Introduction to Devops

**Agenda**

What is software development

Types of model

Waterfall model

Agile model

Why devops

What is devops

Devops lifecycle

Devops tools

What is software development

Software development is the process of transforming customer requirements into a complete software product

Types of model

Waterfall Model

Types of SDLC (software development life cycle )model

1.Waterfall model

2.Agile model

Waterfall model

Requirement

gathering requirement

Design

create high level design

Implentation

Deploying the software

Testing

conducting various testing activities

Deployment

Installing and delivering software

maintaince Providing on going support bug fixex and update

Agile Model

Deployment

Build

Coe review

Testing

Deployment

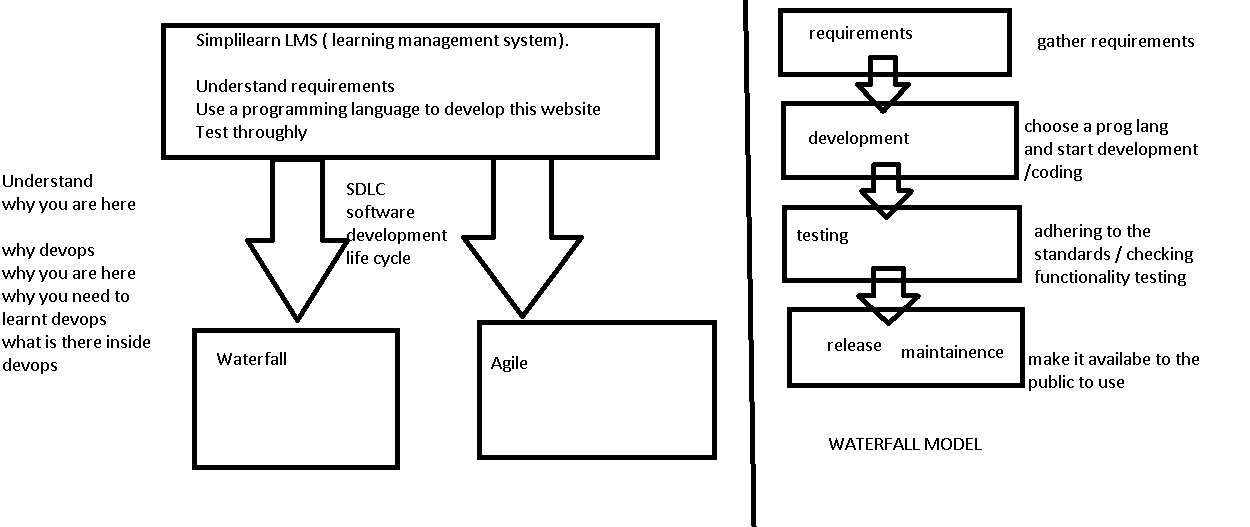
Monitoring

Delivery

Waterfall model

1. Waterfall model was among the irst development models which followed SDLC

2. The waterfall model follows a linear sequential model of development i.e until the first stage is not finished, the next stage will not start

* ****

**Advantages**

Clear Objective

Specific deadline

Well understood milestone

**Disadvantages**

Working product is not available until the last stage in lifecycle.

