UPES

Xebia

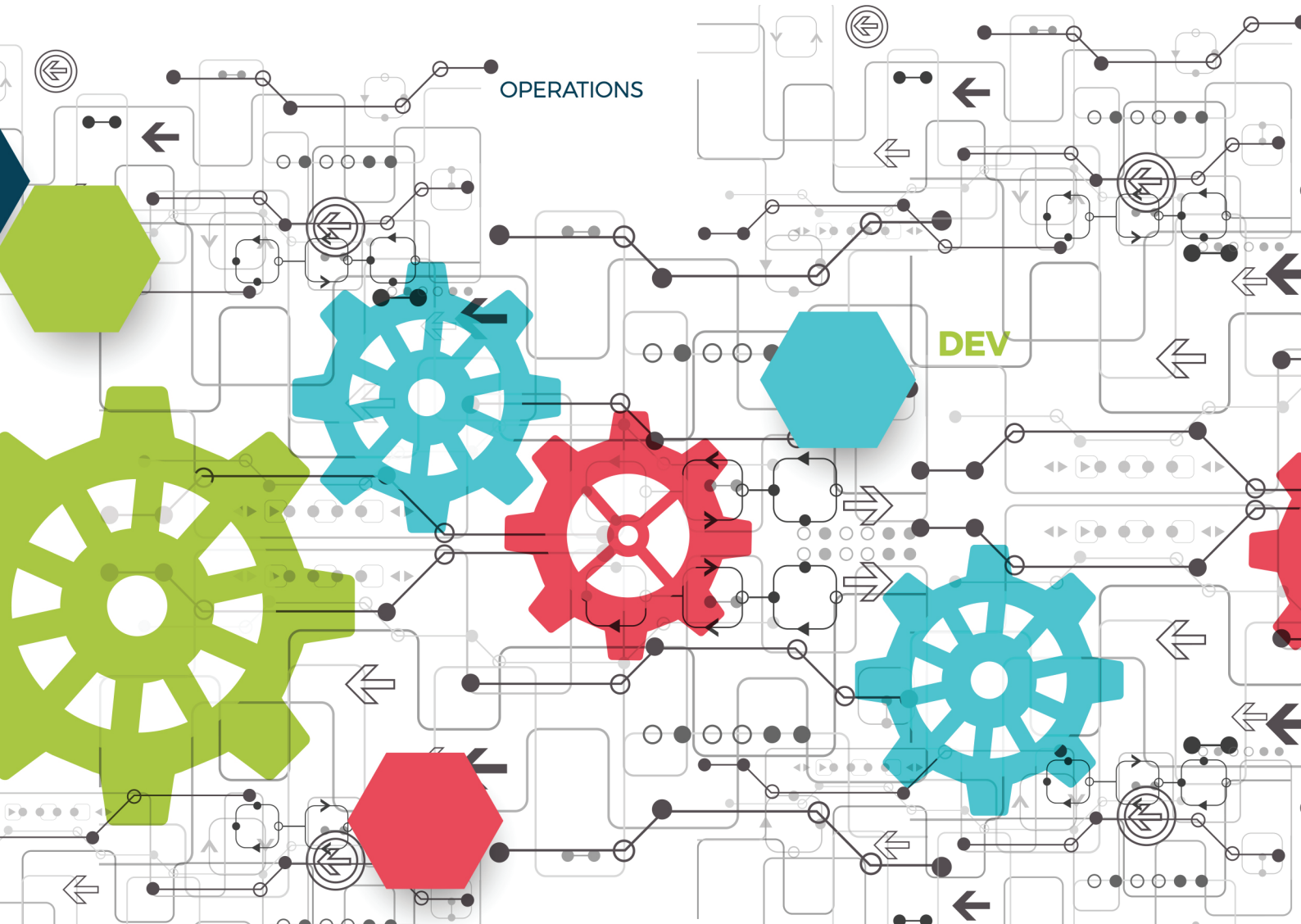
YEAR 3

B.Tech Computer Science and Engineering in DevOps

Application Containerization

MODULE 2

Introduction to Containerization



Contents

MODULE 2

Introduction to Containerization

Facilitator Notes:

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

You will learn about the 'Introduction to Containerization' in this module.

Module Objectives

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

- Explain the Docker architecture
- Identify the different environments: (Dev, QA and Prod)
- Learn about overcoming issues with different environments
- Describe virtual machines for dev/deployments
- Define containers for dev/deployments
- Discuss the advantages and disadvantagesof containerization

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:

- Explain the Docker architecture
- Identify the different environments: (Dev, QA and Prod)
- Learn about overcoming issues with different environments
- Describe virtual machines for dev/deployments
- Define containers for dev/deployments

- Discuss the advantages and disadvantages of containerization

Module Topics

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

- Docker architecture
- Different environments: (Dev, QA and Prod)
- Overcoming issues with different environments
- Virtual machines for dev/deployments
- Containers for dev/deployments
- Advantages and drawbacks of containerization

Facilitator Notes:

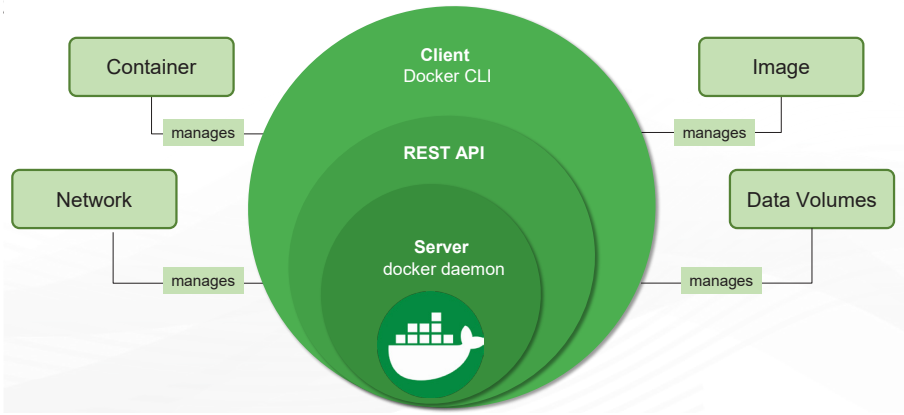
Inform the participants about the topics that they will be learning in this module.

