Software Engineering (CEN5035)
Chapter - 5 & 6 Reading Review Exercises

Name: Vikas Chaubey**, UFID:** 3511 5826**, Email:** vikas.chaubey@ufl.edu

Qns.1: Explain the fundamental difference between virtualization using a VM and virtualization using a container.
Answer:

Virtualization: Virtualization is a concept which is used in cloud computing. Virtualization is the process of creating a software-based virtual representation of software applications, servers, storage and networks. It reduces infrastructure expenses and improves efficiency and agility for all size businesses. Virtualization is used for deployment of software applications.

Virtualization Using Virtual Machines (VM): A virtual machine is a computing resource which is used for deployment of software applications and to run programs. The main component of virtualization process is virtual server, Virtual servers include individual operating system and software packages such as binaries and libraries and software application together. These virtual servers are setup on an underlying physical computer hardware. This physical computer runs its own host operating system. A hypervisor is used as a middle layer between the host operating system and virtual servers. This hypervisor emulates the system hardware, and virtual servers running different operating system make use of hypervisor in order to communicate with host operating system. The advantage of using this method is that many virtual machines running different operating systems with different software applications can run on one single system with isolation.

Virtualization Using Containers: In the case of virtual machines each virtual server is packaged with individual separate operating system. Hence virtual machines are very heavy in terms of size and require more computing resources to function. This adds overhead in memory and complexity of the system. Containers can be used to reduce this overhead. Container is a lightweight virtualization technology. Container size is usually in megabytes compared to virtual machines which has size in gigabytes. Containers allow the different servers to share a single operating system and yet provide effective isolation among software applications running within different containers. Containers are setup on a single hardware machine which has its own operating system, these containers share the kernel of the host operating system, binaries and libraries too with read only access. Hence there is no need to have a hypervisor in this system, also the virtual servers do not need to incorporate separate operating system. Hence containers are very light weight and more portable compared to virtual machines.

Differences:

1) The fundamental difference between VM and containers is that, virtual machine is a virtualization technology which virtualizes the computer hardware using the hypervisor on the other hand containers virtualizes the operating system so that multiple applications can run on single host operating system.
2) Virtual machines are heavy weight while containers are light weight since containers do not incorporate individual operating systems with software applications.
3) Containers reduce management, memory and complexity overheads in the system, but virtual machines increase complexity and memory overhead.
4) Even though containers provide good isolation, the isolation provided by containers is not as good as Virtual machine isolation.
5) The time required to build a container compared to a virtual machine, is very fast because they are really small. Development, testing, and deployment can be done faster as containers are small.
6) The applications built using containers are extremely portable that means they could be moved as a single element and their performance remains unchanged on different host systems.
7) Since containers do not use hypervisor hence it can make use of available system resources more efficiently, the performance of a Containers is higher because of higher density and no overhead wastage of resources.

8) Containers provides very good scalability, containers have the ability that it can be deployed in several physical servers, data servers, and cloud platforms.

Qns.2: Explain what is meant by IaaS and PaaS. Explain why the distinction between these categories of services is becoming increasingly blurred and why they may be merged in the near future.
Answer:

Infrastructure as a service (IaaS): It is a cloud computing service in which a service provider provides infrastructure such as compute service, network services or storage service on rent or lease to the customers. a customer or user can then use this rented or leased infrastructure to run an operating system or a software application on the cloud without worrying about the maintenance and operating costs of the physical servers. It eliminates the need for a customer to manually setup and manage their own physical servers in order to run their software application. The service provider maintains the infrastructure so that it is always up and running. In this system Infrastructure is provided to the customer as a service. Other advantage of the IaaS is that customers can get rent or lease servers which are nearest to the geographic locations close to their end users. IaaS service ensures the access to software applications during a disaster or outage. With IaaS, a system could be easily scaled up or scaled down as per number of users change. It also allows the user to concentrate and focus on the core business activities instead of the infrastructure and its maintenance.

