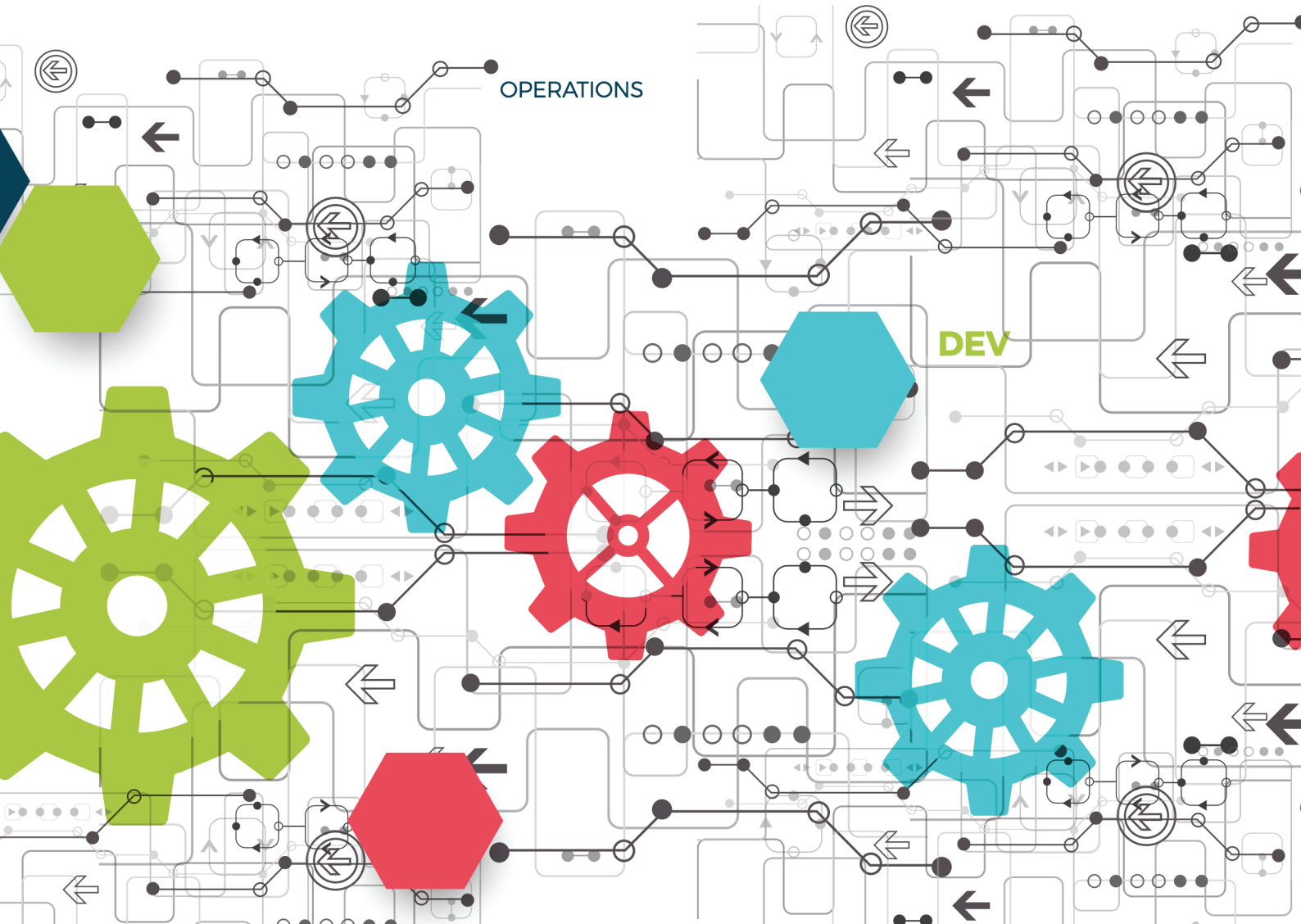
UPES

Xebia

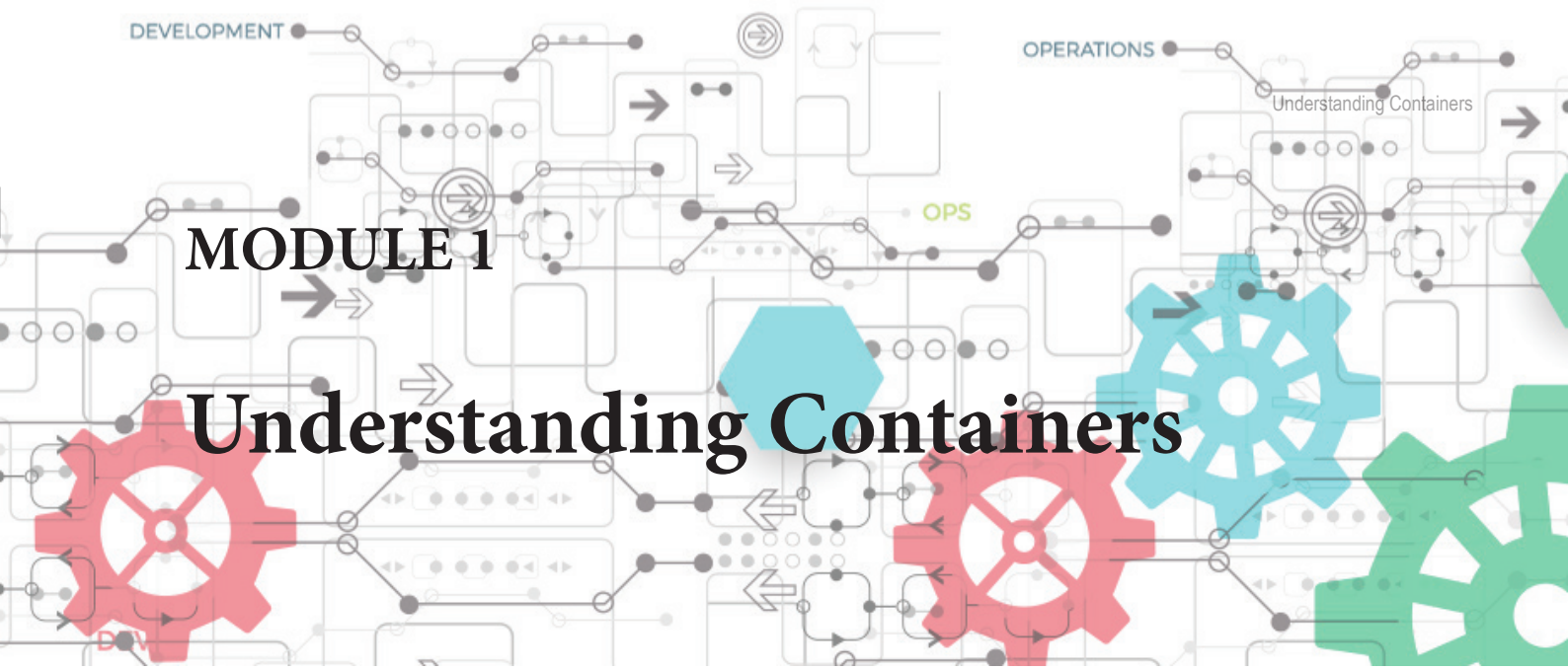
B.Tech Computer Science and Engineering in DevOps

Application Containerization

MODULE 1
Understanding Containers



Contents



MODULE 1

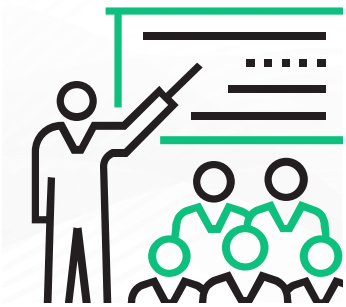
Understanding Containers

Facilitator Notes:

Welcome the participants and give them an overview of the module. Tell them that they will learn about the 'Understanding Containers' in this module.

You will learn about the 'Understanding Containers' in this module.

Module Objectives



At the end of this module, you will be able to:

- Define transporting "goods" analogy
- Explain virtualisation and comparison with virtual machines
- Describe containerization platform, images and runtime
- Identify the journey of container technology
- Discuss the chroot system call
- Explain FreeBSD Jails, LinuX Containers (LXC) and Docker

Facilitator Notes:

Explain the module objectives to the participants.

