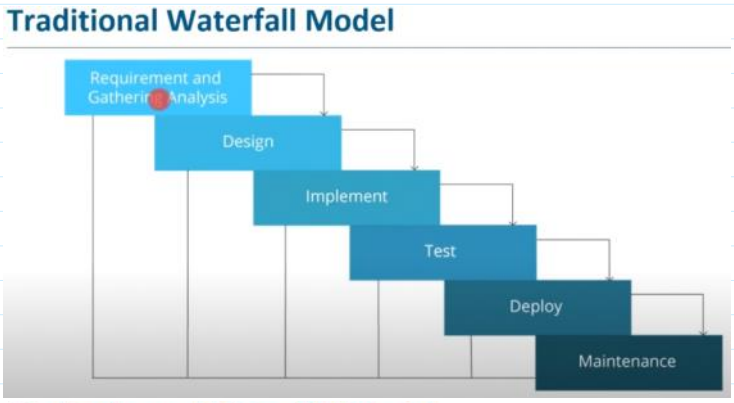
DevOps Tutorial For Beginners | DevOps Tutorial | DevOps Tools Tutorial | Edureka

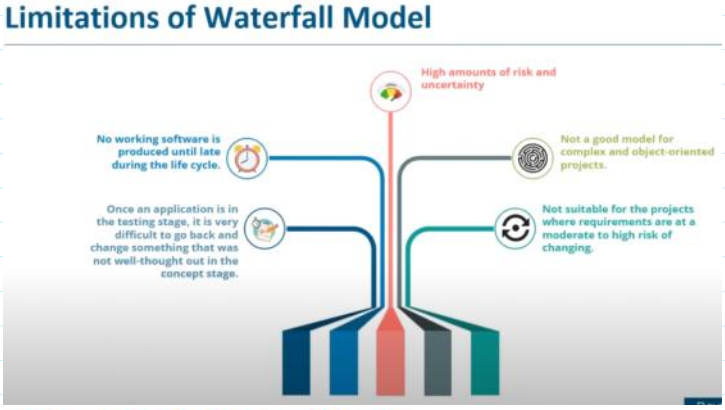
Monday, April 20, 2020 6:12 PM

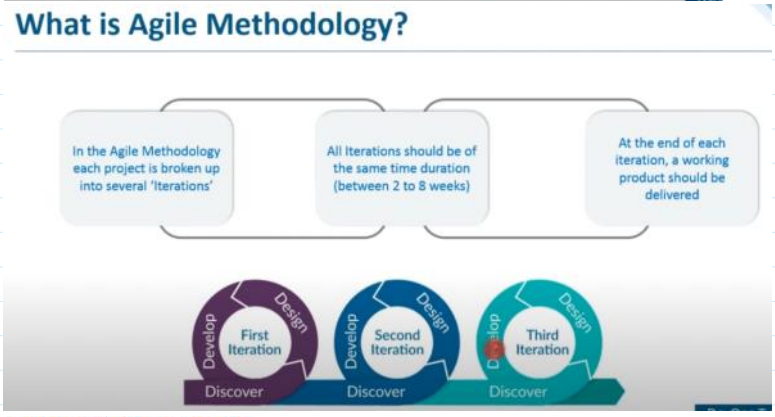
DevOps Tutorial For Beginners | DevOps Tutorial | DevOps Tools Tutorial | Edureka
edureka!



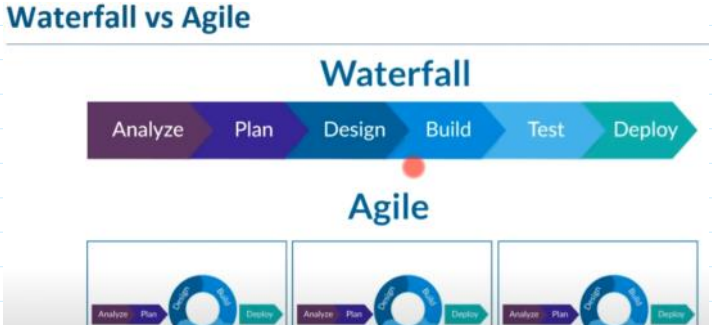
1. Why We Need DevOps

- Waterfall model: linear sequential methodology
- Waterfall does not allow for iteration
- Disadvantages:
  - Difficult to go back and change something once testing is reached
  - Not optimum to use where project requirements are prone to changes
  - Not good for complex OO projects

