**Devops:**

- **Ansible:**
- **Chef**
- **CICD**
- **Docker & K8S**

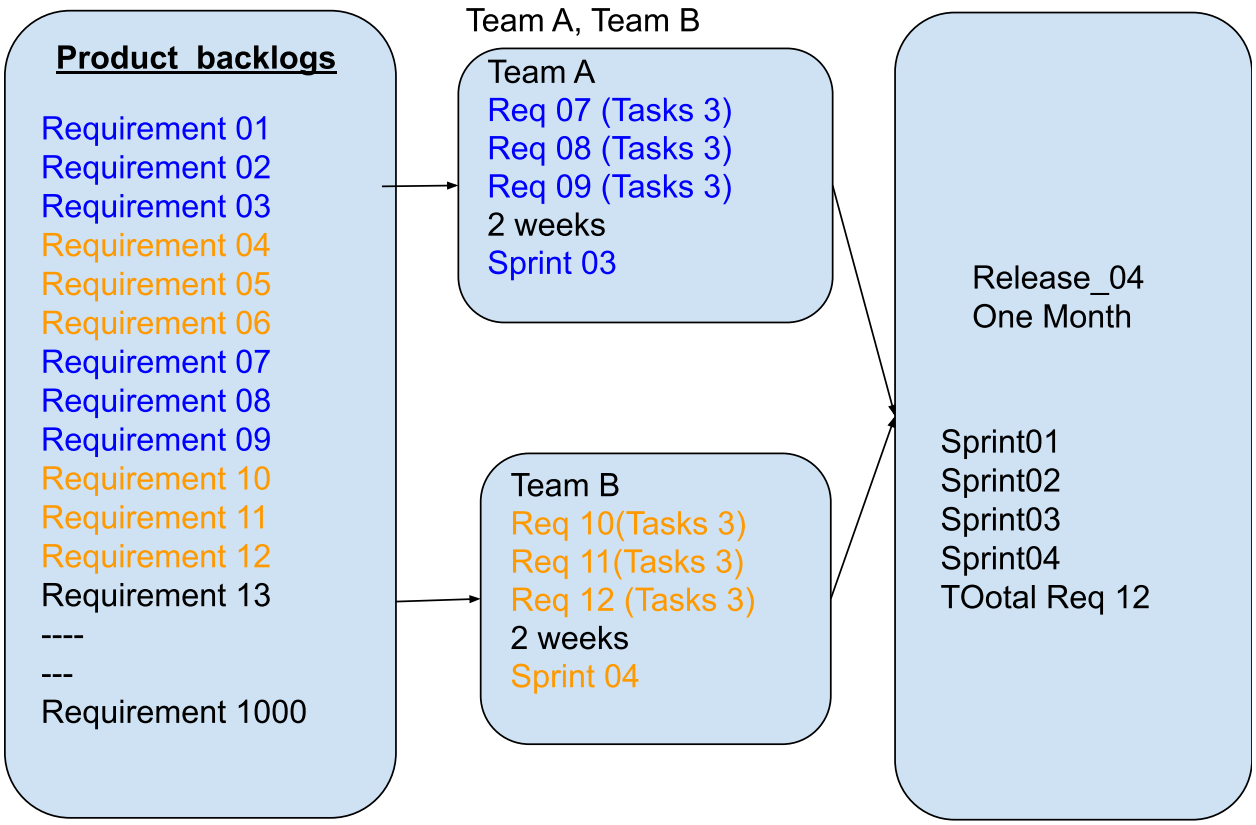
***CICD: 4 weeks:***

1) **Version control : GIT/GiThub**
2) **Build tool : Maven**
3) **Automation CI : Jenkins**
4) **Artifactory CD : Jfrog**
5) **Code Analyzer : sonarQube**

