**What is DevOps:**

* DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



* Devops is a software development approach which involves continuous development, continuous testing, continuous integration, continuous deployment and continuous monitoring of the software throughout its development lifecycle.
* It is a combination of tools, philosophies and practices which allows faster, reliable software releases and frequent releases as well.
* Devops integrated developers and operations teams to improve collaboration and productivity by automating infrastructure, automating workflows and continuously measuring application performance
* Devops is mainly used for fast deployment and delivery and fast feedback as well. We used tools for this. Getting the code from developer, test it and deploy.

**Integration:**

* Generally, we use Jenkins to compile the code. If there is any error, then the tool raises an alarm or sends an email alert. If no issues, it goes to the immediate testing. We run test cases written by testing team. If it is success, we need to create package and package will be deployed in DEV environment and ready for testing again. The process up to package creation is called integration process.
* There will be continuous integration with devops.
* CI process is automated with Jenkins

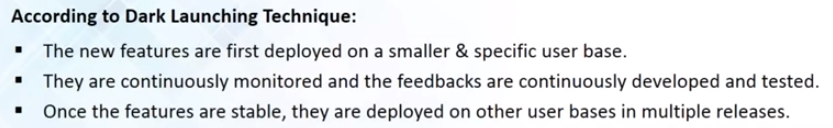
**Continuous Deployment or Delivery:**

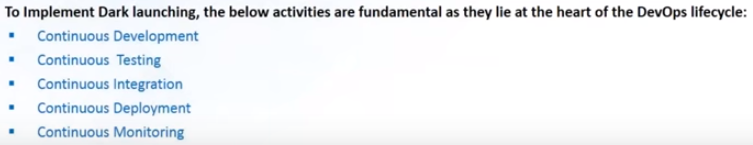
* Deploying the package in server is called deployment/delivery. With devops we can able to do CD.
* Deploying the code in PROD needs an approval. The process with approval is called continuous delivery and the process without approval is called deployment which means deploying in test environment.
* Because of microservices, the need of docker is increasing

**Incremental Development:**

Developing the code in small pieces but not at one go is called incremental development.