Platform as a service (PaaS): It is a cloud computing service in which a service provider provides hardware infrastructure and software tools such as databases, libraries, frameworks etc. as a service to the customers on rent or lease over the internet. Like IaaS service, PaaS cloud service also includes infrastructure such as storage services , servers for deployment and networking services. but PaaS service additionally includes, development tools, business intelligence (BI), middleware services, database management systems etc. PaaS service is designed in order to support the complete software development life lifecycle of web applications which includes development phase, software testing, deployment, managing, and support phases. A software developer can make use of those software tools to develop software applications fast and then infrastructure could be used to deploy these software applications. The PaaS service provider maintains the infrastructure as well as software tools so that it is always up and running and users do not need to worry about the infrastructure and do not need to set up their own software systems such as databases for their web application.

IaaS and PaaS Are Merging:
As the advent of cloud technology and its increased use in the software industry around the world, PaaS was favored more against IaaS because it makes development more convenient for software developers. however, IaaS services are still in demand for certain organization which are more concerned about software security and hence they want to implement their own software solution from scratch by just leasing infrastructure. But as the time passing by demand for integrated IaaS and PaaS services is increasing, A survey done on the current cloud users in the industry concluded that more than 90 percent of the users who require a IaaS service will prefer to choose a cloud service provider which also provides PaaS services. hence this demand for integrated IaaS and PaaS services is driving the next norm of cloud infrastructure adoption and it is speculated that in future most cloud service providers will integrate these two services in order to compete with other cloud service providers to maintain their user share in the market. Some of the leading IaaS companies like Amazon Web Services (AWS) has already started integrating these services, Amazon AWS company which started in 2006 as IaaS cloud service are adding PaaS like features in their services. Also, AWS has market competition with other PaaS-accommodating suppliers such as Google App Engine and Microsoft Azure. Hence it could be seen this demand and market competition will force future cloud suppliers to provide the services of IaaS and PaaS in integrated form.

Qns.3: What are the benefits to software product vendors of delivering software as a service. In what situations might you decide not to deliver software in this way.
Answer:

Software as a Service (SaaS): SaaS is a method of software delivery, in this model a software vendor hosts its software applications on the cloud and provides the software to the users as a service. The software can be accessed by user by simply using internet and web browser either from a computer or mobile. SaaS is a web-based delivery of software to the user.

Benefits to software product vendors of delivering software as a service:

1) **Continuous Cash Flow:** In the SaaS service customer pays for either monthly or yearly subscription of the service or they pay as they use the software service that ensures a continuous cash flow for the product vendors throughout the year as opposed to one time buy models where vendors income varies.
2) **Maintaining Updated Software:** Since the software is managed by the vendors hence it is easy to update the product, this also eliminates the need of maintaining several versions of the same product. The most recent version of the product could be provided to the users. This also helps in maintaining a consistent software code base.
3) **Continuous Deployment:** With SaaS software products could be deployed using continuous deployment, that means as soon as new updates or bug fixes are available, these new versions of products could be readily deployed, this improves software performance and reliability.
4) **Increased User Base with Payment flexibility:** Since SaaS follows a subscription model ,Convenient payment methods and flexibility can attract wide range of users for example the cost of one time buy for photoshop is more than 800$, many users cannot afford to buy the product in one go because it is very expensive but adobe also provides photoshop as a SaaS service through adobe creative cloud, where monthly subscription fees is just 15$.this is more affordable hence more number of users can be encouraged to use it.
5) The vendors could easily maintain early and light version of software for users, this could help in getting feedback from the users and help in fixing bugs in upcoming versions of the product. The lighter versions of software could be maintained for user trials.
6) **Collection of Statistical Data for improvements:** Vendors can collect data from users which could be used to improve the software and its marketing. This data can also be used to market other products to the users by analyzing their preferences or user pattern.

Scenarios when SaaS is not appropriate for software delivery:

1) **User data regulations:** Certain countries have very strict user data regulations for vendors in that case if the vendors not guarantee the compliance with those regulations various businesses would be reluctant to use the software service.
2) **Data transfer and Slow networks**: If the software application requires a lot of to and fro data communication from the client end to the software server and in that case if the user is using a slow internet service, then the software response and performance is limited.
3) **User Data Privacy and Security**: Various businesses such as banks , financial firms , defense organizations deal with very sensitive data hence such organizations may be reluctant to share their data with the software vendors that's why these organizations might not trust the software vendors for their data privacy and security. The user might prefer to implement their own infrastructure, deployment and security layers to use the software.
4) **Data Exchange among different platforms:** If software needs to exchange data with other cloud-based services or local software applications, then it becomes a challenge until and unless other cloud services provide APIs or other means to connect separate applications to exchange data.