You will learn about the following topics in this module:

- Docker architecture
- Different environments: (Dev, QA and Prod)
- Overcoming issues with different environments
- Virtual machines for dev/deployments
- Containers for dev/deployments
- Advantages and drawbacks of containerization

2.1 Docker Architecture

Docker uses a client-server architecture.



Facilitator Notes:

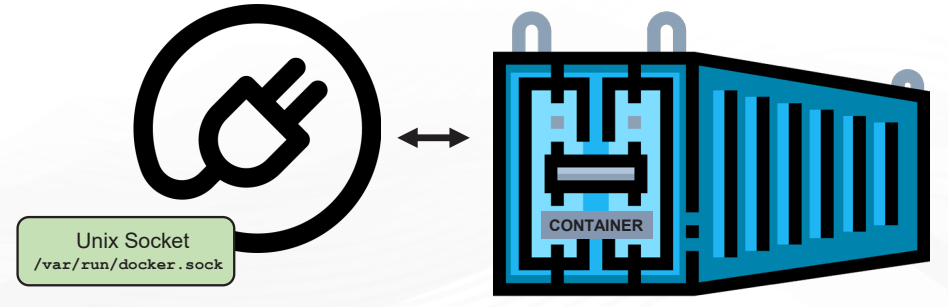
Explain how docker use client-server architecture.

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

We'll dive deep into each component and what role they play in the docker lifecycle in the upcoming slides.

2.2 Docker Daemon (Container Platform)

The Docker daemon listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.



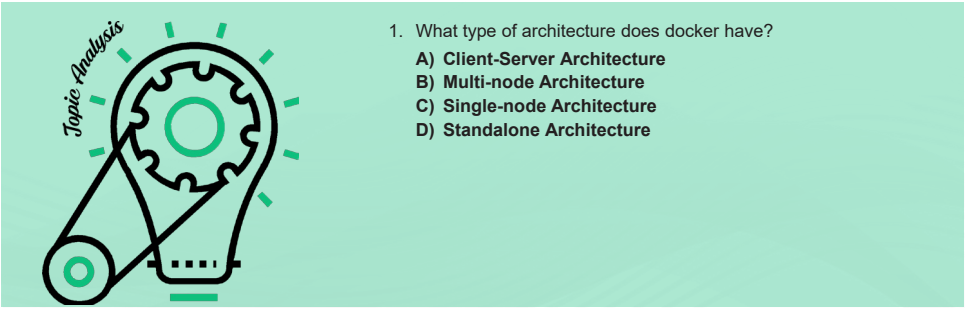
Facilitator Notes:

Explain how docker daemon works as a container platform.

As we have discussed in our previous module, container platform 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.

In case of Docker as a container technology, docker daemon plays the key role of container platform. This is the component which does the rest of the magic and knows how to talk to the kernel, makes the system calls to create, operate and manage containers, which we as users of docker don't have to worry about.

What did you Grasp?

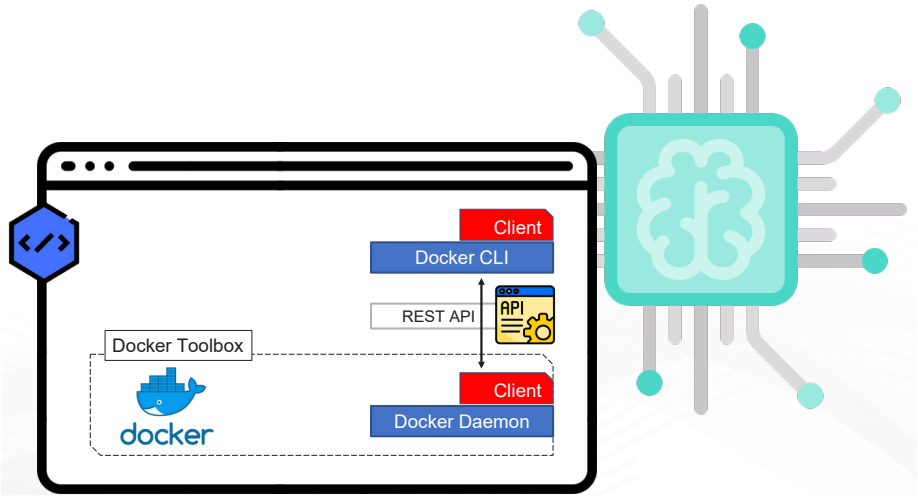


Facilitator Notes:

Answer: A. Client-Server

2.3 Docker Rest API

Let's look at the Docker's engine.



Facilitator Notes:

Discuss how Docker Rest API helps docker client and daemon to communicate.

In any client-server architecture, we need a framework which could enable the communication amongst the two systems with below mentioned features.