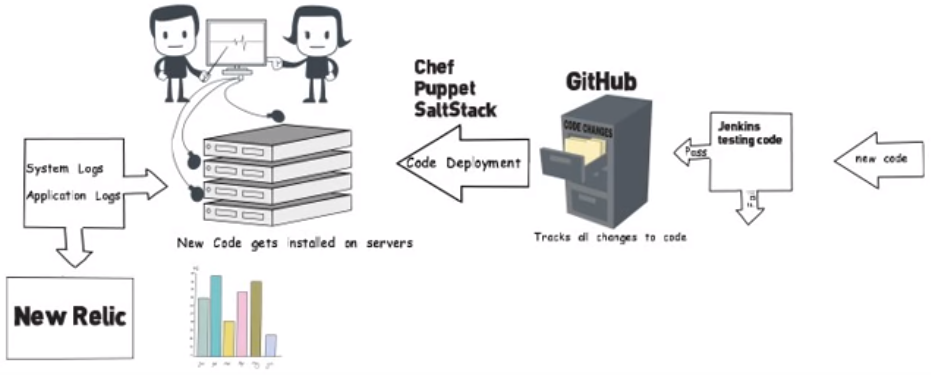
**Dark Launching Technique:**



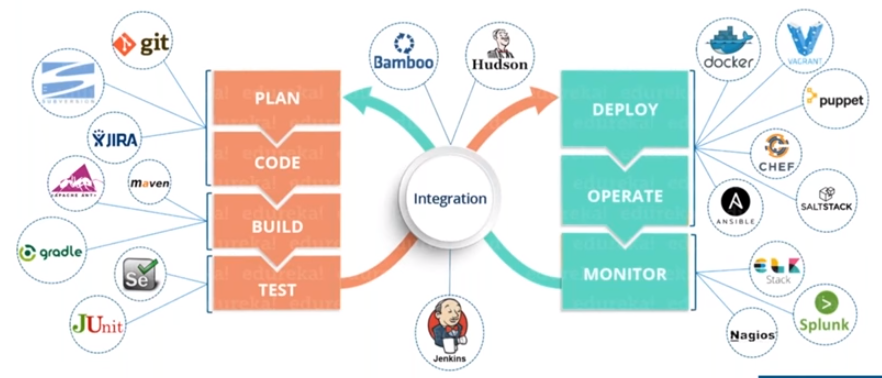


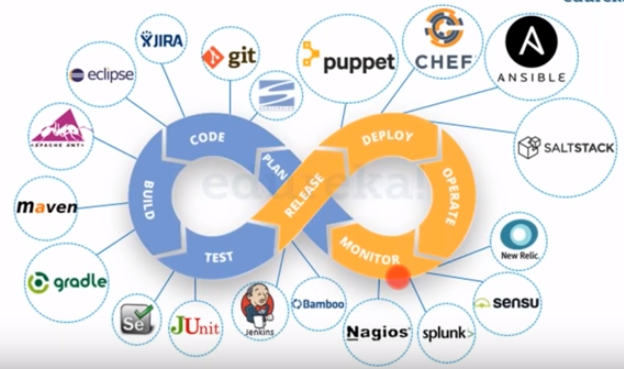
**Devops Tools:**

* Git
* Maven
* Tomcat web server
* Jenkins (Advanced concepts)
* Docker (Advanced concepts)
* Docker, swarm (advanced concepts)
* Ansible (advanced concepts)
* Nagios
* SonarQube
* Nexus
* Splunk
* Linux essentials for devops

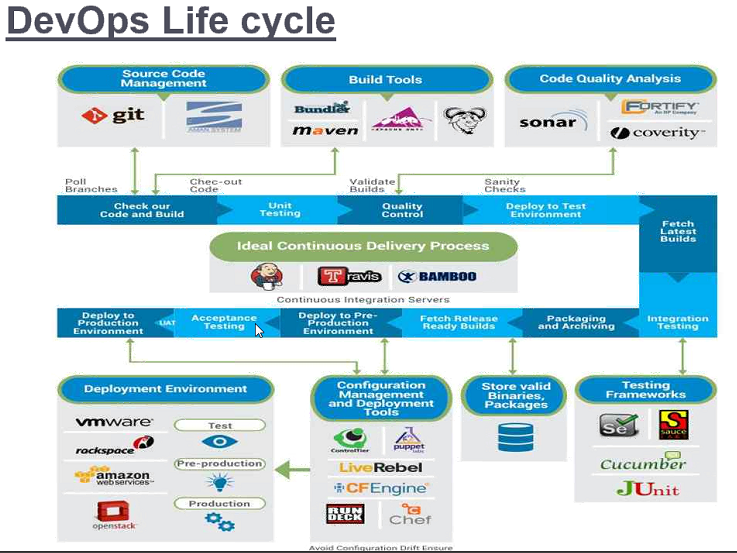


* Configuration management tools are ansible, puppet, chef. Installing packages and configuring, we used these tools to automate.
* Monitoring tool is Nagios
* **Example**: If we have 100 servers and we got to know that there is something wrong, we no need to go and check all the servers if we have installed Splunk in all the servers. It collects all the logs from 100 servers and places in Splunk as single file. We can just go there and search for the error in that file. Splunk is for log management.