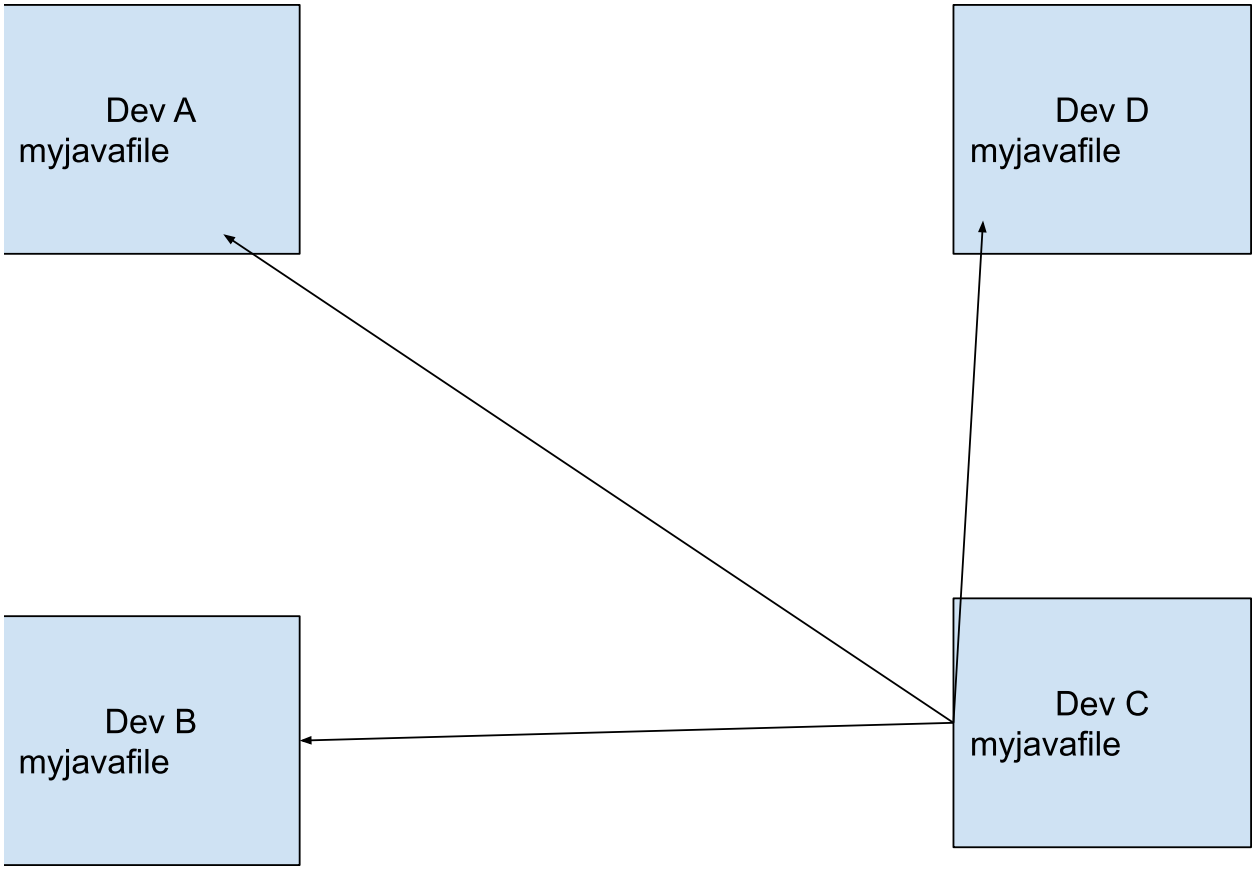
**What is SDLC?**
**SDLC phases?**
**SDLC models?**
**What is a waterfall ?**
**What is Agile ?**
**Waterfall vs Agile?**
**Sprint?**
**Release?**
**Sprint lifeCycle?**

