DevOps Notes

Date: 31/01/2019

Why is DevOps in needed?

- Before DevOps the development and operation team worked in complete isolation.
- Testing and Deployment were is isolated activities done after design-built hence they consumed more time than actual build cycles.
- Without using DevOps, team members are spending a large amount of their time in testing, deploying, and designing instead of building the project.
- Manual code deployment leads to human errors in production.
- Coding and operation teams have their separate timelines and are in synch causing further delays.

There is a demand to increase the rate of software delivery by business stakeholders.

How is DevOps different from traditional IT.

- Let's compare traditional software waterfall model with DevOps to understand the changes DevOps bring.

We assume the application is scheduled to go live in 2 weeks and coding is 80% done.
We assume the application is a fresh launch and the process of buying servers to ship the code has just begun.

Old Process	DevOps
After placing an order for new servers, the Development team works on testing, the Operation team works on extensive paperworks as required in enterprises to deploy the infrastructure.	After placing an order for new servers Development and Operations team work together on the paperwork to set-up the new servers. This results in better visibility of infrastructure requirement.
Projection about failover, redundancy, data center locations, and storage requirements are skewed as no inputs are available from developers who have deep knowledge of the application.	Projection about failover, redundancy, disaster recovery, data center locations, and storage requirements are pretty accurate due to the inputs from the developers.
Operation team has no clue on the progress of the development team. Operation team develop a monitoring plan as per their understanding.	In DevOps the operation team completely aware of the progress the developers are making. Operations team interact with developers and jointly develop a monitoring plan that caters to the IT and business needs. They also use advance Application Performance Monitoring (APM) Tools.
Before go-live the load testing crashes the application. The release is delayed.	Before go-live the load testing makes the application a bit slow. The development team quickly fixes the bottlenecks. The application is released on time.

Why is DevOps Used?

- DevOps allows agile Development Teams to implement continuous Integration and continuous Delivery. This helps them to launch products faster into the market.
- Other important reasons are:

- **Predictability:** DevOps offers significantly lower failure rate of new release.
- **Reproducibility:** version everything so that earlier version can be restored any time.
- **Maintainability:** Effortless process of recovery in the event of a new release crashing or disabling the current system.
- **Time to Market:** DevOps reduces the time to market up to 50% through streamlined software delivery. This is particularly the case for digital and mobile application.
- **Grater Quality:** DevOps helps the team to provide improved quality of application development as it incorporates infrastructure issues.
- **Reduced Risk:** DevOps incorporates security aspects in the software delivery lifecycle. It helps in reduction of defects across the lifecycle.
- **Resiliency:** The operational state of the software system is more stable, secure and changes are auditable.
- **Cost Efficiency:** DevOps offers cost efficiency in the software development process which is always an aspiration of IT companies management.
- **Breaks large code base into small pieces:** DevOps is based on the agile programming method. therefore , it allows breaking large code bases into smaller and manageable chunks.

When to adopt DevOps?

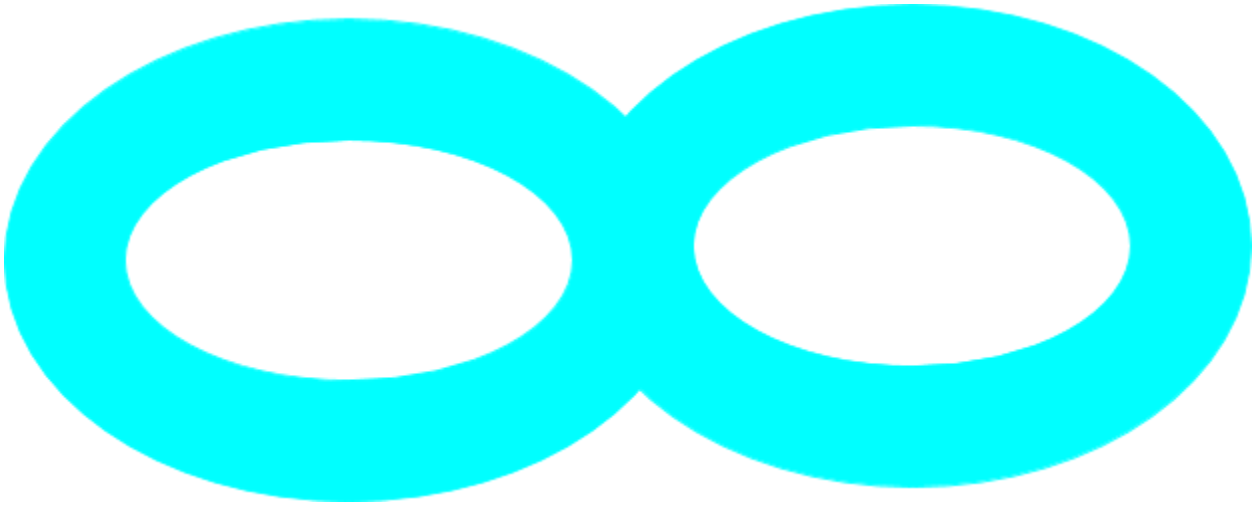
- DevOps should be used for large distributed applications such as E-commerce sites or applications hosted on a cloud platform.

