the dill down

WITH AHMAD & JAMES

AGENDA

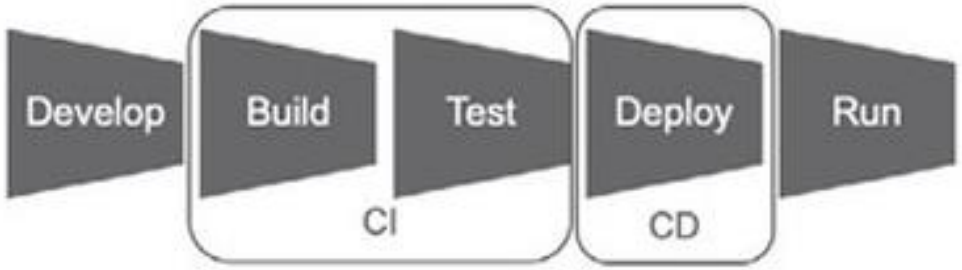
- DevOps & CI/CD Background
- Fabric/Power BI- Git Integration & Source Control

DevOps & CI/CD Background

DevOps focuses on continuous integration (CI) and continuous delivery (CD) of software by automating integration, test and deployment of code. This merging of software development and IT operations ("**DEV**elopment" and "**OP**erationS") reduces time to deployment, decreases time to market, minimizes defects, and shortens the time required to resolve issues."

Source: The DataOps Cookbook: Methodologies and Tools that Reduce Analytics Cycle Time While Improving Quality by Christopher Bergh, Gil Benghiat & Eran Strod

DevOps Process



DevOps & CI/CD Background

CI continuously builds, integrates and tests new code in a development environment.

CD is an automated approach to deploying or delivering software

Source: The DataOps Cookbook: Methodologies and Tools that Reduce Analytics Cycle Time While Improving Quality by Christopher Bergh, Gil Benghiat & Eran Strod

DevOps & CI/CD Background

Guiding principles of DevOps

Use a Version Control System: The artifacts (files) that make this reproducibility possible are usually subject to continuous improvement. Like other software projects, the source files associated with the data pipeline should be maintained in a version control (source control) system such as Git. A version control tool helps teams of individuals organize and manage the changes and revisions to code. It also keeps code in a known repository and facilitates disaster recovery. However, the most important benefit of version control relates to a process change that it facilitates. It allows data-analytics team members to branch and merge.

Branch & Merge: In a typical software project, developers are continuously updating various code source files. If a developer wants to work on a feature, he or she pulls a copy of all relevant code from the version control tool and starts to develop changes on a local copy. This local copy is called a branch. This approach can help data-analytics teams maintain many coding changes to the data-analytics pipeline in parallel. When the changes to a branch are complete and tested, the code from the branch is merged back into the trunk, where the code came from. Branching and merging can be a major productivity boost for data analytics because it allows teams to make changes to the same source code files in parallel without slowing each other down. Each individual team member has control of his or her work environment. They can run their own tests, make changes, take risks and experiment. If they wish, they can discard their changes and start over.

Source: The DataOps Cookbook: Methodologies and Tools that Reduce Analytics Cycle Time While Improving Quality by Christopher Bergh, Gil Benghiat & Eran Strod

DevOps & CI/CD Background

Work with versions

Every version has a description for what the changes in the version do, such as fix a bug or add a feature. These descriptions help the team follow changes in code by version instead of by individual file changes. Code stored in versions can be viewed and restored from version control at any time as needed. Versions make it easy to base new work off any version of code.