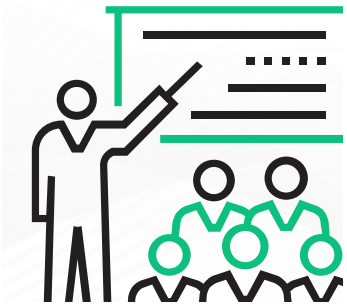
You will be informed about the module objectives.

At the end of this module, you will be able to:

- Define transporting "goods" analogy
- Explain virtualisation and comparison with virtual machines
- Describe containerization platform, images and runtime
- Identify the journey of container technology
- Discuss the chroot system call
- Explain FreeBSD Jails, LinuX Containers (LXC) and Docker

Module Topics

Let us take a quick look at the topics that we will cover in this module:



- Transporting “goods” analogy
- Containerization platform, images and runtime
- Comparison with virtual machines
- The chroot system call
- FreeBSD Jails
- LinuX Containers (LXC)
- Docker

Facilitator Notes:

Inform the participants about the topics that they will be learning in this module.

You will learn about the following topics in this module:

- Transporting “goods” analogy
- Containerization platform, images and runtime
- Comparison with virtual machines
- The chroot system call
- FreeBSD Jails
- LinuX Containers (LXC)
- Docker

1.1 Transporting Goods Analogy

Facilitator Notes:

Explain containers with the transporting goods analogy.

Before 1956, shipping industry was in haphazard because of the way goods were being shipped. Goods were carried by a swarm of workers who carry the goods on their back, climb up the gangplank and stacking them on the ship. The ship holds the all varieties of goods which can be imagined, ranging from tough to fragile. And they were kept at the same place until the ship gets unloaded or goods got deployed on the dock. By then, most of the goods get damaged, perishable goods get perished. Moreover, all of these goods were picked by hands irrespective of their fragility. The results of such strategy were: shipping goods were a disastrous, manual effort, such shipping demerits deter many industries to engage in overseas trade.

Then came containers, due to which goods of the same category were safely stacked inside a container, which led to less damaging of goods, easy shipping, reduction in cost, big cranes loading-unloading the containers with dexterity. Containers changed the overall shipping industry model which attracted other industries to participate in the global trade.

Now, if you compare this analogy with software deployment you can identify the similar flaws, like various services are running on a single server hampering each other because of shared space, deploying them is crucial because of dependencies required on the host systems. What if something like shipping containers can be used in software world, then managing containers will become quite easy.

For that we have container technology in place, where we can put all our dependencies in an isolated environment and ship them over a network without worrying about the configuration on host machines. This makes shipping code among different environment easier.

1.2 Problems in Shipping Industry before Containers

The shipping industry faced the following problems before containers:

- Shipping fragile goods with robust ones
- Shipping of edible food items with raw materials
- Shipping of cars
- Shipping of various goods via different mode of transports like railways, roadways, airways, waterways, etc.
- Loading and unloading of goods

Facilitator Notes:

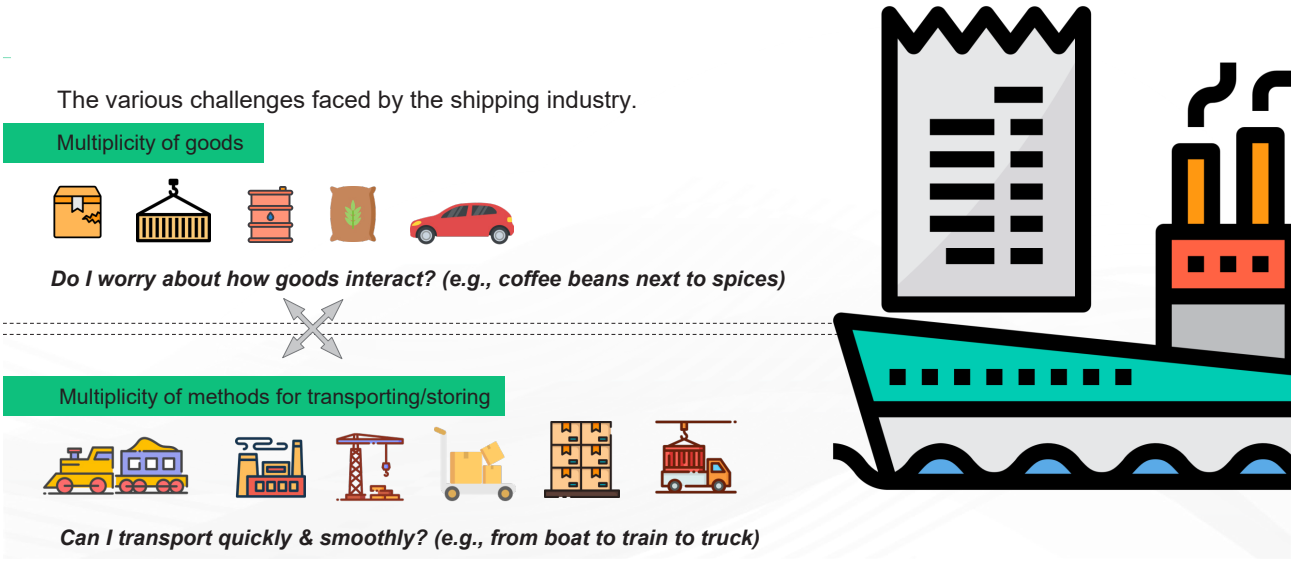
Describe the issues of shipping different types of goods together with different means of transport.

As mentioned earlier, the shipping variety of goods wasn't an easy task. People used to carry them on the back and stack them on ships.

Suppose, a glassware was being stacked up along with the bathroom fittings on the same deck, a sudden swoosh of the waves and you lost the glassware. This will result in huge loss and decrease the reputation of the shipping company. Now, imagine a whole ship stacked up with small packets of variety of goods lying on the deck, the level of chaos is unimaginable. Along with that there won't be much floor area left to carry more goods, which will not be a profitable solution.

Apart from carrying them on the ship, they need to traverse via road and other means of transport, which means they will be moved here and there on the roads which will increase the chances of damage.

1.3 Shipping Industry Challenges



Facilitator Notes:

Inform the participants about the challenges faced by shipping industry.

The one major problem faced by shipping industry was stacking up different types of goods together, which was resulting in damaged goods. So when the containers came into the picture, they ensured that fragile goods to be stored in one container and the same goes for other kinds of goods. Thus, when a glassware was being stacked in one container, all sorts of dependencies was being enclosed in that particular container, like wrapping up glassware in foam and cardboard to avoid breakage. And an organized container stacks allowed better resource utilisation in terms of floor area, since containers can be stacked upon one another.

In the picture, we can see different goods like:

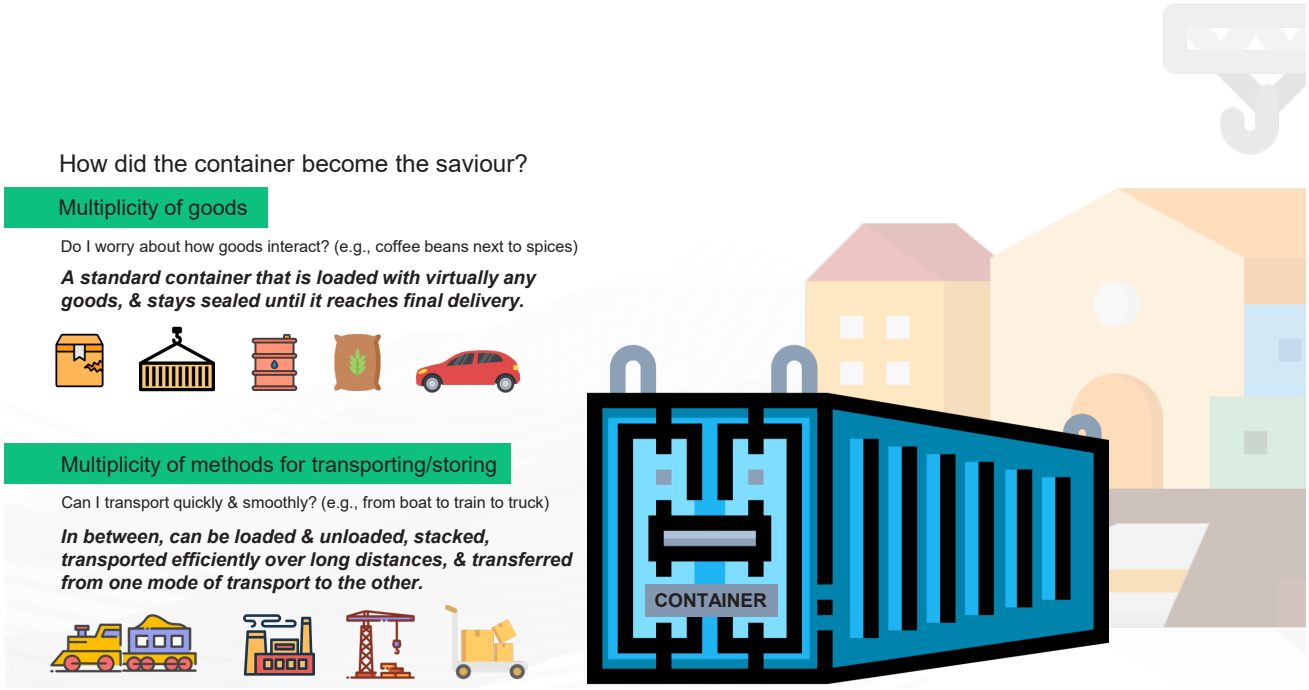
- Cars
- Wine
- Oil tanks
- Piano

- Servers
- Spices
- Fruits

The way to ship them is also different depending upon their fragility. Therefore, the main challenges were as follows:

- How to be sure that the cargo of a port wine company won't damage or contaminate the cargo of another company?
- How to isolate the cargo?
- How to be sure that the cargo is inviolate?
- How to make sure that the cargo is being transported in the best conditions (correct temperature for instance)?

1.4 Container: The Saviour



Facilitator Notes:

Explain how containers solve the problem of shipping different types of goods together and use this analogy to explain shipping off different application stacks.

If we put things in a container isolating them from the other, then it will solve most of the problems. Let's discuss them in detail here:

Challenge-1: How to be sure that the cargo of a port wine company won't damage or contaminate the cargo of another company?

Solution: Since, we have put cargos of different merchants in different containers, there is no chance that things would contaminate each other.

Challenge-2: How to isolate the cargo?

Solution: We can isolate the cargo by keeping different variety of goods in separate containers, which will isolate them from each other.

Challenge-3: How to be sure that the cargo is inviolate?

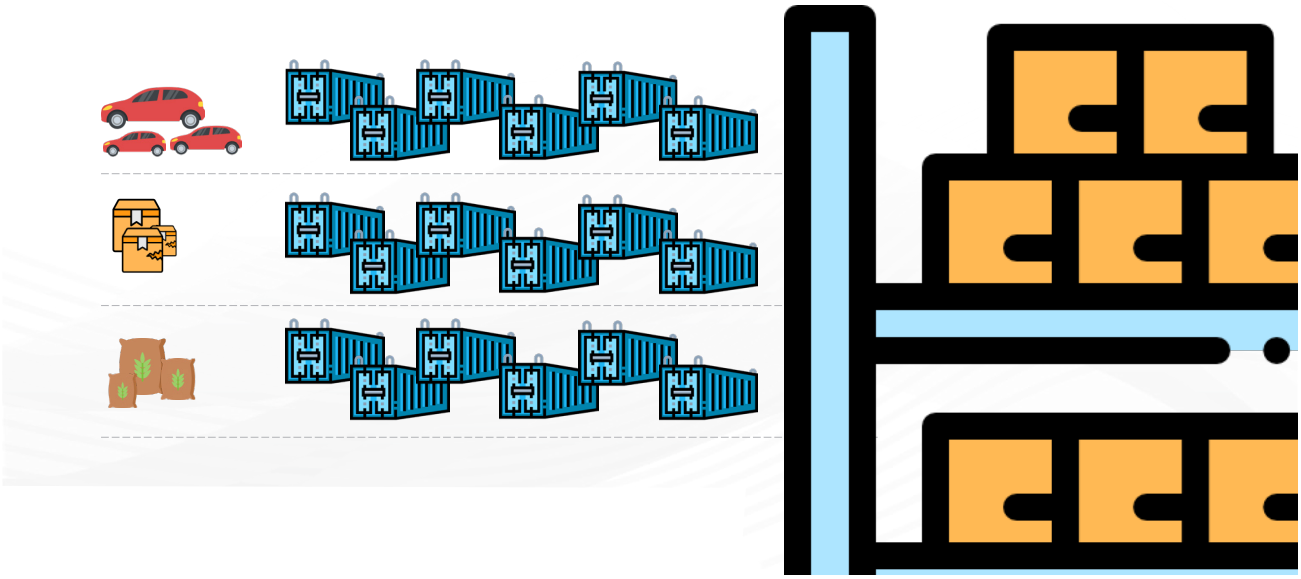
Solution: Each container will have compartments which are stacked up with soft material, which will safeguard the goods from getting damaged.

Challenge-4: How to make sure that the cargo is being transported in the best conditions (correct temperature for instance)?

Solution: Suppose, we have put all the fruits in a single container, so that we can manage the environment inside it, like deep freezing it. Thus, they won't get stale.

1.5 Solution by Containers in the Shipping Industry

Everything falls into place with the help of containers.



Facilitator Notes:

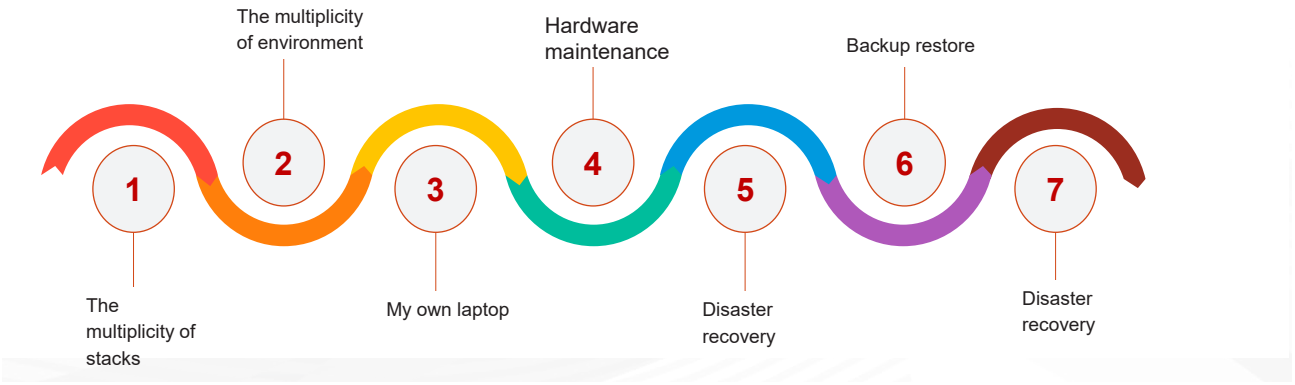
Discuss the how everything works fine with containers.

So, as you can see that everything falls into place with the help of containers. According to the image;

- Now all of the **Wine** will be stacked in one container which can be shipped over ship, trucks or by any other means, and won't get damaged because they are protected with soft material.
- **Oil** containers will not contaminate and smudge the other goods since the oil container has spilled proof shield around it.
- **Cars** won't get scratched as they are nicely placed inside the container locked by the hinges.
- The **Piano** can now be safely shipped without any damage.
- **Fruits** and **spices** are shipped under appropriate temperature, protecting them to get ruined because the container has refrigeration.