**How to create an AWS account?**
**How to create AWS servers?**
**How to use AWS servers?**

**What is SDLC?**

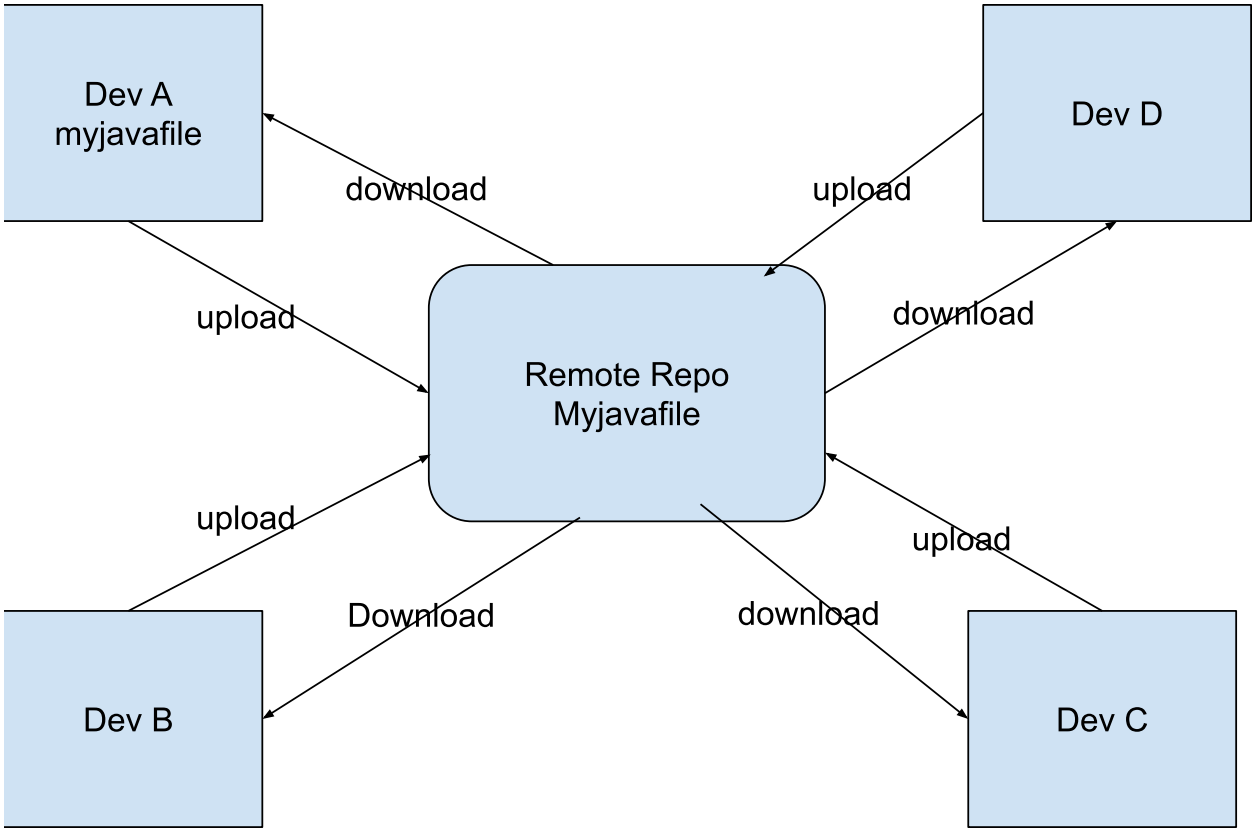



**With Version control**




**In 100% of work**
**80% of time for merge code**
**20% of time for write code**

**With Version control:**



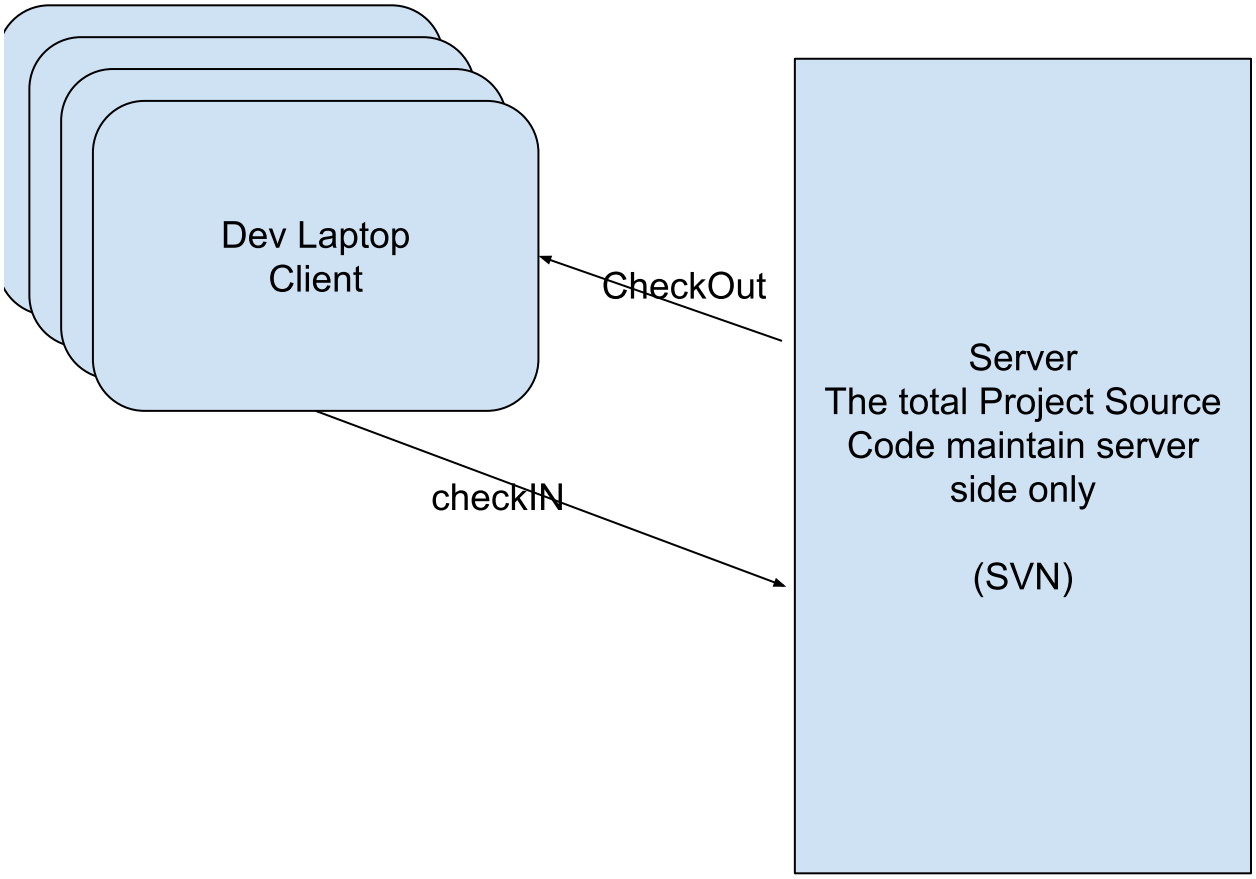

**In 100% of work**
**80% of time for write code**
**20% of time to merge .**

1) **We can able to create versions for every change with time stamp**
2) **We are able to go back to old versions at any point of time.**
3) **We can auto merge ( if it is possible)**
4) **We can able create branches**

