**I-Networking**

# Data Networking Introduction

## **What is Data networking?**

Learning the networking technology can help you understand the internet better. This is the media of connecting one or more persons with each other. By using internet, we can share our stories, knowledge, opinions and experiences with other people. We also can discuss interesting and hot topics with new friends by internet.

Through internet you also can broaden our minds. Well, wow, wow, we know that internet is a network of networks.  
The main network types are : LAN, WAN, PAN, and MAN.  
Are you confused with these network types? What do they all mean?

In this super skill, we will discuss and talk about what the exact LAN, WAN, PAN, and MAN network types are.

# What is Data networking?

As seen previously, the main network types are : LAN, WAN, PAN, and MAN.

### **Personal Area Network (PAN)**

PAN stands for Personal Area Network, and again, it's exactly what it sounds like: a network covering a very small area, usually a small room. The best known wireless PAN network technology is Bluetooth, and the most popular wired PAN is USB.  
You might not think of your wireless headset, your printer or your smartphones as components in a network, but they are definitely talking with each other.  
Many peripheral devices are actually computers in their own right. Wi-Fi also serves as a PAN technology, since Wi-Fi is also used over a small area.

# What is Data networking?

### **Local Area Network (LAN)**

LAN, short for Local Area Network, is a computer network covering a small geographic area with the range of 1-5 km, like the home, office, school, or a group of buildings where there are computers, servers and peripheral devices like printers, scanners, projectors and other storage components. Very often the connections between servers are carried out by [**Ethernet cables**](https://www.fs.com/c/cat5e-cat6-cat7-cat8-904) , and end-devices communicate with each other through wireless connection, that is, Wi-Fi. Some of the most common LAN protocols are Ethernet, Token Ring and Fiber Distributed Data Interface, or FDDI, while most of the wireless protocols in use today are 802.11a, 802.11b, 802.11g and 802.11n.

### **Metropolitan Area Network (MAN)**

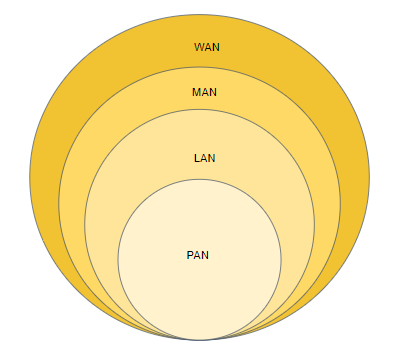
MAN stands for Metropolitan Area Network. As MAN's name suggests, it's often used in cities and other locations covering a range of 50-60 km. MANs are high-speed connection networks that interconnect several local area networks into a single large network with a common bridge. That bridge is called backbone lines which usually established by optical fiber to boost data transfer speed. Simply put, the MAN can be considered as a group of one or more LAN networks connected together through a single cable. RS-232, X-25, Frame Relay, and ATM are the common protocol practice for communication in MAN.

# What is Data networking?

### **Wide Area Network (WAN)**

WAN, short for Wide Area Network, is a computer network that covers a large-scale geographical area with the diameter of roughly 100-1000 km, that is, any network whose communications link cross metropolitan, regional, or national boundaries. Devices involved are more diversified than those applied to the other types, ranging from routers to switches, firewalls modems and so on. Companies such as FS or other worldwide organizations utilize WAN connection between their various branches by communicating via microwave satellites. Some of the most common WAN protocols in use today are Frame Relay, X-25, Integrated Services Digital Network, or ISDN, and Point-to-Point Protocol, or PPP

# What is Data networking?



# What is Data networking?

### **LAN vs MAN vs WAN : How Do They Differ From Each Other?**

With the essence of realizing communication remains the same, the three network types can differ from each other in many aspects. Some of the significant differences are listed in the chart below.

| **Parameters** | **LAN** | **MAN** | **WAN** |
| --- | --- | --- | --- |
| Ownership of network | Private | Private or public | Private or public |
| Geographical area covered | Small | Moderate | Very large |
| Design and maintenance | Easy | Difficult | Difficult |
| Bandwidth | Low | Moderate | High |
| Data rates | High | Moderate | Low |
| Congestion | Less | More | More |
| Application | College, School, Hospital | Small towns, City | Country, Continent |

[#Data networking](https://www.electronics-notes.com/articles/connectivity/data-networks/networking-fundamentals-basics.php)

# OSI Model

When most non-technical people hear the term “seven layers”, they either think of the popular Super Bowl bean dip or they mistakenly think about the seven layers of Hell, courtesy of Dante’s Inferno (there are nine). For IT professionals, the seven layers refer to the Open Systems Interconnection (OSI) model, a conceptual framework that describes the functions of a networking or telecommunication system.

The model uses layers to help give a visual description of what is going on with a particular networking system. This can help network managers narrow down problems (Is it a physical issue or something with the application?), as well as computer programmers (when developing an application, which other layers does it need to work with?). Tech vendors selling new products will often refer to the OSI model to help customers understand which layer their products work with or whether it works “across the stack”.

# OSI Model

7- **Application layer**:Human-computer interaction layer , where applications can access the network services.

6- **Presentation layer**Ensures that data is in a usable format and is where data encryption occurs.

5- **Session layer**Maintains connections and is responsible for controlling ports and sessions.

4- **Transport layer**  
  
Transmits data using transmission protocols including TCP and UDP.

3- **Network layer**  
  
Decides which physical path the data will take.

2- **Data Link layer**  
  
Defines the format of data on the network.

1- **Physical layer**  
  
Transmits row bit stream over the physical medium.

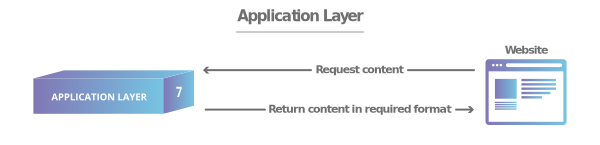
# OSI Model

Each layer of the OSI model handles a specific job and communicates with the layers above and below itself. [**DDoS attacks**](https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/) target specific layers of a network connection; [**application layer attacks**](https://www.cloudflare.com/learning/ddos/application-layer-ddos-attack/) target [**layer 7**](https://www.cloudflare.com/learning/ddos/what-is-layer-7/) and protocol layer attacks target layers 3 and 4.

### **Why does the OSI model matter?**

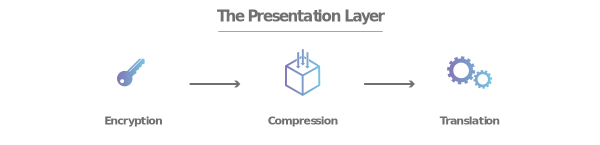
Although the modern Internet doesn’t strictly follow the OSI model (it more closely follows the simpler Internet protocol suite), the OSI model is still very useful for troubleshooting network problems. Whether it’s one person who can’t get their laptop on the Internet, or a web site being down for thousands of users, the OSI model can help to break down the problem and isolate the source of the trouble. If the problem can be narrowed down to one specific layer of the model, a lot of unnecessary work can be avoided.

# Application layer



This is the only layer that directly interacts with data from the user. Software applications like web browsers and email clients rely on the application layer to initiate communications. But it should be made clear that client software applications are not part of the application layer; rather the application layer is responsible for the protocols and data manipulation that the software relies on to present meaningful data to the user. Application layer protocols include [**HTTP**](https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/) as well as SMTP (Simple Mail Transfer Protocol is one of the protocols that enables email communications).

# Presentation layer



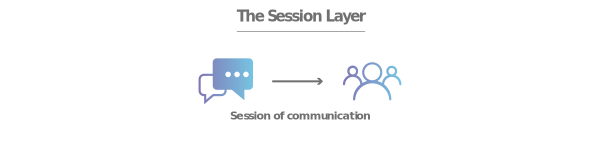
This layer is primarily responsible for preparing data so that it can be used by the application layer; in other words, layer 6 makes the data presentable for applications to consume. The presentation layer is responsible for translation, encryption, and compression of data.

Two communicating devices communicating may be using different encoding methods, so layer 6 is responsible for translating incoming data into a syntax that the application layer of the receiving device can understand.

If the devices are communicating over an encrypted connection, layer 6 is responsible for adding the encryption on the sender’s end as well as decoding the encryption on the receiver's end so that it can present the application layer with unencrypted, readable data.

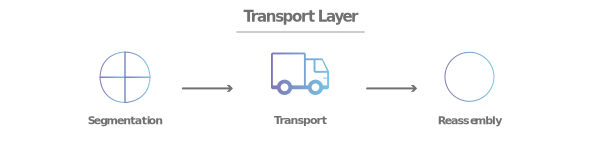
Finally the presentation layer is also responsible for compressing data it receives from the application layer before delivering it to layer 5. This helps improve the speed and efficiency of communication by minimizing the amount of data that will be transferred.

# Session layer



This is the layer responsible for opening and closing communication between the two devices. The time between when the communication is opened and closed is known as the session. The session layer ensures that the session stays open long enough to transfer all the data being exchanged, and then promptly closes the session in order to avoid wasting resources.  
The session layer also synchronizes data transfer with checkpoints. For example, if a 100 megabyte file is being transferred, the session layer could set a checkpoint every 5 megabytes. In the case of a disconnect or a crash after 52 megabytes have been transferred, the session could be resumed from the last checkpoint, meaning only 50 more megabytes of data need to be transferred. Without the checkpoints, the entire transfer would have to begin again from scratch.

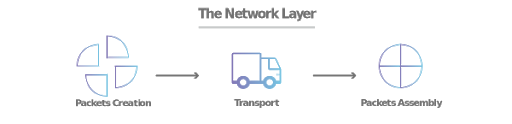
# Transport layer



Layer 4 is responsible for end-to-end communication between the two devices. This includes taking data from the session layer and breaking it up into chunks called segments before sending it to layer 3. The transport layer on the receiving device is responsible for reassembling the segments into data the session layer can consume.

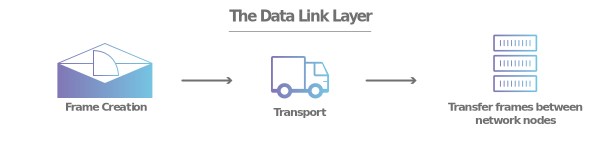
The transport layer is also responsible for flow control and error control. Flow control determines an optimal speed of transmission to ensure that a sender with a fast connection doesn’t overwhelm a receiver with a slow connection. The transport layer performs error control on the receiving end by ensuring that the data received is complete, and requesting a retransmission if it isn’t.

# Network layer



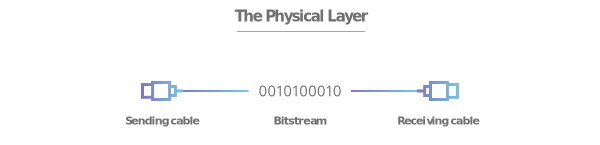
The network layer is responsible for facilitating data transfer between two different networks. If the two devices communicating are on the same network, then the network layer is unnecessary. The network layer breaks up segments from the transport layer into smaller units, called packets, on the sender’s device, and reassembling these packets on the receiving device. The network layer also finds the best physical path for the data to reach its destination; this is known as routing.

# Data link layer



The data link layer is very similar to the network layer, except the data link layer facilitates data transfer between two devices on the SAME network. The data link layer takes packets from the network layer and breaks them into smaller pieces called frames. Like the network layer, the data link layer is also responsible for flow control and error control in intra-network communication (The transport layer only does flow control and error control for inter-network communications).

# Physical layer



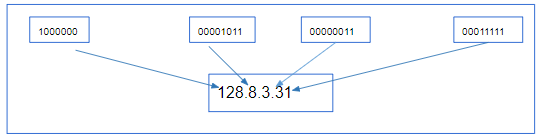
This layer includes the physical equipment involved in the data transfer, such as the cables and switches. This is also the layer where the data gets converted into a bit stream, which is a string of 1s and 0s. The physical layer of both devices must also agree on a signal convention so that the 1s can be distinguished from the 0s on both devices.

[#HTTP](https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/)[#OSI Model](https://www.javatpoint.com/osi-model)[#application security osi model](https://www.imperva.com/learn/application-security/osi-model/)

# IP addressing Basics

IP address is an address having information about how to reach a specific host, especially outside the LAN. An IP address is a 32 bit unique address having an address space of 232.

Generally, there are two notations in which IP address is written, dotted decimal notation and hexadecimal notation. Let's talk about Dotted Decimal Notation.

**Dotted Decimal Notation:**  
Here is how binary octets convert to decimal: The right most bit, or least significant bit, of an octet holds a value of 20. The bit just to the left of that holds a value of 21. This continues until the left-most bit, or most significant bit, which holds a value of 27. So if all binary bits are a one, the decimal equivalent would be 255 as shown here:  
1 1 1 1 1 1 1 1  
128 64 32 16 8 4 2 1 ⇒ (128+64+32+16+8+4+2+1=255) ⇒ Notice 128 = 2^8 , 64 = 2^7 ….  


[#IP Adressing basics](https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html)

IP addressing Basics: Classful Addressing

The 32 bit IP address is divided into five sub-classes. These are:

* Class A (1.0.0.1 to 126.255.255.254)
* Class B (128.1.0.1 to 191.255.255.254)
* Class C (192.0.1.1 to 223.255.254.254)
* Class D (224.0.0.0 to 239.255.255.255)
* Class E (240.0.0.0 to 255.255.255.254)

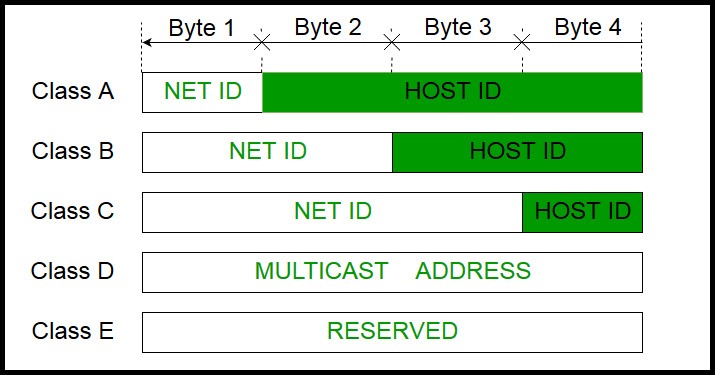
Each of these classes has a valid range of IP addresses. Classes D and E are reserved for multicast and experimental purposes respectively. The order of bits in the first octet determine the classes of IP address.