When not to adopt DevOps?

- It should not be used in a mission-critical application like bank, power and other sensitive data sites, such applications need strict access controls on the production environment, a detailed change management policy, access control policy to the data centers.

DevOps Lifecycle

- DevOps is deep integration between Development and Operations



- Understanding DevOps is not possible without knowing DevOps LifeCycle.

Here is a brief information about the continuous (DevOps LifeCycle)

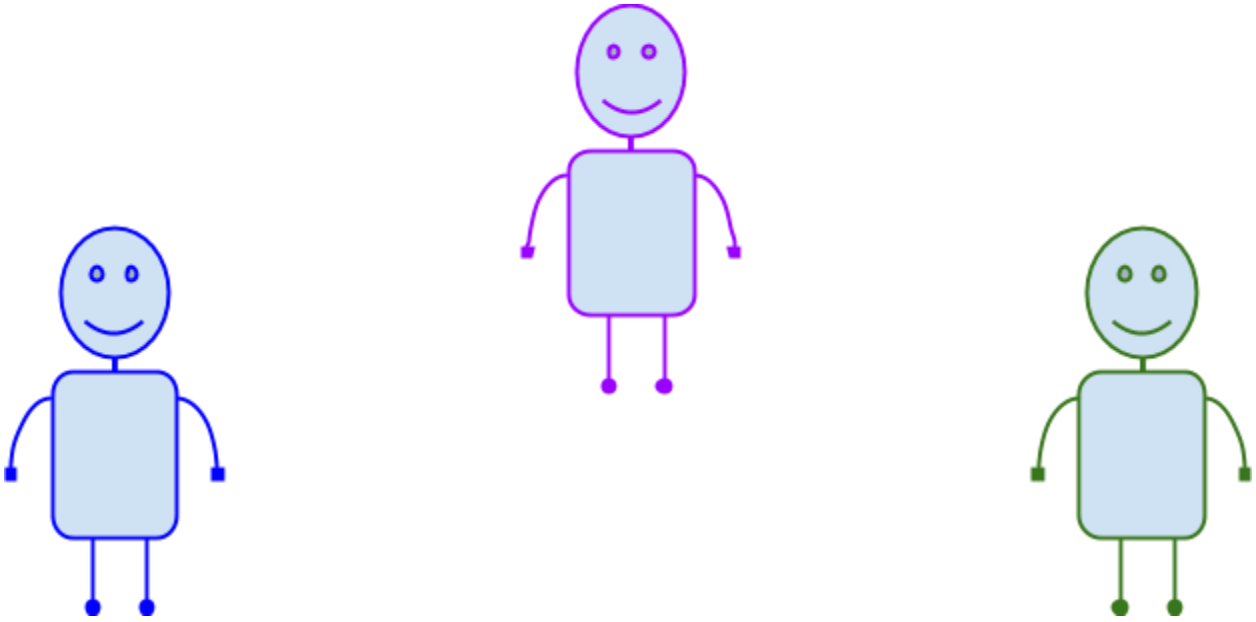
- **Development**
  - In this DevOps stage the development of software takes place constantly, In this phase, the entire development process is separated into small development cycles. This benefits DevOps team to speed up software development and delivery process.
- **Testing**
  - QA team use tools like selenium to identify and fix bugs in the new piece of code.
- **Integration**
  - In this stage new functionality is integrated with the prevailing code, and testing takes place. Continuous development is only possible due to continuous integration and testing.
- **Deployment:**

- In this phase, the deployment process takes place continuously,. It is performed in such a manner that only changes made any time in the code should not affect the functioning of high traffic website.