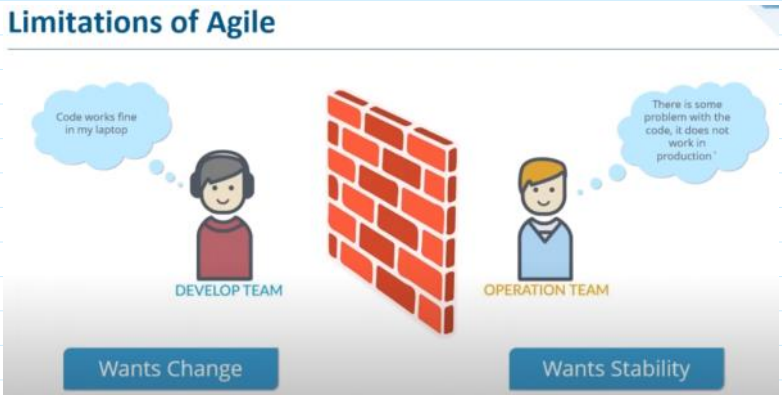




- Agile is a practice that promotes the continuous development and testing throughout the software development life cycle of the project



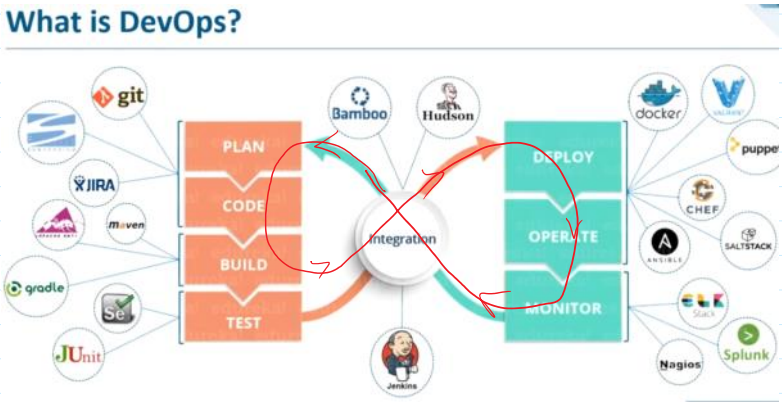
- In the Agile methodology, the design - build - test part is happening continuously.
- There are several testing that happens in this stage. Once final testing is done it is deployed.



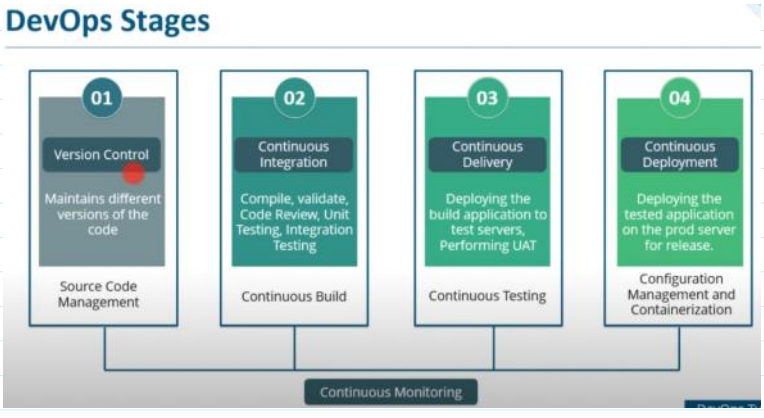
1. The development part in an Agile method is continuous, but the deployment was not continuous. Dev team and Ops have conflict
2. Notice that the design-build-test segment part of the cycle is continuous, but deployment was still linear

What type of project is suitable for waterfall model and agile model
- Projects that have moderate risk of changes in requirement