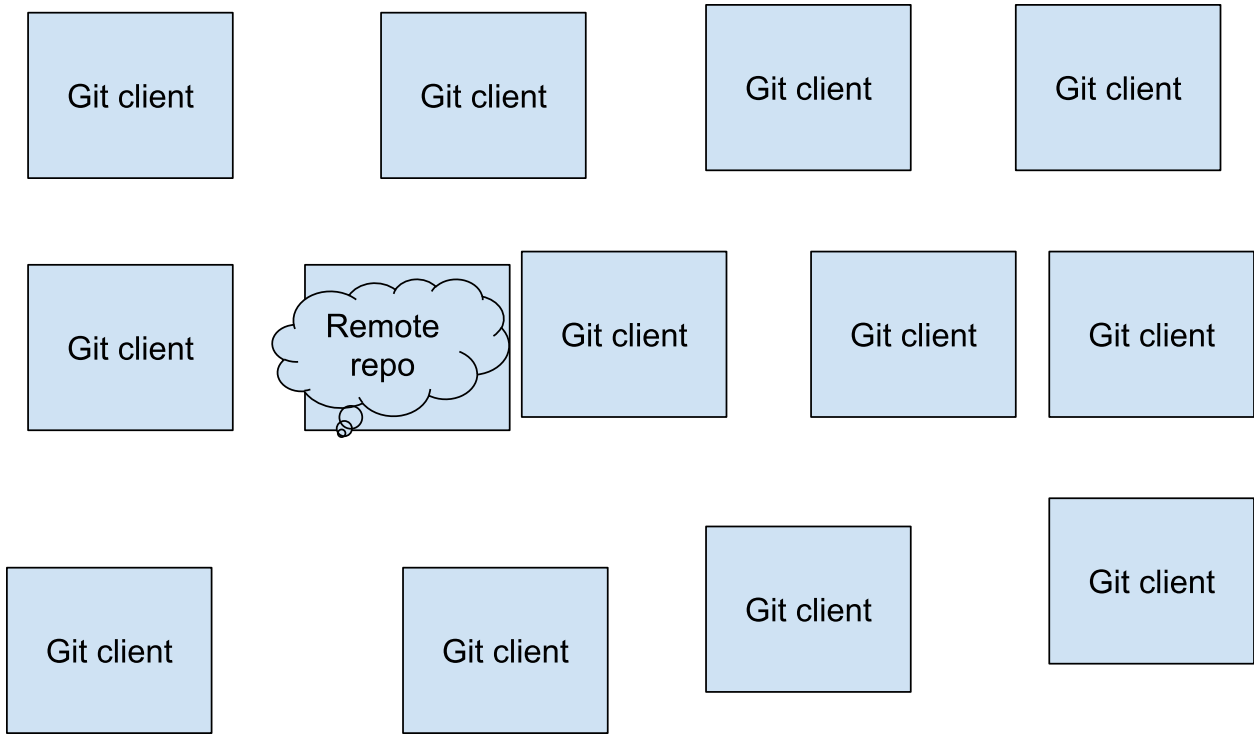
**Version control are two types:**

1) **Client and server ARK**
2) **Distributed ARK**

**Client  and server ARK:**




**Distribution ART: GIT**



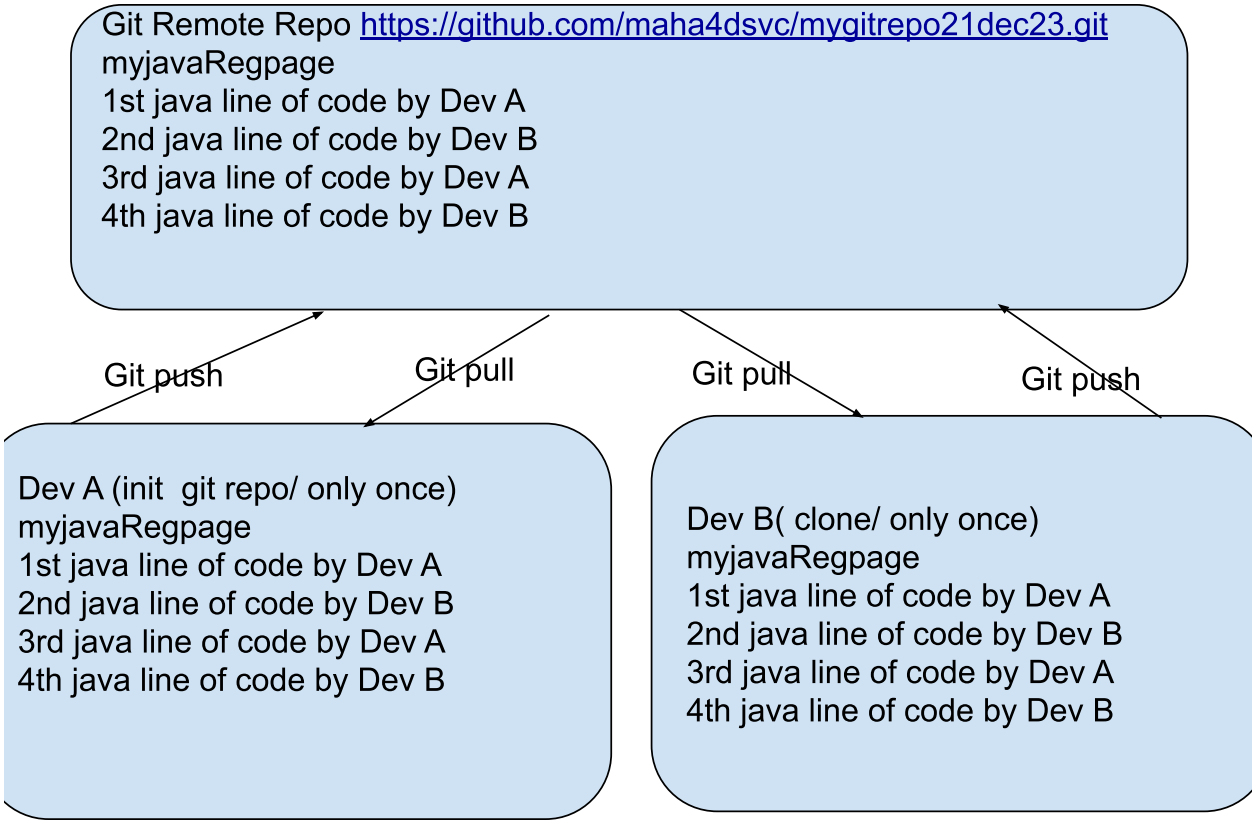

**How can we work with git ?**

1) **Create github account and login**
2) **Create a token, it is just like password**
3) **Install git on laptop**
4) **Create a Remote repo**
5) **Create a folder and open and open git bash and execute below commands (ONLY ONCE)**

```
echo "# myGitRepo19dev23" >> README.md
git init
git add README.md
git commit -m "first commit"
git config --global user.name "maha"
git config --global user.email "maha@gmail.com"
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/maha4dsvc/myGitRepo19dev23.git
git push -u origin main
(username/ passwd /token)
```

6) **For Every change , change may be create /delete/ update /edit**
   **git add -A**
   **git commit -m “comment”**
   **git push**

