DHL Version Control Document

Contents

[1. DHL Project structure in GitHub 2](#_Toc41249667)

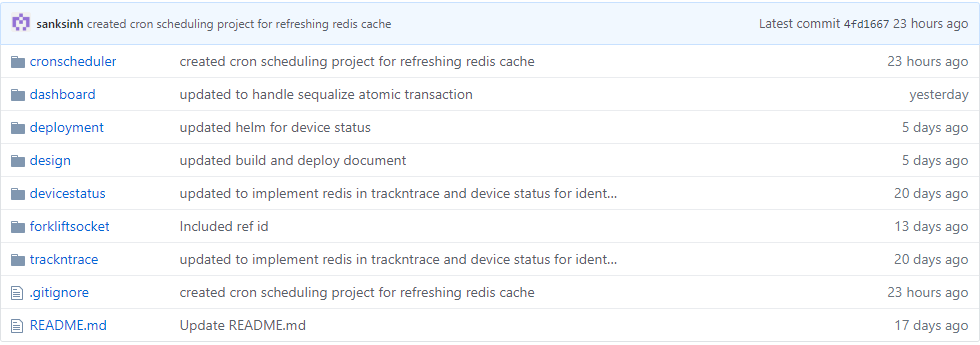
[2. Version control flow 3](#_Toc41249668)

[2.1 Master Branch (Production release) 4](#_Toc41249671)

[2.2 Hotfix Branch (Bug Fix release) 4](#_Toc41249672)

[2.3 Feature Branch (New Enhancement) 5](#_Toc41249673)

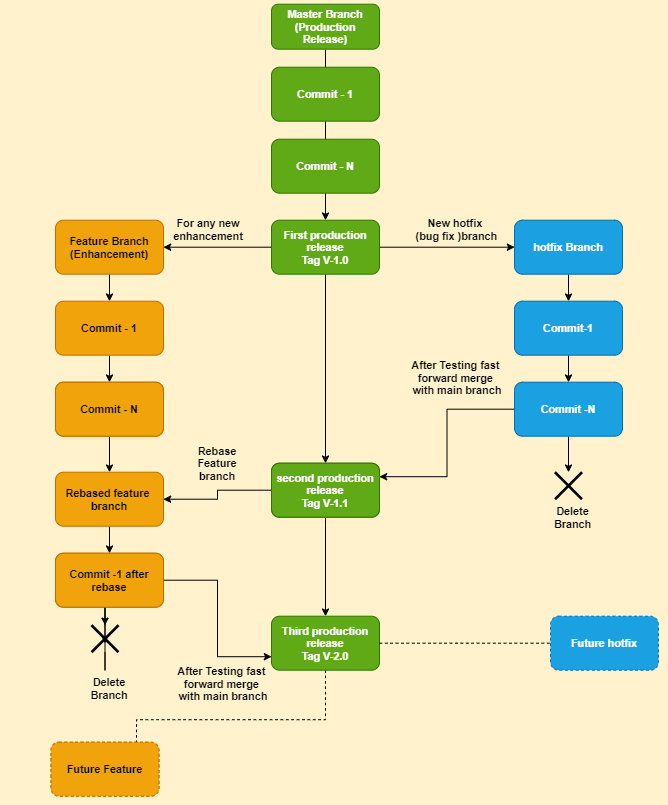
# DHL Project structure in GitHub



Following project included to implement DHL NGFS solution

* Dashboard : DHL will use dashboard UI to access shipment and asset detail
* Trackntrace : Trackntrace will determine shipment and device status
* Devicestatus :Devicestatus will use to get forklift cone device status
* Forkliftsocket :- This will give real time current location of forklift inside warehouse
* Deployment:- Deployment script for deploying all microservices

# Version control flow



Following branch will be created in DHL:



## Master Branch (Production release)

Following activities can be done in Master branch

* Master branch will be production release branch where user will keep committing all changes till first production release
* Developer will tag this branch with release name e.g V-1.0 while releasing to production deployment

Following command will be used to tag Master with release version e.g V-1.0

git checkout master

git tag -a v1.0 -m " version 1.0"

git push origin v1.0

* Developer will create new branch for any bug fixes post production e.g hotfix branch
* Developer can also create any new enhancement branch e.g feature branch based on requirement .This branch life will be longer than hotfix branch
* Developer can roll back to stable version of release tag any time if any issue found after future release and want to build and deploy from stable release

Following command will be used to rollback to stable version of production release

git checkout master

git checkout V1.0

## Hotfix Branch (Bug Fix release)

Following activities can be done in Hotfix branch

* Hotfix branch will be used to fix any issue unearth while production release .Developer will create temporary hotfix branch out of latest production release master branch

e.g to create branch :-

Following command will create and switched to a new branch "hotfix"

git checkout -b hotfix

* Developer will keep committing till issue get resolved and tested successfully
* Developer will use dev/staging deployment env to test the fixes
* After successfully test developer will merge back to master branch

Following command will merge "hotfix" to master

git checkout master

git merge hotfix

* Master branch will be tagged with next minor release version e.g V-1.1

Following command will be used to tag Master with release version e.g V-1.1

git checkout master

git tag -a v1.1 -m " version 1.1"

* Developer will also rebase to any existing feature branch where other developer working on enhancement so that all latest fixes will be available to feature branch also
* After merging to master branch developer will delete this branch

git branch -d hotfix

git push origin --delete hotfix

## Feature Branch (New Enhancement)

Following activities can be done in Feature branch

* Feature branch will be used to do any new enhancement .Developer will create temporary Feature branch out of latest production release master branch
* Developer will keep committing till new enhancement get completed
* Developer will use dev/staging deployment env to test new enhancement
* Enhancement branch will keep rebasing from Master branch for any bug fixes included in minor release version so that it will be in sync with all issued fixed

Following command will be used to rebase Master with release fearure branch

git checkout feature

git rebase master

* After successfully tested Enhancement branch will be merged with Master branch for next release version e.g V-2.0

git checkout master

git merge feature

git tag -a v2.0 -m " version 2.0"

* After merging to master branch developer will delete this branch

git branch -d feature

git push origin --delete feature