The Solution: DevOps - A methodology!
- DevOps bridges the gap between development and operation
- Term of for a group of concepts that have catalyzed into a movement and are rapidly spreading throughout the technical company



- DevOps is a practice of operation and development engineers participating together throughout the entire software life cycle - from design, development, deployment.
- DevOps is characterized by operation staff making use of the many of the same techniques as developers for their systems work

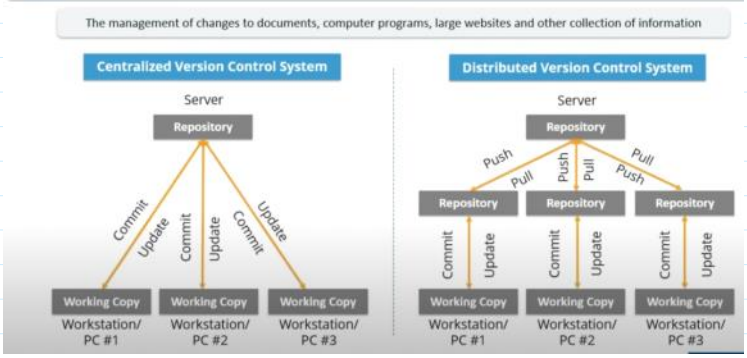


1. Version control - maintaining different version of the code. How will I manage source code? Which developer made a commit that is creating a code? Tool: GIT
2. Continuous Integration - building application continuously. If there is any changes made in the source code, a server should be able to pull that code and prepare that build. Not just compilation. Tools: Jenkins
3. Continuous Delivery - once the app is build, it will be deployed for testing.
4. Continuous Deployment - not a good practice. Multiple checks that you want to do, therefore this is not preferred to be automated

- Once live, it will be continuously monitored. Tools: splunk

Jenkins: Whenever a change is made, it will create a build to be deployed

Source Code Management

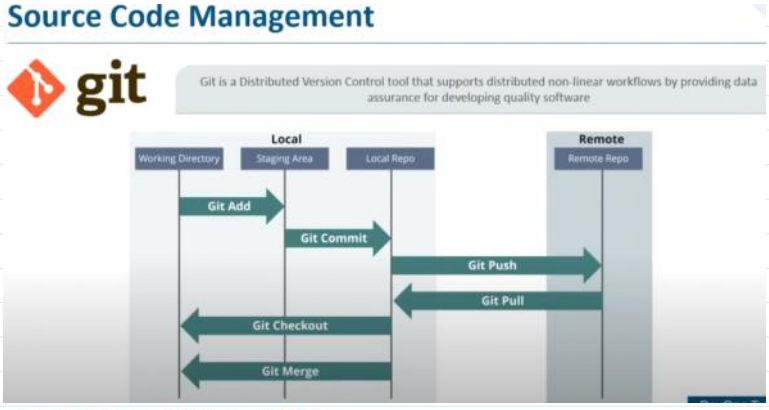


Two types of approach

1. centralized distribution
   - Uses central server to store all the files and enables team collaboration.
   - Works in a single repository to which users can directly access the central server.
   - Every developer has a working copy, if they want to make any changes, they can make a commit or update in the shared repository.
   - The repository can be local or remote which is directly connected to the programmers workstation
   - Every action is made directly on the central server of the central repository
   - **Advantages:** commit information (ID, message, time, place, programmer responsible) is visible which makes it easier to revert back if the new commit has a bug.
   - **Disadvantages:** not available locally, meaning that you always have to be connected to a network to perform an action. Also, once the server crashes of gets hacked, everything goes down. Many companies do not prefer this approach

2. Distributed
   - These systems do not necessarily require a central repository. Each contributor will have their own local repository which is a copy of the main central server.
   - They can update their copy of their repository using pull operation and they can affect changes using push operation
   - **Advantages:** all operations beside push and pull happens very fast because it access the local hard drive and does not need internet connection. Commit to changes to local repository, review it, and once all finished, multiple changes can be pushed onto the main repository at once. If data is lost in the main repository, it can be retrieved from the local repository