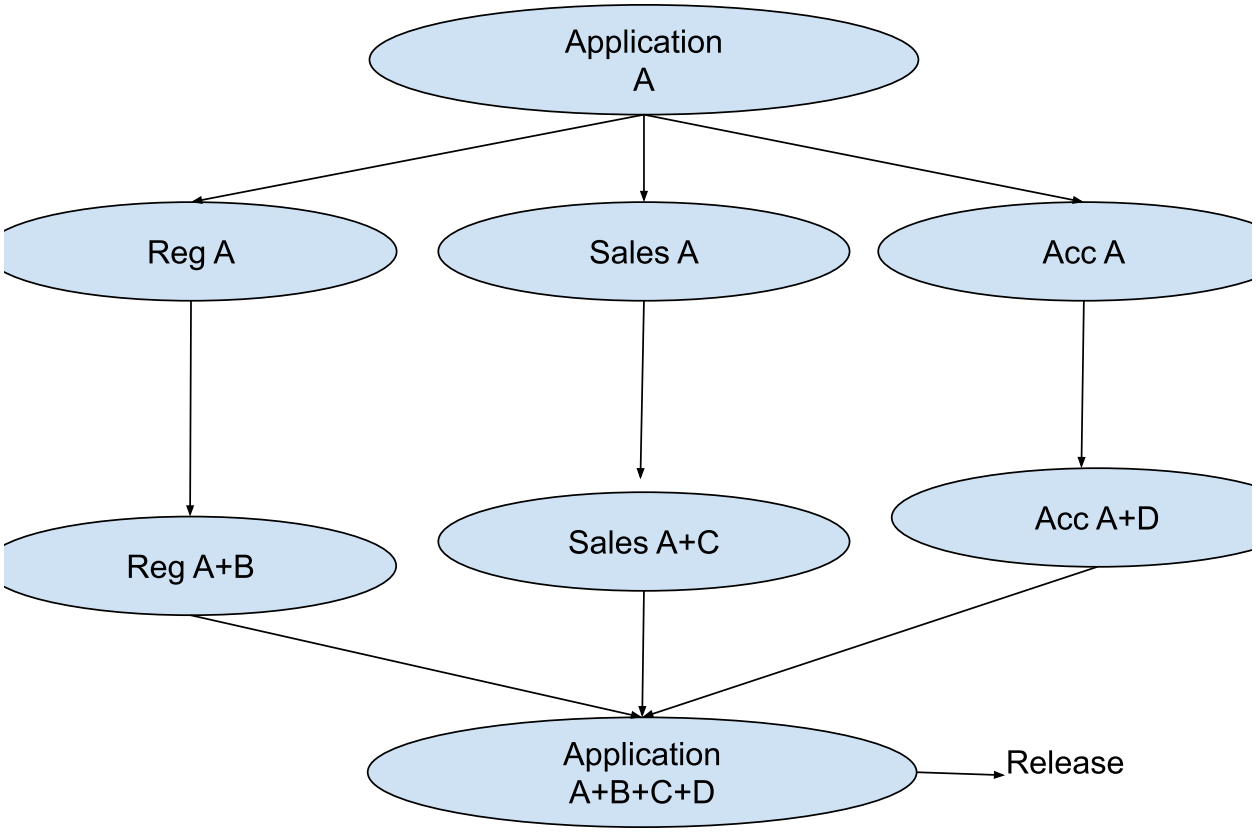
**Working directory : any change (create /delete/update)**
**Staging Area : git add -A**
**Local Repo : git commit -m “comment “**
**Remote Repo : git push**