1.6 Challenges in the Software Industry

The various challenges in the software industry are as follows:



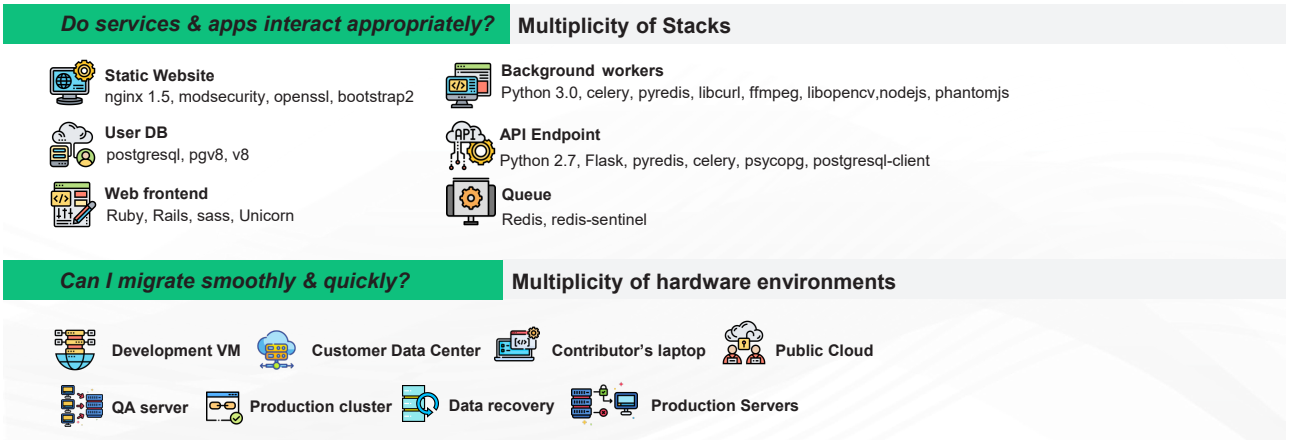
Facilitator Notes:

Discuss the various issues in the software industry.

The various challenges in the software industry are as follows:

- **The multiplicity of stacks:** In most of the software organization, the tech stack is quite diverse in terms of Applications (JAVA, Python, Go, PHP, .net), Platform (Linux, Windows, Mac), Middleware (Caching, Queueing, etc.), Databases (SQL, NoSQL), etc. It happens most of the time that keeping multiple dependencies on a single server could end up in disaster because there won't be any isolation that would keep them separated.
- **The multiplicity of environment:** To make the system robust, each of the applications undergoes various environments to prove its mettle like Development, Staging, UAT, Pre-prod and then production. It is necessary to meet the requirements of the application in each environment to function ideally. And, to achieve that we need a mechanism that will control such operations, avoiding any error due to manual intervention like Software Configuration Management.
- **My own laptop:** One of the most heard statements is "It is working on my laptop, not on prod". To provide the same kind of environment, the code required should be uniform in all environments and platform.
- **Hardware maintenance:** When our system is running on VMs or bare-metal, the underlying hardware sometimes goes under maintenance, which results in downtime if our infra is not dynamic.
- **Disaster recovery:** When a VM goes down, it takes time to recover from that disaster even if you have the best strategy in place, but the duration of the recovery is long enough to have a downtime.

The various challenges in the software industry are as follows:



Facilitator Notes:

Discuss the various issues in the software industry.

- **Backup restore:** If we consider the stateful applications, the major concern in their life cycle is to have a consistent backup if they are running on VMs, but if they go down you need to have a restore plan altogether, but this too cause downtime irrespective of how good is your restore strategy.
- **Disaster recovery:** When a VM goes down it takes time to recover from that disaster even if you have the best strategy in place, but the duration of the recovery is long enough to have a downtime.

1.7 Problems in Software Industry Before Containers

The chaos in the software industry while managing diverse stack in different environments:

	Development VM	QA Server	Single Prod Sever	Onsite Cluster	Public Cloud	Contributor Laptop	Customer Servers
Static Website	?	?	?	?	?	?	?
Background Workers	?	?	?	?	?	?	?
Web Front End	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

Facilitator Notes:

Explain the chaos in the software industry while managing diverse stack in different environments.

As you can see the Matrix in the picture, we have stacks like:

Tech Stack

- **Static website:** A website that will be hosting static content written in HTML with a few images.
- **Web frontend:** Also called a client-side application end written in HTML, CSS, and Javascript for a web application.
- **Background workers:** They are simple independent threads in the application running in the background, they are literally the workers which helps in serving the application.
- **Users DB:** A database where my application is ingesting data and retrieving data when my user is accessing the application.
- **Analytics DB:** A database from which all sorts of reports are generated for analysing the business progress.
- **Queue:** It is a stream-processing software platform, to publish and subscribe to streams of records, similar to a message queue or enterprise messaging system and store streams of records in a fault-tolerant durable way.

We will discuss about platforms in the next slide.

The chaos in the software industry while managing diverse stack in different environments:

	Development VM	QA Server	Single Prod Sever	Onsite Cluster	Public Cloud	Contributor Laptop	Customer Servers
Static Website	?	?	?	?	?	?	?
Background Workers	?	?	?	?	?	?	?
Web Front End	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

Facilitator Notes:

Explain the chaos in the software industry while managing diverse stack in different environments.

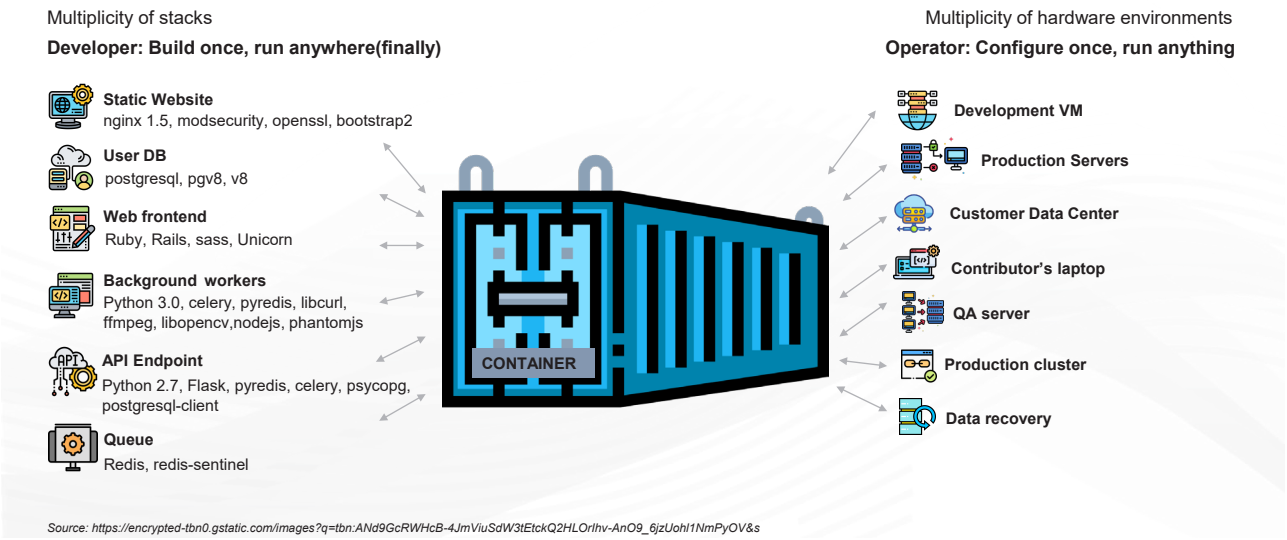
Platforms

- **Contributor laptop:** The system where the code has been developed, i.e., the local system of the developer, e.g.: Laptop and PC.
- **Development VM:** The first environment where the developer runs the code after developing it on the local system. It has other dependencies required for an application to function properly.

- **QA server:** A QA Server usually refers to a machine that handles the QA process, and runs software that helps create environments that can test different code branches, as part of the QA process. This can range from switching environments and checking out a branch, to rebuilding entire machines that match production environments and deploying code to them. The basic principle of a QA Server is to help create QA environments for testing.
- **Single prod server:** A production server may be a dedicated machine, virtual server, basic PC or multiple machines dispersed geographically. For small businesses and simple applications, all the activities involved in deployment may be conducted on a single computer. In enterprise-level software deployment, multiple servers are typically used for the stages required to create and work on software and deliver applications to end-users.
- **Onsite cluster:** It is a platform where the client runs the code once it is delivered to them, after all, sorts of testing done by the delivery team.
- **Public cloud:** The public cloud is defined as computing services offered by third-party providers over the public Internet, making them available to anyone who wants to use or purchase them. They may be free or sold on-demand, allowing customers to pay only per usage for the CPU cycles, storage, or bandwidth they consume.
- **Customer servers:** It is a platform where the application will finally be used by the end-user.

We have to make sure that all the cells should be functional, i.e, the application if it works on the local developer machine should work without any issue on fellow developers and should work without any issues on the production server. One of the standard things that usually there is a difference in the server operating system and the operating system of a local developer.

