**DevOps**:

**Topic**:

1. Overview | Introductions
2. Git and gihub(VCS)
3. Maven
4. Docker
5. Kubernets
6. Prometheus/Grafana
7. ELK
8. Ansible
9. Jenkins
10. Terraform

**Pre Requestions**

**Scripting Language:** Python, PowerShall Script(linux)

**LBA**: Linux Based

Linux Application: Ubuntu, centOS, Red Hat, Windows, MackOS

Web Server : Apache , nginx

**Important Link**:

DevOps by Simple-Learn: <https://www.simplilearn.com/tutorials/devops-tutorial/devops-tools>

DevOps By AWS: <https://aws.amazon.com/devops/what-is-devops/>

Git Installation: <https://git-scm.com/download/linux>

Git Materials Repo(Trander Repository): [GitHub - onlineTrainingguy/DevOpsNotes: It contains Devops related Notes](https://github.com/onlineTrainingguy/DevOpsNotes)

Git Materials by: <http://www.devopsworld.co.in/p/git-commands.html>

Session: 1

**What is SDLC:**

It is the process of to design, develop and deliver the software. The Software Development Life Cycle (SDLC) is **a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time**. The goal of the SDLC is to produce superior software that meets and exceeds all customer expectations and demands

**SDLC Life Cycle:** Requirement => Planning & Analysis => Design => Development (Implementation) => Testing (Verification) => Deployed/Operations => Maintenance

**Why SDL Important:**

1. It provides a standardized framework that defines activities and deliverables
2. It aids in project planning, estimating, and scheduling
3. It makes project tracking and control easier
4. It increases visibility on all aspects of the life cycle to all stakeholders involved in the development process
5. It increases the speed of development
6. it improves client relations/trust and decrease the project risk and gap

**Popular SDLC Modal**

1. **Waterfall**:
   1. Waterfall represents the oldest, simplest, and most structured methodology. Each phase depends on the outcome of the previous phase, and all phases run sequentially
   2. Waterfall model is oldest/first SDLC model and it’s followed the Liner sequential model
   3. Until the first stage is not finished, the next stage will not start.
   4. Best for small project where requirement & vision are clear.
   5. LC: Requirement = > Analysis => Design => Coding & Unit/Dev Testing(Implementation) => Testing ( Integration & System Testing) => Deployed/Operation => Maintenance of Software
   6. Objective & Advantage :
      1. Requirements are very well documented, clear and fixed
      2. Product definition is stable
      3. Simple and easy to understand and use
      4. Easy to arrange tasks
   7. Disadvantage
      1. Working Product is not available until the later stage in lifecycle
      2. Poor model for large & complex project
      3. Cannot accommodate changing requirement
      4. High amounts of risk and uncertainty.
2. **Agile(**Scrum, XP & Canban**)**: The Agile methodology is **a way to manage a project by breaking it up into several phases**. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating.

The Agile methodology is a way to manage a project by breaking it up into several phases. The agile methodology produces ongoing release cycles, each featuring small, incremental changes from the previous release. At each iteration, the product is tested.

It work best when

* 1. You can't estimate the time you'll need and don't know the full scope of requirements
  2. You can afford to iterate and don't need to deliver fully functional software at once
  3. Want divide Project into multiple module/sprint
  4. Want to deliver incremental version of software for every sprint or max in 15 days.
  5. Clients don't have a fixed budget/schedule

There is different implementation mythologies

* + 1. Scrum
    2. Canban
    3. XP

Advantage of Ajile Methodology

1. You can deploy software quicker, so your customer can get value sooner rather than later
2. You can detect and fix issues and defects faster
3. You can get immediate feedback (which also improves team morale)
4. Developers can improve their skills based on QA feedback
5. Welcome with new changes/feedback
6. Best for large/mid label project

DisAdvantage of Ajile Methodology

1. Not suitable for handling complex dependency in project
2. Knowledge transfer to colleagues cab be difficult since there is littlie documentation
3. Success of the project depends heavily on customer interaction
4. It's more difficult to measure progress than it is in Waterfall because progress happens across several cycles
5. Agile demands more time and energy
6. No suitable for small /less buggiest
7. **Lean** **Methodologies** :

In simple Term: **improve efficiency by eliminating waste**.

Lean methodology is **a way of optimizing the people, resources, effort, and energy of your organization toward creating value for the customer**. It is based on two guiding tenets, continuous improvement and respect for people.