**What is branche?**

**What is an application?**

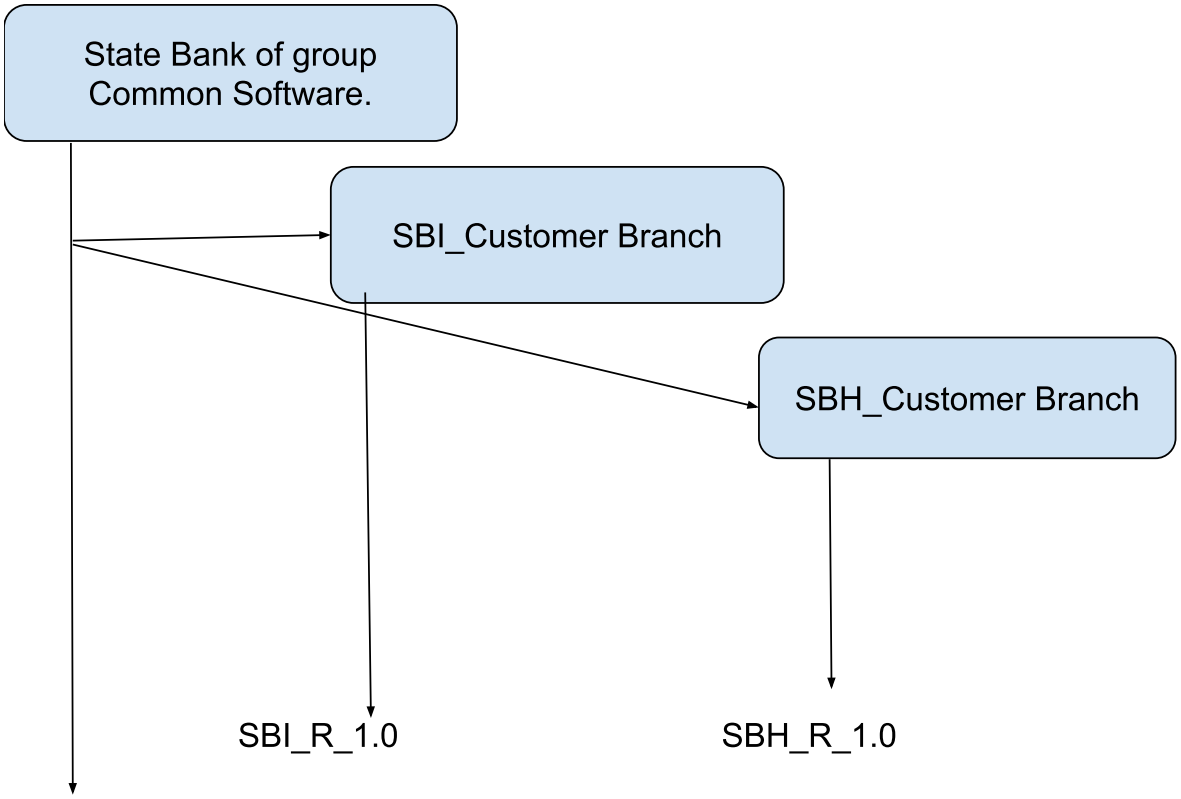
- **Reg and Login**
- **Sales and payments**
- **Account and DashBoard**



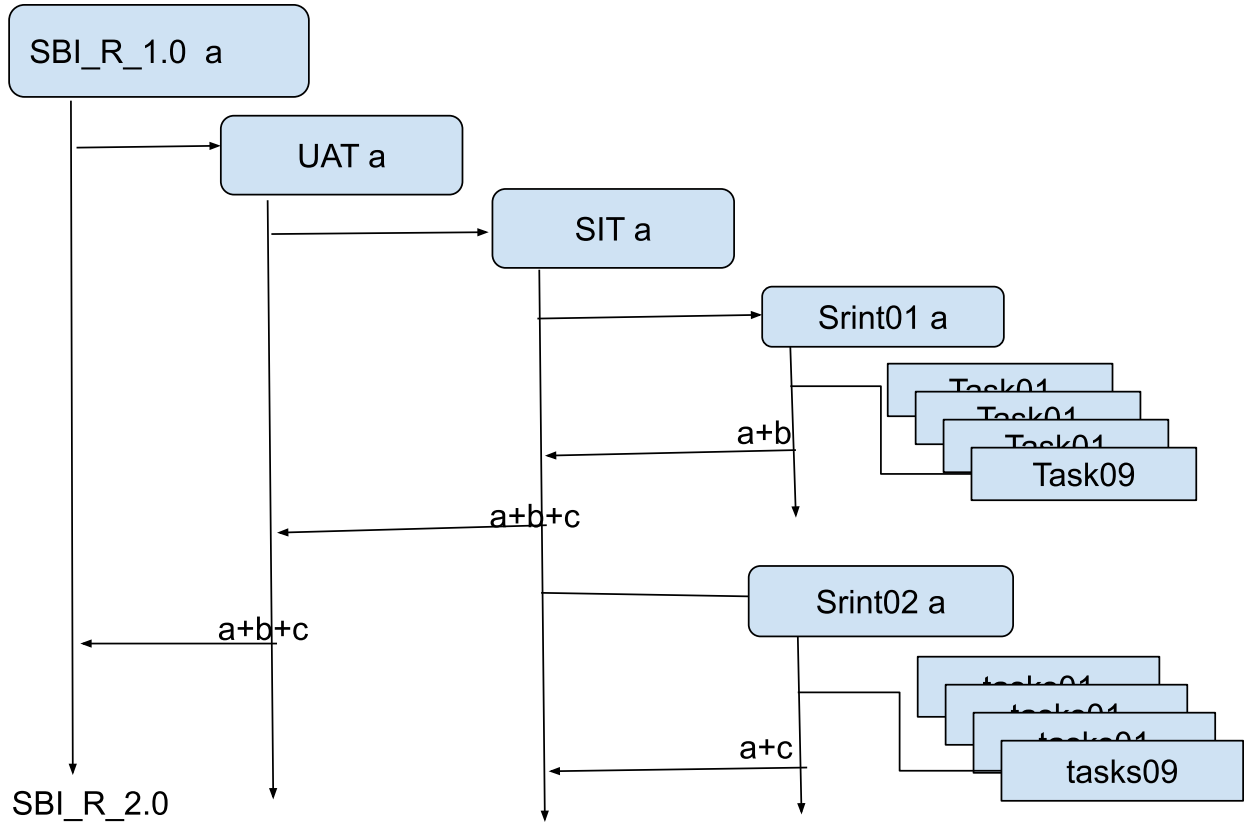

**Branches are Two types:**

1) **Customer branch**
2) **Release branch**

1) **Develop A separate software for each hospital**
2) **Devops A common software all hospital, alter according hospital and deliver**