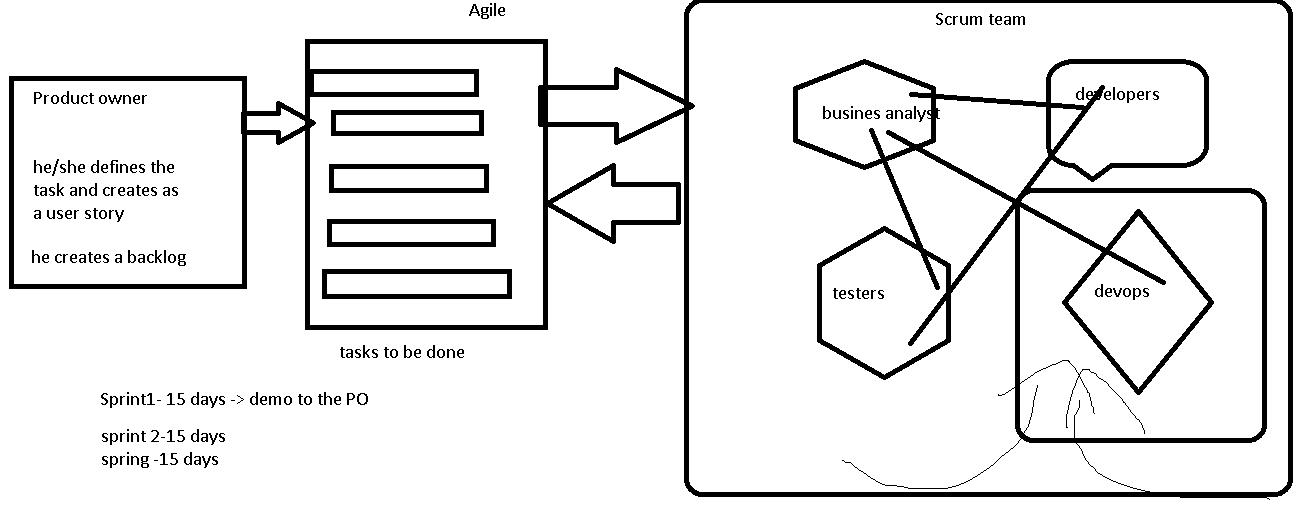
Poor model for large and complex project

High risk and uncertainty

**Agile Model**

**1 To overcome the challenges faced in the waterfall model we came up with the agile methodology**

**2 Agile methodology believes in creating shorter development lifecycle**

* ****

**Advantage**

1. **Customer satisfaction is high**
2. **Less planning required**
3. **Functionality can be created and tested quickly.**

**Disadvantage**

**1.knowledge transfer to colleagues can be difficult**

**2. success of the project depends heavily on customer interaction**

**Why Devops**

**Software company**

**Developer operation**

**Developer scenario**

Developer job is to develop applications and pass his code to the operations team the developer used to run the code on his system and then forward it to operations team, but the code runs fine on the developers system and hence he says “It is not my fault ”

Operations team :- The operations team job is to test the code and provide feedback to developers in case of bugs . if all goes well the operations team uploads the code to the buid servers. The operations when tried to run the code on their system it did not run. The peration then marked as faulty and used to forward this feeback to the developer.

**Summarize**

**This led to a lot of back and forth between the developer and the operations team hence impacted efficiency. This problem was solving using devops.**

**What is devops**

**Devops is a software development methodology which improves the collaboration between developers and operations team using various automation tools. These automation tools are implemented using various stages which are a part of the devops lifecycle**

**How devops works**

**Continuous development continuous deployment**

**Plan Release**

**Code continuous integration Deploy**

**Build Operate**

**test Monitor**

continuous testing continuous monitoring

continuous development - Continuous development, also known as continuous improvement or continuous delivery, is an approach used in software development and project management that emphasizes the constant and iterative delivery of value to customers or users.

continuous Integration-Continuous Integration (CI) is a software development practice that involves frequently integrating code changes from multiple developers into a shared repository. The primary goal of CI is to catch integration issues and conflicts early in the development process, ensuring that the software remains in a working state.

Continuous Deployment-Continuous Deployment (CD) is an extension of continuous integration and continuous delivery (CI/CD) that focuses on automating the release and deployment of software changes into production environments. It involves automatically deploying every validated code change that passes through the CI process, making it immediately available to end-users.

Continuous testing-Continuous Testing (CT) is a software testing approach that integrates testing activities throughout the entire software development and delivery lifecycle. It aims to provide rapid and continuous feedback on the quality of the software as it evolves, supporting the principles of continuous integration and continuous delivery.

Continuous monitoring -Continuous Monitoring (CM) is an ongoing process of monitoring and analyzing the performance, availability, security, and compliance of systems, applications, and infrastructure in real-time. It involves the collection, analysis, and interpretation of various data points to ensure that systems and services operate optimally and meet predefined standards.