Code Together

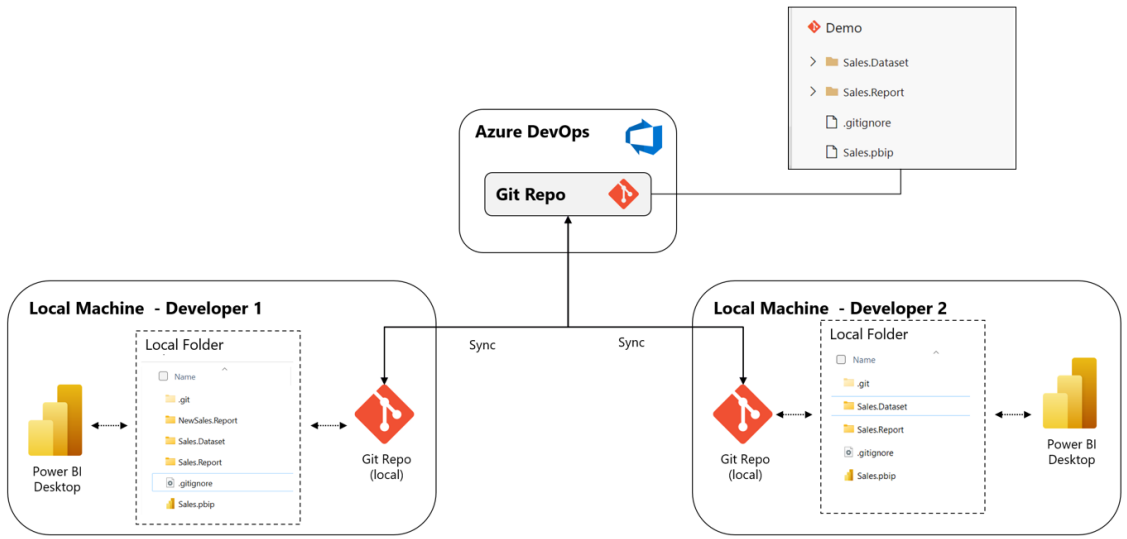
Version control synchronizes versions and makes sure that changes don't conflict with changes from others. The team relies on version control to help resolve and prevent conflicts, even when people make changes at the same time.

Keep a History

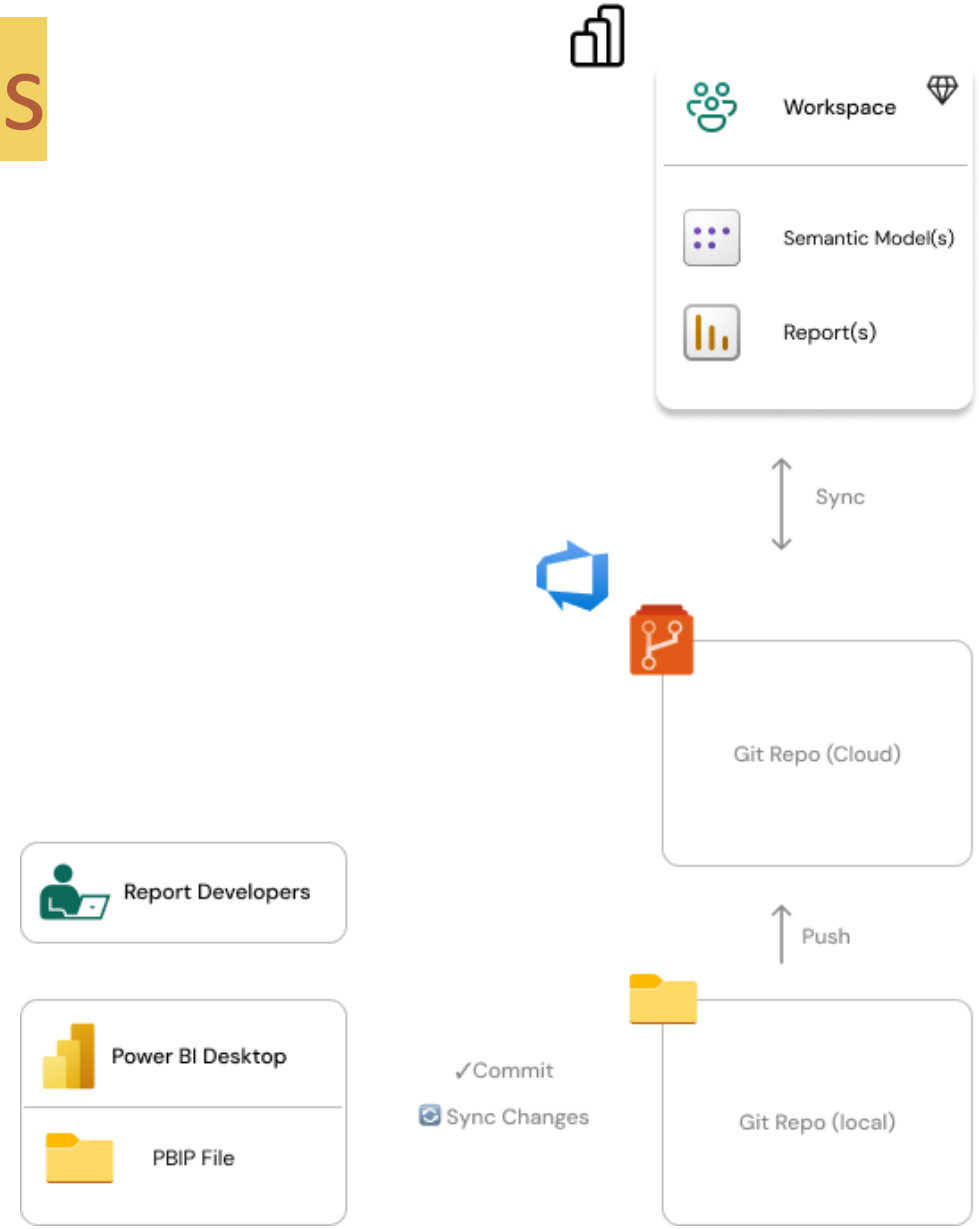
Version control keeps a history of changes as the team saves new versions of code. Team members can review history to find out who, why, and when changes were made. History gives teams the confidence to experiment since it's easy to roll back to a previous good version at any time. History lets anyone base work from any version of code, such as to fix a bug in a previous release.