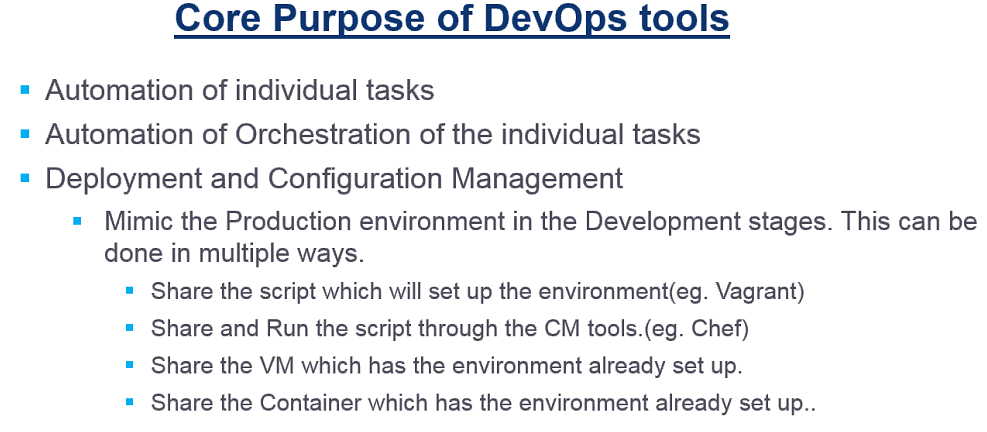
* We do continuous integration everyday



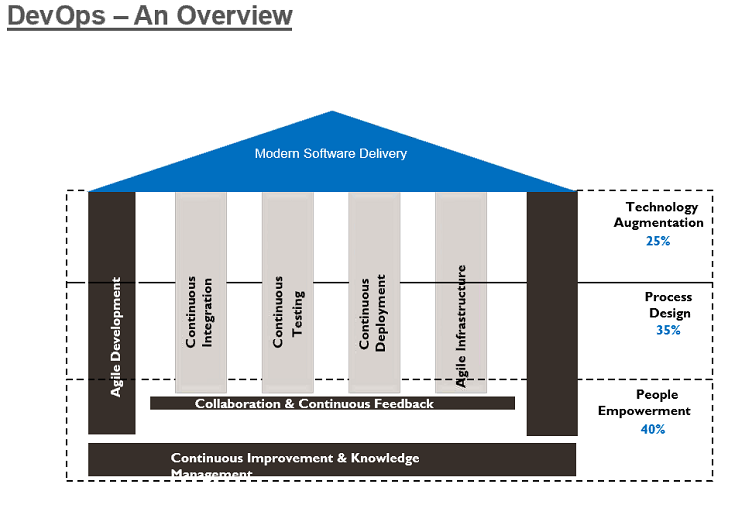
* In devops, if we write any scripts for build or test or code analysis, it needs to be shared in common repo git, so that others can get those too
* The packages, binaries will be deployed into artifactory or some other repos
* We can store test, preprod(stage), production environment packages in artifactory
* Configuration management tools can pick up the files from artifactory and put it on respective environment
* In chef, we write cook books and uploaded in chef server, then the clients will automatically download them and run it. We no need to do it manually like vagrant
* Chef is a domain specific language, we don’t need of shell scripts. It is high level language
* With high level language, we can use it to any os, but assemble level language is used for particular CPU
* Here, developers should take care of production and admins needs to have coding knowledge as well
* Automation might be on individual tasks or orchestration tasks
* Jenkins is orchestration automation tool

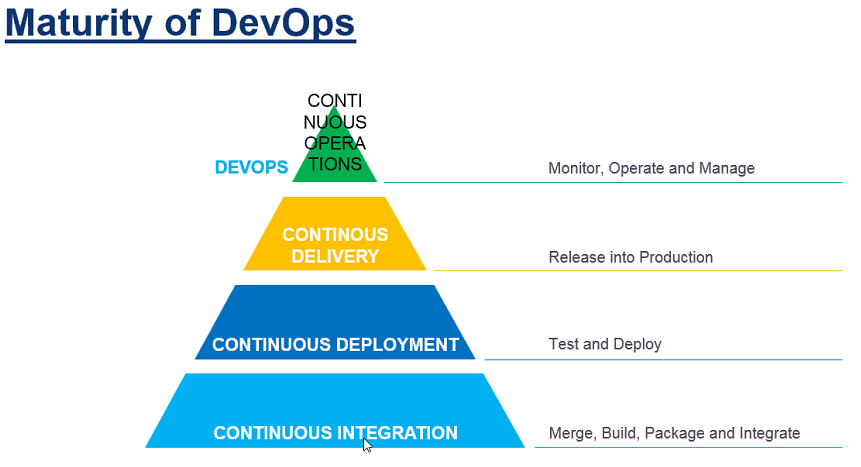


* Mimic production means we can copy the errors from production to preprod and solve there. Then we can move it to production