1.8 Put that in Container!



Source: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRWHcB-4JmViuSdW3tEtckQ2HLOrIhv-AnO9_6jzUohl1NmPyOV&s

Facilitator Notes:

Discuss how putting the application in a container solves the issues faced by the software industry while managing diverse stack in different environments.

The problems which we discussed can be solved if we put things inside a container. Suppose, we have two application which needs to be run on the same VM, one of them runs on python2.7 and other runs on python3. While running both of the applications, we have injected the versions of python as environment variables. If the environment variables are overlapped then both of the applications will suffer.

And If we compare this respect to resource utilisation, if there is no isolation at the process level, that means if any of the application starts consuming more resources it will try to kill the other application, which will bring vulnerability to the system.

Solution

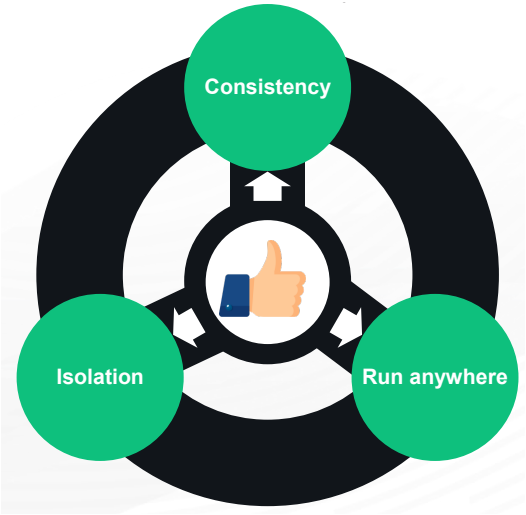
We can put those in separate containers like:

- Container-1: It will have binaries of python2.7 along with the application-1 code.
- Container-2: It will have binaries of python3 along with the application-2 code.

With this, the python version will not clash as both of the applications will be completed isolated because of containers. And we can put the limits on containers with respect to CPU and memory utilisation.

1.9 Solution by containers in the Software Industry

Keeping everything in container solves our problems easily on the factors below:



Facilitator Notes:

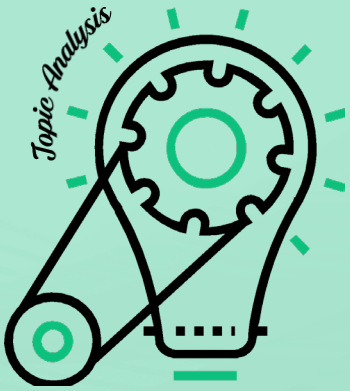
Explain how putting the application in a container solves the issues faced by the Software Industry and fils all the cells of the Matrix.

Putting everything in container solves our problems easily on the factors below:

- **Consistency:** The main benefit of container is that it is consistent among all the environments since it creates predictable environments that are isolated from other applications. A container includes all software dependencies which are needed by the application, such as specific versions of programming language runtimes and other software libraries. All this is consistent no matter where the application is getting deployed. This results in productivity DevOps and Dev teams spend less time debugging and diagnosing differences in environments, and more time shipping new functionality for users.

- **Run anywhere:** Containers are able to run anywhere, enhancing Continuous Delivery and Deployment on Linux, Windows, and Mac operating systems on virtual machines or bare metal, on a Contributor's machine or in data centers on-premises; and in the public cloud as well.
- **Isolation:** Containers virtualize CPU, memory, storage, and network resources at the OS-level, providing developers with a sandboxed view of the OS logically isolated from other applications.

What did you Grasp?



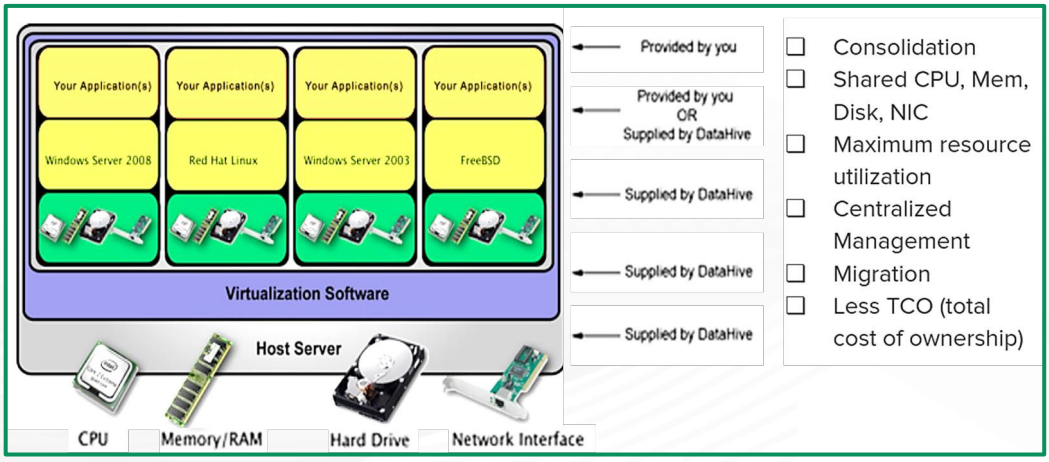
1. Which is the key-factor which helped shipping and software industry to overcome the problems?
   A) Isolation
   B) Availability
   C) Consistency
   D) Performance

Facilitator Notes:

Answer: A. Isolation

1.10 Virtualisation

Virtualization is a technology to run multiple same or different operating systems which are completely isolated from each other.



Facilitator Notes:

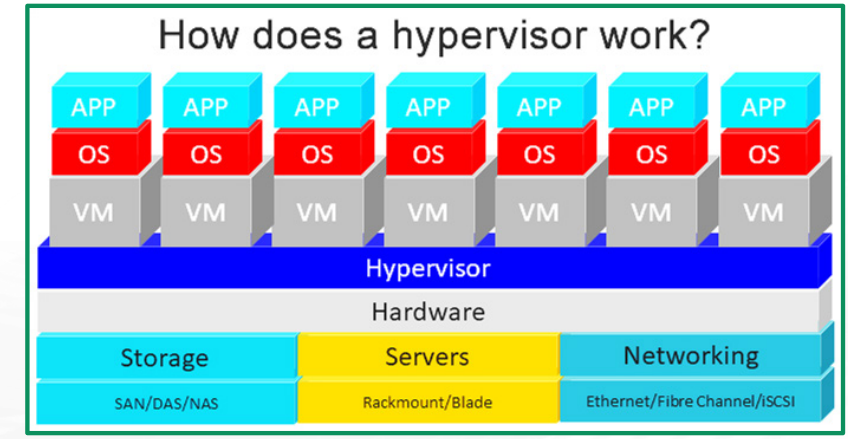
Inform the participants about virtualization works in the software industry.

We have discussed how containers help in the software industry, but learning how virtualization works will help us to understand the core difference between the two.

Virtualization is a technology to run multiple same or different operating systems which are completely isolated from each other.

“Virtualization refers to the creation of virtual machines which have an independent Operating Systems, but the execution of software running on the virtual machine is separated from the underlying hardware resources. Also, it is possible that multiple virtual machines can share the same underlying hardware.”

1.11 What is Hypervisor?



Source: https://www.data-storage.uk/articles/how-does-a-hypervisor-work/

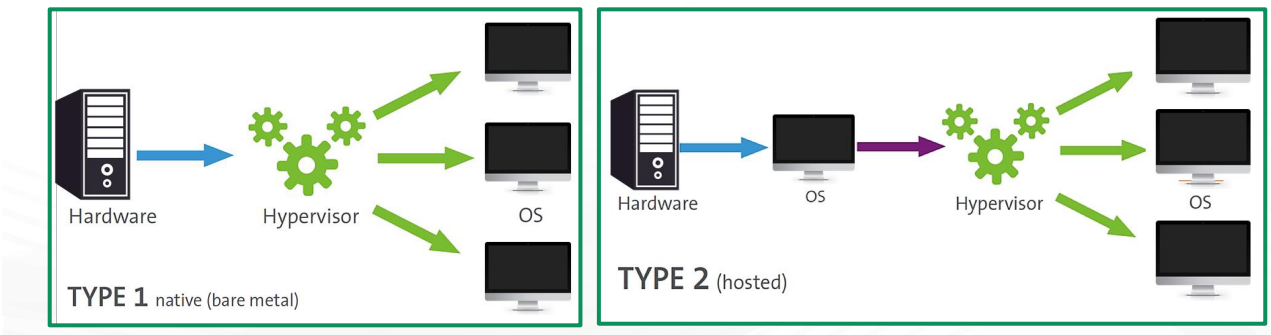
Facilitator Notes:

Describe what is hypervisor in virtualisation.