Source Code Management

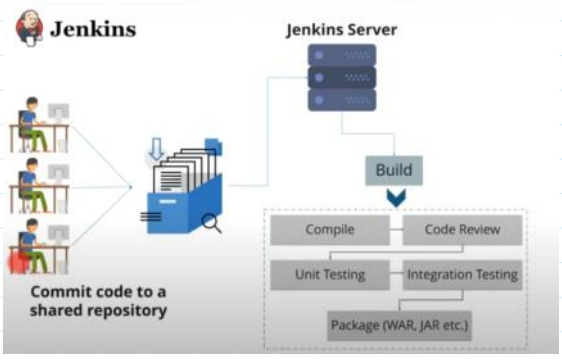


Git Commands

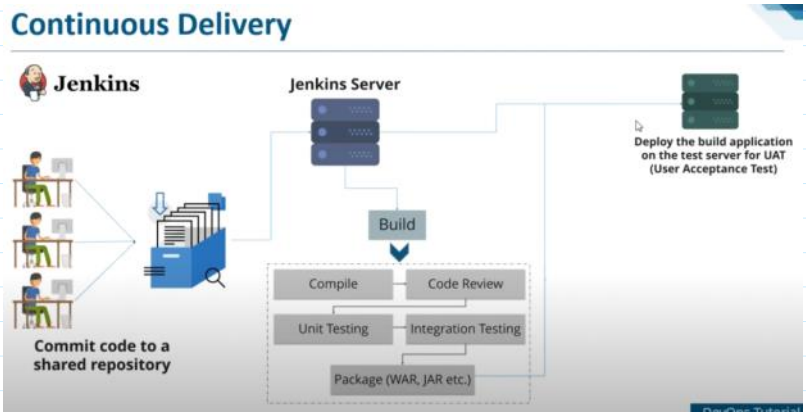
- Git init
- Git add <filename>
- Git commit -m "<message>"
- Git status
- Git push origin master
- Git pull origin master

See git-basic-commands html file

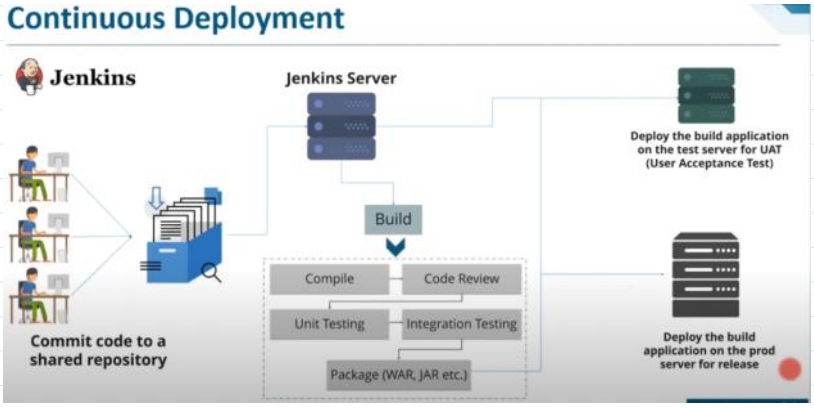
Continuous Integration



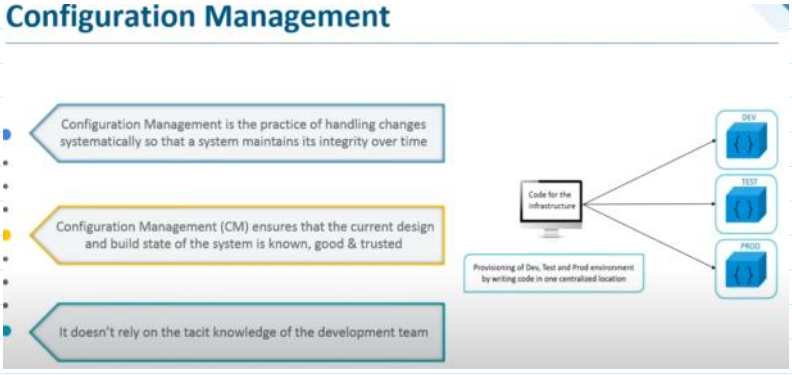
- The moment there are any changes made to the repository, Jenkins will pull the code and prepare the build which includes compilation, code review, testing, packaging.
- Jenkins has various tools in order to perfom all these actions, and has over 2500 plugins