IPv4 address is divided into two parts:

* Network ID
* Host ID

The class of IP address is used to determine the bits used for network ID and host ID and the number of total networks and hosts possible in that particular class. Each ISP or network administrator assigns IP address to each device that is connected to its network.

# IP addressing Basics



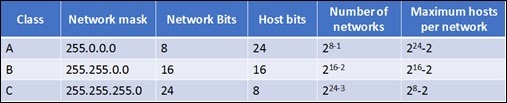
IP addressing: Network mask

* A network mask helps you know which portion of the address identifies the network and which portion of the address identifies the node. Class A, B, and C networks have default masks, also known as natural masks, as shown here:

Class A: 255.0.0.0  
Class B: 255.255.0.0  
Class C: 255.255.255.0  
An IP address on a Class A network that has not been subnetted would have an address/mask pair similar to: 8.20.15.1 255.0.0.0. In order to see how the mask helps you identify the network and node parts of the address, convert the address and mask to binary numbers.  
8.20.15.1 = 00001000.00010100.00001111.00000001  
255.0.0.0 = 11111111.00000000.00000000.00000000  
Once you have the address and the mask represented in binary, then identification of the network and host ID is easier. Any address bits which have corresponding mask bits set to 1 represent the network ID. Any address bits that have corresponding mask bits set to 0 represent the node ID.  
8.20.15.1 = 00001000.00010100.00001111.00000001  
255.0.0.0 = 11111111.00000000.00000000.00000000  
-----------------------------------  
net id | host id

netid = 00001000 = 8  
hostid = 00010100.00001111.00000001 = 20.15.1

* The following table summarizes the hosts and networks for the different classes of address:



[#IP addressing](https://www.geeksforgeeks.org/introduction-of-classful-ip-addressing/)

# Understanding subnet

Subnetting allows you to create multiple logical networks that exist within a single Class A, B, or C network.  
If you do not subnet, you are only able to use one network from your Class A, B, or C network, which is unrealistic.  
Each data link on a network must have a unique network ID, with every node on that link being a member of the same network. If you break a major network (Class A, B, or C) into smaller subnetworks, it allows you to create a network of interconnecting subnetworks. Each data link on this network would then have a unique network/subnetwork ID. Any device, or gateway, that connects n networks/subnetworks has n distinct IP addresses, one for each network / subnetwork that it interconnects.

In order to subnet a network, extend the natural mask with some of the bits from the host ID portion of the address in order to create a subnetwork ID. For example, given a Class C network of 204.17.5.0 which has a natural mask of 255.255.255.0, you can create subnets in this manner:

204.17.5.0 - 11001100.00010001.00000101.00000000  
255.255.255.224 - 11111111.11111111.11111111.11100000  
--------------------------|sub|----

By extending the mask to be 255.255.255.224, you have **taken three bits (indicated by "sub" )** from the original host portion of the address and used them to make subnets. With these three bits, it is possible to create eight subnets(2^3). With the remaining five host ID bits, each subnet can have up to 32(2^5) host addresses, 30 of which can actually be assigned to a device since host ids of all zeros or all ones are not allowed (it is very important to remember this). So, with this in mind, these subnets have been created

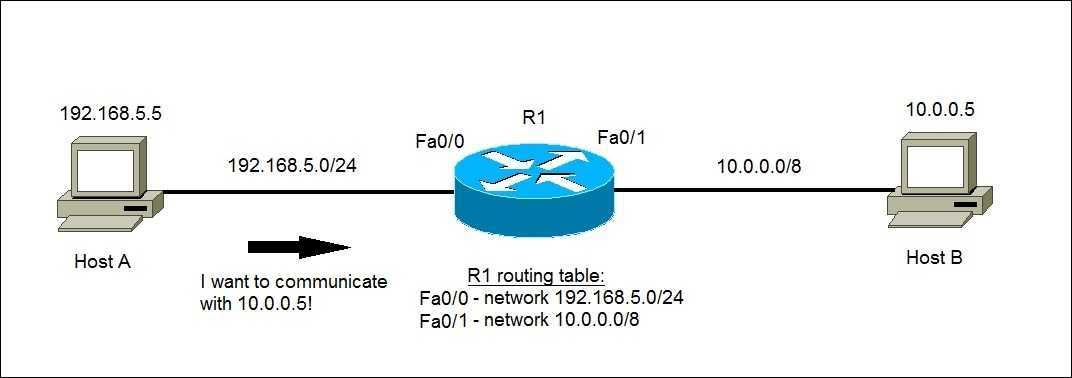
# Understanding subnet

204.17.5.0 255.255.255.224 host address range 1 to 30  
204.17.5.32 255.255.255.224 host address range 33 to 62  
204.17.5.64 255.255.255.224 host address range 65 to 94  
204.17.5.96 255.255.255.224 host address range 97 to 126  
204.17.5.128 255.255.255.224 host address range 129 to 158  
204.17.5.160 255.255.255.224 host address range 161 to 190  
204.17.5.192 255.255.255.224 host address range 193 to 222  
204.17.5.224 255.255.255.224 host address range 225 to 254

# IP Routing

The term IP routing refers to the process of taking a packet from one host and sending it to another host on a different network. The routing process is the usually done by devices called routers. You probably have this device at home, providing you with the Internet access.

Routers examine the destination IP address and make their routing decisions accordingly. To determine out which interface the packet will be forwarded, routers use routing tables which list all networks for which routes are known. Consider the following example:



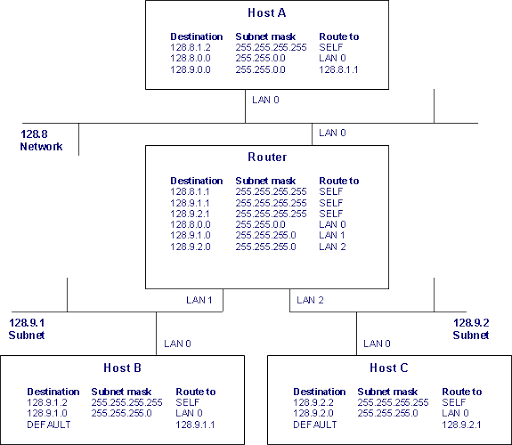
Networks in the internet are connected to each other via routers. Routers carry traffic from one network/subnet to another. Routers maintain a routing table to decide how to route the IP packets. Each routing entry consists of the **destination address, subnet mask** and **"route to" field**. When a message needs to be routed to an IP address, the following steps are followed:

1. The destination IP address is masked with the subnet mask and then compared with the destination field for all entries in the routing table.
2. This comparison may yield a match with more than one entry the entry with the longest subnet mask will be selected. E.g. , a packet destined for 128.8.1.2 reaching Host A would match the entries corresponding to 128.8.1.2 and 128.8.0. The entry corresponding to 128.8.1.2 will be selected, as it has a longer subnet mask.
3. Once an entry has been selected, the "route to" field is consulted and the action taken depends on the contents of this field:

* If the "route to" field contains SELF the packet is meant for this node. The IP packet is passed to the OS for application processing
* If the "route to" field contains a LAN interface id, the packet is destined for a LAN that is directly connected to the router/host. In this case, the packet is routed directly on the LAN.
* If the "route to" field contains an IP address, the packet is forwarded to the IP address specified. Further routing of the packet will be carried out by the specified IP address.

### **Note:**

* IP routing also supports a default entry. If the packet does not match any other entry, it is routed according to the default entry and the router drop the packet.



[#IP Routing](https://www.ibm.com/support/knowledgecenter/ssw_aix_72/network/tcpip_routing_intro.html)

IP Routing

**Multiple IP Addresses:**

Another important aspect of internets is a node in the internet can have multiple IP addresses. There will be one IP address per interface. For example, the Router in the figure above has three IP addresses, viz. 128.8.1.1, 128.9.1.1 and 128.9.2.1.

**Routing of a Packet from Host A to Host C:**

Here we will trace the path taken by an IP packet sent from Host A to Host C. Routing related fields in the Ethernet MAC header and IP header are shown.

**Host A originates an IP packet towards Host C:**

1. Application sends a message to Host C by sending it to 128.9.2.2 IP address (Host C's IP address).
2. This IP address matches the entry corresponding to 128.9.0.0. The "route to" field for the selected entry contains another IP address - 128.8.1.1. This is the IP address of the Router.
3. The IP routing table is accessed again for 128.8.1.1.
4. The entry that matches 128.8.1.1 contains LAN 0 interface id. This specifies that the destination node is directly connected to the host.
5. This packet is passed to the device driver.
6. Device driver consults the ARP cache to identify the Ethernet MAC address corresponding to the 128.8.1.1. (ARP is covered in another article).
7. Ethernet frame is sent to the MAC address found by ARP.

IP Routing

The packet sent on the 128.8 LAN is:  


Router send the IP packet to Host C

1. Router receives the Ethernet frame and passes it to the IP layer.
2. IP routing table is consulted and a matching entry is found corresponding to 128.9.2 subnet.
3. Packet is routed on the LAN 2 interface.
4. Host C's MAC address is found from the ARP cache.
5. Ethernet frame is addressed to Host C MAC Address.

The packet sent over the 128.9.2 LAN is:  


Host C receives the IP packet

1. Host C receives the Ethernet frame and passes it to the IP layer.
2. IP routing table is searched and a match is detected with 128.9.2.2 entry.
3. The "route to" field contains SELF, so the message is passed to the higher layer for delivery to the application.

IP Routing

The packet sent on the 128.8 LAN is:  


Router send the IP packet to Host C

1. Router receives the Ethernet frame and passes it to the IP layer.
2. IP routing table is consulted and a matching entry is found corresponding to 128.9.2 subnet.
3. Packet is routed on the LAN 2 interface.
4. Host C's MAC address is found from the ARP cache.
5. Ethernet frame is addressed to Host C MAC Address.

The packet sent over the 128.9.2 LAN is:  


Host C receives the IP packet

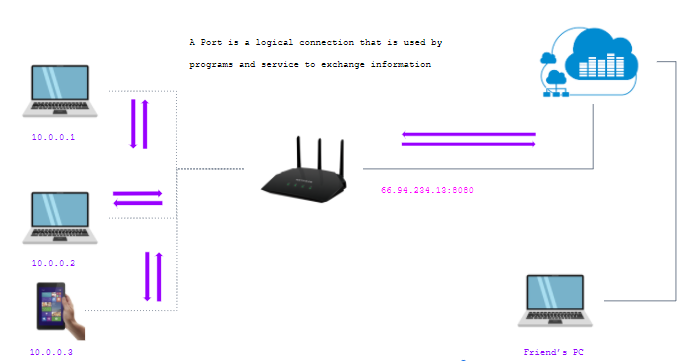
1. Host C receives the Ethernet frame and passes it to the IP layer.
2. IP routing table is searched and a match is detected with 128.9.2.2 entry.
3. The "route to" field contains SELF, so the message is passed to the higher layer for delivery to the application.

[#NAT](https://www.youtube.com/watch?v=FTUV0t6JaDA) [#Port Forwarding](https://www.youtube.com/watch?v=2G1ueMDgwxw) [#DHCP](https://www.youtube.com/watch?v=e6-TaH5bkjo)

# Network Service Port Forwarding

A port forward is a way of making a computer on your home or business network accessible to computers on the internet, even though they are behind a router. It is commonly used in gaming, security camera setup, voice over ip, and downloading files. After you have forwarded a port you are said to have an open port.  
A Port is a logical connection that is used by programs and service to exchange information.  
Example: Your friend is going to connect your Desktop using RDC (Remote Desktop Connection)

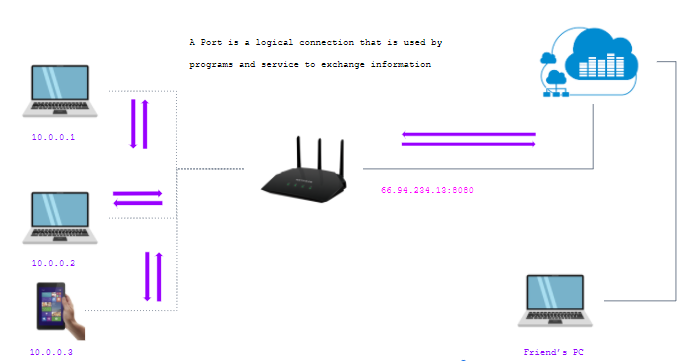
* Without any port forwarding , your router does not know what to do with this request.
* we are going to tell our router to send or forward any requests that come in the port (8080 for example ) and send the request to our computer.
* we need to log into the router’s configuration page by typing the router’s internal @IP. Address using q web browser.