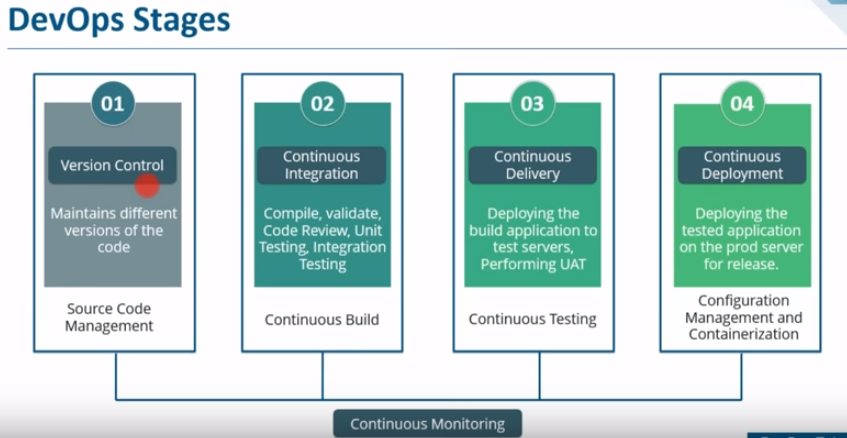


* If a person sets an environment, he can share the vagrant script to others, so that others can get same environment by running the script manually
* But I chef, we can do it automatically
* Or we can share the virtual machine. VM is heavy in terms of memory
* We might use some other tools in real time, but the concept is same



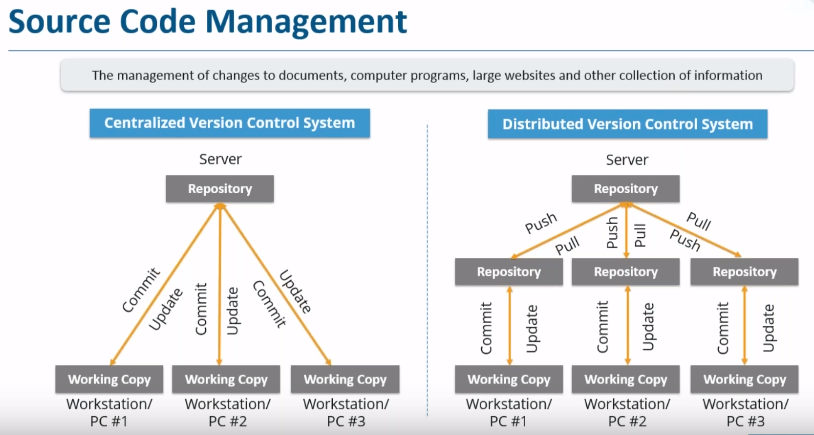


**Devops Stages:**



* Multiples developers might be developing multiple code. For example, developer A committed a code and that code causing an error. How could we know which developer committed the code and at what time the code has been committed? And where was that code editing happened? All these questions can be answered once we use version control softwares like git, subversion etc.
* Continuous integration is building the application continuously. We do compile, validate, code review, unit testing, integration testing and package creation under this.

**Source code management:**



* Suppose multiple developers working for an application and a bug has occurred. How will we know which commit has occurred the error and how to revert to previous code? That’s why source code management tools were introduced. One is centralized version control system, and another is distributed version control system.
* CVCS stores all the files in central server and enabled the team collaboration. It works in a single repository which allows used to access central server.
* Developers can update or commit in shared repository which connects to each developer workstation. Every operation gets performed on central repository.
* SVN (Subversion), CVS, IBMs Clear Case, Microsoft TFS, Git are source code management tools
* SVN is centralized source code management tool.
* Git is DSCM tool. It is fast & distribute. Git is comprehensive tool, having lots of features.
* SCM tools are referred as version control tools

**Advantages:**

* If anyone commit in repository. There will be always a commit id and commit message. So, we can know which person made commit and at what time, so that we can rollback if we want

**Disadvantages**:

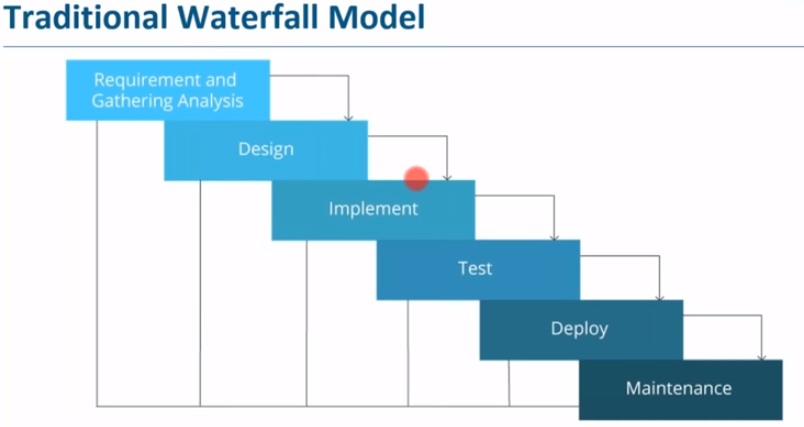
* First, it is not locally available. We need to be connected to some sort of network. If something happened all the data will be lost.