Qns.4: Using an example, explain why EU data protection rules can cause difficulties to companies that offer software as a service.
Answer:

SaaS and EU Data protection Legislation: Under the 1995 data protection directive which is now transposed into a national law in 27 different European member states. The transfer of user's personal data such as name, address, IP addresses and Credit card details, is not allowed outside the European Economic Area (EEA) which includes European union member states, Iceland and Norway. The transfer can only happen if certain conditions are met. Transfer of data means the handling of user's data in some manner in non-EEA countries. Over the years EU data protection laws are updated to be more effective and efficient. The EU General Data Protection Regulation (GDPR) which was passed in EU parliament in 2014, replaces the 1995 Data Protection Directive 95/46/EC and it reshapes the way organizations across the region approach data privacy.

Generally, when a European company signs up to a SaaS cloud service then it is also responsible for the handling of user data by the SaaS provider. Generally, SaaS provider should be held responsible for data handling, but EU data protection law binds the user company to ensure that the SaaS service is compliant with the data protection policy of the union. And the data is not transferred to non-EEA countries or servers without fully complying to the data protection laws.in that case if the vendors do not guarantee the compliance with those regulations various businesses would be reluctant to use the software service. Compliance with the GDPR data regulations increases the cost of SaaS services significantly. Which is the main challenge for SaaS services. For example, AWS services compliant with GDPR regulations are much costlier. Apart from that complying to GDPR regulations makes it hard to run the fully compliant SaaS business.

In a gist GDPR has following regulations for SaaS services:

1) Collection of data should be done when necessary it should be kept as short as possible and it should be shared with as few entities as possible.
2) The data handling policy should be transparent and comply with GDPR
3) Users should be allowed to choose what they want to share and what they do not want to share. They should be able to retrieve their data and even able to delete it permanently.
4) In case of security breach, the SaaS provider needs to notify the individual and authorities.

Complying to all these rules might pose serious challenges in front of the SaaS providers. Below are some examples where SaaS service can violate GDPR's data regulations:

1) Many websites use google analytics in order to collect visitor statistics. Although google analytics does not take user data but it utilizes randomly generated client IDs, but once this client ID is connected with the service it is not anonymous to the service because users IP address is sent to google as part of network request , while google claims that it does not store the IP address , the service is still considered to be violating the data protection rules.
2) Another issues is access logs and error reporting, these logs are very important for problem debugging and defending application against cyber-attacks hence they cannot be ignored when a user accesses the SaaS platform their IP address user agent and Http data is collected and stored before even taking the customer consent, that violates the GDPR data regulations.
3) If SaaS service uses third party payment gateways, then how user's payment information is being handled becomes important for the SaaS service which is integrating the payment gateway with the service.
4) In case where service incorporates single Sign on (login with Gmail, Facebook accounts) with the application then in that case it is important to understand how this user information is being utilized by those domains. That impacts the SaaS service compliance directly.

Qns.5 What are the key issues that have to be considered when deciding on whether to implement a multi-tenant or multi-instance database when software is delivered as a service?
Answer:

Multitenant Database: In a multi-tenant database system all the users are served by a single instance of multitenant database. A multitenant database is partitioned so that users can access and store their own data. The schema of multitenant data base is universal. For all users and is defined by the SaaS service. Each user has its own tenant identifier or unique ID which makes sure that user is logically isolated from the data of other users.

Multi Instance Database: In case of Multi instance database, all users are served by a separate individual copy of the database. Each user can store and access their data from their respective database copy. Separation of user database copies provide data isolation among users.

Issues to be considered:

1) **Security:** Multitenant databases contains all the user information in the single instance hence there are chances of data leakage from one customer to another. However, in the case of multi instance data base since all users have their own database instance there is no chance of data leakage among users. similarly, in case of multitenant database if the database is compromised because of some security breach then whole user information is compromised however in case multi instance database it will not happen.
2) **Maintainability:** Multitenant databases are easy to maintain because they have single instance hence if any kind of update or patch fix has to be done on the database system it could be done easily, however in the case of multi instance database update and patch fixes will have to be done on all instances which will take more effort.
3) **Resource Availability:** Multitenant databases will use the system compute resources effectively because only a single instance of database is maintained. on the other hand, multi instance database could prove to be resource heavy and might add extra resource overhead.
4) **Flexibility:** Multi instance databases are more flexible because individual instance database schema can adapt to users need. On the other hand, multitenant database is rigid because there is only one single schema for all the users, and it cannot adapt to individual users requirements.
5) **Complexity:** Multi instance databases are less complex than multitenant databases because in multi-tenant databases all the user information is stored in a single instance of database.
6) **Cost:** Setting up a multi instance system is costlier than a multi-tenant system. In order to setup multiple instances of databases multiple Virtual machines are required which increases the associated cost.
7) **Scalability:** Multi instance systems are more scalable than multi-tenant system. If the user load increases, then in case multi instance database new instances could be provisioned to handle that additional user load hence multi instance systems are more scalable. Also, different servers could be customized as per the needs of the users.
8) **Transaction requirements:** If the database is supposed to support ACID transactions i.e. the database is supposed to be consistent all the time then in that case multi-tenant database system should be preferred or VM based multi instance database could be used.
9) **Database Size:** For large data base a multi-tenant database should be preferred as it could be very costly and effortful to implement multi instance database.
10) **Interoperability:** if user has different schemas then in order to improve interoperability the multi instance database should be used. which allows to customize individual schemas of the user specific instances of database.

Qns.6: What are the advantages of using services as the fundamental component in a distributed software system?
Answer:

Software services: Distributes software system is developed using various types of software technologies, programing languages and platforms for example in a web application back end could be written using JAVA, .Net while the front end could be written using Angular JS , while the system can also use AI components which are written in python or JULIA programming languages. In that scenario where the software system components are so diverse and contains so many heterogeneous applications it becomes a challenge to obtain a correspondence among these components so that they can communicate with each other and work together. That's is where software services come into picture, software services enable different software components in distributed software systems to communicate with each other and function together. A software service is a software component which could be accessed over internet, for a given input a service produces an output. This software service could be accessed by other components of distributed system through an interface and hence the implementation of the service is hidden. The software services do not maintain any internal state. Since there is no local state hence software services could be dynamically allocated from one virtual server to another and they could be replicated across various servers.
Example of web services: Soap based web services, Rest based web services.

Advantages of using software services in distributed software systems:

1) **Interoperability, Application and Data integration:** software services in a distributed system which has various diverse and heterogeneous components based out of different platforms and technologies, provide interoperability among those components so that despite having different technology base they can still communicate and function together to achieve a common goal. This interoperability is achieved using vendor, platform and language independent XML technologies and HTTP as transport protocol for communication which enables different software components to communicate with each other.
2) **Software modularity and flexibility:** software services are loosely coupled and hence they increase the modularity and flexibility of the software system
3) **Reduced Integration Cost:** Software services reduce the integration cost of diverse technical components in the distributed software system.by using service APIs any type of component can communicate with other components in the system, separate integration of components is not required.
4) Software services encourages and make use of existing technology and enables current technologies to function together.
5) **Encapsulation and security:** software services are encapsulated that means their implementation is abstract and no visible from outside. The services are accessed through an interface hence the components which are using them do not get to know about the inner implementation of the software service. That improves security.
6) **Software Services are standard:** software services are built upon HTTP and XML protocols which are standard hence present a wide use case.
7) **Versatility:** software services are versatile i.e. they can be used by software components or other software services as well as by human being through a web-based client interface. A client can combine multiple software services as per requirements.
8) **Code -Reuse:** software services could be used by other software services, this eliminates the need to implement a customized functionality separately in a software component if any other software service already implements this functionality, instead software component can simply call that software service and use it.

Qns.7: What are the principal problems with multi-tier software architectures? How does a microservices architecture help with these problems?

Answer: A multi-tier architecture is a client server architecture in software systems in which functionality of the software system is physically divided and separated using different layers. There are various possibilities of multi-tier software architecture such as 1- tier, 2- tier, 3 tier and n- tier software system. For example, a 3 tier architecture has a presentation layer (User interface), business logic layer, data retrieval layer (data base).