A Port is a logical connection that is used by programs and service to exchange information  


### **Note:**

* 66.94.234.13 ⇒ Public IP adress
* 8080 ⇒ Port Number

# Network Service Port Forwarding

A Port is a logical connection that is used by programs and service to exchange information  


### **Note:**

66.94.234.13 ⇒ Public IP adress  
8080 ⇒ Port Number

# Network Service : DHCP

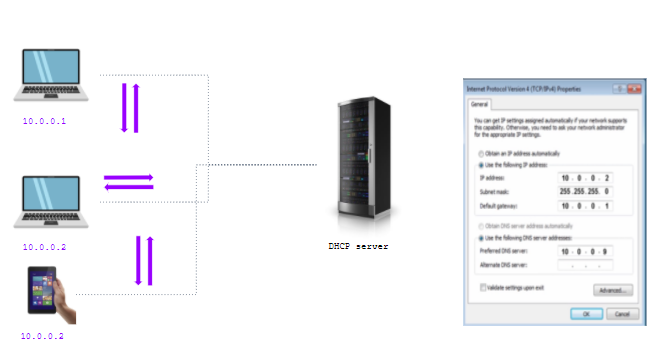
Definition : DHCP stands for dynamic host configuration protocol and is a network protocol used on IP networks where a DHCP server automatically assigns an IP address and other information to each host on the network so they can communicate efficiently with other endpoints.

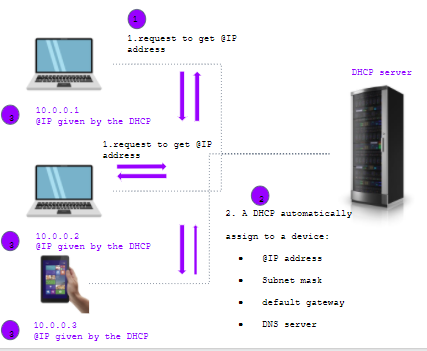
In addition to the IP address, DHCP also assigns the subnet mask, default gateway address, domain name server (DNS) address and other pertinent configuration parameters. Request for comments (RFC) 2131 and 2132 define DHCP as an Internet Engineering Task Force (IETF)- defined standard based on the BOOTP protocol.  
DHCP simplifies IP address management

The primary reason DHCP is needed is to simplify the management of IP addresses on networks. No two hosts can have the same IP address, and configuring them manually will likely lead to errors. Even on small networks manually assigning IP addresses can be confusing, particularly with mobile devices that require IP addresses on a non-

permanent basis. Also, most users aren’t technically proficient enough to locate the IP address information on a computer and assign it. Automating this process makes life easier for users and the network administrator.

# DHCP with static @IP address

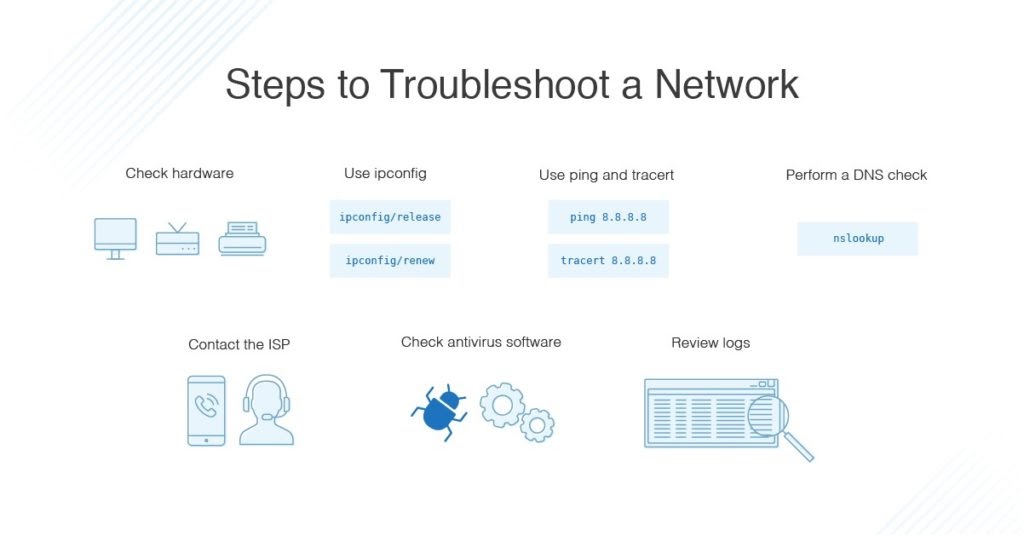
* When we create a static @IP manually we can a problem with a large internet connection also an @IP address conflict because each host must have a unique @IP address  
  
* host2 and host3 can’t access to the network because he have the same @IP address (10.0.0.2)
* A dynamic @IP address is where a device get the I.P address from a DHCP address.



# Troubleshooting

How to Troubleshoot a Network ?  
Issues can arise at numerous points along the network. Before you start trying to troubleshoot any issue, you want to have a clear understanding of what the problem is, how it came up, who it’s affecting, and how long it’s been going on. By gathering the right information and clarifying the problem, you’ll have a much better chance of resolving the issue quickly, without wasting time trying unnecessary fixes.

# Troubleshooting Steps



We always start troubleshooting using these simple network troubleshooting steps to help diagnose and refine the issue.

1. Check the hardware. When you’re beginning the troubleshooting process, check all your hardware to make sure it’s connected properly, turned on, and working. If a cord has come loose or somebody has switched off an important router, this could be the problem behind your networking issues. There’s no point in going through the process of troubleshooting network issues if all you need to do is plug a cord in. Make sure all switches are in the correct positions and haven’t been bumped accidentally.Next, turn the hardware off and back on again. This is the mainstay of IT troubleshooting, and while it might sound simplistic, often it really does solve the problem. Power cycling your modem, router, and PC can solve simple issues—just be sure to leave each device off for at least 60 seconds before you turn it back on.
2. Use ipconfig/ifconfig. Open the command prompt and type “ipconfig” (without the quotes) into the terminal. The Default Gateway (listed last) is your router’s IP. Your computer’s IP address is the number next to “IP Address.” If your computer’s IP address starts with 169, the computer is not receiving a valid IP address. If it starts with anything other than 169, your computer is being allocated a valid IP address from your router.  
   Try typing in “ipconfig /release” followed by “ipconfig /renew” to get rid of your current IP address and request a new one. This will in some cases solve the problem. If you still can’t get a valid IP from your router, try plugging your computer straight into the modem using an ethernet cable. If it works, the problem lies with the router
3. Use ping and tracert. If your router is working fine, and you have an IP address starting with something other than 169, the problem’s most likely located between your router and the internet. At this point, it’s time to use the ping tool. Try sending a ping to a well-known, large server, such as Google, to see if it can connect with your router. You can ping Google DNS servers by opening the command prompt and typing “ping 8.8.8.8”; you can also add “-t” to the end (ping 8.8.8.8 -t) to get it to keep pinging the servers while you troubleshoot. If the pings fail to send, the command prompt will return basic information about the issue.  
   You can use the tracert command to do the same thing, by typing “tracert 8.8.8.8”; this will show you each step, or “hop,” between your router and the Google DNS servers. You can see where along the pathway the error is arising. If the error comes up early along the pathway, the issue is more likely somewhere in your local network.
4. Perform a DNS check. Use the command “nslookup” to determine whether there’s a problem with the server you’re trying to connect to. If you perform a DNS check on, for example, google.com and receive results such as “Timed Out,” “Server Failure,” “Refused,” “No Response from Server,” or “Network Is Unreachable,” it may indicate the problem originates in the DNS server for your destination. (You can also use nslookup to check your own DNS server.)
5. Contact the ISP. If all of the above turn up no problems, try contacting your internet service provider to see if they’re having issues. You can also look up outage maps and related information on a smartphone to see if others in your area are having the same problem.
6. Check on virus and malware protection. Next, make sure your virus and malware tools are running correctly, and they haven’t flagged anything that could be affecting part of your network and stopping it from functioning.
7. Review database logs. Review all your database logs to make sure the databases are functioning as expected. If your network is working but your database is full or malfunctioning, it could be causing problems that flow on and affect your network performance.

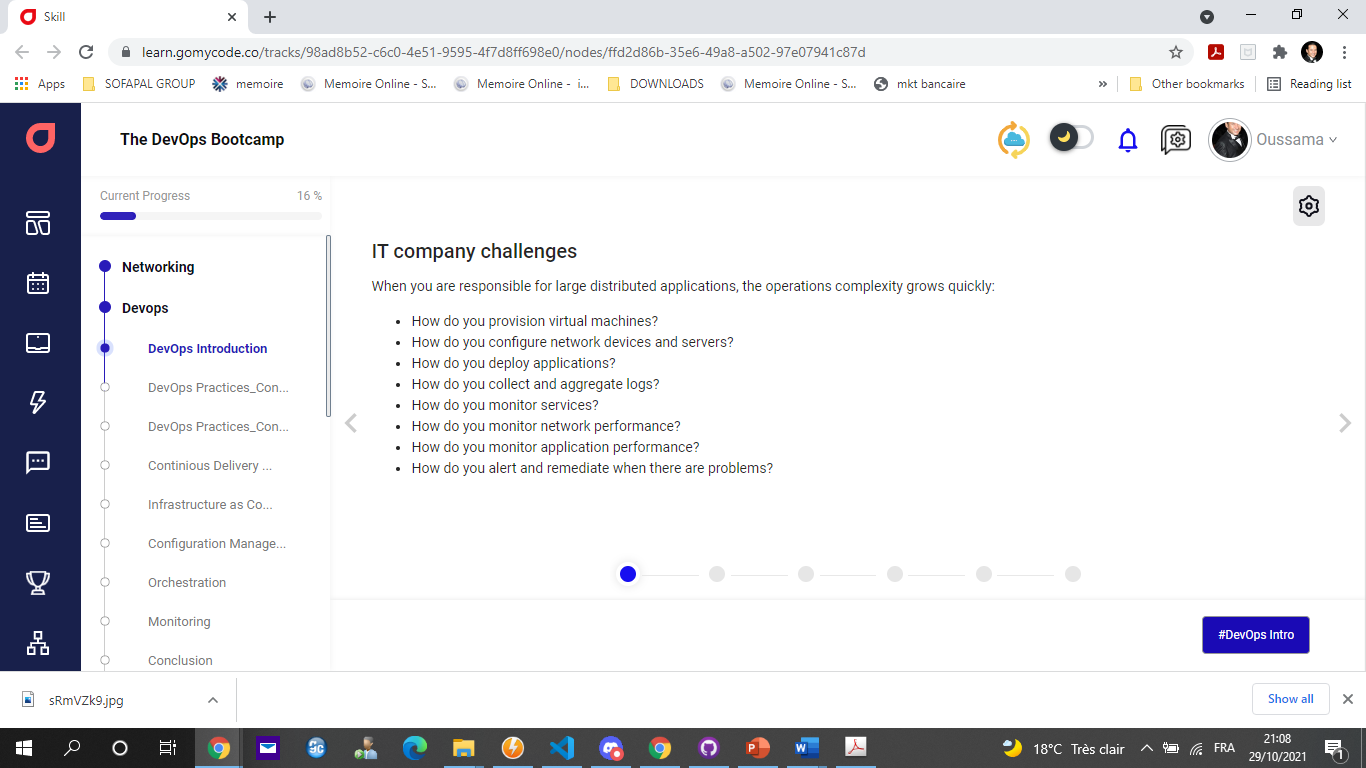
[#Subnet](https://www.techrepublic.com/article/subnet-a-class-a-network-with-ease/) [#Troubleshooting](https://quizizz.com/admin/quiz/58e6a2bf30df97b93b54ec32/network-troubleshooting?msftTeamsAuthNotify=false)