- **Monitoring:**
  - In this phase, Operation team will take care of the inappropriate system behavior or bugs which are found in production.
- **DevOps workflow:**
  - Workflow provide a visual overview of the sequence in which input is provided. It also tells about actions are performed and output is generated for an operations process

How is DevOps different from Agile?

- Stakeholders and communication chain a typical IT process



- Agile addresses gaps in Customer and Developer communication.
- DevOps addresses gaps in Developers and IT Operations communication.

Agile	DevOps
Emphasize breaking down barriers between developers and management.	DevOps is about software development and Operations teams.

Addresses gap between customers requirements and development teams.	Addresses the gap between development and Operation team.
Focuses more on functional and non-functional readiness.	It focuses operational and business readiness
Agile development pertains mainly to the way development is thought out by the company.	DevOps emphases on deploying software in the most reliable and safest ways which aren't necessarily always the fastest.
Agile developments puts a huge emphasis on training all team members to have varieties of similar and equal skills. So that when something goes wrong any team member can get assistance from any member in the absence of the team leader.	DevOps likes to divide and conquer spreading the skill set between the development and operation teams. It also maintains consistent communication.
Agile development manages on sprints. It means that the time table is much shorter (less than a month) and several features are to be produced and released in the period.	DevOps strives for consolidated deadlines and benchmarks with major releases, rather than smaller and more frequent ones.

DevOps Principles:

- Here, are Six Principles which are essential when adopting DevOps.
- **Customer Centric Action:**
    - DevOps team must take customer-centric action for that they should constantly invest in products and services.
- **End-to-End Responsibility:**
    - The DevOps team need to provide performance support until they become end-of life. This enhances the level of responsibility and the quality of the products engineered.
- **Continuous Improvement:**
    - DevOps culture focuses on continuous improvement to minimize waste. It continuously speeds up the improvement of product or services offered.
- **Automate Everything:**

    - Automation is vital principle of DevOps process. This is not only for the software development but also for the entire infrastructure landscape.
- **Work as one Team:**
    - In the DevOps culture role of the designer, developer and tester are already defined. All they needed to do is work as one team with complete collaboration.
- **Monitor and Test everything:**
    - It is very important for DevOps team to have a robust monitoring and testing procedures.

Who is DevOps Engineer?

- A DevOps Engineer is an IT Professional who works with software developers system operators and other production IT staff to administer code releases. DevOps should have hard as well as soft skills to communicate and collaborate with development, testing and operations teams.
- DevOps approach need frequent, incremental changes to code versions, which means frequent deployment and testing regimens. Although DevOps engineers need to code occasionally from scratch. It is important that they should have the basics of software development languages.
- A DevOps engineer will work with development team staff to tackle the coding and scripting needed to connect elements of code, like libraries or software development kits.

Roles, Responsibilities and skills of a DevOps Engineer?

- DevOps engineers work full-time. They are responsible for the production and ongoing maintenance of a software applications platform.
- Following are some expected roles, responsibilities and skills that is expected from DevOps Engineer
    - Able to perform system troubleshooting and problem-solving across platform and application domains.
    - Manage project effectively through open, standards-based platforms.
    - Increase project visibility thought traceability.

- Improve quality and reduce development cost with collaboration.
- Analyse, design and evaluate automation scripts and systems.
- Ensuring critical resolution of system issues by using the best cloud security solutions services.
- DevOps Engineer should have the soft skill of problem-solver and quick learner.

