**SESSION - 74**

**Branching Strategy**

**==================**

what are the branches

how you bring changes from DEV to PROD

other branches to main/master branch

git flow -> master develop release feature and hotfix --> waterfall

feature branch --> main merge --> agile devops microservices

trunk based --> main branch

what are the long lived branches

what are short lived branches

longlived branches --> master develop

short lived branches --> feature release and hotfix

main/master --> PROD

**develop** --> active developments going on

**========**

source: main/master

destination: main/master

**short-lived branches**

**=====================**

**feature-1**

**========**

source: develop

destination: develop

clone --> build --> unit test cases --> scan --> create image --> push image --> deploy into DEV

successfully merge into master

**feature-2** --> until you merge to develop

**=========**

1. pull the changes from develop

2. merge/rebase

3. raise PR and merge to develop

Release = New features + Bug fixes + improvements

**Release branch** --> release-1.3

**===============**

source: develop

destination: master and develop

lifetime is until that release is successfully sent to PROD. You get the changes to master/main

deploy into environments DEV, QA, UAT and test the application

deploy to PROD. If successfully deployed then you merge these changes into master/main and develop

you can delete release branch when it is successfully deployed into PROD.

waterfall model

product based has to support multiple versions at a time --> 20, 19, 18, 17

git checkout release-18

**hotfix**

**=======**

SLA --> Service Level Agreement

priority-1 --> max 4 hours --> business is getting affected

source: main

destination: develop and main

hotfix-emergency-ticket-price --> approval from chairman/CEO

test in develop once and then deploy to PROD. you need to get changes into main and develop both

# **Branching Strategy**

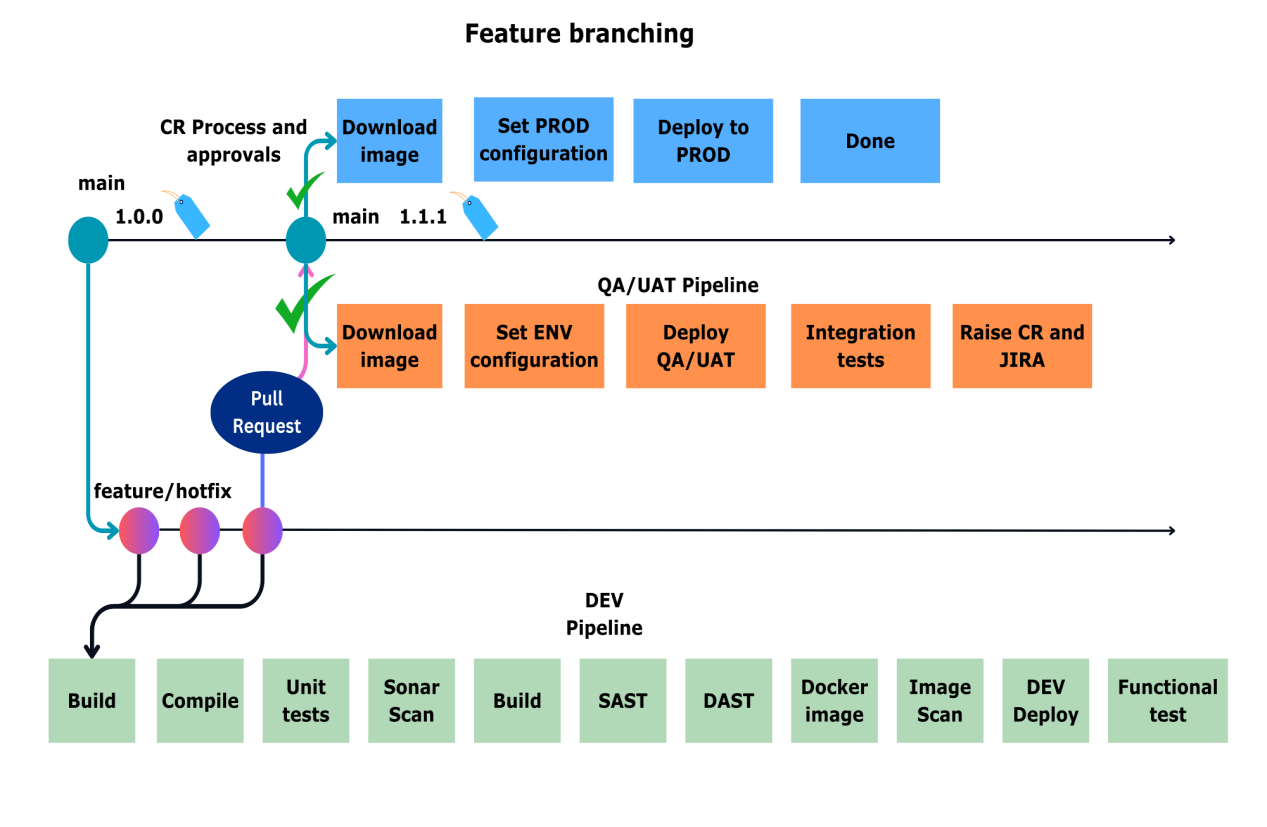
A well-defined branching strategy is crucial for maintaining a clean, organized, and manageable codebase.

