NOTEBOOKS BASICS :-

Magic Commands :-

- Specific to databricks notebooks.
- Starts with % symbol (% symbol will be the 1st character in a cell for magic command being used).
- We can use 1 magic command per cell.

Language Magic Commands :-

- Used to execute a cell containing code of a different language than the default notebook language.

Ex :- %python , %sql , %scala , %md (↓ markdown)

%run command :-

- Using %run command, we can run a notebook from another notebook.
- Relative path is provided to the other notebook to be run (called reference notebook).
- Reference notebook(s) runs as a part of current notebook, and so all the temporary views and all local declarations of current notebook will

be available in the reference notebook, and vice versa.

Ex: `%run ../folder/file_name`

display() function :-

display() is a function specific to Databricks, When we have a tabular data, returned by a Python cell, we can use this function to get a preview similar to an sql table in look.

Properties of display() function :-

- Previews results upto 1000 records.
- Provides button to download data as CSV.
- Allows rendering plots.

Databricks Repos :-

It is a feature of databricks that takes all of the practices involved in managing workspace assets, and applies them to a similar env. that is backed by revision control using git.

⇒ Databricks repos, by its Git Integration, supports Git Versioning, CI/CD Integration.

⇒ Databricks supports following Git Service Prov.:-
Az. DevOps, GitHub, GitLab, Bitbucket.

⇒ Supported operations include :-

a) Cloning a repo:- Creates a local working copy of the remote git repo.

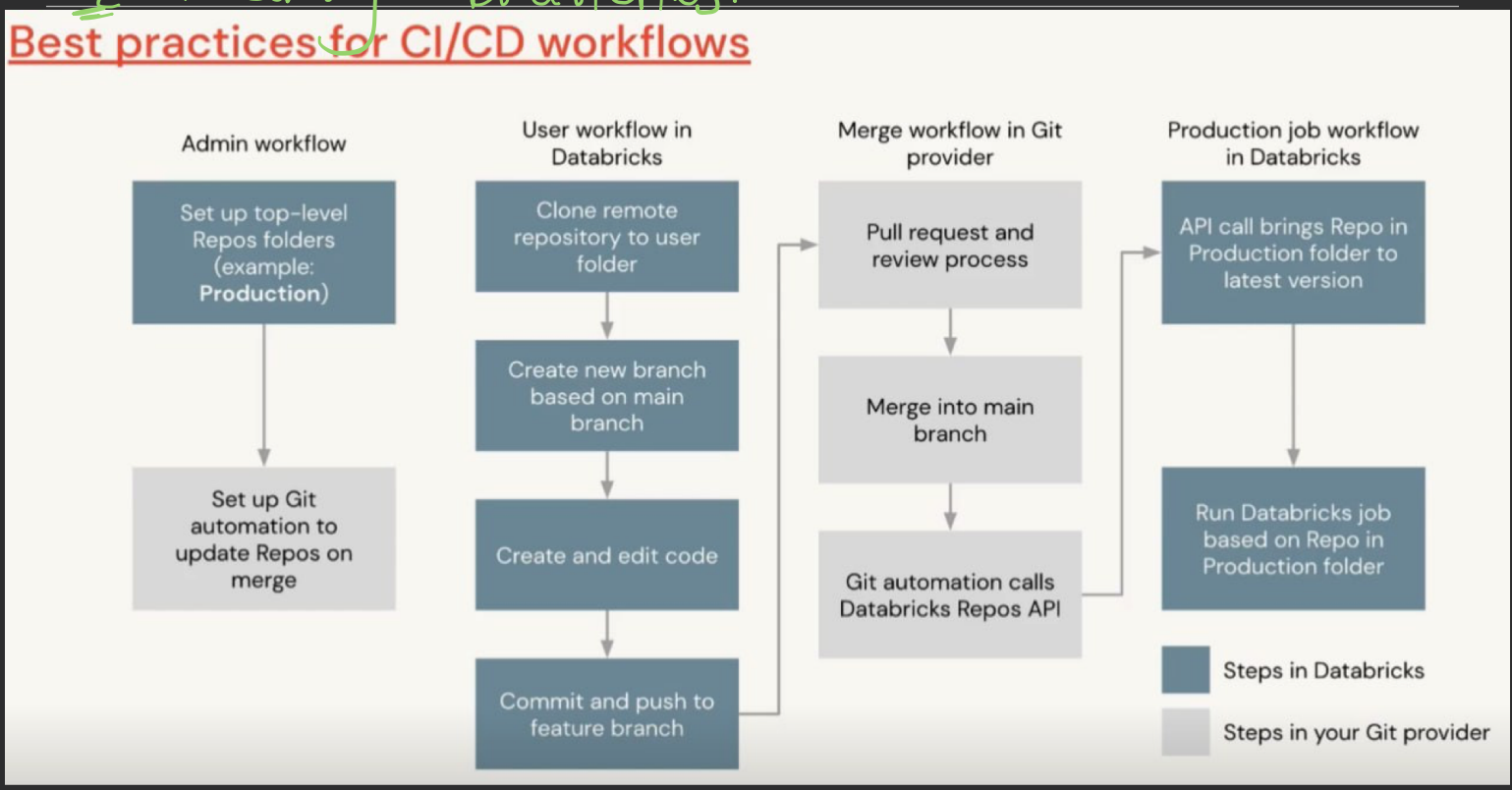
b) Pulling:- Synchronises upstream code changes of remote repo with our local repo.

c) Adding new items, moving & renaming items.

d) Committing & Pushing.

e) Creating branches.

Best practices for CI/CD workflows



⇒ Using Databricks Repos :-

⇒ At first, we need to configure Git Integration in the workspace, and have a repository ready.

⇒ 2 information are required to connect Databricks to a git service :-

(i) Username , (ii) Personal Access Token.

Personal Access Tokens can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over basic authentication

⇒ Git Integration in Databricks :-

open the Data Science & Engg. Workspace in Databricks → Settings → User Settings

↓

Git Integration

⇒ Select the Git provider (Ex :- GitHub)
⇒ Enter username & the Personal Access Token.

≥ Each Databricks repo maps to a Git Repository
So, in order to setup a Databricks repo, at first, we must make a Git Repository.

⇒ Create a GitHub repository, and copy its URL

⇒ In Databricks, go to repos tab → Add repo → Clone remote Git repo option

Paste the repo URL, and rest fields will be auto-filled.
Click Create.
The Databricks repo will be created, and the underlying GitHub repo will be cloned in Databricks platform.

⇒ Under the Databricks repo, we can now create folders, create or import or modify notebooks.
After all the above changes, we should go to the Repo (top-left corner), add Comments, and then Commit & Push the changes.

Best Practise:-

≥ If collaborating, it is advised to Pull changes first, before our modification, such that we always work on the updated version.

≥ Modifications must always be done on feature branches

Code in DBFS = Local Copy

Code in GitHub Repo. = Remote Copy.