Principal problems with multi-tier software architectures:

1) **Performance:** As the components and layers in the multi-layer software system increase the communication endpoints in the software system increase that is a greater number of components need to communicate with other components in other layers over the network. N- tier architecture introduces extra latency due to network calls among several tiers.
2) **Security:** in multi-tier architecture each layer needs to be protected and secured against the possible security concerns. In order for the whole system to be resilient and secure from outside attacks each layer of the system needs to be secured.
3) **Maintainability:** As the multi-tier architecture involves a greater number of layers and software components in the system, maintaining the software system becomes difficult as the number of layers increase hence the complexity also increases.
4) **Reliability:** In order to function as per desired specification all, the layers in the multi layered architecture has to work in sync, if one layer fails to function properly the whole system faces issues. Hence reliability is also important among all the layers of multi-tier system.
5) **Scalability:** As the user load on the system increases the whole system has to be scaled up that includes scaling up all the layers of software system for example increasing UI servers to handle more load, have more back end servers and more number of database servers to handle increased load.
6) **Updates:** when a new update has to be made to the system all layers in the system has to be updated in order to reflect the change.

Microservices Architecture: A microservice architecture is an architectural style which structures a software application as a collection of microservices. This architecture solves many problems associated with multi-tier software system. a microservice is a small scale, stateless, self-contained process which run and function separately and have a single responsibility.

Advantages over multi-tier software architecture:

1) A microservice architecture is not a layered architecture all microservices exist as separate entities hence this type of software architecture is highly maintainable and testable.
2) Another advantage is that when a change in the software system has to be made then only the respective microservice has to be updated not the whole system.
3) Scalability is another important advantage, sometimes software systems are required to scale a specific functionality in the system hence in this scenario multi-tier architecture has to scale up the whole system but in case of microservices architecture a specific microservice could be scaled up without changing the remaining system.
4) A microservice architecture is not a layered architecture hence it does not have single point of failure as opposite to multi-layer architecture where if one layer fails the whole system is affected, in case of microservice architecture if one service fails other services can still function without any issue. Hence this architecture is more reliable.
5) Loose coupling of microservices makes the software system more modular and flexible.
6) Microservices are containerized hence they are isolated from other services and components which increases security. that also means those services could started and stopped without effecting other components in the system.

Qns.8: Explain why each microservice should maintain its own data. Explain how data in service replicas can be kept consistent.
Answer:

why each microservice should maintain its own data?
The main advantage of a microservice architecture is that it improves the agility of the software system. When a software system is decomposed into microservice architecture then ideally microservices are supposed to be developed and deployed independent of each other. In order for microservices to be independent they have to be loosely coupled. In order to make micro services independent and loosely coupled they should not be dependent on each other especially in terms of data. Hence each micro service needs to maintain its own persistent data and it should only be accessible through API.

Few ways to keep microservice data private:

1) Using private tables in data base for individual services
2) Using individual schema for each service which is private to that particular service
3) Using a separate database server for each service.

However, in real world the data independence is impossible. There are always overlaps between data used among the services. Hence software application should be properly designed with adequate measures to maintain data consistency. In order to keep data consistent certain following measures can be taken:

1) Designing microservices with as little data sharing as possible
2) If data sharing is unavoidable then keeping data access limited to read only
3) If services are replicated, then using the mechanism to keep the database copies among the replicas consistent.

how data in service replicas can be kept consistent?

Replica data Inconsistency: when several replicas of the same microservice execute concurrently. These replicas might update their own copy of database. Eventually all the databases need to be made consistent so that all the replicas are working with the same data.

Solution: Replica data consistency problem could be resolved using concept of "eventual consistency", eventual consistency ensures that the databases of replicas will become consistent eventually at one point. Eventual consistency could be implemented by using transaction logs. Whenever a microservice replica makes a database change then this change is recorded on the pending updates log. other replicas can look at this log and can update their own database instances to make their databases consistent.

Other Replica data inconsistency solutions:

1) **Compensating transactions:** replica data inconsistency could be resolved using compensating transactions, it's a transaction which reverses the previous operation hence in case of any microservice failure of ordering service the compensating transaction could be created in order to undo the previous operation.
2) **Reconciliation:** If the system which is responsible for calling the compensation trigger crashes in middle of the process in that case. In order to find crashed transactions, apply compensation or resume operations, the data needs to be reconciled from multiple services. This technique is used in financial system, reconciliation ensures that two records are in agreement with each other at certain point of time. Using this technique, the data from multiple microservices could be reconciled on any specific action trigger.