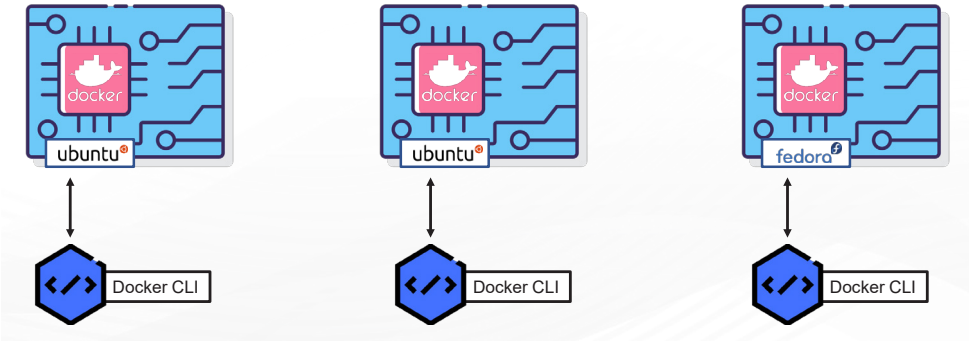
- Simplicity, first and foremost
- Provide useful abstractions. Next, consider the concept of abstraction.
- Discoverable
- Consistent and symmetrical
- Follow the principle of least astonishment

And the Rest API used by docker is apt for the requirement.

Docker Engine on a whole exposes an API for us and also interfaces with kernel. It is the underlying client-server technology that builds and runs containers using Docker components and services. DE comprises of Docker daemon, a REST API and the CLI that talks to the Docker daemon through the API.

2.4 Docker CLI

CLI is a command line program that accepts text input to execute operating system functions.



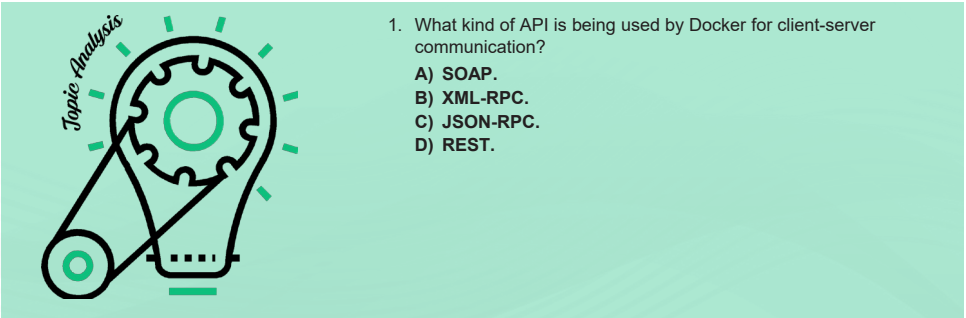
Facilitator Notes:

Inform the participants how Docker CLI makes request via Docker Rest API to communicate with daemon.

CLI is a command line program that accepts text input to execute operating system functions. In case of a docker, we needed a flexible command line tool which could manage the lifecycle of docker entities like image, container, volume, network, etc.

In simple terms, we can say that docker-cli is the utility which we use to run any docker commands, e.g., docker run (docker container run), docker images, docker ps, etc. It allows us to run these commands which a human can easily understand.

What did you Grasp?

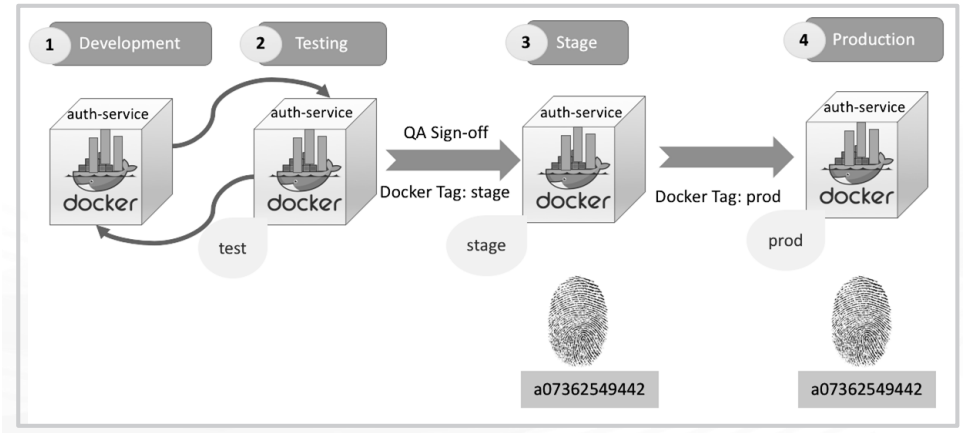


Facilitator Notes:

Answer: D. Rest API

2.5 Different Environments: (Dev, QA and Prod)

The term environment refers to the collection of hardware and software tools a system developer uses to build software systems.



Source: https://blog.imaginea.com/wp-content/uploads/2016/11/Screenshot-2016-11-10-16.19.17.png

Facilitator Notes:

Discuss what are environments in the software industry.

In software development, the proper functioning of a piece of code holds utmost priority. So, to test its proper functioning, it goes through many phases where it is tested in all sorts possible and these phases are called environment in the software development.

The term environment refers to the collection of hardware and software tools a system developer uses to build software systems. As technology improves and user expectations grow, an environment's functionality tends to change as per the requirement.

The main three environments are as follows:

- Development
- Testing
- Stage
- Production

2.5.1 Development Environment

A development environment is a collection of procedures and tools for developing, testing and debugging an application or program. The development environment normally has three server tiers, called development, staging and production.

Facilitator Notes:

Explain what is development environment in the software industry.

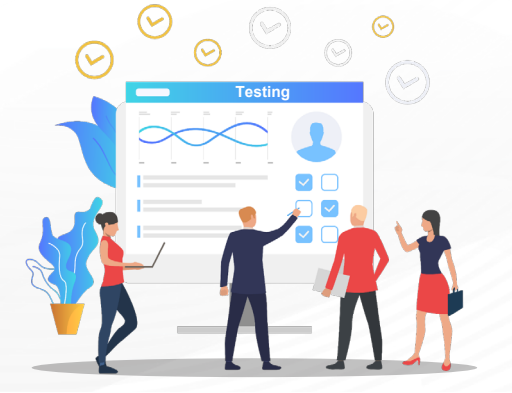
A development environment is one where everything begins, like the origin of the code base, it is usually your local desktop. In this environment, we do all types of change, update, remove and addition in the code. Development environment consists of all branches and commits, which are simultaneously being used by co-developers.