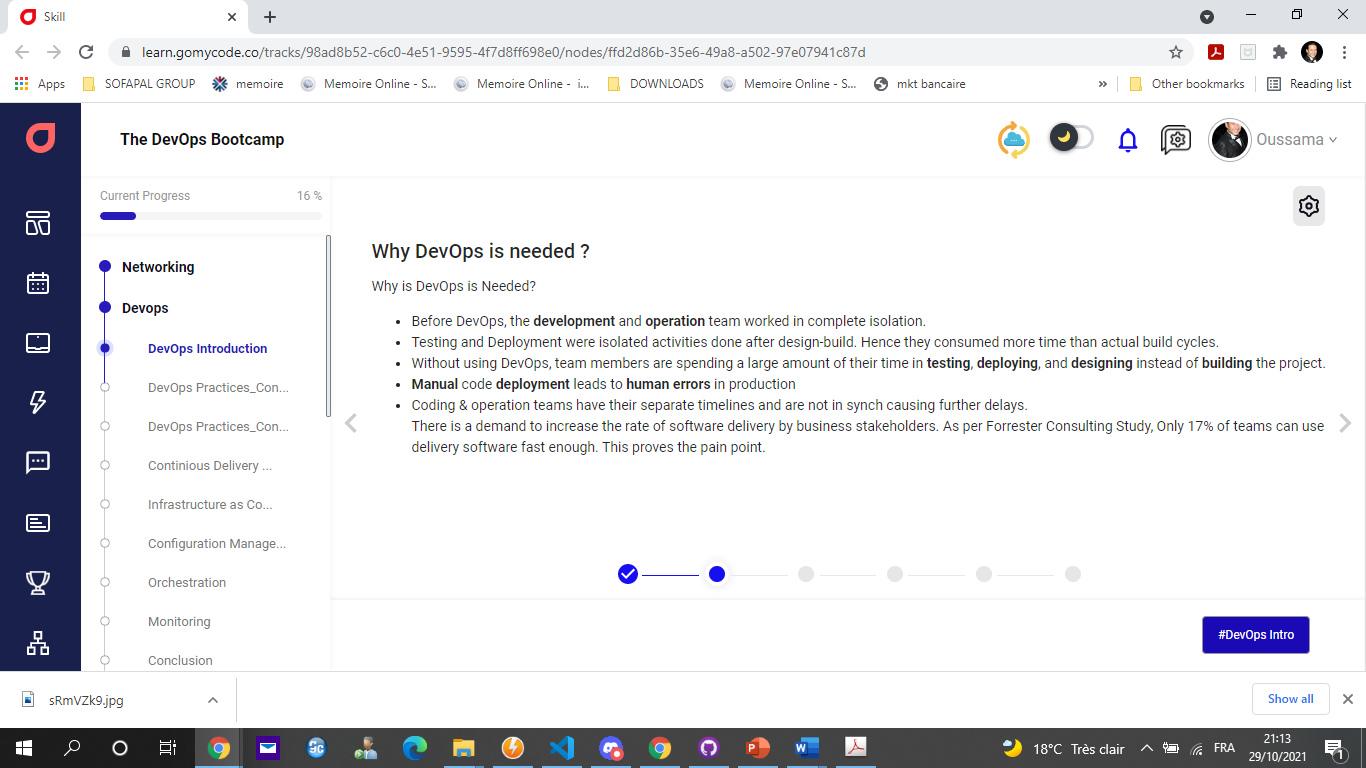
Recap

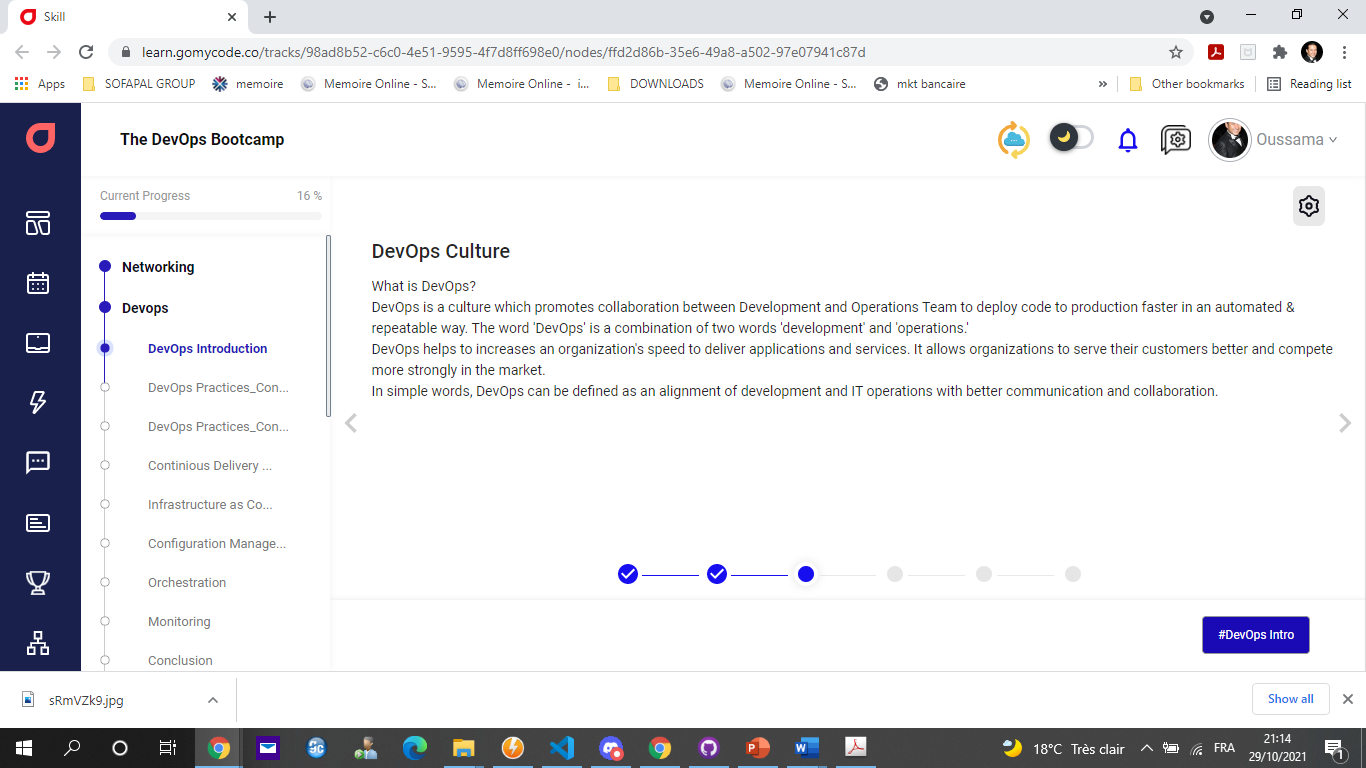
In this chapter we learnt about:

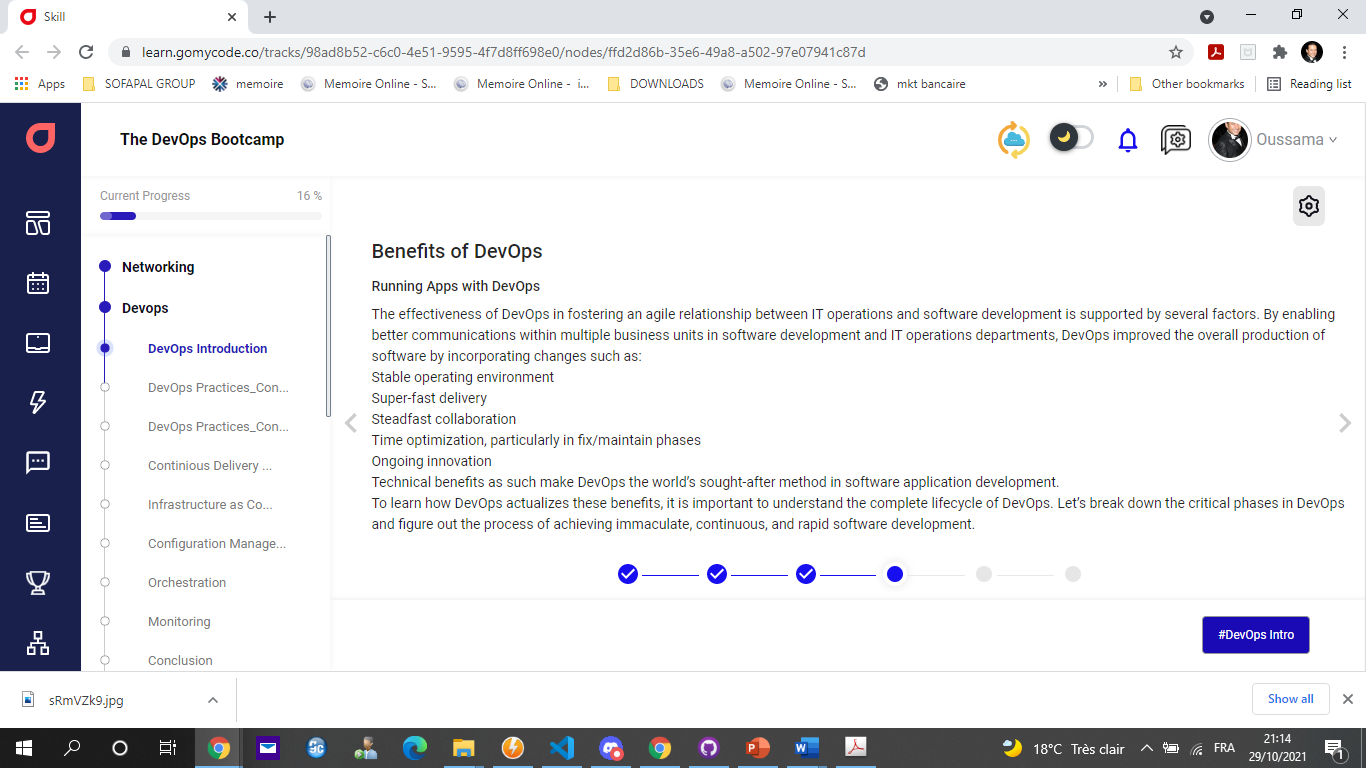
* The different type of network depend to geographic area .
* The OSI model and Layers
* The differents network class (A,B,C)
* The subnet and addressage
* The DHCP, NAT and troubleshooting

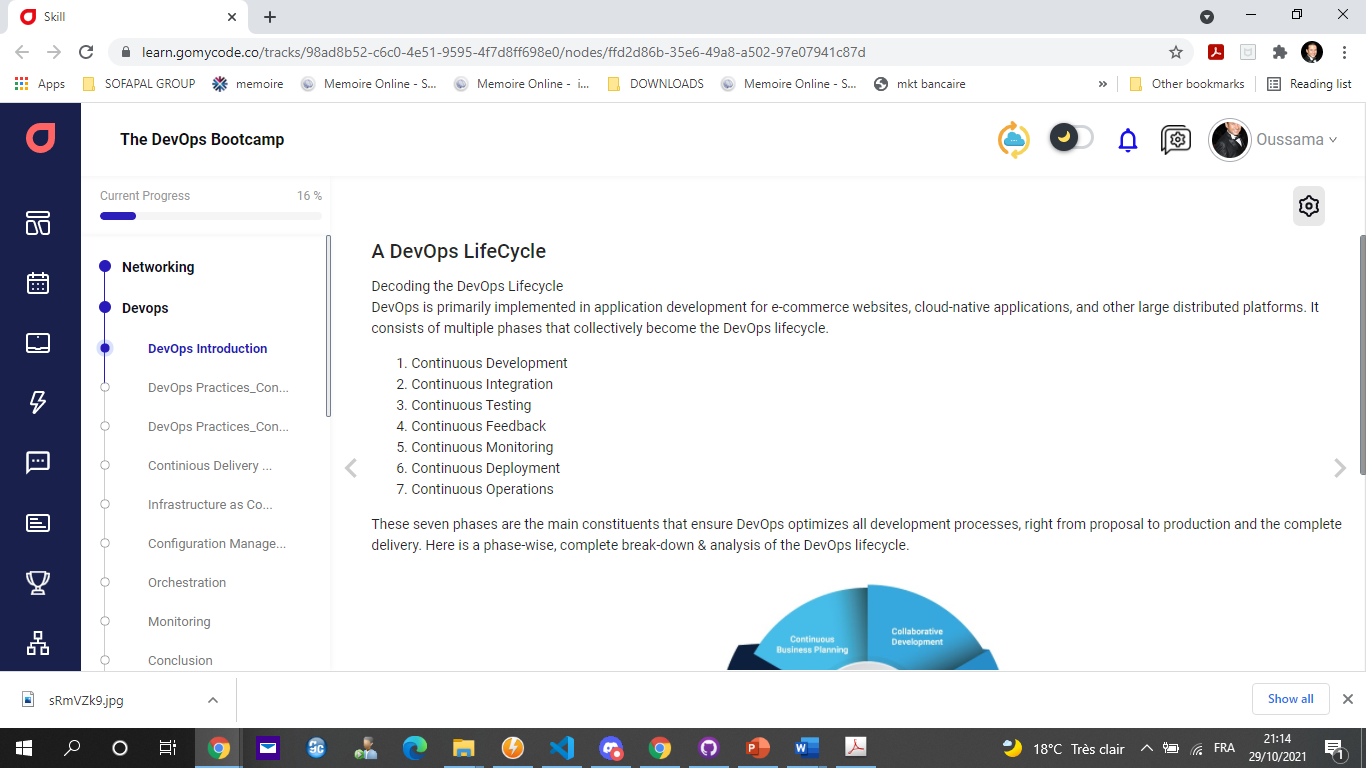
**II-Devops**

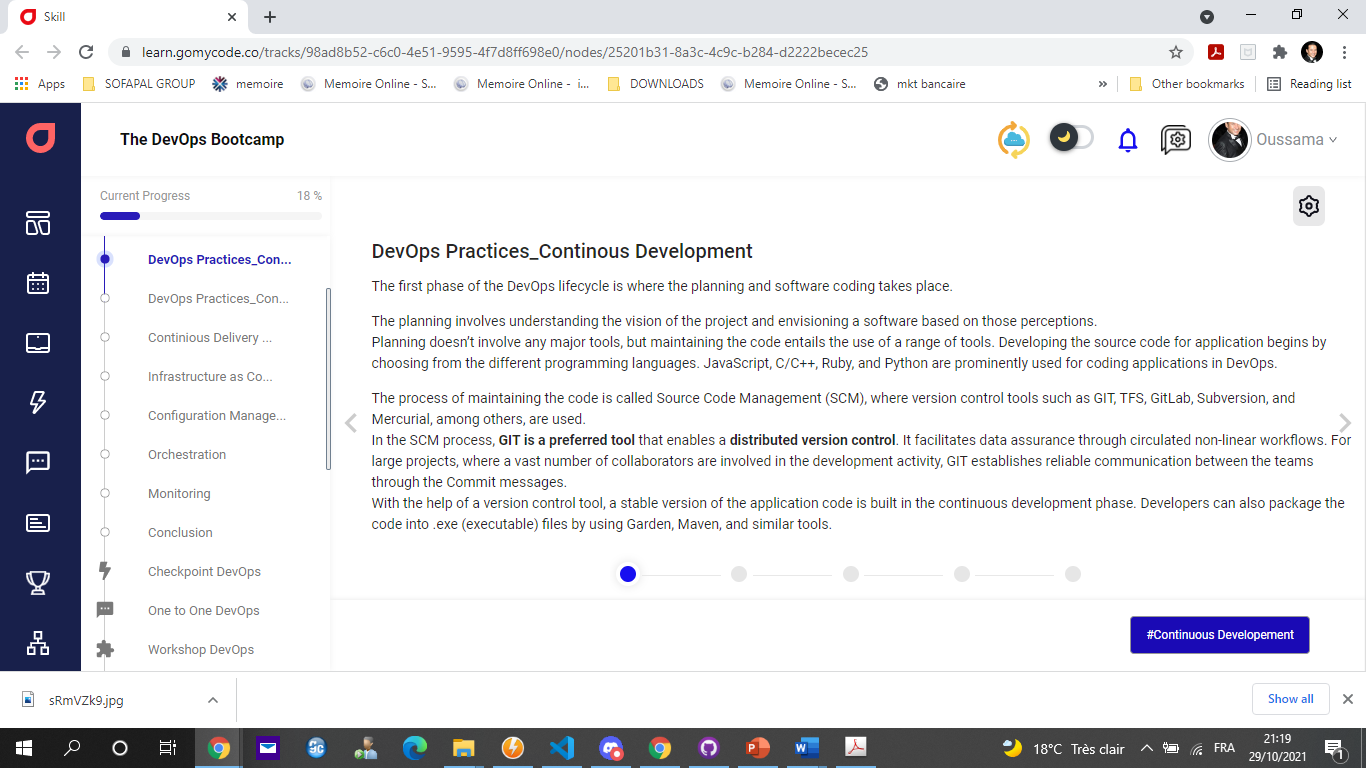












DevOps Practices\_Continous Development

The main benefits of continuous development include:

**Faster Delivery of New Feature**  
With continuous development, new features are developed and deployed quickly with automated processes in place. As a result, the speed that companies are able to test new iterations is drastically improved. Automated tests quickly pinpoint incompatibilities so bugs can be identified and addressed almost immediately. By releasing new features rapidly, companies are able to generate a return on investment considerably faster than they would using other deployment methods.