DevOps Automation Tools:

- It is vital to automate all the testing processes and configure them to achieve speed and agility. This process is known as DevOps automation.
- This difficulty faced in large DevOps team that maintain large huge IT infrastructure can be classified briefly into six different categories.
- **Infrastructure Automation:**
  - **AWS:** being cloud service you do not need to be physically present in the data center. Also they are easy to scale on-demand there are no up front hardware costs. It can be configured to provision more servers based on traffic automatically.
- **Configuration Management:**
  - **CHEF:** It is useful DevOps tool for achieving speed, scale and consistency. It can be used to easy out complex tasks and perform configuration management with this tool, DevOps team can avoid making changes across ten thousand servers. Instead they need to make changes in one place which is automatically reflected in other servers.
- **Deployment Automation:**
  - **Jenkins**: This tool facilitates continuous integration and testing. It helps to integrate project changes more easily by quickly finding issues as soon as a built is deployed.
- **Log Management:**
  - **Splunk:** This is a tool solves the issues like aggregating, storing, and analyzing all logs in one place.
- **Performance Management:**
  - **App Dynamic**: It is DevOps tool which offers real-time performance monitoring. The data collected by this tool helps developers to debug when issues occur.

- **Monitoring:**
    - **Nagios:** It is also important to make people are notified when infrastructure and related services go down. Nagios is one such tool for this purpose which helps DevOps teams to find and correct problems.

Introduction of git

Before understanding git understand Version Control System.

- **Version Control System:**
    - Version Control System (VCS) is a system that help software developers to work together and maintain a complete history of their work.
- **Listed below are the functions of a VCS**
    - Allows developers to work simultaneously.
    - Does not allow overwriting each others changes.
    - Maintains a history every version.
- **Following are the types of VCS**
    - Local Version Control Systems
    - Centralized Version Control System (CVCS)
    - Distributed / Decentralized Version Control System (DCCS)
- **Version Control System Tools:**
    - Git
    - Concurrent Version System
    - Subversion
    - Mercurial

What is Git?

- Git is a distributed version controls tool that supports distributed non-linear workflows by providing data assurance for developing quality software.

Features of Git

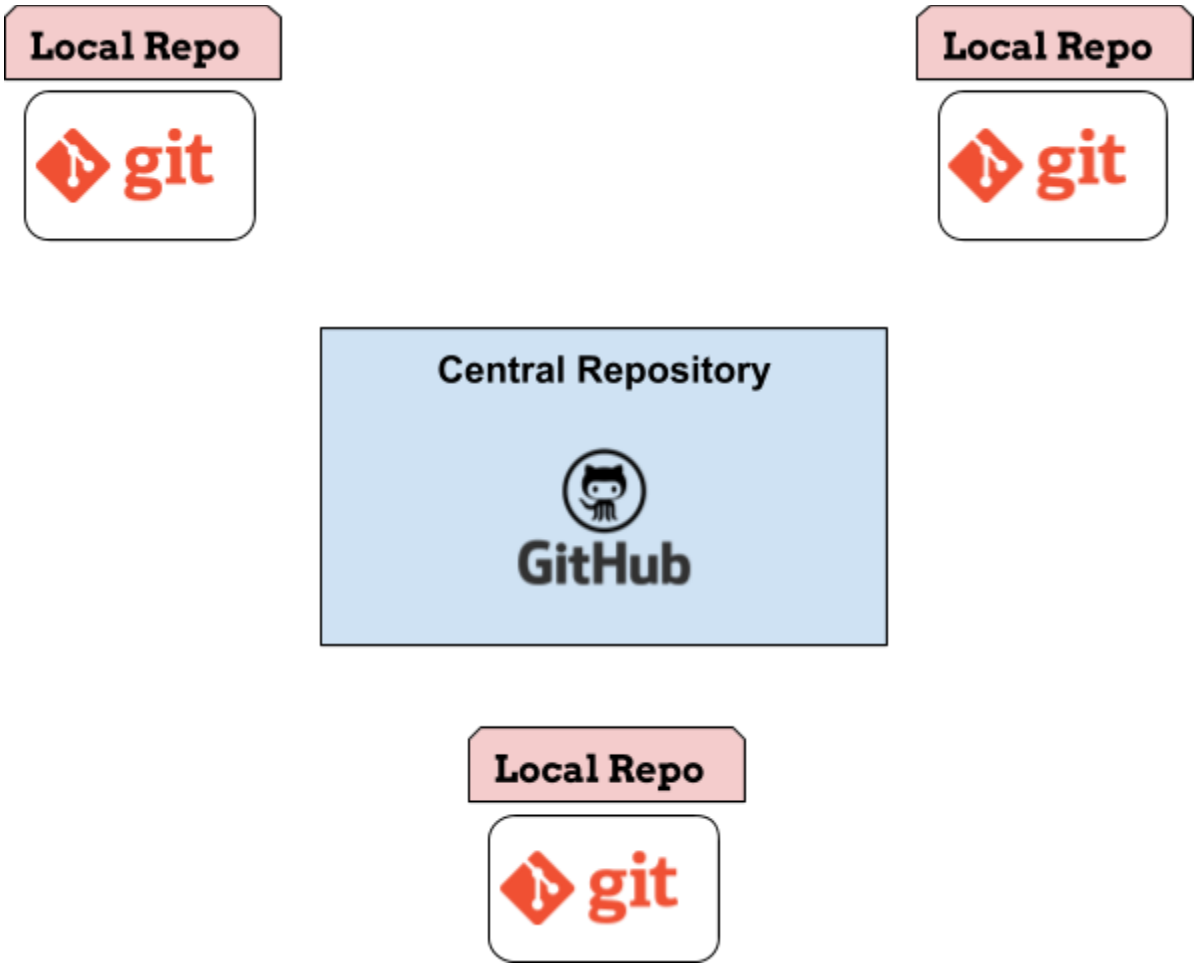
- **Distributed:**
    - Allows distributed development of code.