**Tools For Each DevOps Stage**

* Continuous Development – git /Github
* Continuous Build -Maven
* Continuous Integration –Jenkins,Argo CD
* Continuous Testing - Selenium
* Continuous Deployment - Ansible / puppet
* Continuous Code review -Sonarqube
* Continuous Delivery - Docker , Kubernetes , Terraform
* Continuous Monitoring – promtheus and grafana

**GIT**

**Version control With GIT**

**Agenda**

* 1. What is Version Control?
  2. Types of Version Control System
  3. Introduction to GIT
  4. GIT Lifecycle
  5. Common GIT commands
  6. Merging in GIT
  7. Resolving Merge Conflicts
  8. Git Workflow
  9. What is Forking?
  10. Git setup

What is Version Control?

* Version control is a system that records/manages changes to documents,

computer programs etc over time. It helps

us tracking changes when multiple people work on the same project.

Advantages of Version Control:

* Versioning is Automatic
* Team collaboration is simple
* Easy access to previous versions
* Only modified code is stored across different versions, hence saves storage

Types of Version Control System:

1. Central VCS
2. Distributed VCS

Centralized VCS:

* Centralized Version Control system has one single copy of code in the central server.
* If the central server crashes, all data from the project will be lost.
* Developers will have to “commit” their changes in code to this central server.
* “Committing” a change is to recording the change in the central system.

Distributed VCS:

* In this VCS, one does not necessarily rely on a central server to store all the versions of a projects file.
* There is not threat, even if the server crashes.
* Every developer “clones” a copy of the main repository on their local system.
* This also copies, all the past versions of the code on the local system too.
* Therefore, the developer need not be connected to the internet to work on the code.

Introduction to GIT:

Why Git?

* It ensures there are no code conflicts between developers.
* Developers can revert their project to the older versions of the code.

What is Git?

* Git is a version control system that is used to manage & keep track of code.
* Everytime we commit or save, git basically takes a pictures of what files look like at that moment & stores a reference to that snapshot.
* **Create GitHub Account**
* <https://github.com/>
* Please give your email , username and choose a password.
* Verify the email and you should be able to access .

Git Lifecycle:

The lifecycle stages are as follows;



1. Working Directory:

* The place where your project resides in your local disk.
* The project can be tracked by git, using the command “git init” & it automatically creates a hidden .git folder.

1. Staging Area:

* Staged files are the files that are ready to be committed to the repository we are working on.
* To get Git to track them, we need to stage them, or add them to the staging environment. Using command “git add --all”.

1. Commit:

* Once the files are selected & are ready in staging area, they can now be saved in repository.
* Saving file is known as doing a commit.
* The command is “git commit –m ‘message’”.

Common Git Commands:

* **git init:** To create new repository & to initialize a git repository on local system.
* **git clone [url]:** To download the remote repository to our local system.
* **git status:** To check status of the files.
* **git add:** Add changes to the staging area.
* **git commit:** Commit changes to the local repository.(git commit –m “message”)
* **git remote:** Connecting local repository to the remote repository.(git remote add origin “<url>”) .Here remote repository is on github.
* **git push:** To push the changes to your repository(git push origin <branch-name>). If the branch is master, hence use command (git push origin master).
* **git pull:** To pull changes from remote repository(github).
* **git branch:** Branches allow you to work on different parts of projet without impacting the main branch. Also we can work on different projects without interfering with each other.