**Release branch:**




**git checkout -b <new Branch name> : for Creating New branch**
**git checkout <old branch name> : for switch to old branch**
**git merge : merge branches**

**How does git work?**
**Git cat-file**
**-t**
**-e**
**-s**
**-p**

**Git stash**
**1st day of java lines of code completed**
**2nd day of java lines of code completed**
**3rd day of java lines of code completed**
**4th day of java lines of code completed**

**AS Dev , he doing task and complete with in 5days**
**MyjavaRegpage**
**Complete  manager work**
**1st day of java lines of code completed**
**2nd day of java lines of code completed**
**3rd day of java lines of code completed**
**4th day of java lines of code completed**
**5th day of java lines of code completed**

**Git Cherry-pick  c1, c2 c3**
**Git Rebase**

1) **It is only for reuse**
2) **It is not maintain any history**

**Git Restore**
**Git reset**
**Git Revert**

**Git client**

**Working directory : any change (create /delete/update) : git restore <file name>**
**Staging Area : git add -A : git restore --staged <filename>**
**Local Repo : git commit -m "comment " : git reset HEAD~1**

1) **It is not maintain any history**
2) **We can only reset from Head**
3) **We can't reset any middle of commit**

**Default : change back to working directory : git reset HEAD~1**
**Soft : change back to staging areas : git reset --soft HEAD~1**
**Hard : change will be remove : git reset --hard HEAD~1**

**Remote Repo : git push : git revert <commit id>**

1) **It is maintain history**
2) **We can revert any commit**

**Git Client**

**Target folder = maven workspace**
**Mvn clean = delete workspace**
**Install = move war file from workspace to local repo**

**mvn clean compile = download + delete + compile**
**mvn clean test = download + delete + compile +test**
**mvn clean package = download + delete + compile +test +package**
**mvn clean install = download + delete + compile +test +package + install**
**mvn clean deploy = download + delete + compile +test +package + install + deploy**

**Install jenkins:**

1) **Create aws ec2 server with ubuntu 20 and login**
2) **Install open jdk11**
   **#** `sudo apt-get install openjdk-11-jdk`
3) **Install maven**
   **#** `sudo apt-get install maven`
4) **Install git**
   `sudo apt-get install git`
5) **Install jenkins**

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins
```

6) **Configure jenkins server**
7) **Create a new job and execute.**

## ***Build periodically:***

**By giving Schedule time , my Jenkins job will be triggered without any condition.**

## ***POLL SCM:***

**By giving Schedule time , my Jenkins job will be triggered with conditions.**

***Condition*: when jenkins job triggers, first verify git status, if any new commits on git repo then only jenkins job will trigger or else simple skip.**

**45 mints no commit skip**
**46th mint commit**

## ***GitHub hook trigger:***

**When commit code, execute jenkins job**

**UP /Down stream jobs**

**Job01**

**Job02  upstream job1 and down stream job3**

**job03**

**Artifact (war file)**
**Upload artifact (war file) into Remote repo**
**CD/Build repo/release**

**Artifactory: Jfrog**

1) **Create Aws redhat server with t2.medium**
   **$ sudo -i**
   **# yum update -y**
2) **Install openjdk 11**
   `# sudo yum install java-11-openjdk`

3) **Create Jfrog account for install doc and license**

4) **Install jfrog** https://jfrog.com/artifactory/install/
   # yum install wget
   # wget -O artifactory-pro.rpm
   "https://releases.jfrog.io/artifactory/artifactory-pro-rpms/jfrog-artifactory-pro/jfrog-artifactory-pro-[RELEASE].rpm"

   # sudo yum install ./artifactory-pro.rpm -y

   # sudo systemctl start artifactory.service

5) **Configure jfrog server**
6) **Install artifactory plugin on jenkins server**
7) **Install maven tool**
8) **Configure jfrog server with jenkins server**
9) **Create jenkins job  for upload artifact (WAR FILE )  into jfrog server**

Master and slave