    - Every developer has a local copy of the entire development history and changes are copied from one repository to another.
- **Compatible:**
    - Compatible with existing systems and protocols.
    - SVN repositories can be directly accessed using Git-SVN
- **Non-Linear**
    - Supports non-linear development of software.
    - Includes various techniques to navigate and visualize non-linear development history.
- **Branching**
    - It takes only a few seconds to create and merge branches.
    - Master branch always contains production quality code.
- **Lightweight:**
    - Uses lossless compression technique to compress data on the client's side.
- **Speed:**
    - Fetching data from local repository is 100 times faster than remote repository.
    - Git is one order of magnitude faster than other VCS tools.
- **Open Source:**
    - You can modify its source code according to your needs.
- **Reliable:**
    - On events of system crash, the lost data can be easily recovered from any of the local repositories of the collaborators.
- **Secure**
    - Uses SHA1 to name and identify objects.
    - Every file and commit is checksummed and is retrieved by its checksum at time of check out.
- **Economical**
    - Released under GPL License. It is for free.
    - All heavy lifting is done on client-side, hence a lot of money can be saved on costly servers.

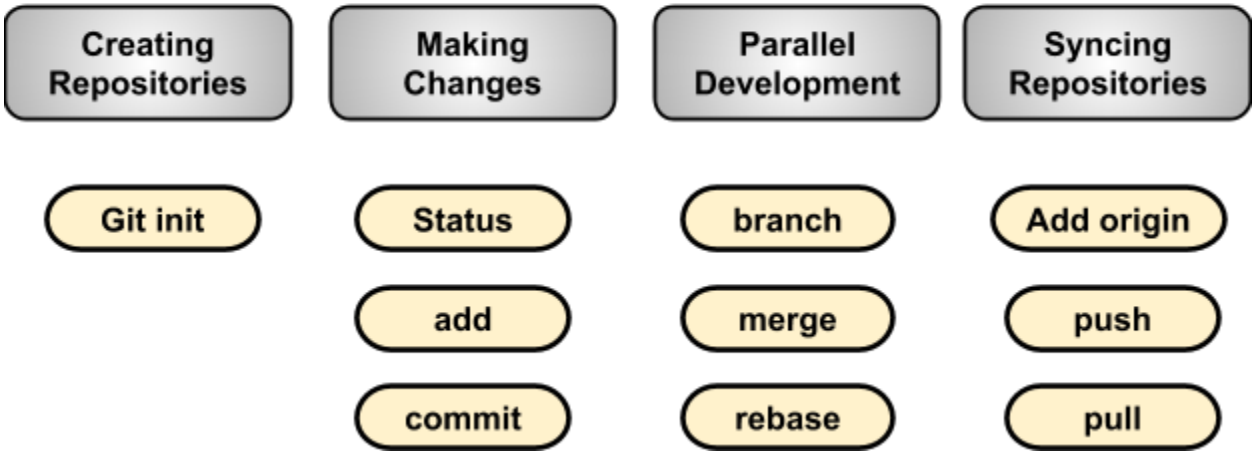
What is a Repository?