Qns.9: What is a timeout and how is it used in service failure management? Explain why a circuit breaker is a more efficient mechanism than timeouts for handling external service failures.
Answer:

Service failure: A microservice could encounter an internal or external failure. An internal failure is a failure which occurs within the service however an external failure is induced by external factors because of which the service might become unresponsive. In order to handle the service failures service failure management is used in the software system.

Timeout: A timer is a counter which is associated with the service requests made by the microservices. The timer starts counting the elapsed time as soon as the request is made by the service. When the counter reaches an already predefined value then the microservice which made the request assumes that service request has failed. This is called timeout in service failure management. The problem with timeout approach of service failure management is that whole systems is delayed by timeout value and hence the systems becomes slow.

why a circuit breaker is a more efficient mechanism than timeouts for handling external service failures?
In case of external failure, the service becomes unresponsive because of external factors, if timeouts are used by the service failure management then service which makes the request has to wait till timeout till the request is considered as failed request. however, the problem with timeouts is that they slow down the software system by introducing halts in the system. This problem with timeouts could be resolved using the circuit breaker , circuit breaker works like electric circuit breaker hence when a failed service is detected circuit breaks the connection to the failed service immediately hence the requesting service does not need to wait till timeout , it immediately knows that request has been failed. This does not introduce halts and delays in the system. Hence circuit breakers can handle the external failures more efficiently.

Qns.10: Why should you use continuous deployment in a microservices architecture? Briefly explain each of the stages in the continuous deployment pipeline.
Answer:

Continuous deployment: It is a software release process in which whenever a new update for the software application is ready for deployment by the development team. And once the the code is committed to the code repository, then deployment process is automatically triggered, this process uses automated testing such as unit testing, integration testing and acceptance tests to validate whether the code base is correct and stable for immediate autonomous release. If the codebase passes all the tests, then the code build is performed, and the built artifacts or deployable build file is deployed on the target server automatically. This whole process is automatic and does not require any human intervention, if continuous deployment process encounters any single failure during testing of codebase then the release process is stopped.

The main advantage of the continuous deployment is that the new updates to the software systems could be readily deployed as soon as updates are available.

Why should we use continuous deployment in a microservices architecture?
Continuous deployment with microservices is very effective and efficient deployment strategy. In a microservice architecture system each microservice is deployed independently hence when any update has to be made to any specific service it could be updated and redeployed without effecting other services. As soon as the new updates are available for the microservices Continuous deployment could be used to automatically deploy those updates. Since microservices are independent of each other the deployment time for individual service is very less compared to monolithic systems hence it makes more sense to readily deploy updates as soon as they are available. Continuous deployment can help in automating this process. Also, now days the development teams are also responsible for deployment of microservices in that case CD pipelines can automate the process and remove the burden of manual deployment from the team. Another major advantage of CD pipeline is that when microservice

system is very large which contains hundreds of microservices in that case manual deployment is very difficult and could be faulty with human errors but with the help of CD pipelines even large deployments can also be done accurately. Another major advantage of using CD is that it runs all types of testing on code base in order to check its validity only if the code base is valid and pass all the tests then the build is deployed on the target server otherwise the release process is stopped and failure notifications are sent to the team. This automation of deployment makes sure that microservice updates are readily deployed as soon as they are available, and it also makes sure that codebase and builds are production ready by running several tests to ensure its performance.

Different stages in the continuous deployment pipeline:

1) A development team commits the code into code repository as soon as the updates are ready.
2) **Unit testing:** As soon as the code is committed into repository continuous pipeline is triggered and unit test cases are run for individual code blocks in the updated code base. If unit testing is successful for all the unit test cases, then the deployment process proceeds to the next stage otherwise it fails**.**
3) **Integration testing:** once the unit testing is successful then the modules which need to be tested together are built then integration testing is done. integration testing ensures whether all modules together are working as expected or not. If integration testing is successful for all the test cases, then the deployment process proceeds to the next stage otherwise it fails.
4) **Build:** In this phase the microservice will be containerized
5) **Deployment:** The containerized service will be deployed on the target server
6) **User Acceptance testing:** This is the final phase of testing performed in the pipeline to ensure that final build is working properly and ready to be moved to production if the build passes the user acceptance testing then the deployment is finalized or if the user acceptance testing fails then release process is stopped and the notification is sent to the deployment team.
7) **Final Release:** if the build passes the user acceptance testing then the current working service is replaced with the new build in the production server which starts giving services.