If taking an example of an application which needs a database for its proper functioning, then there will be a local database configured in the local network so that you do not hamper the real data while development. Since, many co-developers will be working simultaneously on one project, they all will be working on separate branches and connected to the local database.

A basic level of testing is also done in this environment, so that any sort of bug doesn't get squeak into the next environment. Further, you can release your code to the next environment after development in the code.

2.5.2 Testing Environment

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured.



Facilitator Notes:

Explain what is testing environment in the software industry.

This is an environment in which the piece of code is tested in various aspects. A testing environment consist of software and hardware for testing teams to execute test cases. This software and hardware provides flexibility to testing teams so that they can carry out various tests on the code.

Testing can be done with various approaches like:

- Manual Testing
- Automation Testing
- Unit Testing
- Integration Testing
- Smoke-Sanity Testing
- Regression Testing

2.5.3 Staging Environment

Staging environment is similar to production environment in many ways.

Facilitator Notes:

In this sub-module you will explain what is Staging environment in software industry.

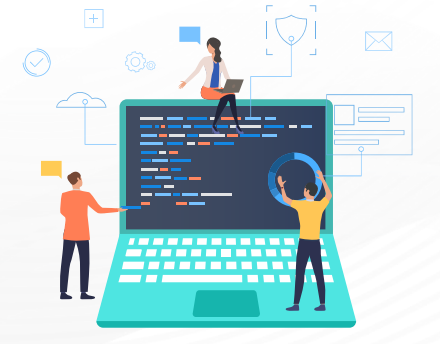
Staging environment is similar to the production environment in many ways. Since the same code is running, usually we take a backup of live environment data to the staging environment. Whenever we are proceeding with any major changes staging environment helps to validate those.

There are various aspects of testing, which can only be done with live data, hence those testings are done in staging environment, since it provides us that flexibility of data migration. And such hardcore testing does provide insights in major changes.

Suppose you have a client to whom you want to show a demo of the major changes you are doing in the code, such demonstrations are also done in staging environment, because it is almost similar to production environment.

2.5.4 Production Environment

Production environment is a term used mostly by developers to describe the setting where software and other products are actually put into operation for their intended uses by end users.



Facilitator Notes:

In this sub-module you will explain what is Production environment in software industry.

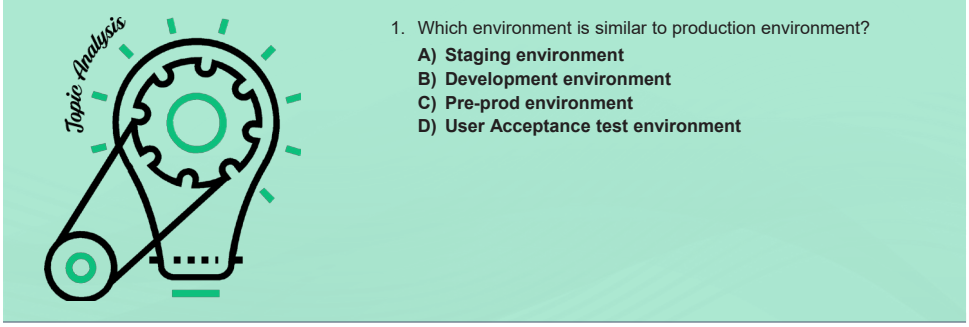
A very frequent lingo used in the software industry is going live, it actually means that we are taking our code to the production environment. It is the environment where the code faces real time traffic and genuine users are hitting your application. This is the most important environment amongst all.

Production environment is the user facing interface, where all users are using the application and an organization is generating revenue. So, it is supposed to be an error free, that is why the code goes through several environment.

There are various methods to deploy code in production, some of the common methods are as follows:

- Recreate
- Ramped (also known as rolling-update or incremental)
- Blue/Green
- Canary
- A/B testing
- Shadow

What did you Grasp?

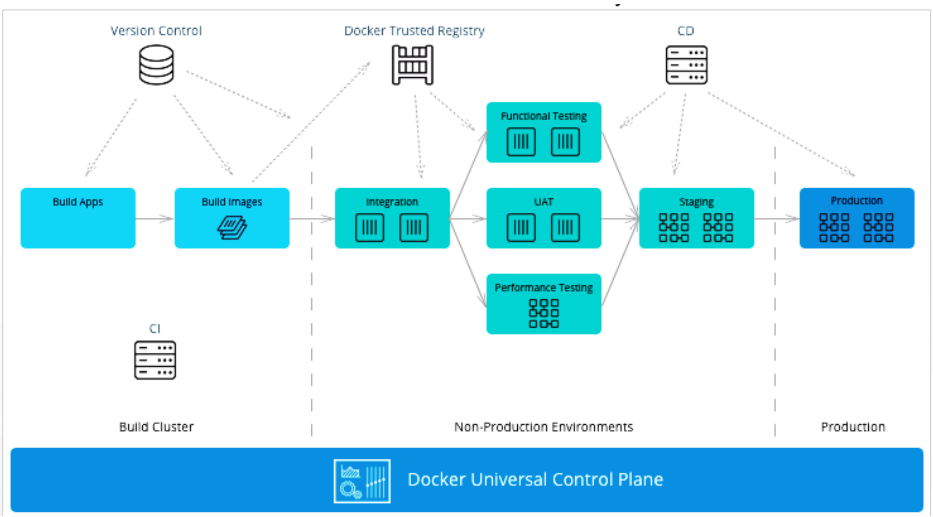


Facilitator Notes:

Answer: A. Staging environment

2.6 Issues with Different Environments

Various issues faced in different environments of the software industry.