DevOps & CI/CD Background

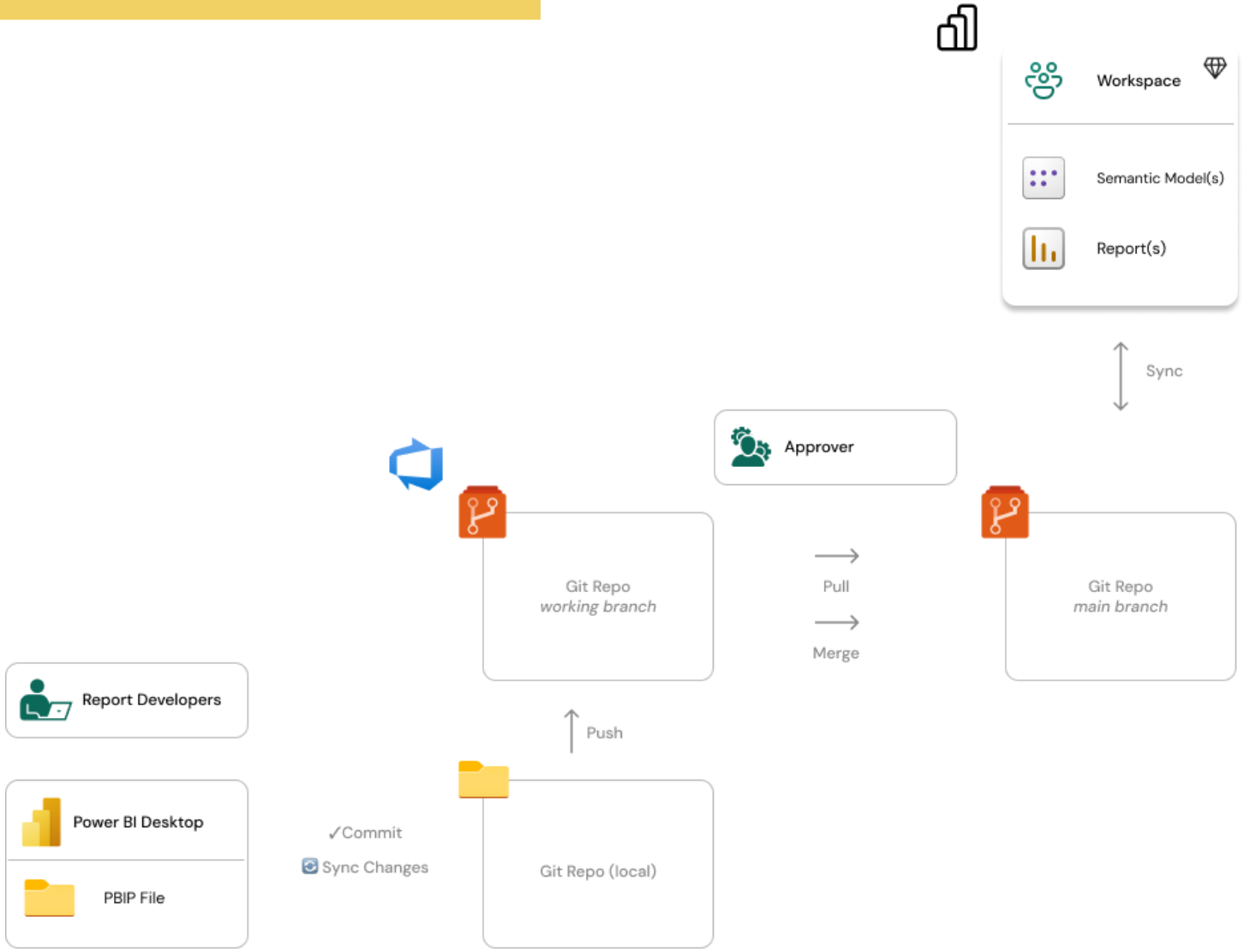
A **Git repository**, or repo, is a folder that Git tracks changes in. There can be any number of repos on a computer, each stored in their own folder. Each Git repo on a system is independent, so changes saved in one Git repo don't affect the contents of another.