**Better Quality Product**  
Continuous development relies on automated tests and user behavior feedback to improve the software on an iterative basis. As a result, those using this approach can respond to feedback much quicker; immediately rolling back the deployment or implementing the necessary fixes. While methods like the waterfall approach require software developers to assume which features are important to the user, continuous development uses feedback and data to best determine the priority of tasks and projects.

**Risk Avoidance**  
Continuous development allows developers to avoid the risk of launching large unsuccessful projects by introducing new concepts in a lean and agile manner. Small iterations can be implemented and tested quickly, and easily pulled back without significant impact if the need arises.

**Less Resource Requirement**  
Large updates can require a considerable amount of labor in coding, deploying and testing. Continuous development relies on the implementation of automated processes to increase overall efficiency and decrease the amount of time and labor that is necessary to properly deploy new software features.

**Increased Productivity**  
Since the feedback loop is expedited, developers are given a full development workflow. By receiving continuous feedback from various stakeholders throughout the process; developers are always provided with further tasks to better optimize the software.

\*

# DevOps Practices\_Continous Development

Please notice that:  
Based on microservice architecture , Devops and continuous development help you to quick implement and maintain your application.  
You can use multiple programming languages at the same time to develop your application when your application is based on microservice architecture.  
Rest API, Message broker are the two methods to assure the communication between microservices.  
Git is a distributed source control management while SVN is a centralized source control management, for more details visit this [link](https://medium.com/faun/centralized-vs-distributed-version-control-systems-a135091299f0)

DevOps Practices\_Continous Development

We will find as below the best frameworks for microservices.

| **Framework** | **Documentation** |
| --- | --- |
| Spring boot with spring cloud | <https://spring.io/projects/spring-boot> |
| GoMicro(Golang Microservices Framework) | <https://github.com/micro/go-micro> |
| Flask (Python microservices framework) | <https://palletsprojects.com/p/flask/> |
| Molecular(Node js microservices framework) | <https://moleculer.services/> |
| Quarkus microservices framework | <https://quarkus.io/> |
| [Swoft](https://github.com/swoft-cloud/swoft) (PHP microservices framework) | <https://en.swoft.org/> |

\*

# DevOps Practices\_Continous Integration

The source code gets modified **several times**, and these frequent changes happen on a weekly or a daily basis. Code integration, the next phase, is the core of the entire DevOps lifecycle. In continuous integration, new codes that support **add-on functionalities are built and integrated into the existing code.**

In this phase, **bugs in the source code are detected early on**. To generate new code that brings more functionalities to the application, developers run tools for **unit testing, code review, integration testing, compilation, and packaging**.

The continuous integration of this new code into the existing source code helps reflect the changes that end-users would experience with the updated code.

**Jenkins** is popularly used as a reliable DevOps tool for procuring the updated source code and constructing the build into.exe format. These transitions occur seamlessly, and the updated code is packaged and proceeded to the next phase, which is either the production server or the testing server.

# DevOps Practices\_Continous Integration

* Benefits of a Continuous Integration Process

The practices mentioned above allow the development team to focus more on planning and developing the applications they work on rather than on working on the underlying tools and processes or losing time on manual release processes or deployments. It ensures end users can quickly benefit from higher quality new features.

* Improve Developer Productivity

Developers can focus on coding as they are provided a full development workflow. Developers can continuously receive feedback for their work from various stakeholders: customers, QA, product managers, and others.

* Automate the Software Release Process

Developers can automate the building, testing and release process, making you dev team more efficient and cohesive.

* Identify Bugs

Developers can quickly identify bugs in the code and address them early on through various tests that have been put in place prior to deployment.

* Deliver Updates Faster

Continuous Delivery and Continuous Deployment provides the dev team with the capability to deliver applications and updates frequently and easily with a click of a button.

<https://www.guru99.com/top-20-continuous-integration-tools.html>

# DevOps Practices\_Continous Integration

| **Tools** | **Documentation site** | **Price** | **Features** |
| --- | --- | --- | --- |
| Jenkins | <https://www.jenkins.io/> | Open source/free | 1. Continuous Integration and Continuous Delivery |
|  |  |  | 2. Easy Installation |
|  |  |  | 3. Easy configuration |
|  |  |  | 4. Distributed |
|  |  |  | 5. Extensible |
|  |  |  | 6. Plugins |
|  |  |  |  |
| Gitlab CI | <https://gitlab.com/> | community and enterprise edition | 1. secure registry for docker images |
|  |  |  | 2. integrate git svc , build and deployment tools |
|  |  |  |  |
| Azure DevOps | <https://azure.microsoft.com/en-us/services/devops/> | enterprise edition | 1. Define a CI/CD pipeline and manage releases with |
|  |  |  | 2. multiple environments using [Azure Pipelines](https://azure.microsoft.com/en-us/services/devops/pipelines/). |
|  |  |  | 3. Target any service on Azure including Azure Kubernetes Service (AKS), Azure Virtual Machines, or Azure Functions. |
|  |  |  | 4. Automate workflows with [GitHub Actions](https://github.com/features/actions) |
|  |  |  | 5. Host [Jenkins](https://docs.microsoft.com/en-us/azure/jenkins/overview/) workloads, extend existing configuration, or simplify CI/CD with Jenkins plug-ins for Azure. |
|  |  |  |  |
| TeamCity | <https://www.jetbrains.com/teamcity/> | Free | 1. Extensibility and Customization |
|  |  |  | 2. Provides better code quality for any project |
|  |  |  | 3. Project level cloud profiles |
|  |  |  | 4. Comprehensive VCS integration |

\*

# DevOps Practices\_Continious Delivery and Continuous Deployment

Conventionally, the phase of continuous deployment takes place before continuous monitoring. But, developers make sure that this phase is always active in the DevOps lifecycle, especially after the application goes live and starts receiving high traffic.

In this phase, the finalized application code is deployed to the production servers. Configuration Management is a key process in this phase, and it carries out the precise deployment of application code on all servers. Consistency in the application’s performance and functional conditions is established and curated. Code is released to the servers, updates are scheduled for all servers, and these configurations are kept consistent throughout the production process. Ansible, Puppet, and Chef are some of the effective **DevOps tools** used for Configuration Management, where they frequently execute the quick and continuous deployment of new code.

Containerization tools are used to achieve continuous deployment through the Configuration Management process. Vagrant, a containerization tool, develops coherence in different environments - from development and testing to staging and production. Similarly, the scalability of continuous deployment is handled by tools like Docker. These tools nullify all sorts of production failures and system errors by replicating and packaging the software couplings from testing, staging, and development phases. Ultimately, the application runs smoothly on different computers.

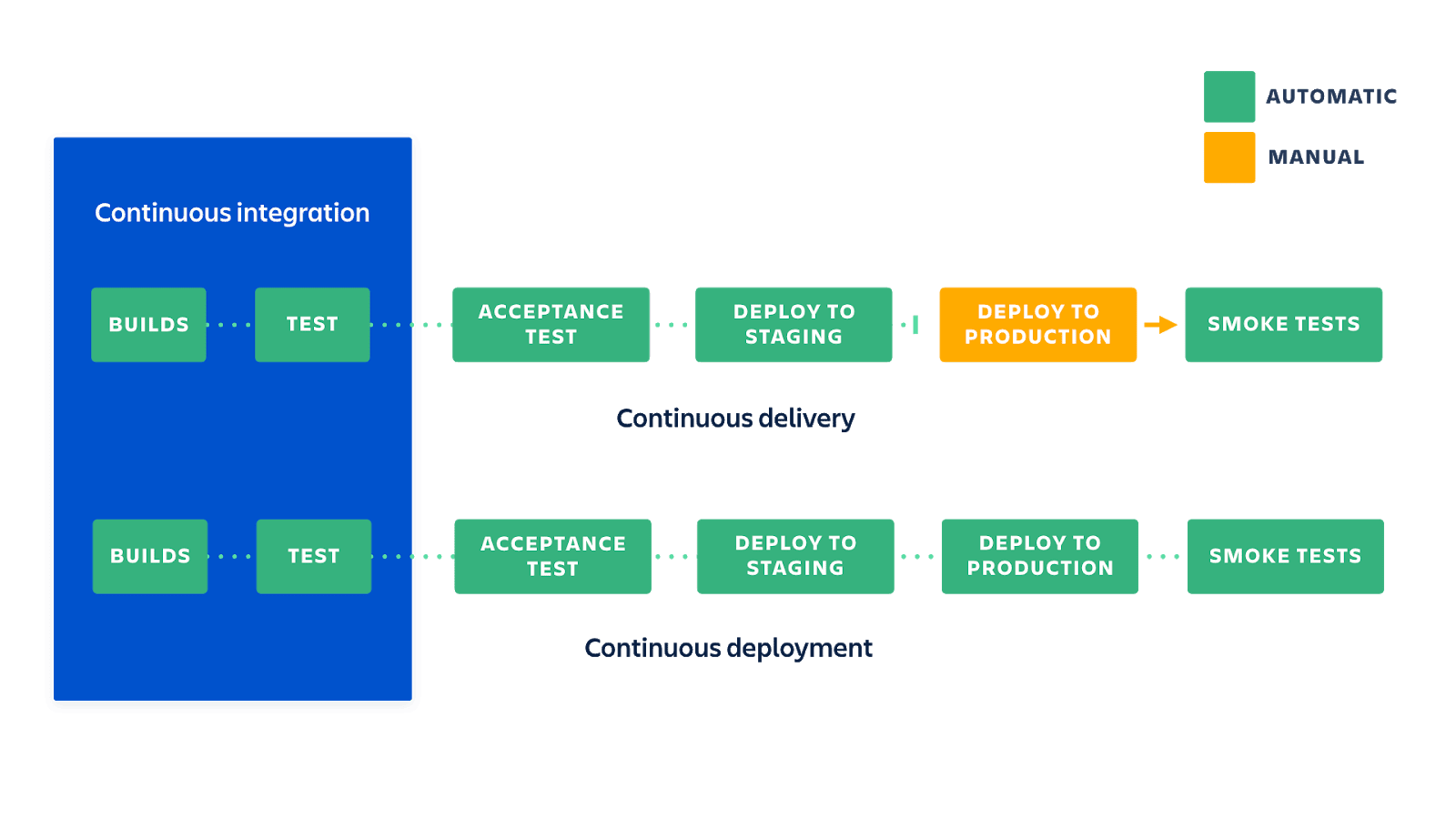
<https://www.softwaretestinghelp.com/best-continuous-delivery-tools/>

<https://www.g2.com/categories/continuous-delivery>

<https://xebialabs.com/the-ultimate-devops-tool-chest/continuous-delivery/>

# DevOps Practices\_Continous Integration vs Delivery vs Deployment

The main difference between continuous delivery and continuous deployment is the deploy to production step are manually for continuous delivery and automatic for continuous deployment.



\*

# Infrastructure as Code

* [Wikipedia defines IaC](https://en.wikipedia.org/w/index.php?title=Infrastructure_as_code&oldid=903249795) as follows :

Infrastructure as code is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

* As far as definitions go, this one isn’t bad, but it’s somewhat wordy. Let’s try and rewrite a simpler version:

Infrastructure as code (IaC) means to manage your IT infrastructure using configuration files.

<https://medium.com/microservices-architecture/top-10-microservices-framework-for-2020-eefb5e66d1a2>

<https://www.thorntech.com/2018/04/15-infrastructure-as-code-tools/>

<https://testmozusercontent.com/q/1245195/student?question_id=40421128>

# Infrastructure as Code\_The Pain of Managing IT Infrastructure

Historically, **managing IT infrastructure was a manual process**. People would physically put servers in place and configure them. Only after the machines were configured to the correct setting required by the OS and applications would those people deploy the application. Unsurprisingly, **this manual process would often result in several problems**.

The first big problem is **cost**. You’d have to hire many professionals to perform the necessary tasks at each step of the process, from network engineers to hardware maintenance technicians. All of those people need to be paid, obviously, but they also need to be managed. That leads to more management overhead and adds more complexity to communication inside the organization. The result? Dollars go away. And we didn’t even mention building and maintaining your own data centers, which would increase the costs by orders of magnitude.

The next big problems are **scalability and availability**. But in the end, it all comes down to speed. Since manual configuration is so slow, applications would often struggle with spikes in access, while the system administrators would be desperately trying to set up servers to manage the load. This necessarily impacts availability. If the organization didn’t have backup servers or even data centers, then the application could be unavailable for long periods.

Last but not least on our list of problems comes **inconsistency**. If you have several people manually deploying configurations, discrepancies aren’t going to be unavoidable.