Source: https://success.docker.com/api/images/.%2Frefarch%2Fdev-pipeline%2Fimages%2Fgeneral-workflow.png

 |

Facilitator Notes:

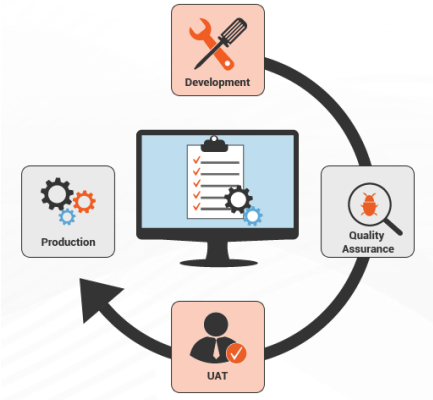
Discuss what kind of issues are faced in different environments of the software industry.

The importance of qualitative testing in an IT industry is increasing day by day with many new technologies and software coming into the market. It is very important for the companies to pay equal attention to testing as they pay to the development of the product. The deployment of the software must be bug-free and for that proper testing environment should be set up.

An IT test environment management means managing and controlling the whole testing environment in order to conduct error-free testing. It includes the management of multiple components like hardware, software, databases, applications, etc. There are some problems faced while managing the test environment and these problems raise several issues in the testing of the product.

2.6.1 Difference in Testing and Production Environment

The major difference among the both environments is that of user interaction, production is used by users whereas test env is used for testing so that no failure occurs while user experience.



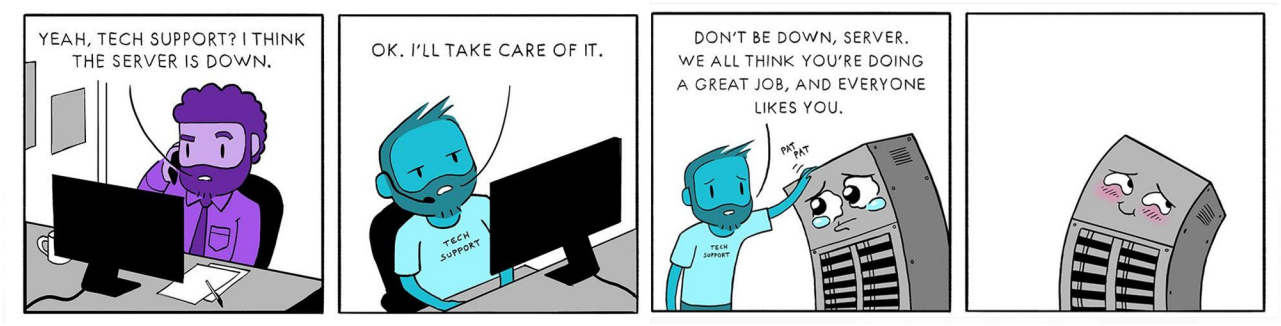
Facilitator Notes:

Explain the difference in testing and production environment.

The difference in the testing environment and the production environment is quite wide. Some of the key differences are as follows:

- The main difference is the users are only face production and testing environment is only for running the test cases and finding issues in the code.
- The software and hardware configuration are same in both the environments, in that sense both of the environments are same.
- We need testing environment, so that there won't be any failure in the production environment as we have tested the code in all aspects in the testing environment.

2.6.2 Maintenance of Testing Environment



Source: https://i.redd.it/nvfde0q2jrl11.jpg

Facilitator Notes:

Inform the participants about the maintenance of testing environment.

The other big problem faced by the current IT industry is the poor maintenance of the devices of the testing environment. There should be a proper maintenance in the management of the infrastructure and the assets of the testing premise, so that they function properly at the time of testing.

Poorly managed hardware, softwares and electronic devices will either need repair or replacement in regular interval and if not done causes delay in the testing process.

If any hardware needs any replacement or repairing, then it will require a whole procedure where you have to raise ticket for the approval of the higher authorities due to increase in the budget of the testing environment. All this will eventually cause the product deployment to get delayed day by day.

2.6.3 Budget Constraints

Having Multiple environment is a costly issue. Sometimes, the companies due to lack of time constraint or budget, assign the developers the responsibility of testing the developed product.

Facilitator Notes:

Explain the participants about budget constraints.

Sometimes, the companies due to lack of time constraint or budget, assign the developers the responsibility of testing the developed product. This thing creates problems for the product being deployed as the application developers have a narrow perspective of the product than the professional testers who have several years of experience.

Having a separate environment for testing and staging purpose is a costly issue and a lot of companies make a mistake by not opting for this. This doesn't allow the testing of the product to be done on different platforms. Because of which mistakes, the product contains bugs, and errors in the code and the deployment results in a failure.

2.6.4 Miscommunication

Source: https://cdn4.vectorstock.com/i/1000x1000/33/03/miscommunication-vector-843303.jpg

