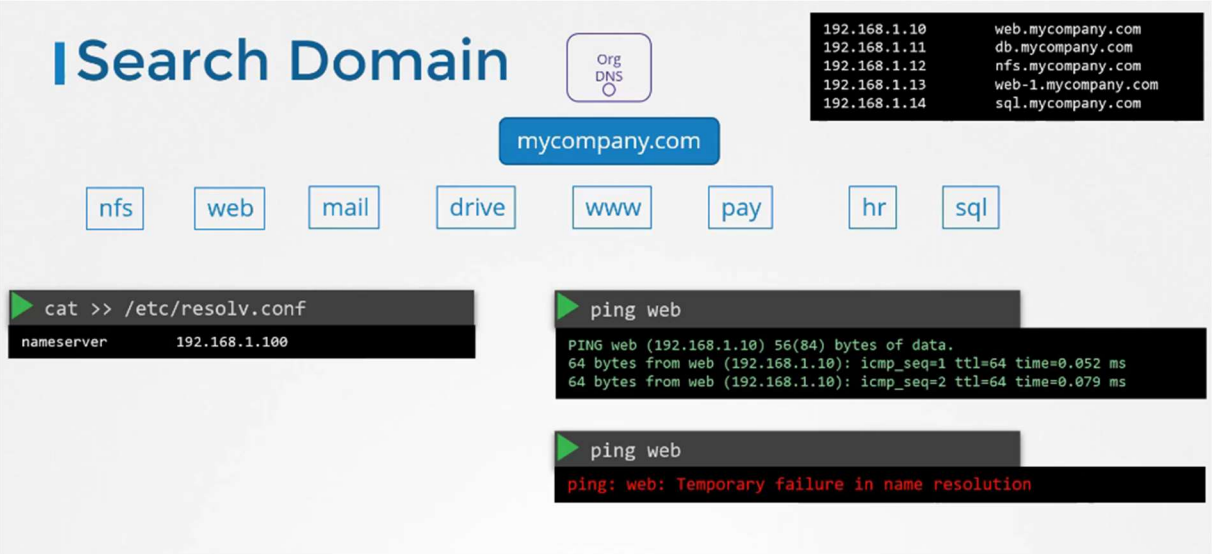
On the internet, the IP address of the server serving apps.google.com may be resolved with the help of multiple DNS servers. A root DNS server looks at your request and points you to a DNS server serving .coms. A .com DNS server looks at your request and forwards you to Google. And Google's DNS server provides you the IP of the server serving the apps applications.

In order to speed up all future results, your organization's DNS server may choose to cache this IP for a period of time, typically few seconds up to few minutes. That way, it doesn't have to go through the whole process again each time.



So that was out in the public. What about your organization? Your organization can have a similar structure, too. For example, your organization could be called as mycompany.com and have multiple subdomains for each purpose. The www for external facing website, mail.mycompany.com for accessing your organization's mail, drive for accessing storage, pay.company.com for accessing the payroll application, hr for accessing HR application, et cetera. All of these are configured in your organization's internal DNS server.

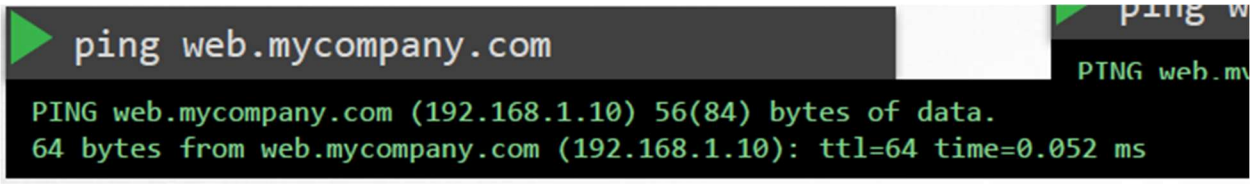
The reason we discussed all of this is to understand another entry in the /etc/resolv.conf file. Remember, this is the file where we configure the DNS server to be used for our host. With that, we were able to resolve servers in your organization with just their names like web. We have now introduced more standard domain names like web.mycompany.com, or db.mycompany.com, et cetera. Now, when you ping web, you can no longer get a response. Of course, this is because we are trying to ping web, but there is no record by the name web on my DNS server. Instead, it is web.mycompany.com. So you have to use web.mycompany.com.



Now I can understand if someone outside our company wants to access our web server, he would have to use web.mycompany.com, but within our company, your own company, you want to simply address the web server by its first name web, just like how you address other members in your family simply by their first names, which is not the case when someone outside your family addresses them using their full names. So what do you do to configure web to resolve my web.mycompany.com?

You want to say, when I say web, I mean web.mycompany.com. For that, you make an entry into your host's **/etc/resolv.conf** file called search and specify the domain name you want to append. Next time you try to ping **web**, you will see it actually tries **web.mycompany.com**.

Now, your host is intelligent enough to exclude the search domain if you specified a domain in your query like **web.mycompany.com**

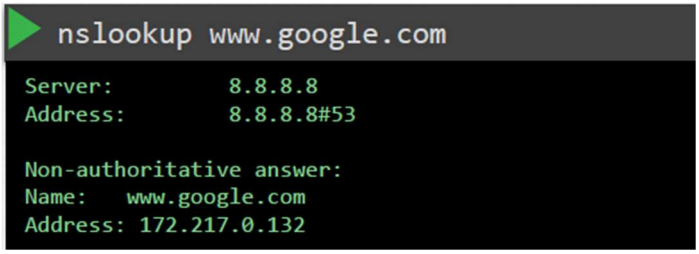


You may also provide additional search domains like this. So it would mean when I say web, I mean web.mycompany.com, or web.prod.mycompany.com. So your host will try searching all of these domain name when you look for a host name.

Finally, a word about record types. So how are the records stored in the DNS server? We know that it stores IP to host names. That's known as A records. Storing IPv6 to host names is known as AAAA records. Mapping one name to another name is called CNAME records. For example, you may have multiple aliases for the same application, like a food delivery service may also be reached at eat or hungry. That's where a CNAME record is used. Name to name mapping. There are many more,

A	web-server	192.168.1.1
AAAA	web-server	2001:0db8:85a3:0000:0000:8a2e:0370:7334
CNAME	food.web-server	eat.web-server, hungry.web-server

Now, ping may not always be the right tool to test DNS resolution. There are a few other tools as well, such as nslookup. You can use nslookup to query a host name from a DNS server. But remember, nslookup does not consider the entries in the local etc host file. So if you add an entry into the local etc host file for your web application, and if you try to do an nslookup for that web application, it is not going to find it. The entry for your web application has to be present in your DNS server. nslookup only queries the DNS server.



The same goes with dig. dig is another useful tool to test DNS name resolution. It returns more details in a similar form as is stored on the server.

```
dig www.google.com
```

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28065
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         245     IN      A       64.233.177.103
www.google.com.         245     IN      A       64.233.177.105
www.google.com.         245     IN      A       64.233.177.147
www.google.com.         245     IN      A       64.233.177.106
www.google.com.         245     IN      A       64.233.177.104
www.google.com.         245     IN      A       64.233.177.99

;; Query time: 5 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Mar 24 04:34:33 UTC 2019
;; MSG SIZE  rcvd: 139
```