CARD – 1 : NEW FEATURE WHEN IN SPRINT

Objective: To create a feature branch local and remote in-line with the GIT Branching Model

Frequency: As and when required to start a new feature

Sr.No.	COMMAND	PURPOSE
1	`git checkout develop`	Switch local branch to develop
2	`git pull`	Get latest updates from upstream develop

3	`git new-feature-web dev1-US0021-createUser`
	OR
	`git new-feature-auth dev2-US0031-authNotify`
	OR
	`git new-feature-cms dev3-US0041-credentialRevoke`
		Depends on your workstream cell, use one of the 3 git aliases;
		- `new-feature-web`
		- `new-feature-auth`
		- `new-feature-cms`
		to create a new feature.
		Note: the feature naming convention is :
		<Developer Initials>-<JIRA Issue no:>-<Brief Feature Desc>

CARD – 2 : EVERYDAY – PART 1

Objective: To keep your feature branch on remote server updated with your local changes

Frequency: Multiple times a day

WHILE YOU ARE ON YOUR FEATURE BRANCH

Sr.No.	COMMAND	PURPOSE
1	`git status`	To ensure the working branch is clean and there are no pending files waiting for commit

2	`git add .`
	OR
	`git add *.java`
	OR
	`git add <filename1> <filename>`
		If status does not show a clean working branch, use this to add un-staged files for commit
		Note: "`git add .`" is used to stage all files

3	`git commit`	Commit staged files to local feature branch
4	`git push`	Push local commit to upstream feature branch

CARD – 3 : EVERYDAY – PART 2

Objective: To keep your feature branch up-to-date with other completed features on develop branch.

Frequency: Once a day – to avoid one big merge at the end of the feature or at the end of the sprint.

WHILE YOU ARE ON YOUR FEATURE BRANCH

Sr.No.	COMMAND	PURPOSE
1	**Execute all commands from 'EVERYDAY - PART 1':** To keep your feature branch on remote server updated with your local changes	
2	**git checkout develop** **git pull** **git checkout <name-of-feature>** **git pull** **git pull origin develop**	Update all local branch references from upstream server Get all changes from develop branch onto your feature
3	**git mergetool**	If the above step results in merge conflict(s), resolve conflicts manually using this command or by using a tool of your choice e.g. tortoisemerge.
4	**git status**	To check staged/unstaged/modified files
5	**git commit**	Commit staged files to local feature branch
6	**git push**	Push local commit to upstream feature branch

CARD – 4 : FINISHING A FEATURE

Objective: To finish and merge your feature back into develop branch.

Frequency: As and when work on a feature finishes.

Sr.No.	COMMAND	PURPOSE
1	**Execute all commands from 'EVERYDAY - PART 1':** To keep your feature branch on remote server updated with your local changes	
2	**Execute all commands from 'EVERYDAY - PART 2':** To keep your feature branch up-to-date with your local feature	
3	**Execute feature Jenkins job:** Make sure this builds successfully, if any issues in build that requires code changes, re-run steps 1, 2 & 3	
4	**git checkout develop** **git pull** **git checkout <name-of-feature>** **git pull** **git pull origin develop**	Update all local branch references from upstream server Just to make sure there is nothing to update from develop, if there is any then re-run steps 1, 2 & 3
5	**git checkout develop**	Switch to the local develop branch
6	**git pull**	Update local develop with latest changes to upstream develop
7	**git flow feature finish <name-of-feature>**	To finish your feature
8	**git push**	Push any local changes to upstream repository