# Infrastructure as Code\_Infrastructure as Code Benefits

You’ve just read about the problems caused by a manual approach to infrastructure management. We’ve told you how cloud computing is a solution to some of those problems, but not all. Then, we wrapped up by arguing that IaC is the final piece of the puzzle.

Now we’re going to dive into some of the benefits your organization will reap by adopting an IaC solution:

**Speed**

The first significant benefit IaC provides is speed. Infrastructure as code enables you to quickly set up your complete infrastructure by running a script. You can do that for every environment, from development to production, passing through staging, QA, and more.  
IaC can make the entire software development life cycle more efficient.

**Consistency**

Manual processes result in mistakes, period. Humans are fallible. Our memories fault us. Communication is hard, and we are in general pretty bad at it. As you’ve read, manual infrastructure management will result in discrepancies, no matter how hard you try. IaC solves that problem by having the config files themselves be the single source of truth. That way, you guarantee the same configurations will be deployed over and over, without discrepancies.

**Accountability**

This one is quick and easy. Since you can version IaC configuration files like any source code file, you have full traceability of the changes each configuration suffered. No more guessing games about who did what and when.

# Infrastructure as Code\_Infrastructure as Code Tools

[**Terraform**](https://www.terraform.io/) is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

The key features of Terraform are:

* Infrastructure as Code: Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.
* Execution Plans: Terraform has a "planning" step where it generates an execution plan. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.
* Resource Graph: Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform builds infrastructure as efficiently as possible, and operators get insight into dependencies in their infrastructure.
* Change Automation: Complex changesets can be applied to your infrastructure with minimal human interaction. With the previously mentioned execution plan and resource graph, you know exactly what Terraform will change and in what order, avoiding many possible human errors

# nfrastructure as Code\_Infrastructure as Code Tools:

**AWS CloudFormation**

Similar to Terraform, AWS CloudFormation is a configuration orchestration tool that allows you to code your infrastructure to automate your deployments.

Primary differences lie in that CloudFormation is deeply integrated into and can only be used with AWS, and CloudFormation templates can be created with YAML in addition to JSON.

CloudFormation allows you to preview proposed changes to your AWS infrastructure stack and see how they might impact your resources, and manages dependencies between these resources.  
To ensure that deployment and updating of infrastructure is done in a controlled manner, CloudFormation uses Rollback Triggers to revert infrastructure stacks to a previous deployed state if errors are detected.

You can even deploy infrastructure stacks across multiple AWS accounts and regions with a single CloudFormation template. And much more.

# Infrastructure as Code\_Infrastructure as Code Tools

**Azure Resource Manager and Google Cloud Deployment Manager**

If you’re using Microsoft Azure or Google Cloud Platform, these cloud service providers offer their own IaC tools similar to AWS CloudFormation.

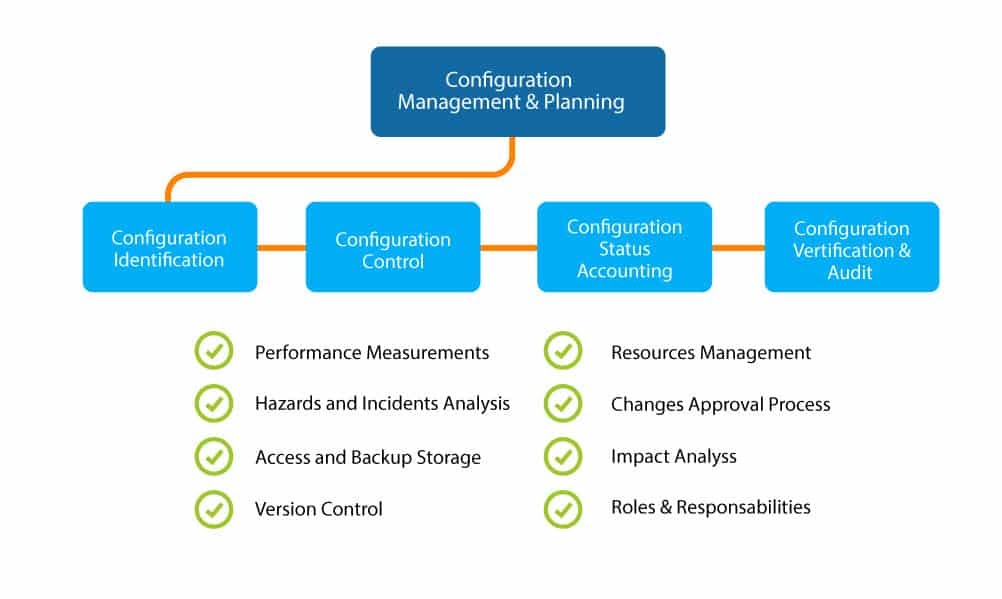
[**Azure Resource Manager**](https://azure.microsoft.com/en-us/features/resource-manager/) allows you to define the infrastructure and dependencies for your app in templates, organize dependent resources into groups that can be deployed or deleted in a single action, control access to resources through user permissions, and more.

[**Google Cloud Deployment Manager**](https://cloud.google.com/deployment-manager/) offers many similar features to automate your GCP infrastructure stack. You can create templates using YAML or Python(jinja2), preview what changes will be made before deploying, view your deployments in a console user interface, and much more

\*

# Configuration Management

Configuration Mnagement is the discipline of ensuring that all software and hardware assets which a company owns are known and tracked at all times—any future changes to these assets are known and tracked. You can think of configuration management like an always up-to-date inventory for your technology assets, a single source of truth.



# Software Configuration Management Tasks

* Identification

– tracking multiple versions to enable efficient changes

* Version control

– control changes before and after release to customer

* Control change

– authority to approve and prioritize changes

* Configuration auditing

– ensure changes made properly

* Reporting

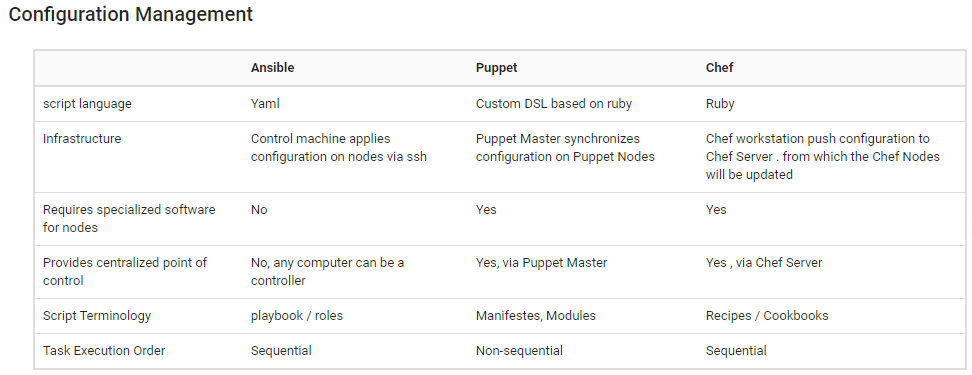
– tell others about changes made

<https://www.toptal.com/devops/interview-questions>

<https://hackr.io/blog/devops-interview-questions>

<https://www.ibm.com/cloud/learn/application-modernization>

<https://newrelic.com/resources/ebooks/enterprise-guide-continuous-application-modernization>



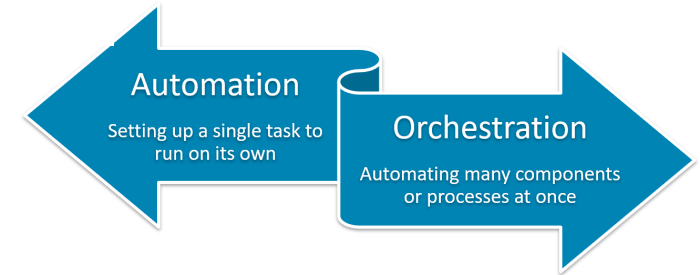
\*

# Orchestration

Orchestration is the automated configuration, management, and coordination of computer systems, applications, and services. Orchestration helps IT to more easily manage complex tasks and workflows.

IT teams must manage many servers and applications, but doing so manually isn’t a scalable strategy. The more complex an IT system, the more complex managing all the moving parts can become. The need to combine multiple automated tasks and their configurations across groups of systems or machines increases. That’s where orchestration can help.

[**Automation**](https://www.redhat.com/en/topics/automation) and [**orchestration**](https://www.redhat.com/en/topics/automation/what-is-orchestration) are different, but related concepts. Automation helps make your business more efficient by reducing or replacing human interaction with IT systems and instead using software to perform tasks in order to reduce cost, complexity, and errors.



In general, automation refers to automating a single task. This is different from orchestration, which is how you can automate a process or workflow that involves many steps across multiple disparate systems. When you start by building automation into your processes, you can then orchestrate them to run automatically.

<https://devopscube.com/devops-tools-for-infrastructure-automation/>

<https://www.plutora.com/ci-cd-tools/orchestration-scheduling-tools>

# Orchestration tools:

Today’s IT brings complex deployments and challenges. You’ve got to deal with clustered applications, multiple datacenters, public, private, and hybrid clouds, and applications with complex dependencies. You need a tool that can orchestrate your processes simply and ensure that all tasks happen in the proper order.

[Cloud](https://www.redhat.com/en/topics/cloud) orchestration can be used to provision or deploy servers, assign storage capacity, create virtual machines, and manage networking, among other tasks. There are many different orchestration tools that can help you with cloud orchestration. [Red Hat® Ansible® Automation Platform](https://www.redhat.com/en/technologies/management/ansible) is one option.  
If you’re after container orchestration, [Kubernetes](https://www.redhat.com/en/topics/containers/what-is-kubernetes) is an [open source](https://www.redhat.com/en/topics/open-source/what-is-open-source-software) platform that automates [Linux container](https://www.redhat.com/en/topics/containers/whats-a-linux-container) operations. It eliminates many of the manual processes involved in deploying and scaling containerized applications. There are also managed services available for orchestrating containers.

# Orchestration

Orchestration benefits :

* Decreasing IT costs – or better using the budget towards innovation and new projects.
* Decreasing friction among teams.
* Increasing – and perhaps improving – productivity.
* Standardizing processes and products across the spectrum so that they are more consistent and reliable.

\*

# Monitoring

Monitoring the performance of an application is of key importance for [application developers](https://www.cuelogic.com/blog/should-you-hire-a-fullstack-developer-or-a-devops). In this phase, developers record data on the use of application and continuously monitor each functionality. “Server not reachable” or “low memory” are some of the common system errors resolved in this phase.

Continuous monitoring helps in sustaining the availability of services in the application. It also determines the threats and root causes of recurring system errors. Security issues get resolved and problems are automatically detected and fixed.

Compared to the software development teams, the IT operations teams are more involved in this phase. Their role is pivotal in supervising user activity, checking the system for unusual behavior, and tracing the presence of bugs.

Sensu, ELK Stack, NewRelic, Splunk and Nagios are the key DevOps tools used in continuous monitoring. These tools enable complete control in overseeing the performance of the system, the production server, and the application. The operations team can actively engage in increasing the reliability and productivity of the applications with the help of these tools.

When major issues are detected in this phase, the application is swiftly rerun through all the earlier phases of the DevOps lifecycle. That is how finding a resolution to all sorts of issues becomes faster in this phase.

<https://dzone.com/articles/devops-tools-for-monitoring>

<https://www.toptal.com/devops/interview-questions>

# Recap

In this chapter we learn about:

* The DevOps cycle
* The continuous development logic and tools
* The continuous integration logic and tools
* The continuous delivery and deployment logic and tools
* The continuous deployment tools
* The continuous monitoring tools

III-**CI/CD**

# Continuous Integration

# CI-defined

“Continuous Integration is a software development practice where members of a team integrate their work frequently,  
usually each person integrates at least daily - leading to multiple integration per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible”

**Martin Fowler**

# What does it really mean ?

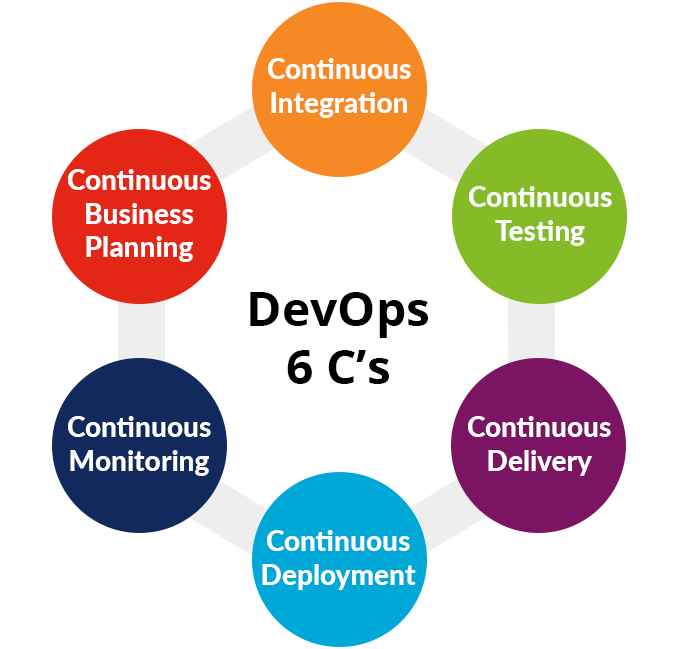
At a regular frequency (ideally at every commit), the system is:

**Integrated**  
All changes up until that point are combined into the project.

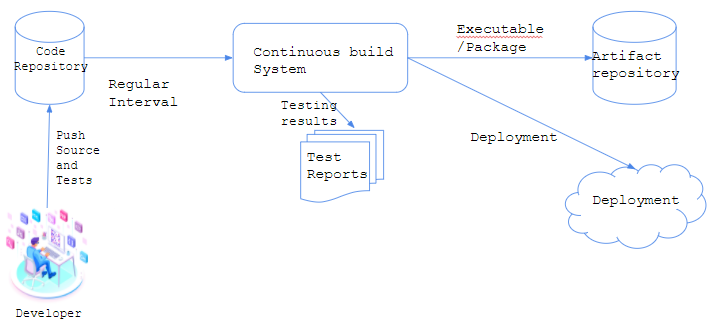
**Built**  
The code is compiled into an executable or package.