(git branch <new branch name>)

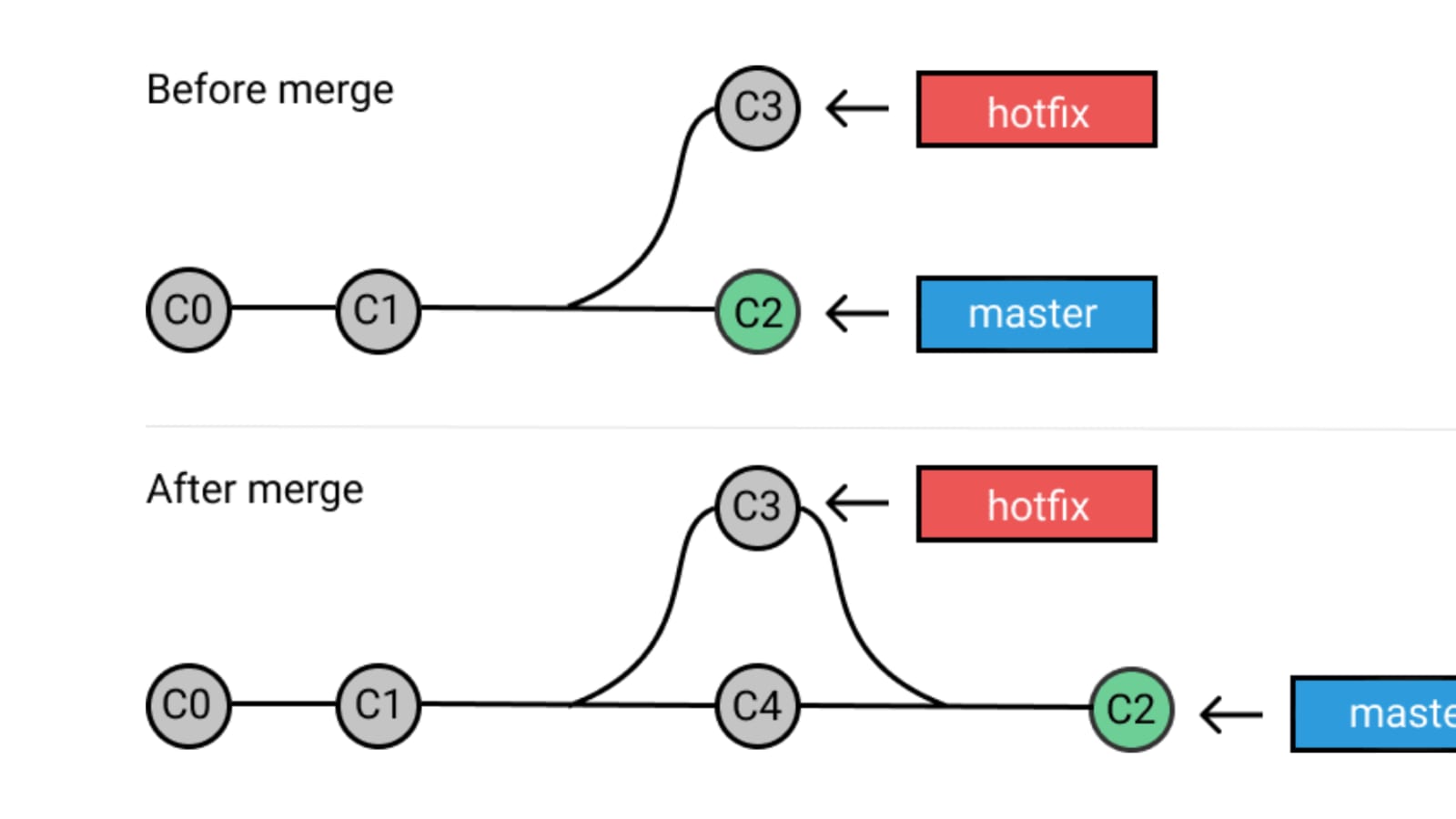
* **git checkout <branch name>:** It is used to switch to the new branch.
* **git log:** It is used to view the history of commits for a repo.
* **git Stash:** Used when we want to save our work without committing the code. It takes our modified tracked files & saves them on stash of unfinished changes, that we can reapply at any time.
* **git revert<commit-id>:** It reverts the commit that has already been pushed & made public.
* **git diff:** It helps in checking the difference between two versions of a file.