- A directory or storage space where your project can live. It can be local to a folder on your computer, or it can be a storage space on github or another online host. You can keep code files, text files, image files, your name as a repository.

- There are two types of repositories:
- **Central Repository**
    - Typically Located on Remote Server
    - Exclusively consists of .git repository folder
    - Meant for team to share and exchange data.
- **Local Repository**
    - Typically located on local machine
    - Reside as a .git folder inside your projects root.
    - Only admin of the machine can work with this repository.


Local Repo
git
Local Repo
git
Central Repository
GitHub
Local Repo
git

Git Operations and Commands

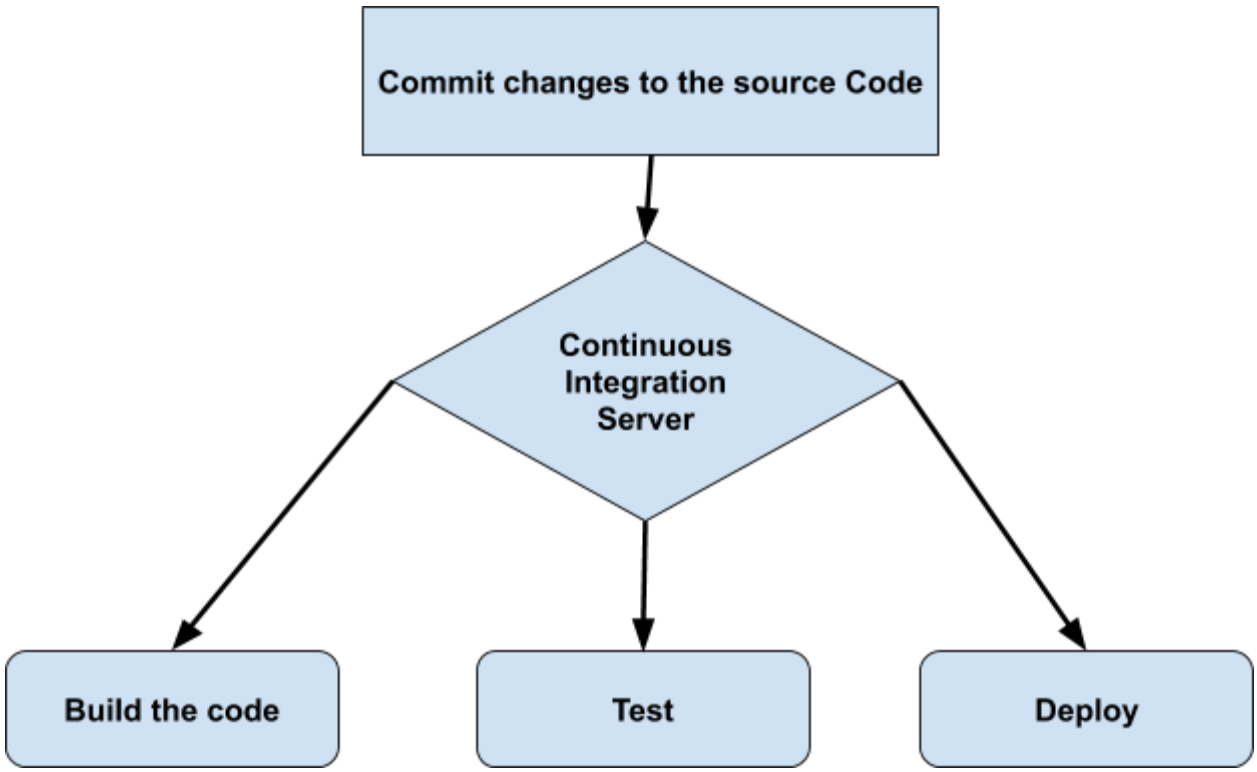


- # Continuous Integration
- **Problems before Continuous Integration**
    - Developers have to wait till the complete software is developed for the test results.
    - If the test fails then locating and fixing bugs is very difficult. Developers have to check the entire source code of the software.
    - Software delivery process was slow.
    - Continuous feedback pertaining to things like coding or architectural issues, build failures, test status etc.
- **Continuous Integration**
    - Since after every commit to the source code an auto build is triggered and then it is automatically deployed on the test server.
    - If the test results shows that there is a bug in the code then the develops only have to check the last commit made to the source code.
    - This also increases the frequency of new software releases.
    - The concerned teams are always provided with the relevant feedback.