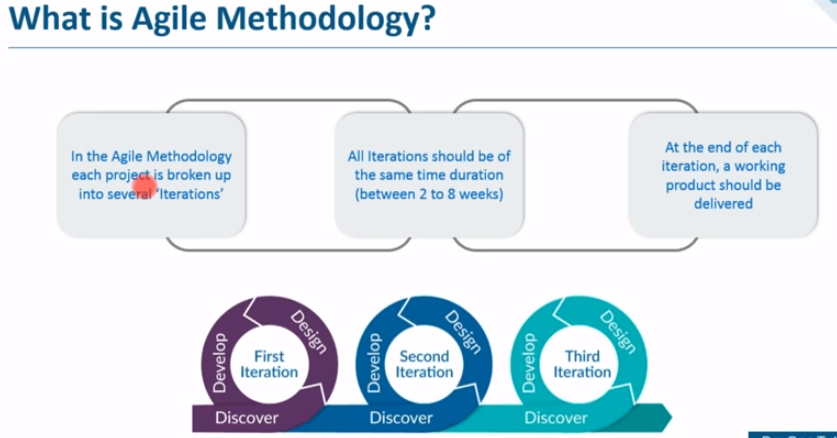
**DVCS:**

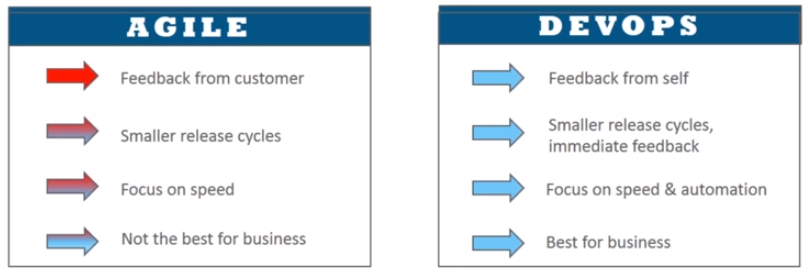
* In DVCS, everyone has their own local repository or clone of main repository. They can commit or update in local repository. They can get the copy from main repository called pull and update the changes is called push.
* Here the tool need not to access main repository all the time. It can use hard disk and there is no need of network connectivity.

**Waterfall vs Agile vs Devops methodology:**

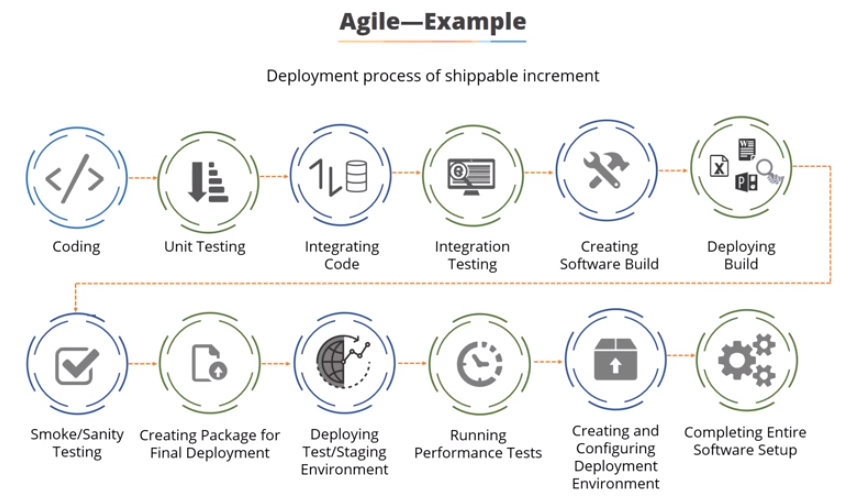
* In waterfall methodology, the phases were like requirement gathering, architecting, developing the code, test it and deliver it. It was taking very long time to do this all because the application goes to the next stage only when the last stage completes.
* For example, if we found a bug while testing the code, it is very difficult to go back and change the code
* Later in agile methodology, the software would not deliver at one go. It broken into smaller pieces and deliver it.
* Agile methodology is a practice that promotes the continuous iteration of development and testing throughout the software development lifecycle of the project.
* Agile is old and devops is recent one. Both are fast
* In agile, there are people developing, testing and operating. But in devops it is like merging the roles and the devops engineer even involves with operations and monitor the application performance.