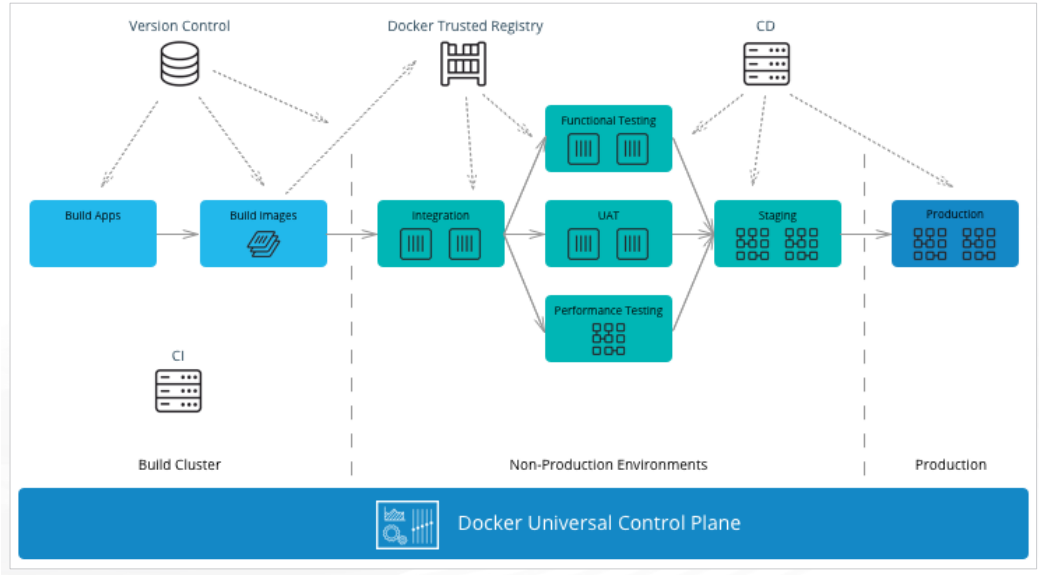
Facilitator Notes:

Explain the role of miscommunication faced in different environments of software industry.

One of the major mistake done in an IT environment management is miscommunication. A gap in communication causes misunderstanding among testing and the business requirements which eventually results in the failure of deployment of the product.

Lack in proper defect tracking tool is also a big problem faced in the IT environment management. This creates a problem as important information about the bugs might not be identified or it might get lost and eventually results in failure.

2.6.5 Overcoming issues with different environments



Source: https://success.docker.com/api/images/.%2Frefarch%2Fdev-pipeline%2Fimages%2Fgeneral-workflow.png

Facilitator Notes:

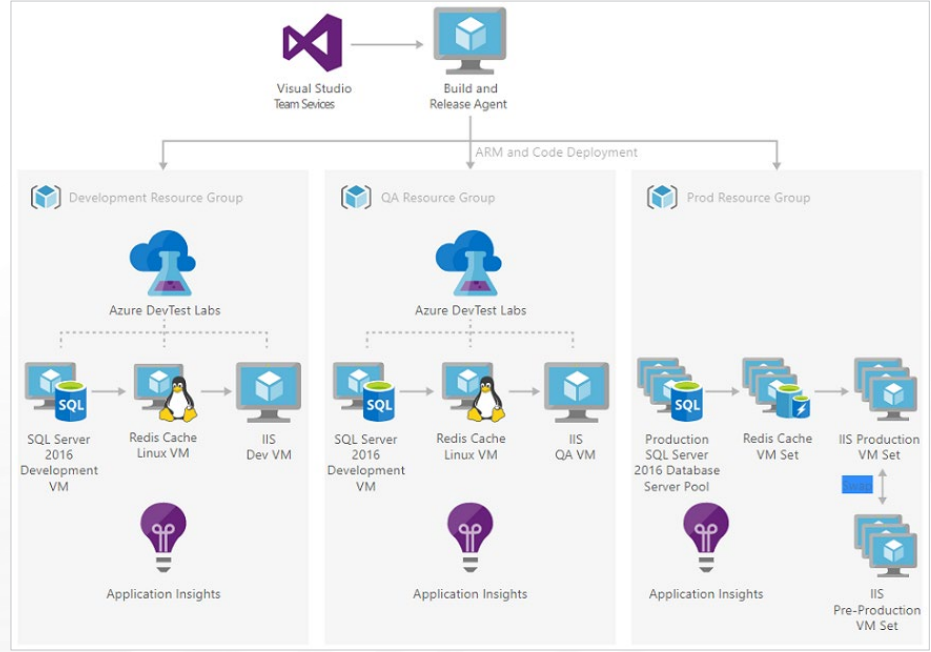
Discuss how container technology fixed issues which we faced in different environments of the software industry.

Containers are the easy way out to overcome the issues we face in different environments.

- **Difference in environments:** Containers are prominent because of a unique tendency to encapsulate everything inside them, which solves our problem of different environments. Since all the dependencies are inside the container, hence there is the least possibility that there will be overlapping of some sort of binaries or software packages which could compromise the services or applications.
- **Maintenance:** As we know that shipping of containers from one machine to another is quite flexible, hence if your systems are undergoing any maintenance, there won't be an issue, because you can ship your containers to different systems without compromising the production. Moreover, If we are using any sort of an orchestrator, it will manage the tenancy as well which will surely decrease the hassle.
- **Budget constraints:** Unlike metal machines, containers are very well capable of managing the resources in a very frugal way. For example, suppose you are running an application on a bare metal system of the 2 CPU and 4GBs of RAM and system works fine, but in reality the application only consumes what is needed, which is around 400 MBs of RAM the rest of the compute resource is for managing the load, hence in the meantime, your resources are being wasted, but in case one of the containers they take what they want, hence the resource utilisation is optimal and results in being cost effective.

2.7 Virtual Machines for Dev/Deployments

Let's understand how development and deployment work with virtual machines.



Source: https://d1x2i8adp1v94i.cloudfront.net/wp-content/uploads/2017/12/AZDPS-Implementing-Develop-and-Test-Scenarios-with-Microsoft-Azure-Virtual-Machines.png.webp

Facilitator Notes:

Describe how development and deployment work with virtual machines.

Before getting started to this topic, let us understand what is development and deployment.