The hypervisor is a software layer which lies between hardware and operating systems and helps to interact with hardware and resources also provides an interface to share the available resources to virtual containers. Some of the common examples are VMware, Virtualbox, etc.

1.12 Types of Hypervisors

The two types of hypervisors are as follows:



Source: https://www.flexiant.com/2014/02/05/what-does-a-hypervisor-do/

Facilitator Notes:

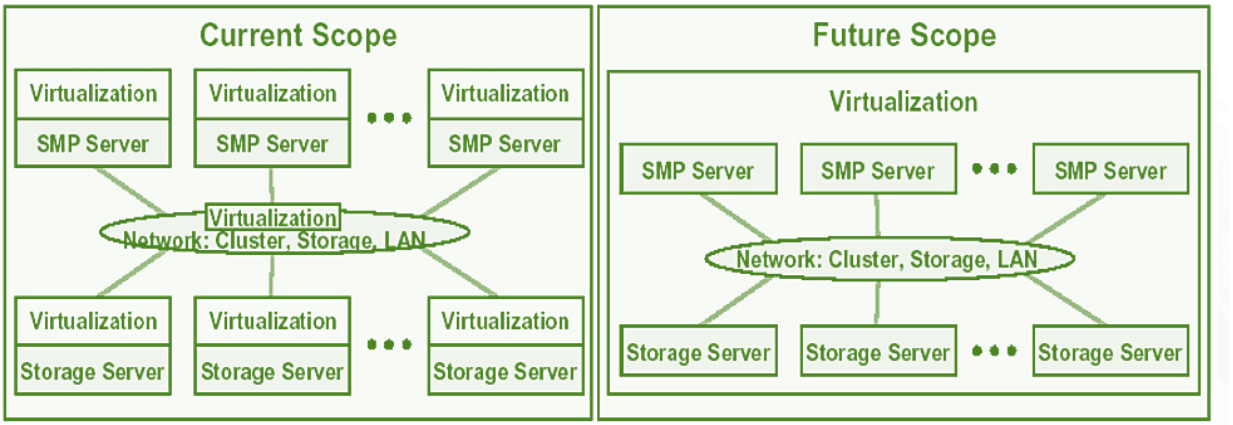
Inform the participants about the different types of Hypervisor in Virtualisation.

We know what Hypervisors are and what they do, but there are few hypervisors, which work with a slight differently to achieve the same objective. Let's classify them into two. They are as follows:

- **Type-1(Native):** Also called Bare Metal Hypervisor. In this type, the hypervisor sits on the top of hardware layer directly and provides interface to provision VMs.
- **Type-2(Host):** In this type, the Hypervisor is installed on an OS that means it needs a host OS to sit on, like in development people use virtualbox for development

1.13 Scope of Virtualisation

Virtualization has a wide scope in the Software Industry in terms of all the components required to keep our whole ecosystem up and running.



Source: https://www.slideshare.net/VaibhavSawant1/green-computing-1-2

Facilitator Notes:

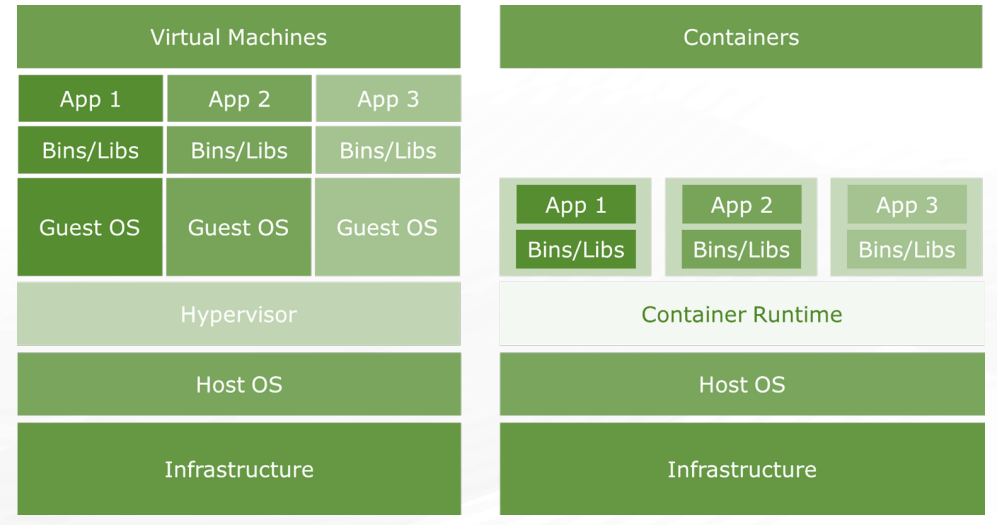
Explain the scope of Virtualisation.

Virtualization has a wide scope in the software industry in terms of all the components required to keep our whole ecosystem up and running.

- **Server:** When we say server, it literally translates to a system that is serving something, and in the software industry, it could be a frontend application, backend application, a static website that can be served. And that server will be a virtual machine.
- **Storage:** Storage is an integral part of any system, If we talk about an application, it needs a database that will store its data for processing. The database which will be doing that will append the data in a file system of a server and again that server will be a virtual machine.
- **Application:** In any business model, there is more than one application that will be serving the purpose all those applications are hosted on a virtual machine.

1.14 Containers vs Virtual Machines

Let's look at the differentiation between containers and machines.



Source: https://i2.wp.com/www.dvoconsult.com/wp-content/uploads/2019/04/containers-vs-virtual-machines.jpg?fit=1024%2C536&ssl=1

Facilitator Notes:

Discuss the difference between virtualisation and containerisation.

If looked from the end user perspective, containerisation is no different from virtualisation, since both provides an isolated environment to serve application on a single platform, but in the context of operations and development containers are way better than VMs.

Let's differentiate them with different parameters:

- **GuestOS:** VMs run on a hypervisor on which a new kernel is loaded into the resources allocated to it. Whereas, container share the OS and kernel of the guest and loads into the physical memory.
- **Networking:** VMs can be linked to the physical switches and NICs since hypervisor has capability for that. Whereas, containers use Virtual NICs for their networking, features like NIC bonding does not exist in containers.
- **Security:** VMs provide complete Isolation in terms of hardware and software, containers provide isolation with the help of namespaces.
- **Lifecycle:** VMs load a complete OS into its memory which itself takes time to spin up new VMs, whereas containers doesn't need to load the whole OS and because of extraction at software level the lifecycle is very quick which helps in spinning up new containers just a game of seconds.

What did you Grasp?

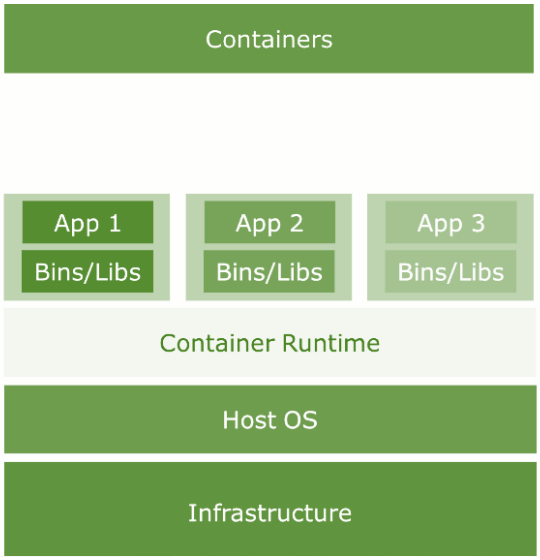
1. What is the main difference between Containerisation and Virtualization?
   - **A) Lifecycle**
   - **B) Isolation**
   - **C) Extraction at software level**
   - **D) Extraction at hardware level**

Facilitator Notes:

Answer: A. Extraction at software level

1.15 Containerisation Platform, Runtime and Images

Let's look at the containerisation ecosystem.



Source: https://i2.wp.com/www.dvoconsult.com/wp-content/uploads/2019/04/containers-vs-virtual-machines.jpg?fit=1024%2C536&ssl=1

Facilitator Notes:

Discuss what kind of components are generally present in containerisation ecosystem.