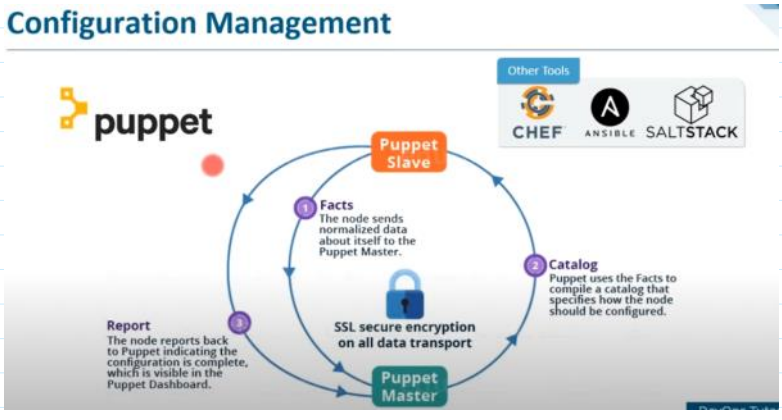
- CD is taking CI to the next step
- Once build is successful, and Jenkins will deploy it for user acceptance test
- **Advantage:** if there are any build failure, then we do not need to go to the entire cycle to determine which commit had the bug. This allows us to determine errors error
-



- Continuous Deployment: taking the build application that have been tested and deploying that for release in an automated fashion
- This is not a good practice because there must be certain checks, or perhaps must be marketed first before a release.

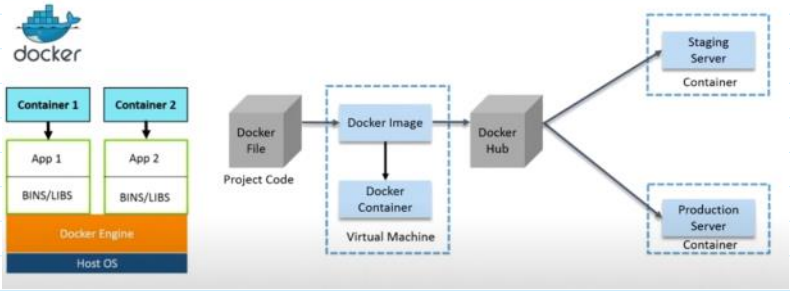


- Scenario: in my workstation I have upgraded a tool and coded using that environment, but once deployed in the test servers, there are some dependency issue because the environment used in the test servers are outdated
- Helps know what components to change once requirements change
- Helps to revert back to the previous version of the new version was not compatible



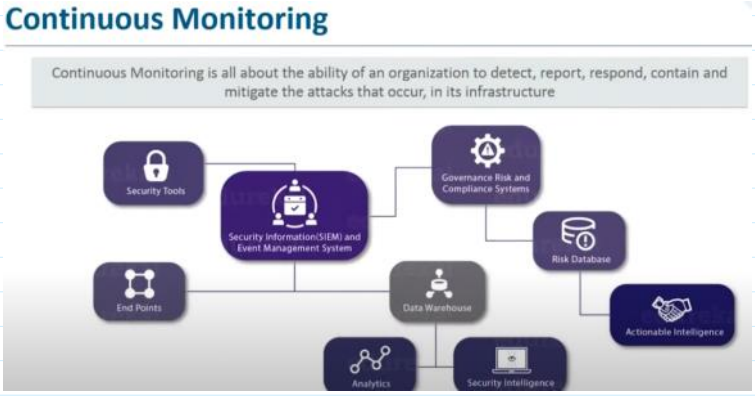
- Used for deploying, managing, configuring servers.
- Puppet can define distinct configuration for each host and continually check if for conformity, alterations of configurations on the host
- Can help with dynamic scaling up and scaling down of machines
- Provides control over all configured machines, so a centralized change get propagated to all automatically
- It follows a slave/master architecture, in which slaves pull from the central sever for any changes in the configuration. "Is there any changes happening in the configuration from the master?" If yes, it will pull that configuration and will deploy in that particular node.
- Ansible saltstack follow a push configuration where the master will push the code during changes. Chef and puppet use a pull configuration from the master.