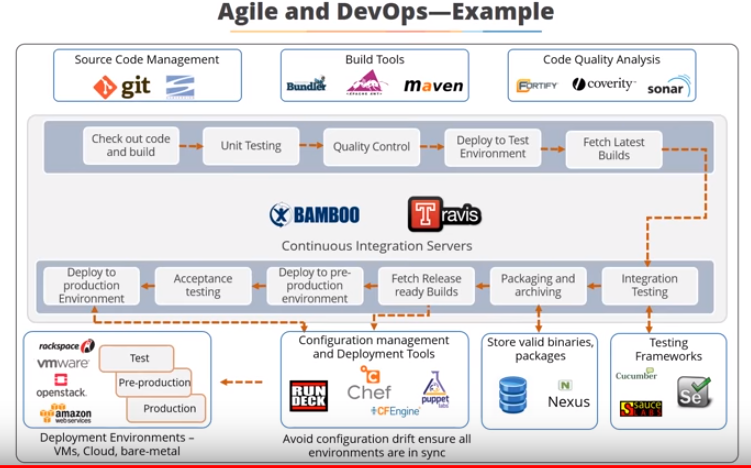






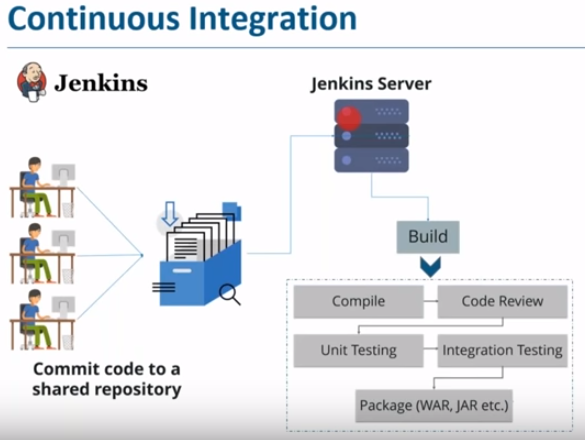
**Examples for Agile & DevOps process:**





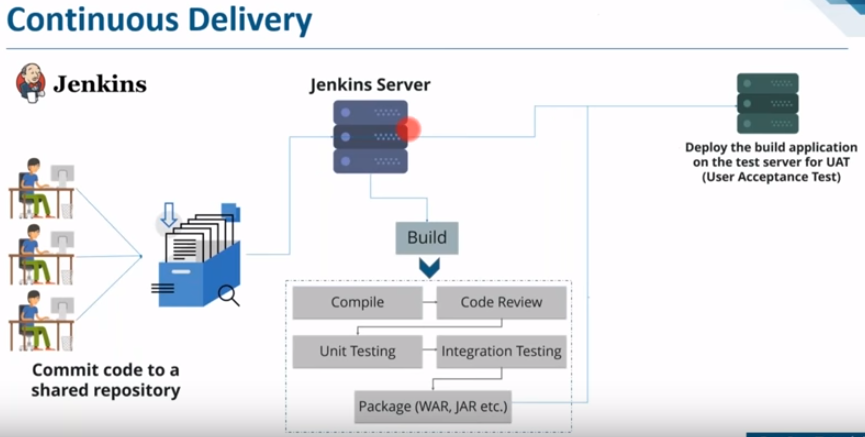
**Continuous integration:**

* The moment developer made changes to the code, Jenkins will pull the code, it will build it and deploy it to the test server for end user testing
* If there is an error in build, we can know which commit caused the error. We don’t need to go to the entire source code of the application
* Even in testing, if we found any issue we can know which commit caused the issue