We have learned about the VMs that how it needs a layer of Hypervisor to work on to spin up the VMs. Since, containers are similar to VMs in many ways, they too need some engine on which they can run on.

There are majorly three components in the ecosystem, namely, Platform, Runtime and Images. If we map these components to virtual machines we might find some similarities.

- Hypervisor in virtualisation is equivalent to the container platform because it helps the container to interact with host just like the hypervisor.
- VM in virtualisation is equivalent to runtime in containerisation since both serves an application.
- ISO files in virtualisation are equivalent to Images in containerisation since both of them read these files from the file system to spin up VMs and containers respectively.

1.16 Container Platform

Containerization platform: It can be defined as a technology to isolate processes from each other, in such a manner that processes run like they are running in a normal operating system which is enforced by the container runtime.

Source: https://1.bp.blogspot.com/-kx-LtOGIwqU/XOzPWiztcVI/AAAAAAAAIvE/qURId5PRX40aV-TFJ-B7GA8t5CqA-UF_3gCEwYBhgL/s1600/docker.png

Facilitator Notes:

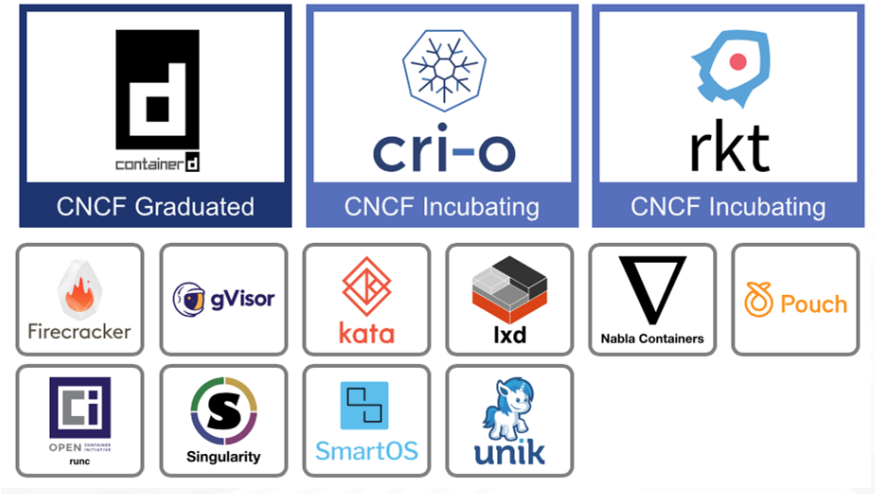
Describe what is container platform.

Containerization Platform: It can be defined as a technology to isolate processes from each other in such a manner that processes run like they are running in a normal operating system which is enforced by the container runtime. They ideally share the resources of a single host machine without having the ability to see each other's or the host's processes and resources because of namespaces and cgroups. A platform also has a control over the runtime and the lifecycle of the images.

It can also be defined as a tool or technology, which provides the functionality to spin up containers with the extraction at the software level and not at the hardware level like virtual Machines. **Examples:** LXC, Rkt, Docker.

1.17 Container Runtime

Container runtime: It is a Container execution environment, which ensures the allocation of limited shares of resources (e.g., CPU, memory, disk) to the containerized application, also exposes the services and APIs and tools for managing containers.



*Source: https://miro.medium.com/max/2486/1*OnB-Ah-FehjzQfhu5-BHmQ.png*

Facilitator Notes:

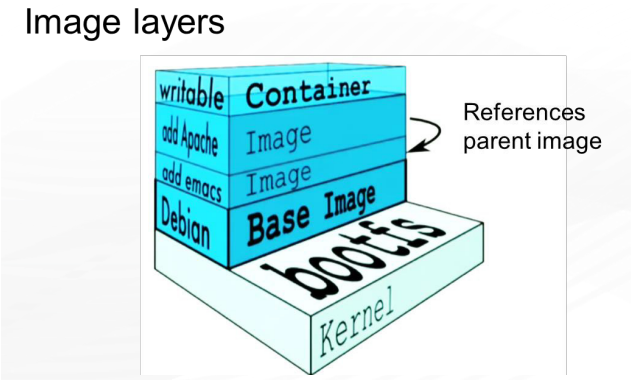
Explain container runtime.

Container runtime: It is a container execution environment, which ensures the allocation of limited shares of resources (e.g., CPU, memory, disk) to the containerized application, also exposes the services and APIs and tools for managing containers. It acts as a layer between the host machine and the containers fulfilling all sorts of requirements like port and volume mapping needed to achieve containerized services.

Examples: LXD, Docker daemon, Rkt process.

1.18 Container Images

Image: Images are readable files which will be used to spin up containers. An image defines the file system and execution parameters for the container. Images can be layered, composable, depending on the format of the runtime.



Source: https://s3.amazonaws.com/dev.assets.neo4j.com/wp-content/uploads/20160112150631/image-layers-docker.png

Facilitator Notes:

Explain what are container images.

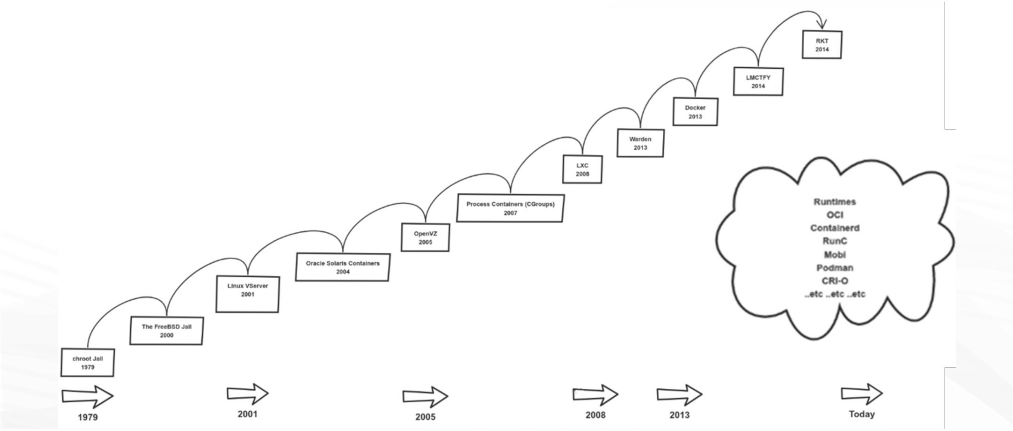
Image: Images are readable files which will be used to spin up containers. An image defines the filesystem and execution parameters for the container. Images can be layered, composable, depending on the format of the runtime.

If need to be defined in terms of application, it can be stated that an image has all the dependencies encapsulated to fulfill the requirement. For example an image of node.js application will have node binaries and application code.

Examples: Docker image, appc, LXC image format.

1.19 Journey of Container Technology

2D representation of time and container technology



Source: https://medium.com/faun/the-missing-introduction-to-containerization-de1fbb73efc5

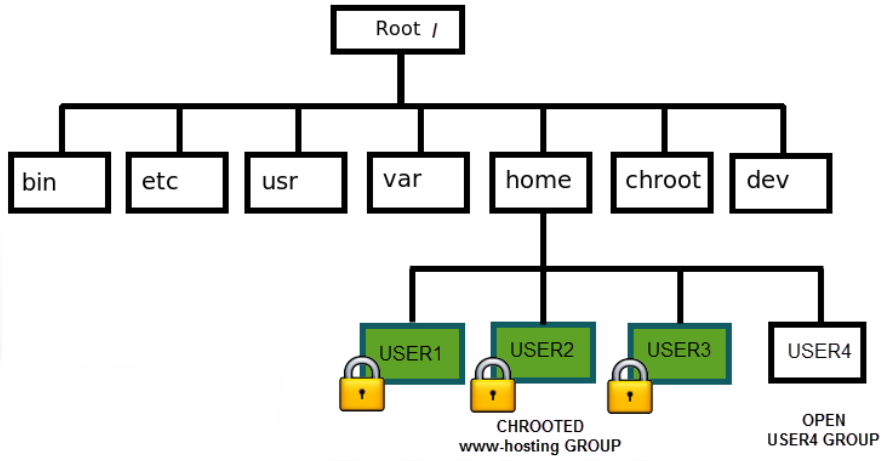
Facilitator Notes:

Inform the participants about the journey of containers.