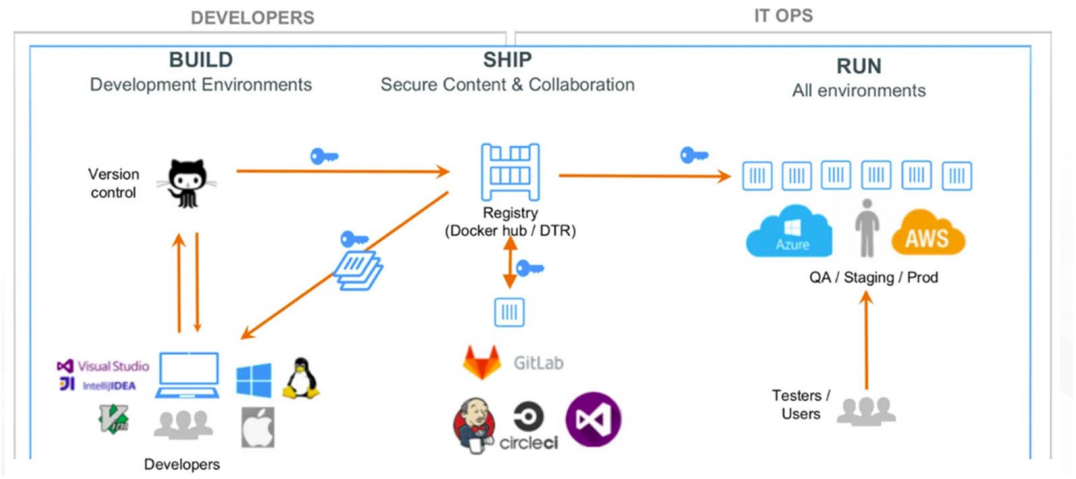
- When a software developer integrates new functionality in a piece of software code that process is called software development. Development as it literally translates to developing a new functionality or growing further. In the modern software industry, development is also related to continuous integration in which developers are encouraged to develop code in small chunks rather than putting all of the functionality at once.
- Deployment means when a piece of software surpasses the development phase and in some case the testing phase, then it enters to deployment phase where it will be used in a live environment, used by multiple user and serving its sole purpose.

Typically in the case of development and deployment in virtual machines, the code is developed in an environment which has all the dependencies which are needed for the proper functioning of code. When that code goes build and proceeds further for deployment those dependencies need to be the same in the other environment as well.

To ensure consistency we need software configuration management in place to handle such dependencies.

2.8 Containers for Dev/Deployments

Now, let's discuss how development and deployment work with containers.



Source: https://i0.wp.com/www.docker.com/blog/wp-content/uploads/4fa92c35-5a00-4e7a-929e-e5ae4b99701a.jpg?fit=1600%2C902&ssl=1

Facilitator Notes:

Explain how development and deployment work with containers.

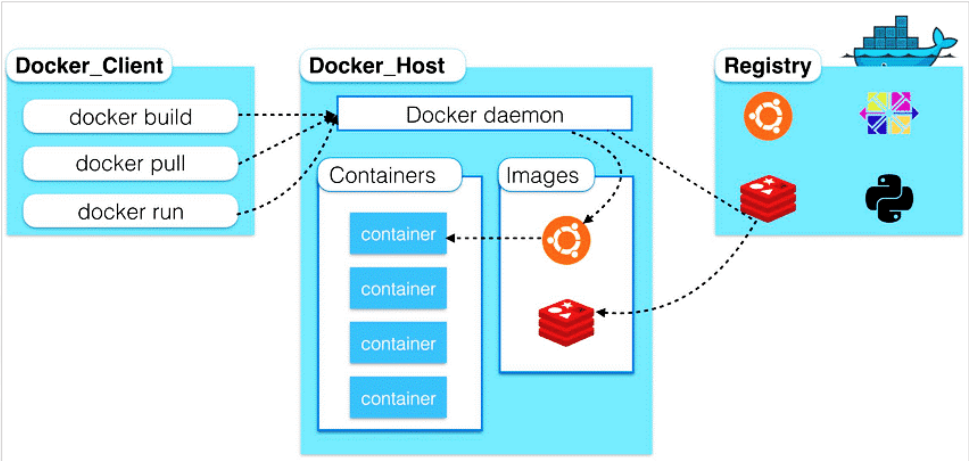
Development in containers:

In our legacy models, the code is developed in a local development environment, where it is build and the dependency is resolved in the local machines and other dependency like database or any other middleware is also placed somewhere in the local network. If we take an example of JAVA based application after building the code with maven, the artifact generated is a .JAR or .WAR file, which is then to be deployed onto the production servers with their on live databases and middlewares.

But in case of Docker, that .JAR or .WAR file is replaced by the images and that image is our artifact, which will be further deployed in the production servers which we be having a container platform.

2.9 Docker Lifecycle

Let's understand about the Docker lifecycle.



Source: https://www.researchgate.net/profile/Yahya_Al-Dhuraibi/publication/308050257/figure/download/fig1/AS:433709594746881@1480415833510/High-level-overview-of-Docker-architecture.png

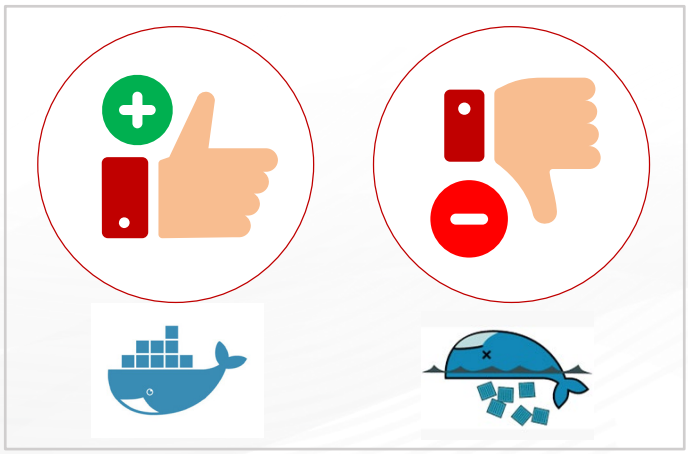
Facilitator Notes:

Explain how development and deployment work with docker using docker lifecycle.