Before Continuous Integration	After Continuous Integration
The entire source code was built and then tested.	Every commit made in the source code is built and tested.
Developer have to wait for test results.	Developers know the test result of every commit made in the source code on the run.
No feedback	Feedback is present

What is Continuous Integration

- Continuous Integration is a development practice in which the developers are required to commit changes so the source code in a shared repository several times a day or more frequently.
- Every commit made it the repository is then built. This allow the teams to detect the problems early.

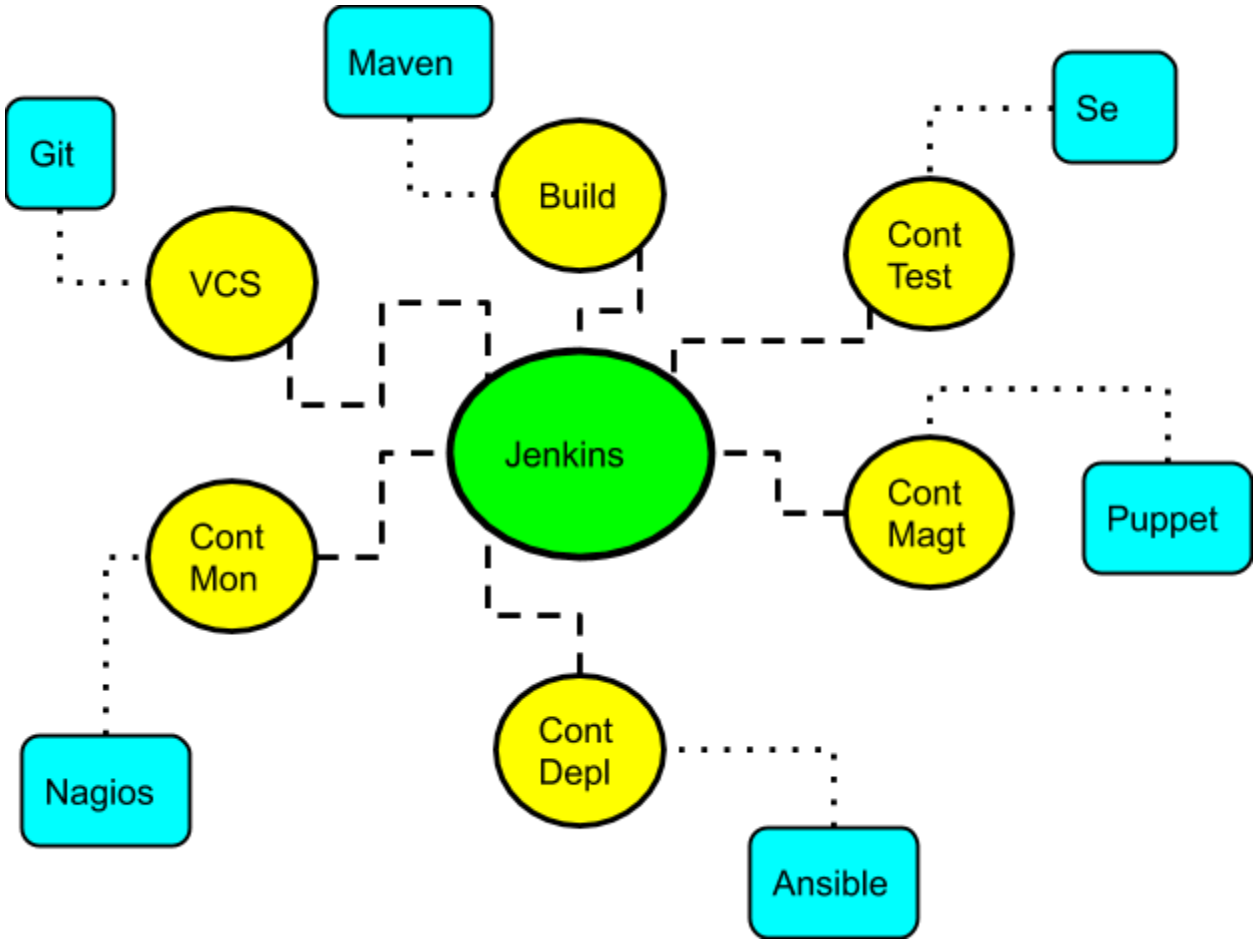


Continuous Integration Tools