(git diff <commit-id of version x> <commit-id of version y>)

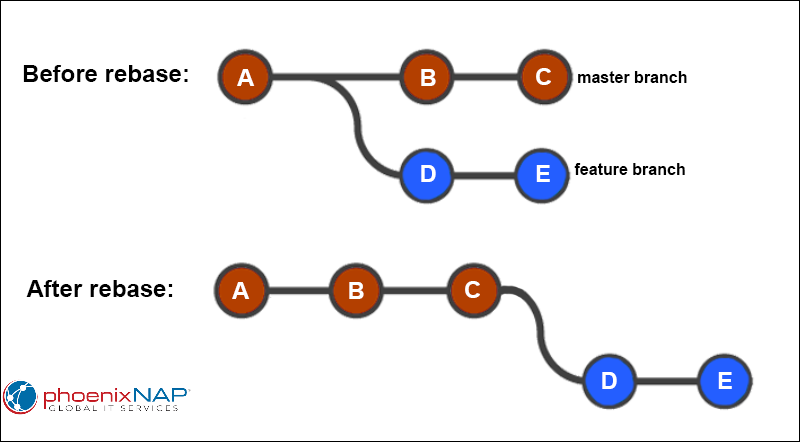
Merging in Git:

* If the developer has finished his code on his branch, then code will have to be combined with master branch. This can be done using two ways.

1. Git Merge
2. Git Rebase

Git Merge: If we want to apply changes from one branch to another branch, then we use merge command. “git merge <source branch name>”

* It’s preferred to switch to the master branch before any branch wants to merge with it.
* Should be used on remote branches, since history does not change.

Git Rebase: It’s an alternative to git merge command. And should be used on local branches, since history does change.

“git rebase <source branch>”

Merge Conflicts:

* Merge conflicts occur when changes are made to the same line of a file. For example if one person edits a file & another person deletes the same file.

Resolving Merge Conflicts:

To resolve merge conflict use the following commands.

* git status: To identify conflicted files.
* git log --merge: It produces the list of commits that are in conflict with source branch.
* git pull --rebase origin master: To know where conflict is.
* git diff: It is used for comparing branches, commits, and files.
* git mergetool: To solve conflict.
* git rebase --continue : To continue next.
* git merge --abort: To abort the conflict.

What is Forking?

A fork is a copy of a repository allows you to freely experiment with changes without affecting the original project.

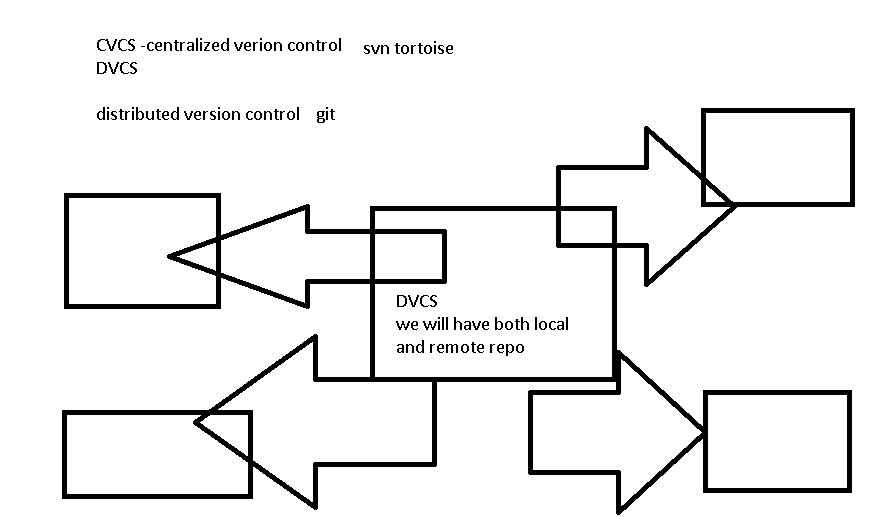
**Forking in Github**

1.Forking is a github concept, it has nothing to do with git software

2.It is used to copy someone else’s repository to your own repo in github

3. The changes made to a forked repository are not reflected in the parent repository.

Git setup:

****