- Stage-1: Development
  - This is the stage where you are developing your code, making changes, and committing in the Version Control System, typically what you do at development stage using the philosophy of Continuous Integration.
- Stage-2: Building code and Dockerizing the application
  - At this stage, you will build the code and create an artifact of it, later you will encapsulate application inside a Docker image from a Dockerfile.
- Stage-3: Pushing Docker image to private registry
  - Once you have created a docker image, you will test that code locally and push it to the private docker registry for deploying it to further environments.
- Stage-4: Deploying it to the testing environment
  - Now, you have a Docker image in your private docker registry, pull the latest docker image in the respective environment and run that image in testing environment for running test cases and other testing further aspects of the tests.
- Stage-5: Going live
  - Once you are satisfied that everything works fine in the previous environments, you can go live running the latest docker image.

2.10 Advantages and Drawbacks of Containerization

Now, we will discuss the advantages and disadvantages of containerization.



Facilitator Notes:

Explain the participants about the advantages and disadvantages of containerization.

We discussed about how containerisation works and now we will discuss what could be the merits and demerits of it. Since, it is subjective that docker could be beneficial for some use cases, but yet it could be a hassle for the application which works just fine on VMs.

2.10.1 Advantages of Containerization

Advantages

Some advantages of Docker are as follows:

- Cost saving
- Standardization and productivity
- Rapid deployment
- Isolution
- Security

Facilitator Notes:

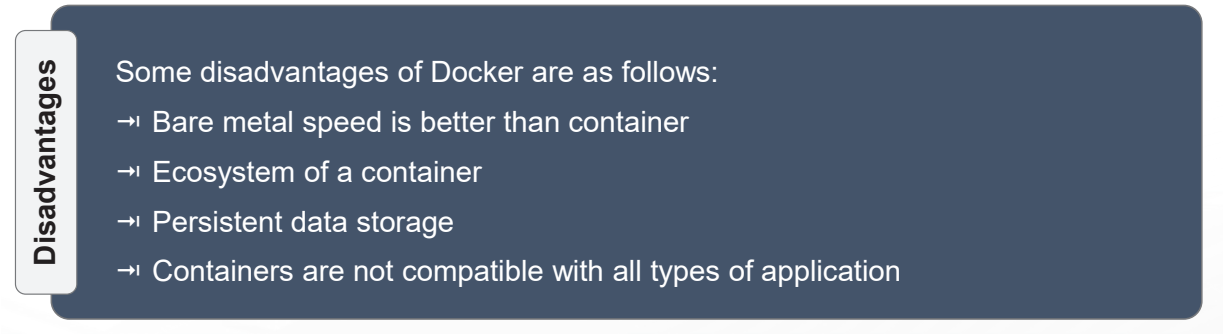
Inform the participants about the advantages of containerization.

Some of the advantages of containerization are as follows:

- **Cost Saving:** Docker uses what it needs, hence the wastage of resources is quite minimal, and having multiple microservices running the return of investment is good in the long run.
- **Standardization and productivity:** Docker brings a standard in whole build and release cycle which is quite reliable and saves time, hence ends up increasing productivity.

- **Rapid deployment:** Docker is widely known for its shipping capabilities, since all sorts of dependencies are encapsulated in one image, hence shipping it and deploying becomes easier.
- **Isolation:** Because of the technologies like c-groups and namespaces, Docker is quite advanced in terms of providing isolation compared to other container technologies. We don't have to worry about the environment anymore or the software configuration of that environment, because a docker image is capable enough to provide the suitable environment to the encapsulated application.
- **Security:** Security is sacrosanct in Docker containers which is a default feature, hence they are quite secure. The security feature can also be taken to another level if you run your processes as non-privileged users inside the container. To meet the security benchmarks, you can add an extra layer of safety by enabling AppArmor, SELinux, GRSEC, or another appropriate hardening system.

2.10.2 Disadvantages of Containerization



Facilitator Notes:

Explain the participants about the disadvantages of containerization.

Some of the disadvantages of containerization are as follows:

- **Bare metal speed is better than container:** Containers are known to consume resources more efficiently than virtual machines. But due to overlay networking, interfacing between containers and the host system, they are still doubted for performance overhead, interfacing between container,s and the host system and so on. To utilise maximum resources and if you want 100 percent performance, then bare metal is a wiser choice than containers.
- **Ecosystem of a container:** Running containers in production needs an orchestrator which is a much needed platform. Usually they are open source, but some container products don't work with other ones majorly due to due to competition among the companies that back them. For example, OpenShift only works with the Kubernetes orchestrator.
- **Persistent data storage:** Containers are designed for running stateless applications which don't need to keep data with them, because once a container dies all of its data is not retrievable unless you save it somewhere else first. There are certain ways to save data persistently in Docker, such as docker volume mount and bind.

- **Containers are not compatible with all types of application:** Containers works well with microservices which are discreet in nature and usually stateless, but if the application is monolithic it won't be wise to run it in docker, since you won't be reaping benefits of docker with monolithic application.

In a nutshell, we learnt:

1. Docker architecture
2. Identified the different environments: (Dev, QA and Prod)
3. Learned about overcoming issues with different environments
4. Virtual machines for dev/deployments
5. Containers for dev/deployments
6. Discussed the advantages and disadvantages of containerization

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:

1. Docker architecture
2. Identified the different environments: (Dev, QA and Prod)
3. Learned about overcoming issues with different environments
4. Virtual machines for dev/deployments
5. Containers for dev/deployments
6. Discussed the advantages and disadvantages of containerization

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