- Jenkins
- Buildbot
- Travis CI
- Bamboo

What is Jenkins

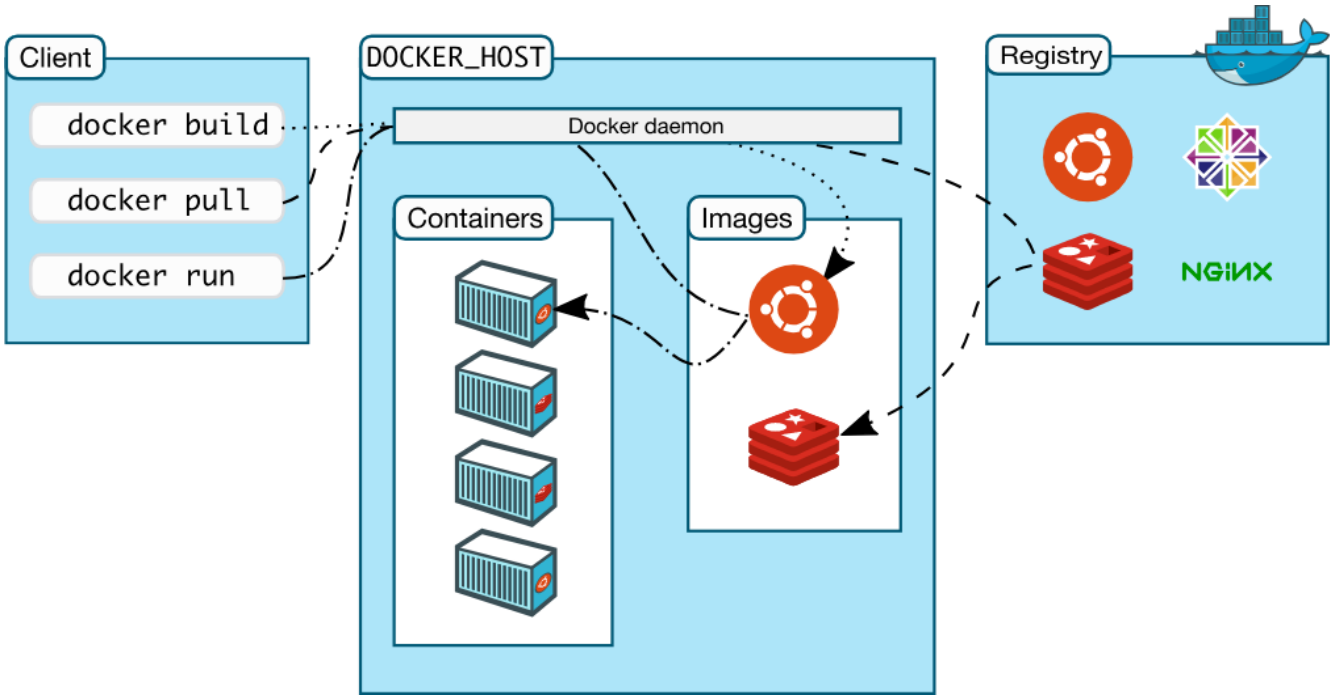
- Jenkins is an open source automation tool written in java with plugins built for continuous integration purpose. Plugins allows integration of various DevOps stages.



Docker

- Docker is the world's leading software container platform.
- We can use docker in
    - Design
    - Development
    - Deployment
    - Testing / Release
- Docker is present in the entire workflow, but its main use is in deployment.

- It works in Deployment
- Docker makes the process of application deployment very easy and efficient and resolves a lot of issues related to deploying applications



Understand Dockers in Easy Way

-