* 1. It is extension of Agile methodologies
  2. Lean Development is a philosophy of increasing quality of software deliver by making use of agile methods

Advantage:

* + - 1. **Cost Decrease**: Eliminate the wastage by analysis the uses of resources
      2. Increase the Quality: By Improving the quality from previous & compotator

**What is JIRA(EPIC, Story, Task):**

**What is Unit Testing(JNIT)**: It’s done by Developer who has knowledge of internal codebase

**What is Integration Testing**(Selenium, Kukumber): It’s done by tester who does not has knowledge of codebase

**DevOps Methodologies =>**

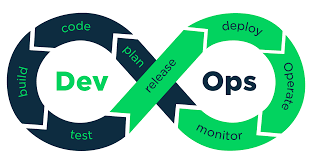
**Dev(Developer) + Ops(Operation)):**

**DevOps** is a set of practices/process that combines software development and IT operations. It aims to shorten the systems development life cycle (SDLC) and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from the Agile way of working.

**DevOps** is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity. It’s enables organizations to deliver better serve their customers and compete more effectively in the market.

**DevOps cultural philosophies**: With DevOps, the two teams work together to optimize both the productivity of developers and the reliability of operations. They strive to communicate frequently, increase efficiencies, and improve the quality of services they provide to customers.

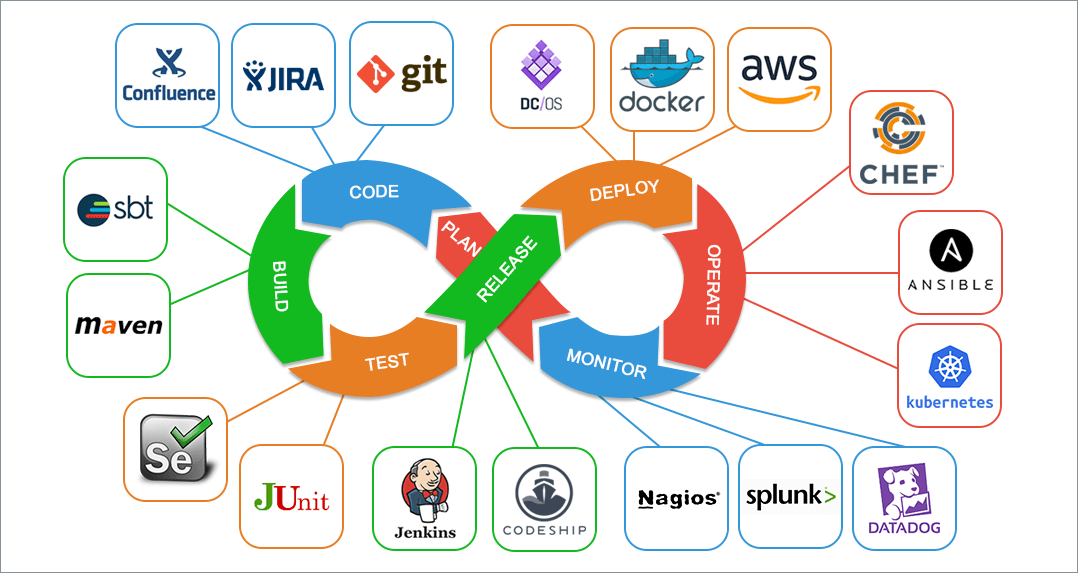
**DevOps Practices**: CI, CD, Micro-services, Infra-structure as code, Monitoring & logging



**Benefit of Using DevOps:**

1. **Speed**: Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results.
2. **Rapid Delivery** : Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers’ needs and build competitive advantage.
3. **Scale**: Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk
4. **Reliability**: Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users.
5. **Improve Collaborations** : Developers and operations teams collaborate closely, share many responsibilities, and combine their workflow

Tools Using in DevOps



**Planing**: Jira

**Code**: git, gitLab,

**Build**: Maven, Gradle, ANT, SBT

**Test**: Selenium, Junit, AZure

**Release**: Jenkin, Bambu , Codeship, OpenStack,

**Deploy**: Docker, AWS,

**Oprate**: Kubernet, Checf, Ansile, ,

**Monitor**: Nagios, Splunk, DataDog

**Continuous integration(CI):** Continuous Integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates

**Varsion Control System => VCS**

**(Git, GitLab, GitHub, SVN, bitbucket)**

**What is Centralize & Distributed Version control system(C-VCS & D-VCS):**

**C-VCS(SVN):**

1. you need to take an update from the master repository and then you get the local copy of the code in your system. So once you get the latest version of the code, you start making your own changes in the code and after that, you simply need to commit those changes straight forward into the master repository. Committing a change simply means merging your own code into the master repository or making a new version of the source code
2. So the basic workflow involves in the centralized source control is getting the latest version of the code from a central repository that will contain other people’s code as well, making your own changes in the code, and then committing or merging those changes into the central repository.
3. So everything is centralized in this model.