**Tested**  
Automated test suites are run.

**Archived**  
Versioned and stored so it can be distributed as is, if desired.

**Deployed**  
Loaded onto a system where the developers can interact with it.  


# Continuous Integration workflow



Developers contribute to create softwares, for that he need a source control version like Git to manage commits and versions.After pushing code to the git repository a continuous build system will launch a build on a regular time interval or using a webhooks.  
Continuous build system responsible to create artifact as a package or an executable binary that we need to register/saved on an artifactory with a specific version or a tag, also he make a list of testing scenarios (unit test or e2e test) that we need to generate a test reports for each build.  
and finally CBS can make a deployment to a web server for example or to a kubernetes cluster.

[#Continuous Integration](https://www.jenkins.io/doc/book/pipeline/) [#Gitlab CI/CD pipelines](https://docs.gitlab.com/ee/ci/)

Improving Your Productivity:

Continuous integration can help you go faster:

* Detect build breaks sooner: when a developer make a new change to the git repository , with continuous integration we launch a new build to test the effects of the new piece on our software , so if something go wrong we can detect who is responsible for that conflicts and we can fix it quickly.
* Report failing tests more clearly: Every build CI system generate a builds reports , so if something go wrong  
  we can generate a report and mention causes.
* Make progress more visible: Every commit we add the new changes/pieces to our software that make progress more visible depend to our sprint (Agil) or our goals.

Continuous Integration Tools:

* Code Repositories :

[SVN](https://subversion.apache.org/) , [Mercurial](http://mercurial-scm.org/) , [Git](https://git-scm.com/)

* Continuous Build Systems:

[Jenkins](https://www.jenkins.io/) , [Bamboo](https://www.atlassian.com/software/bamboo) , [Cercle-CI](https://circleci.com/), … .

* Continuous Test Frameworks:

[Junit](https://junit.org/junit5/) , [CppUnit](https://freedesktop.org/wiki/Software/cppunit/), [PHPUnit](https://phpunit.de/) ...

* Artifact Repositories:

[Nexus](https://www.nexusmods.com/) , [Artifactory](https://jfrog.com/artifactory/), [Archiva](https://archiva.apache.org/)

# Continuous Testing

Today, software development is all about Agile and Continuous Delivery practices that aim to keep up with changing business needs and requirements. How can we ensure quality when the software requirements keep changing?

**What is Continuous Testing?**  
Continuous Testing is defined as a software testing type that involves a process of testing early, testing often, test everywhere, and automate. It is a strategy of evaluating quality at every step of the Continuous Delivery Process.  
The goal of Continuous Testing is test early and test often. The process involves stakeholders like Developer, DevOps, QA and Operational system.

[#Continuous Testing](https://www.guru99.com/continuous-testing.html)

Continuous Testing Tools

1. QuerySurge

[QuerySurge](https://bit.ly/2Ni99Gd) is the smart data testing solution that is the first-of-its-kind full DevOps solution for continuous data testing. Key features include Robust API with 60+ calls, detailed data intelligence & data analytics, seamless integration into the DevOps pipeline for continuous testing, and verifies large amounts of data quickly.  
[Start testing for free](https://bit.ly/2Ni99Gd)

1. Jenkins

Jenkins is a Continuous Integration tool which is written using Java language. This tool can be configured via GUI interface or console commands.  
Download link: [https://jenkins.io/](https://bit.ly/2Ni99Gd)

1. Travis

Travis is continuous testing tool hosted on the GitHub. It offers hosted and on-premises variants. It provides a variety of different languages and a good documentation.  
Download link: <https://travis-ci.org/>

1. Selenium

Selenium is open-source software testing tool. It supports all the leading browsers like Firefox, Chrome, IE, and Safari. Selenium WebDriver is used to automate web application testing.  
Download link: <https://www.seleniumhq.org/>

# Continuous Delivery

Continuous Delivery is a software development discipline\*\* where you build software\*\* in such a way that the software can be **released to production at any time**.  
You’re doing continuous delivery when:  
Your software is deployable throughout its lifecycle  
Your team prioritizes keeping the software deployable over working on new features  
Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them  
You can perform push-button deployments of any version of the software to any environment on demand  
You achieve continuous delivery by continuously integrating the software done by the development team, building executables, and running automated tests on those executables to detect problems. Furthermore you push the executables into increasingly production-like environments to ensure the software will work in production.

[#Continuous Delivery](https://continuousdelivery.com/)

Continuous Delivery

The key test is that a business sponsor could request that the current development version of the software can be deployed into production at a moment's notice - and nobody would bat an eyelid, let alone panic.  
To achieve continuous delivery you need:

* a close, collaborative working relationship between everyone involved in delivery (often referred to as a [**DevOps Culture**](https://martinfowler.com/bliki/DevOpsCulture.html)**)**.
* extensive automation of all possible parts of the delivery process, usually using a [**Deployment Pipeline**](https://martinfowler.com/bliki/DeploymentPipeline.html)

Continuous Delivery is sometimes confused with Continuous Deployment. Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day. Continuous Delivery just means that you are able to do frequent deployments but may choose not to do it, usually due to businesses preferring a slower rate of deployment. In order to do Continuous Deployment you must be doing Continuous Delivery.

[**Continuous Integration**](https://martinfowler.com/articles/continuousIntegration.html) usually refers to integrating, building, and testing code within the development environment. Continuous Delivery builds on this, dealing with the final stages required for production deployment.

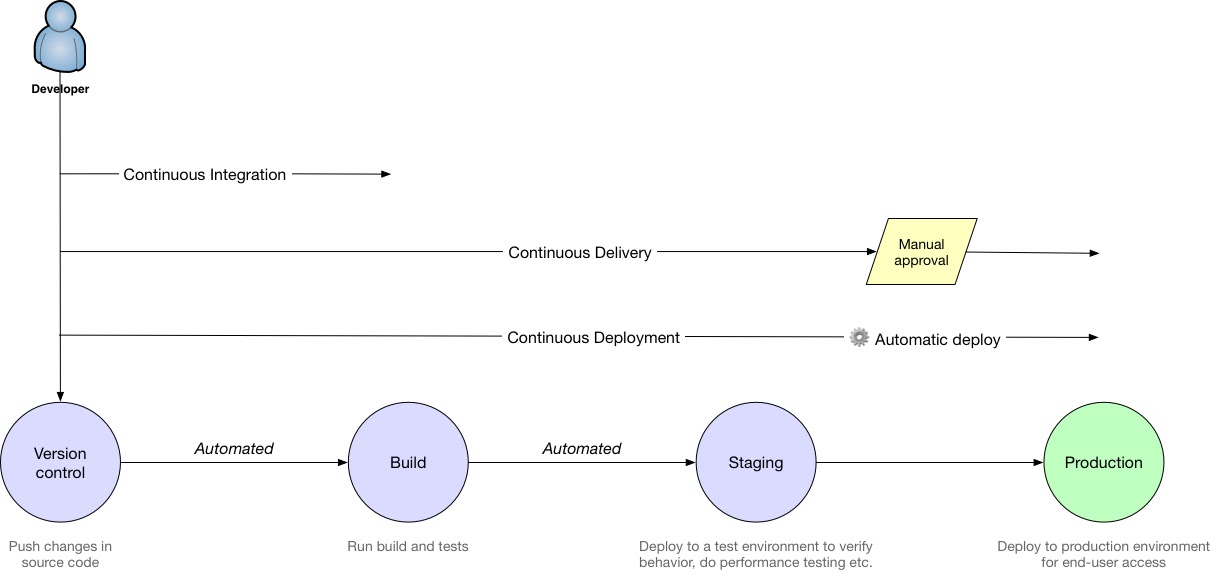
Continuous Delivery\_The Benefits

The principal benefits of continuous delivery are:

* Reduced Deployment Risk: since you are deploying smaller changes, there's less to go wrong and it's easier to fix should a problem appear.
* Believable Progress: many folks track progress by tracking work done. If "done" means "developers declare it to be done" that's much less believable than if it's deployed into a production (or production-like) environment.
* User Feedback: the biggest risk to any software effort is that you end up building something that isn't useful. The earlier and more frequently you get working software in front of real users, the quicker you get feedback to find out how valuable it really is (particularly if you use [Observed Requirements](https://martinfowler.com/bliki/ObservedRequirement.html)).

User feedback does require you to be doing continuous deployment. If you want that, but don't fancy getting new software to your entire user base, you can deploy to a subset of users. In a recent project of ours, a retailer deployed its new online system first to its employees, then to an invited set of premium customers, and finally to all customers.

Continuous Delivery



Continuous integration vs delivery vs deployment:

* **Continuous** **integration** is an automated build and test .
* **Continuous** **delivery** : continuous integration + automated deployment to test environment.
* **Continuous** **Deployment** : continuous delivery + automated deployment to production.

# Continuous Deployment

Continuous deployment is the next step of continuous delivery: Every change that passes the automated tests is deployed to production **automatically**. Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements. There are business cases in which IT must wait for a feature to go live, making continuous deployment impractical. While application feature toggles solve many of those cases, they don’t work in every case. The point is to decide whether continuous deployment is right for your company based on business needs — not on IT limitations.

[#Continuous Deployment](https://www.scaledagileframework.com/continuous-deployment/) [#Understand Gitlab](https://about.gitlab.com/blog/2018/01/22/a-beginners-guide-to-continuous-integration/) [#Quiz](https://www.fullstack.cafe/blog/15-continuous-integration-interview-questions-in-2018)

# Continuous Deployment

Traditional development practices treat deployment and release as the same activity: changes deployed to production are immediately available to users. This motivates behaviors that make applying certain design thinking practices, such as A/B Testing, hard to implement, and serve to inhibit the flow of value. Continuous deployment separates the deployment and release process.

This promotes design thinking practices and the flow of value by:

* Targeting functionality to specific customers  
  Separating deployment from release enables the organization to target customers with specific functionality, allowing the organization to assess the impact of the release before deploying functionality to all customers.
* Promoting experimentation, such as A/B Testing  
  Design thinking practices, such as A/B Testing, require the ability to present different functionality to targeted users, gathering the data that helps create designs optimized for user needs.
* Promoting small batches  
  Prior choices made in the pipeline, such as automated testing, make deploying in small batches easier.
* Releasing on business needs  
  Enterprises tend to release infrequently when deploying a release is complex and error-prone. Conversely, when deployment and release are separated, and investments are made to ensure both are automated and low-risk, enterprises can release on demand, substantially increasing business agility. For example, a release can be in production ahead of a marketing milestone, giving the organization complete flexibility in maximizing all aspects of the delivery of value.

# The Four Activities of Continuous Deployment:

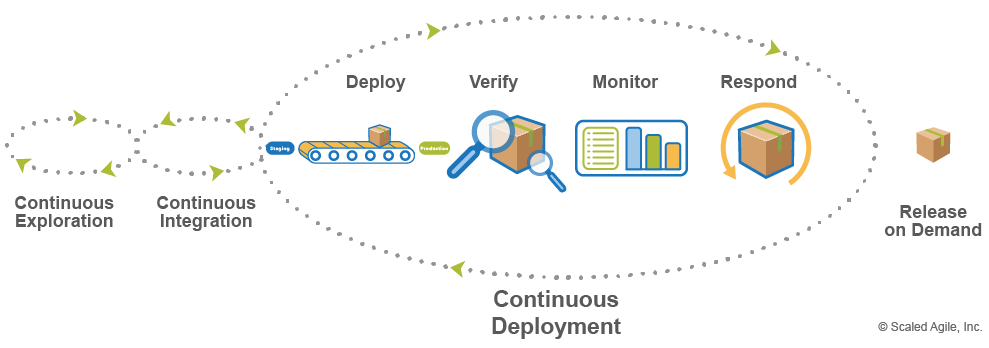
SAFe describes four activities of Continuous Deployment, as illustrated in Figure below:

**Deploy to production** describes the practices necessary to deploy a solution to a production environment.

**Verify the solution** describes the practices needed to make sure the changes operate in production as intended before they are released to customers.

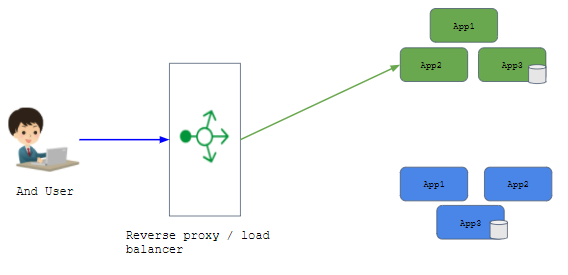
**Monitor for problems** describes the practices to monitor and report on any issues that may arise in production.