Containerization



- Containers: lightweight version of VM machines
- All dependancies are present in a 'container'. All I need is this container to run my application
- Docker Image = template for docker container. With the help of the docker image, we can create as much docker container as we want
- The docker image can be uploaded to docker hub. Any member of the team can pull that image to create their own docker container

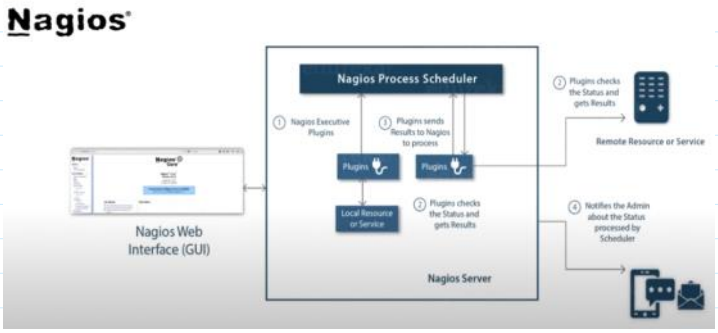
Continuous Monitoring



- Tool resolve any system errors: low memory, unreachable server etc before it has any negative impacts on productivity
- Detects any network or server problems
- Determine root cause of issues
- Maintains availability and security of all the severs
- Monitors and troubleshoot server performance issues
- Allows to plan for infrastructure updates,

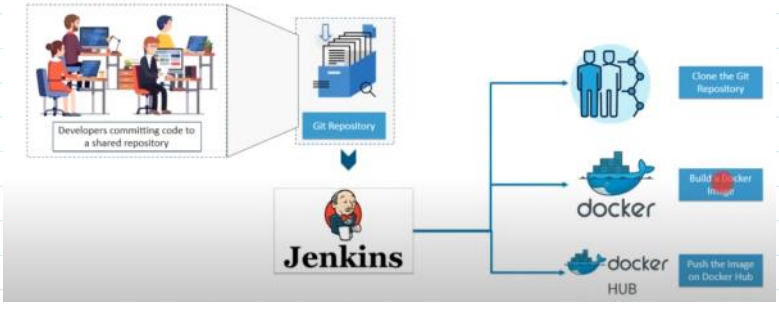
- **Detect, report, respond, contain, mitigate the attacks that occur**

Continuous Monitoring



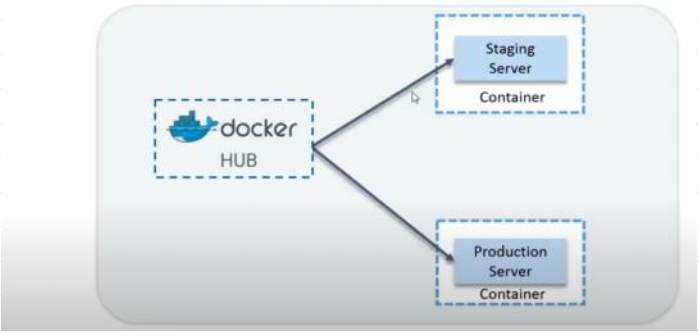
- Runs on a server usually as a deamon or a service
- It periodically runs plugins residing in the same server. The contact hosts or servers on your network or internet locally or remotely
- The plugins check statuses and gets result
- We can see status on the web GUI or SMS

DevOps Use-Case Part - 1



- Devs push commiting code to the repository
- Jenkins will pull the code and will clone the repository
- Jenkins will build a docker image using a docker file
- It will be tested
- Once test is successful, Jenkins will push the docker image on to the docker hub

DevOps Use-Case Part - 2



- Once jenkins has uploaded the docker image in docker hub, any team member can pull this image and create as many containers as they want and perfrom their tests and changes