**D-VCS(Git & Mercurial):**

1. In distributed version control most of the mechanism or model applies the same as centralized. The only major difference you will find here is, instead of one single repository which is the server, here every single developer or client has their own server and they will have a copy of the entire history or version of the code and all of its branches in their local server or machine.
2. So when you start working on a project, you clone the code from the master repository in your own hard drive, then you get the code from your own repository to make changes and after doing changes, you commit your changes to your local repository and at this point, your local repository will have ‘change sets‘ but it is still disconnected with the master repository (master repository will have different ‘sets of changes‘ from each and every individual developer’s repository), so to communicate with it, you issue a request to the master repository and push your local repository code to the master repository. Getting the new change from a repository is called “pulling” and merging your local repository’s ‘set of changes’ is called “pushing“.

| **Sr. No.** | **Key** | **Centralized Version Control** | **Distributed Version Control** |
| --- | --- | --- | --- |
| 1 | Working | In CVS, a client need to get local copy of source from server, do the changes and commit those changes to centeral source on server. | In DVS, each client can have a local branch as well and have a complete history on it. Client need to push the changes to branch which will then be pushed to server repository. |
| 2 | Learning Curve | CVS systems are easy to learn and set up. | DVS systems are difficult for beginners. Multiple commands needs to be remembered. |
| 3 | Branches | Working on branches in difficult in CVS. Developer often faces merge conflicts. | Working on branches in easier in DVS. Developer faces lesser conflicts. |
| 4 | Offline Access | CVS system do not provide offline access. | DVD systems are workable offline as a client copies the entire repository on their local machine. |
| 5 | Speed | CVS is slower as every command need to communicate with server. | DVS is faster as mostly user deals with local copy without hitting server everytime. |
| 6 | Backup | If CVS Server is down, developers cannot work. | If DVS server is down, developer can work using their local copies. |

**GIT:**

1. Git is a distributed version control system for tracking changes in source code during software development. It is designed for coordinating work among programmers/teams, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.  Git is a popular version control system. It was created **by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.**
2. It is used for:
   1. Tracking code changes
   2. Tracking who made changes
   3. Coding collaboration teams

**GITHUB**:

1. GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. **Created by microsoft in 2018**
2. GitHub is a collaboration platform that helps review and manage codes remotely
3. Uses:
   1. Create and use a repository
   2. Start and manage a new branch
   3. Make changes to a file and push them to GitHub as commits
   4. Open and merge a pull request

**GITLAB**:

1. GitLab is **an open source code repository and collaborative software development platform for large DevOps and DevSecOps projects.**
2. The open source software project was created by Ukrainian developer Dmitriy in 2014

**Diffrence b/w GIT & GitHub**

| S.No. | Git | GitHub |
| --- | --- | --- |
| 1. | Git is a software. | GitHub is a service. |
| 2. | Git is a command-line tool | GitHub is a graphical user interface |
| 3. | Git is installed locally on the system | GitHub is hosted on the web |
| 4. | Git is maintained by linux. | GitHub is maintained by Microsoft. |
| 5. | Git is focused on version control and code sharing. | GitHub is focused on centralized source code hosting. |
| 6. | Git is a version control system to manage source code history. | GitHub is a hosting service for Git repositories. |
| 7. | Git was first released in 2005. | GitHub was launched in 2008. |
| 8. | Git has no user management feature. | GitHub has a built-in user management feature. |
| 9. | Git is open-source licensed. | GitHub includes a free-tier and pay-for-use tier. |
| 10. | Git has minimal external tool configuration. | GitHub has an active marketplace for tool integration. |
| 11. | Git provides a Desktop interface named Git Gui. | GitHub provides a Desktop interface named GitHub Desktop. |
| 12. | Git competes with CVS, Azure DevOps Server, Subversion, Mercurial, etc. | GitHub competes with GitLab, Git Bucket, AWS Code Commit, etc. |

SVN:

Murcury

Bitbucket

What is git & github and SVN:

What is Build System:

What is Docker:

What is Selenium & Kubernet:

What s jemkins:

**AWS: How to create EC2 Instance**

1. Login to your Account
2. Select EC2
3. Enter/Type the name of your Instance (e.i: MyTestInstance)
4. Select the Application & OS Image (e.i: OS: Ubuntu, Windows, Red Hat)
5. Select the Instance Type (i.e. : t1.micro, t2.micro, C5.large)
6. Key Pair(Login):
   1. If available than select that
   2. if not create new key pair
      1. Enter the name of key pair
      2. Select the key pair type (e.i. : RSA or ED25519)
      3. Select the Private key file format (e.i : .pem(For use with OpenSSH) or .ppk(for use with PuTTY))
7. Network Setting
   1. Create or select the Firewall(Security group)
8. Configure Storage
   1. Add New Volume
   2. File System
9. Advance Details
   1. Request for spot instance
   2. Domain join directory
   3. IMA Instance Profile setup

How Login & get Access AWS instance from local system

* + - 1. Download & Install PuTTY or mobaxterm
      2. Get PPK or Pemp file from AWS Console
      3. Get IP & Users Details from AWS console