As we can see in this 2D representation of time and container technology, there is a time-lapse of the container journey. The foundation of containers was **Chroot Jail** and by adding more functionality to it born the **FreeBSD Jail**. These were the two major breakthroughs in the initial journey of containers. After that, many organisation created container technology on chroot jail and FreeBSD like Oracle Solaris Container, OpenVZ, LXC, which were doing just fine, but the technology which gained the attention was **Docker** and everything changed post that. It brought a revolution in container technology.

1.20 The Chroot System

Let's look at the Chroot system:



Source: http://linuxaria.com/wp-content/uploads/2014/01/chroot-ori12.png

Facilitator Notes:

Describe the chroot system.

We all are aware about docker by now, but the actual essence of containerisation began way back in 1979 with chroot systems.

As the name implies like most of Linux commands "chroot" changing root. It is considered as one of the first containerization technologies. It allows you to isolate a process and its children from the rest of the operating system. The only problem with this isolation is that a root process can easily exit the chroot.

What did you Grasp?

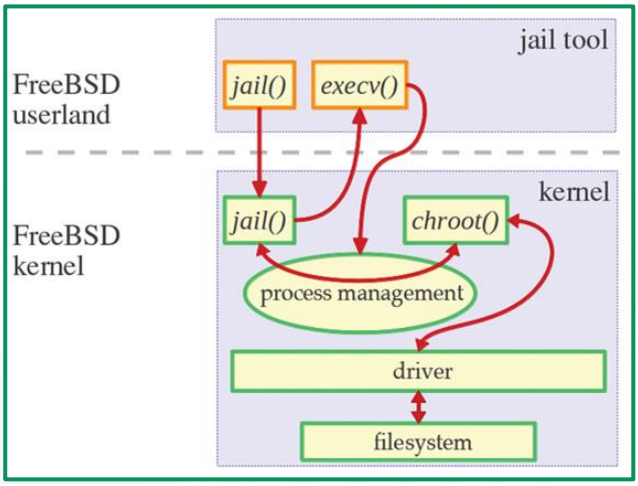
1. What was the demerit of chroot?
   - **A) Root process can easily exit the chroot**
   - **B) Ability to isolate processes**
   - **C) Extraction at software level**
   - **D) C-groups enabling how much resource you can use**

Facilitator Notes:

Answer: A root process can easily exit the chroot

1.21 FreeBSD Jails

Let's look at the FreeBSD Jails:



Source: http://www.admin-magazine.com/Archive/2013/13/How-to-configure-and-use-jailed-processes-in-FreeBSD/(offset)/3

Facilitator Notes:

Inform the participants about the FreeBSD Jails.

Twenty two years after chroot, FreeBSD Jails were introduced. The main intent behind FreeBSD was to overcome the security vulnerability present in chroot file isolation.

FreeBSD was able to isolate processes as well, which were not possible with chroot along with the activities in the file system.

What did you Grasp?

1. What was the advantage of FreeBSD over chroot?
   - **A) Root process can easily exit the chroot**
   - **B) Ability to isolate processes**
   - **C) Extraction at software level**
   - **D) C-groups enabling how much resource you can use**

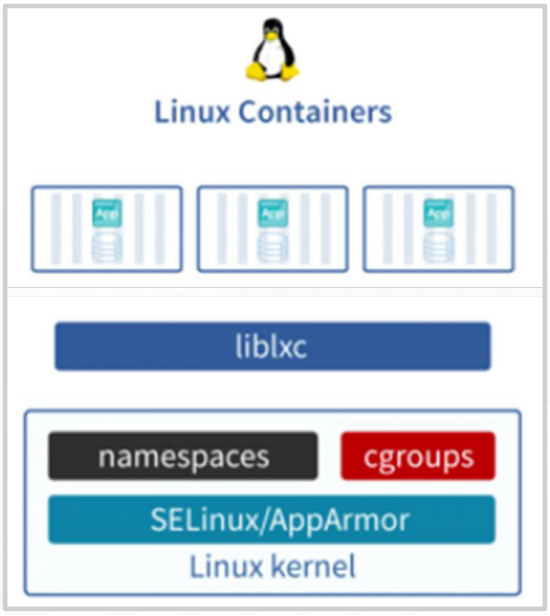
Facilitator Notes:

Answer: B. Ability to isolate processes

FreeBSD was able to isolate processes as well, which were not possible with chroot.

1.22 LinuX Containers (LXC)

Let's look at the LinuX containers (LXC):



Source: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcR2oxv0OImn_QPxFahTuelvfsEz-mYpqUmo8zsVMyH-cbTYZn04raA&s

Facilitator Notes:

Explain the Linux Containers.

Eight years later Linux Containers were introduced, going a step ahead with the isolation of processes, and activities in the file system, LXC introduced c-groups which provide a mechanism for aggregating/partitioning sets of tasks, and all their future children, into hierarchical groups with specialized behaviour. In a simplified way, it limits how much resource you can use.

What did you Grasp?

1. Which technology helped LXC to stand out from the other container utilities?
   - **A) Root process can easily exit the chroot**
   - **B) Ability to isolate processes**
   - **C) Extraction at software level**
   - **D) C-groups**

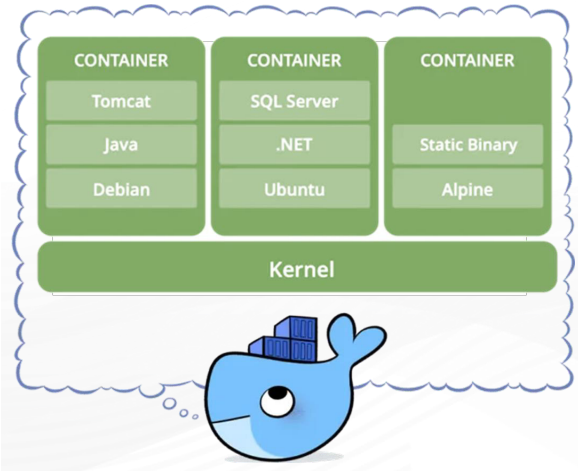
 |

Facilitator Notes:

Answer: D. C-groups

1.23 Docker

In 2013, the first version of Docker was introduced. Developed by Google engineers collaborating with Docker over libcontainer and porting the core concepts and abstractions to libcontainer.



Source: https://i0.wp.com/www.docker.com/blog/wp-content/uploads/011f3ef6-d824-4d43-8b2c-36dab8eaaa72-1.jpg?fit=650%2C530&ssl=1

Facilitator Notes:

Describe the Docker containers.

In 2013, the first version of Docker was introduced. Developed by Google engineers collaborating with Docker over libcontainer and porting the core concepts and abstractions to libcontainer.

The major aspects because of which docker is still running are as follows:

- **Promoting microservice architecture:** As more organisation are moving towards microservice architecture, docker has been fulfilling such requirements. Docker is continuing to prove its capability with enhanced monitoring, storage, security, and delivery chains.
- **Bringing development and operations together:** Docker is the first DevOps tool that's as popular with developers as it is with ops engineers. The reason being, developers can work inside the containers, and ops engineers can work outside the containers in parallel.
- **Consistency in continuous integration:** Containers are able to run anywhere, enhancing Continuous Delivery and Deployment on Linux, Windows, and Mac operating systems on virtual machines or bare metal, on a contributor's machine or in data centers on-premises; and in the public cloud as well.
- **Delivery chains:** It simplifies software delivery chains by providing environment parity and convenience in deploying an app for multiple types of environments. To take advantage of the Docker's benefits in this respect, we need a robust delivery chain.

In a nutshell, we learnt:

1. Transporting “goods” analogy
2. Containerization platform, images and runtime
3. Comparison with virtual machines
4. The chroot system call
5. FreeBSD Jails
6. LinuX Containers (LXC)
7. Docker

Facilitator Notes:

Share the module summary with the audience.

Ask the participants if they have any questions. They can ask their queries by raising their hands.

Now, you have reached the end of the module, In this module, you have learned:

- Transporting “goods” analogy
- Containerization platform, images and runtime
- Comparison with virtual machines
- The chroot system call
- FreeBSD Jails
- LinuX Containers (LXC)
- Docker

Release Notes

B. TECH CSE with Specialization in DevOps

Semester Six -Year 03

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide, Mock exams and relevant lab guide.

Current Release Version.
1.0.0

Current Release Date.
19 Jan 2020

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Jan 2021