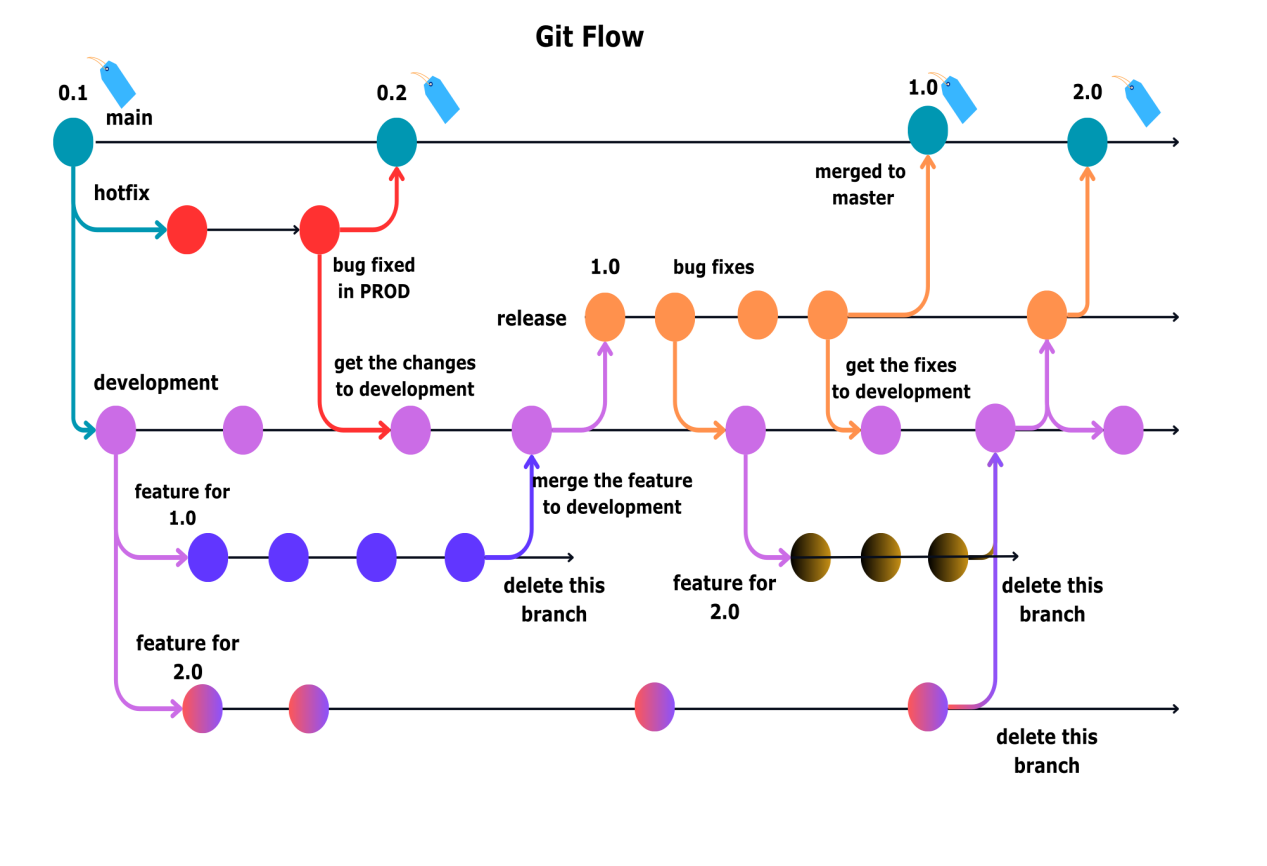
Here are some key reasons why having a branching strategy is important:

* Parallel Development
* Code Stability
* Code Reviews and Collaboration
* CICD

There are 3 popular branching strategies.

* Git flow
* Feature branching
* Trunk based





featuring branching strategy

**master/main and feature**

**========================**

web applications will not have versions

feature -->

clone --> build --> unit test cases --> scan --> create image --> push image

merge to main --> DEV, QA, UAT, SIT, PROD

main --> hotfix --> deploy to DEV -> merge the changes to main -> deploy to PROD

change request

**Microservices**

**==============**

f-1

git stash --> pause the develop

git checkout main

git pull origin main

git checkout -b h1

few commits

deploy to dev

merge to main --> deploy to PROD

checkout f-1

git merge main (local laptop)

github.joindevops.com --> it is in our servers