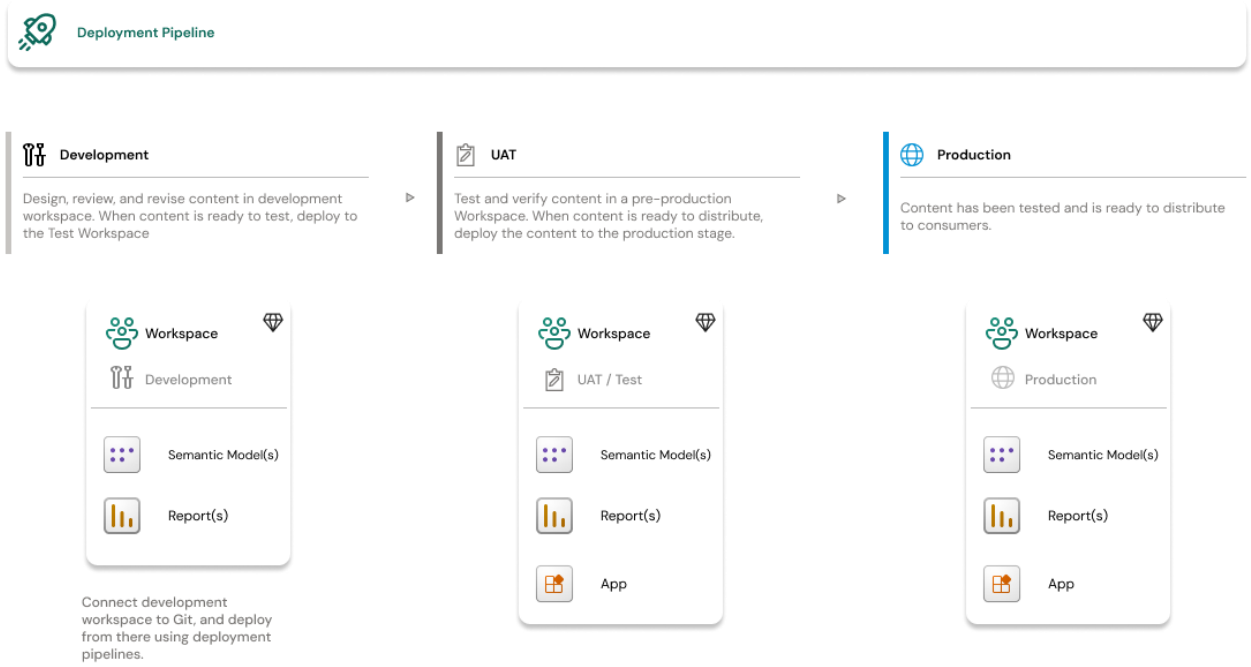
Basic Process

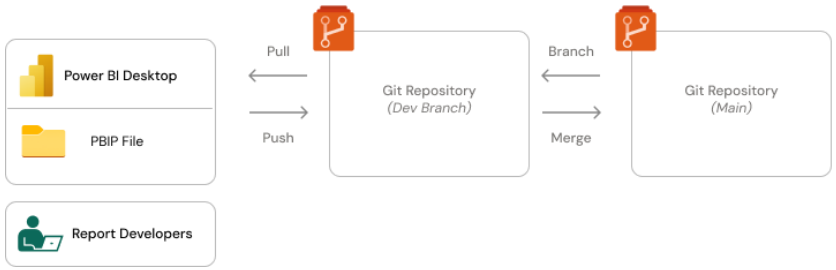


Intermediate Process



End-to-End Process





the

screw up

direct query

Thank you!

See you February 5th 12:00 PST (LinkedIn Event coming soon!)