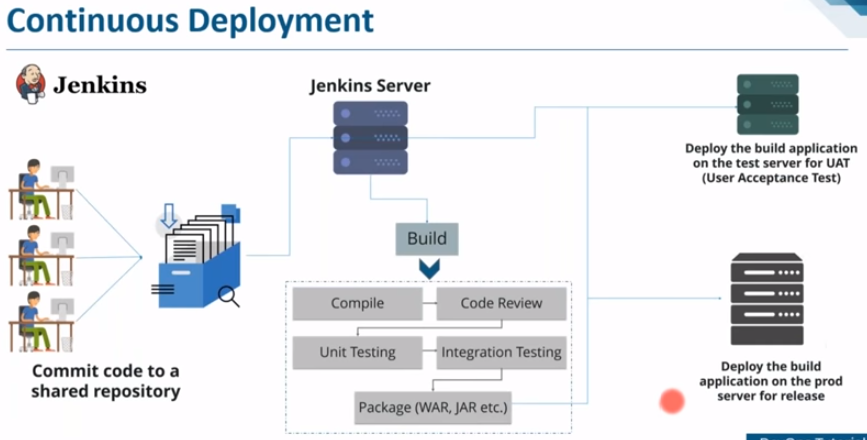
**Continuous delivery:**

* It is nothing but taking the integration to next step



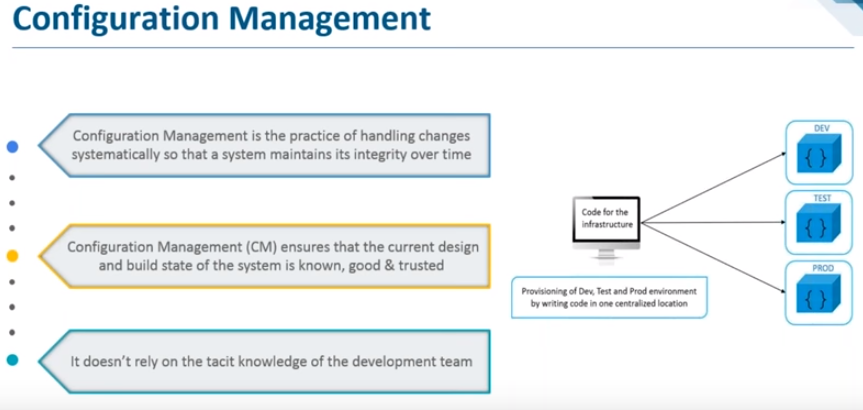
**Continuous deployment:**

* It means deploying the code on prod servers after testing

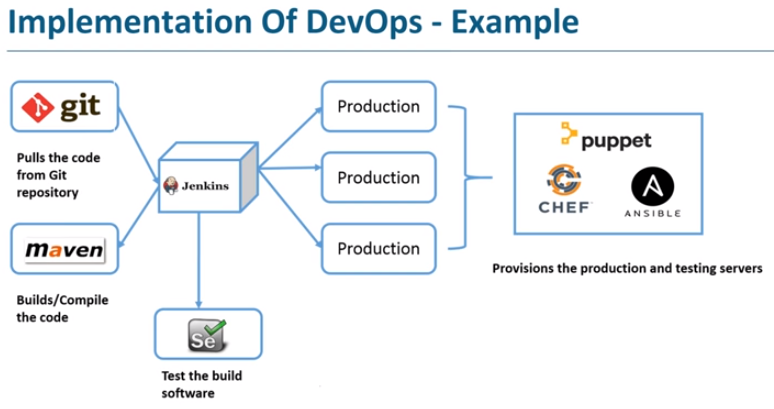


* Checking and functioning the small pieces is called unit testing and combining those multiple pieces and testing them is called integration testing
* Jenkins is a continuous integration tool

**Configuration management:**



Even after deployment, we can revert to old stage of application



**Docker in devops:**

* Developer writes code in docker file. Once the docker file is ready, we can build the docker image. With those images, we can create as many containers we want, containers are nothing but the environment where the application runs
* We can upload the docker image on docker hub. From there various teams like staging or production can pull the docker image and prepare as many containers they want
* Docker is nothing but a git repository of docker images
* Automatic continuous delivery is not a good practice as even after testing the UAT, there might be many things to check before deploying to PROD