**Respond and recover** describes the practices to rapidly address any problems that happen during deployment.



# Continuous Blue/Green Deployment Strategy

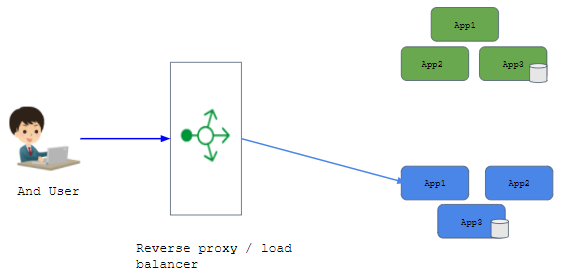
Simply, you have two identical environments (infrastructure) with the “Green” environment hosting the current production apps  
(app1 version1, app2 version1, app3 version1 for example) and a new version to installed “Blue” (app1 version2, app2 version2, app3 version3 for example):



# Continuous Blue/Green Deployment Strategy

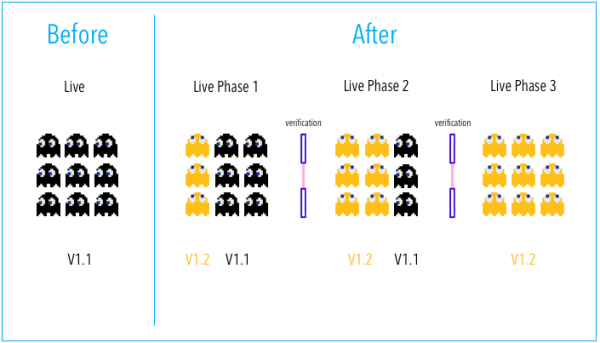
Now, when you’re ready to make a change to app2 for example and upgrade it to v2, you’d do so in the “blue environment”. In that environment, you deploy the new version of the app, run smoke tests, and any other tests (including those to exercise/prime the OS, cache, CPU, etc).

When things look good, you change the load balancer/reverse proxy/router to point to the blue  
environment:



# Continuous Blue/Green Deployment Strategy :

Canary is about deploying an application in small, incremental steps, and only to a small group of people. There are a few  
possible approaches, with the simplest being to serve only some percentage of the traffic to the new application, to a more  
complicated solutions, such as a feature toggle. A feature toggle allows you to gate access to certain features based on specific criteria (e.g., gender, age, country of origin). The most advanced feature toggle I am aware of, gatekeeper, is implemented at Facebook.



[#Continuous Deployment](https://www.scaledagileframework.com/continuous-deployment/)[#Understand Gitlab](https://about.gitlab.com/blog/2018/01/22/a-beginners-guide-to-continuous-integration/)[#Quiz](https://www.fullstack.cafe/blog/15-continuous-integration-interview-questions-in-2018)

Recap

In this chapter we learn about:

* The CI/CD cycle.
* The continuous development and Testing .
* The continuous deployment tools and strategies .
* Jenkins

[#Gitlab CI/CD pipelines](https://docs.gitlab.com/ee/ci/) [#Understand Gitlab](https://about.gitlab.com/blog/2018/01/22/a-beginners-guide-to-continuous-integration/) [#Continuous Deployment](https://www.scaledagileframework.com/continuous-deployment/)[#Continuous Delivery](https://continuousdelivery.com/) [#Quiz](https://www.fullstack.cafe/blog/15-continuous-integration-interview-questions-in-2018) [#Continuous Testing](https://www.guru99.com/continuous-testing.html)

# 4- Containers

# YAML quick start

YAML Ain’t Markup Language ([YAML](http://yaml.org/)) is a serialization language that has steadily increased in popularity over the last few years. It’s often used as a format for configuration files, but its object serialization abilities make it a viable replacement for languages like JSON. This YAML tutorial will demonstrate the language syntax with a guide and some simple coding examples in [Python](https://rollout.io/blog/python-feature-flag-guide/).  
YAML has broad language support and maps easily into native data structures. It’s also easy to for humans to read, which is why it’s a good choice for configuration.

The YAML acronym was shorthand for Yet Another Markup Language. But the maintainers renamed it to YAML Ain’t Markup Language to place more emphasis on its data-oriented features.

---

name: "GoMyCode"

course: "DevOps Introduction"

hours: 50

document-provider: true

chapters:

- networking

- devops

- ci

- containers

- kubernetes

- aws

This is a YAML file example that I’ll explain it on the next slide to understand the different types (integer, float, string , array ) supported by YAML language .

The file starts with three dashes. These dashes indicate the start of a new YAML document. YAML supports multiple documents, and compliant parsers will recognize each set of dashes as the beginning of a new one.

Next, we see the construct that makes up most of a typical YAML document: a key-value pair.  
**name** is a key that points to a string value: **GoMyCode.**

YAML supports more than just string values. The file starts with six key-value pairs.  
They have four different data types.

* **name** and **course** are strings.
* **hours** is a integer-point number.
* **document-provider** is a boolean.

You can enclose strings in single or double-quotes or no quotes at all. YAML recognizes unquoted numerals as integers or floating point.

The seventh item is an array. **chapters** has six elements, each denoted by an opening dash.

I indented the elements in **chapters** with two spaces. Indentation is how YAML denotes nesting. The number of spaces can vary from file to file, but tabs are not allowed. We’ll look at how indentation works below.

{

"name": "GoMyCode",

"course": "DevOps Introdutction",

"hourse": 50,

"document-provider": true,

"chapters": [

"networking",

"devops",

"ci",

"containers",

"kubernetes",

"aws"

]

}

[#Yaml Tutorial](https://gettaurus.org/docs/YAMLTutorial/)[#Docker Quickstart](https://docker-curriculum.com/)

# Outline Indentation and whitespace:

Whitespace is part of YAML’s formatting. Unless otherwise indicated, newlines indicate the end of a field.  
You structure a YAML document with indentation. The indentation level can be one or more spaces. The specification forbids tabs because tools treat them differently.  
Consider this document. The items inside stuff are indented with two spaces.

foo: bar

pleh: help

stuff:

foo: bar

bar: foo

Let’s take a look at how a simple python script views this document. We’ll save it as a file named foo.yaml.  
The PyYAML package will map a YAML file stream into a dictionary. We’ll iterate through the outermost set of keys and values and print the key and the string representation of each value. You can find a processor for your favorite platform here.

import yaml

if \_\_name\_\_ == '\_\_main\_\_':

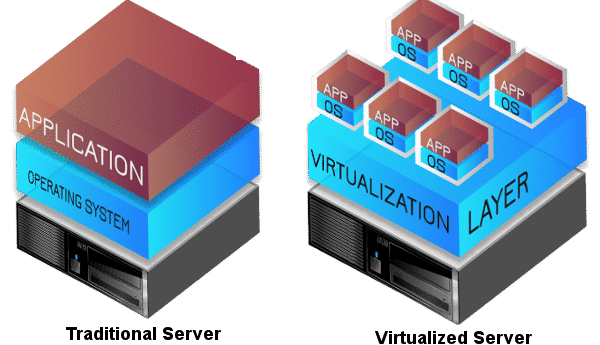
stream = open("foo.yaml", 'r')

dictionary = yaml.load(stream)

for key, value in dictionary.items():

print (key + " : " + str(value))

What is containers:Virtualization vs Docker



Instead of using physical machine or server , now with the virtualization we can use a single machine and make multiple VMs or virtual machines . These be able by using an abstraction layer / virtualization layer called “Hypervisor” after that we can multiple Vms with different operating system , the questions why we need virtualization ?  
Advantages of Virtualization:

* Minimize hardware costs.
* Multiple virtual servers on one physical hardware.
* Easily move VMs to other data centers.
* Conserve power Free up unused physical resources.
* Easier automation. Simplified provisioning/administration of hardware and software.
* Scalability and Flexibility: Multiple operating systems

[#What is a container](https://cloud.google.com/containers) [#Docker Tutorial](https://geekflare.com/docker-tutorials/)

What is containers:Virtualization vs Docker

we listed the list of advantages of virtualisation and how he make it easy to manipulate multiple operating system , portability , reduce resources consumption … despite that he has some limitation:

* Each VM requires an operating system (OS)
* Each OS requires a license(windows case).
* Each OS has its own compute and storage overhead.
* Needs maintenance, updates

What is containers

With VMs, you basically run an entire operating stack in a virtual machine. The host operating system simply acts as a foundation for VMs to run. Everything inside the virtual machines is configured individually for the apps you run in them.

Containers, on the other hand, share the underlying operating system and kernel. This means a single environment can power multiple containers more efficiently. Think of containers as a virtualization method at the process (or application) level, rather than at an OS level.

Containers are more flexible and scalable synchronistically. That is the fundamental reason they are used to power microservice-based apps for this very reason. Rather than bringing the entire system down when an update is ready for deployment, individual containers can be updated independently while maintaining system integrity and availability.

before starting use container let’s define a container as : “Containers provide a standard way to **package** your **application's code**, **configurations**, and **dependencies** into a **single object** , Containers share an operating system installed on the server and run as resource-isolated processes, ensuring quick, reliable, and consistent deployments, regardless of environment. “

The terms container appear with Linux Containers and based on two Linux technologies :

* **cgroup**: Control Groups provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behaviour.
* **namespace**: wraps a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource.

In short:

* **Cgroups** = limits how much you can use;
* **namespaces** = limits what you can see (and therefore use)

What is containers

To understand more how the container working we need to understand the technologies behind it. so we need to understand cgroups and namespaces and which resources managed by this two technologies :

* **Cgroups involve resource metering and limiting:**
  + Memory
  + CPU
  + Block I/O
  + Network
* **Namespaces provide processes with their own view of the system:**
  + Multiple namespaces:
  + Mount - isolate filesystem mount points
  + UTS - isolate hostname and domain name
  + IPC - isolate interprocess communication (IPC) resources
  + PID - isolate the PID number space
  + Network - isolate network interfaces
  + User - isolate UID/GID number spaces
  + Cgroup - isolate cgroup root directory

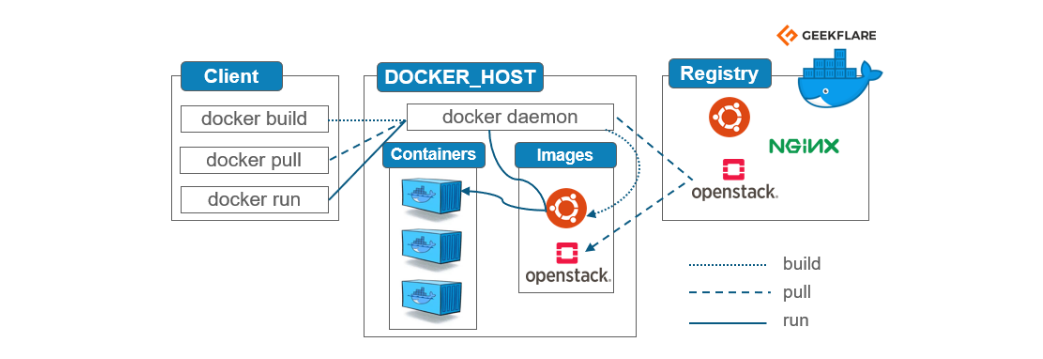
# What is containers:

To resume a container is Standardized packaging for software and dependencies that isolate apps from each other  
and Share the same OS kernel , also container is portable and works with all major Linux and Windows Server

Key Benefits of Docker Containers

* **Speed** :
  + No OS to boot = applications online in seconds
* **Portability** :
  + Less dependencies between process layers = ability to move between infrastructure
* **Efficiency**:
  + Less OS overhead
  + Improved VM density

# Docker Architecture

Below is the simple diagram of a Docker architecture.  


[#Docker architecture](https://geekflare.com/docker-architecture/) [#Docker architecture](https://k21academy.com/docker-kubernetes/docker-architecture-docker-engine-components-container-lifecycle/)

# Architecture\_Docker Engine

Docker Engine is the core part of the whole Docker system.  
Docker Engine is an application which follows client-server architecture. It is installed on the host machine.

There are three components in the Docker Engine:

**Server**  
It is the docker daemon called dockerd. It can create and manage docker images. Containers, networks, etc.

**Rest API**  
It is used to instruct docker daemon what to do.

**Command Line Interface (CLI)**  
It is a client which is used to enter [**docker commands**](https://geekflare.com/docker-commands/).

# Architecture\_Docker client and register

## **Docker Client**

Docker users can interact with Docker through a client. When any docker commands runs, the client sends them to dockerd daemon, which carries them out. Docker API is used by Docker commands. Docker client can communicate with more than one daemon.

## **Docker Registries**

It is the location where the Docker images are stored. It can be a public docker registry or a private docker registry. Docker Hub is the default place of docker images, its stores’ public registry. You can also create and run your own [**private registry**](https://geekflare.com/docker-private-registry-ubuntu/).  
When you execute docker pull or docker run commands, the required docker image is pulled from the configured registry. When you execute docker push command, the docker image is stored on the configured registry.

# Architecture: docker object

When you are working with Docker, you use images, containers, volumes, networks; all these are Docker objects.  
Let’s talk about Docker Image!  
Docker images are read-only templates with instructions to create a docker container. Docker image can be pulled from a Docker hub and used as it is, or you can add additional instructions to the base image and create a new and modified docker image. You can create your own docker images also using a [dockerfile](https://geekflare.com/dockerfile-tutorial/). Create a dockerfile with all the instructions to create a container and run it; it will create your custom docker image.  
Docker image has a base layer which is read-only, and the top layer can be written. When you edit a dockerfile and rebuild it, only the modified part is rebuilt in the top layer.

# Architecture\_Docker container

After you run a docker image, it creates a docker container. All the applications and their environment run inside this container. You can use Docker API or CLI to start, stop, delete a docker container.  
Below is a sample command to run a ubuntu docker container:

rassakra$ docker run -i -t ubuntu /bin/bash

# Architecture\_Docker volume

The persisting data generated by docker and used by Docker containers are stored in Volumes. They are completely managed by docker through docker CLI or Docker API. Volumes work on both Windows and Linux containers. Rather than persisting data in a container’s writable layer, it is always a good option to use volumes for it. Volume’s content exists outside the lifecycle of a container, so using volume does not increase the size of a container.  
You can use -v or –mount flag to start a container with a volume. In this sample command, you are using geek volume volume with geekflare container.

rassakra$ docker run -d --name geekflare -v geekvolume:/app nginx:latest