A Version Control Primer for Databases

Steve Jones (he/him)
Editor, SQLServerCentral
Redgate Software
@way0utwest

Agenda

- Who am I?
- Why use Version Control
- The basics of a versioning software
- Git introduction
- Push/Pull
- Branches
- Merges
- Pull Requests

Steve Jones

Advocate, Redgate Software
Editor, SQLServerCentral
he/him

31 years SQL Server data experience

DBA, developer, manager, writer, speaker in a variety of companies and industries

Founder, SQLServerCentral

Currently the editor in chief, with the goal of helping you learn to be a better data professional every day

14 year Microsoft Data Platform MVP

I have been honored to be recognized by Microsoft for the as a Data Platform MVP working with SQL Server

/in/way0utwest @way0utwest sjones@sqlservercentral.com www.voiceofthedba.com

Keeping track of code is hard

Not really, but it can be….

Versions get confusing

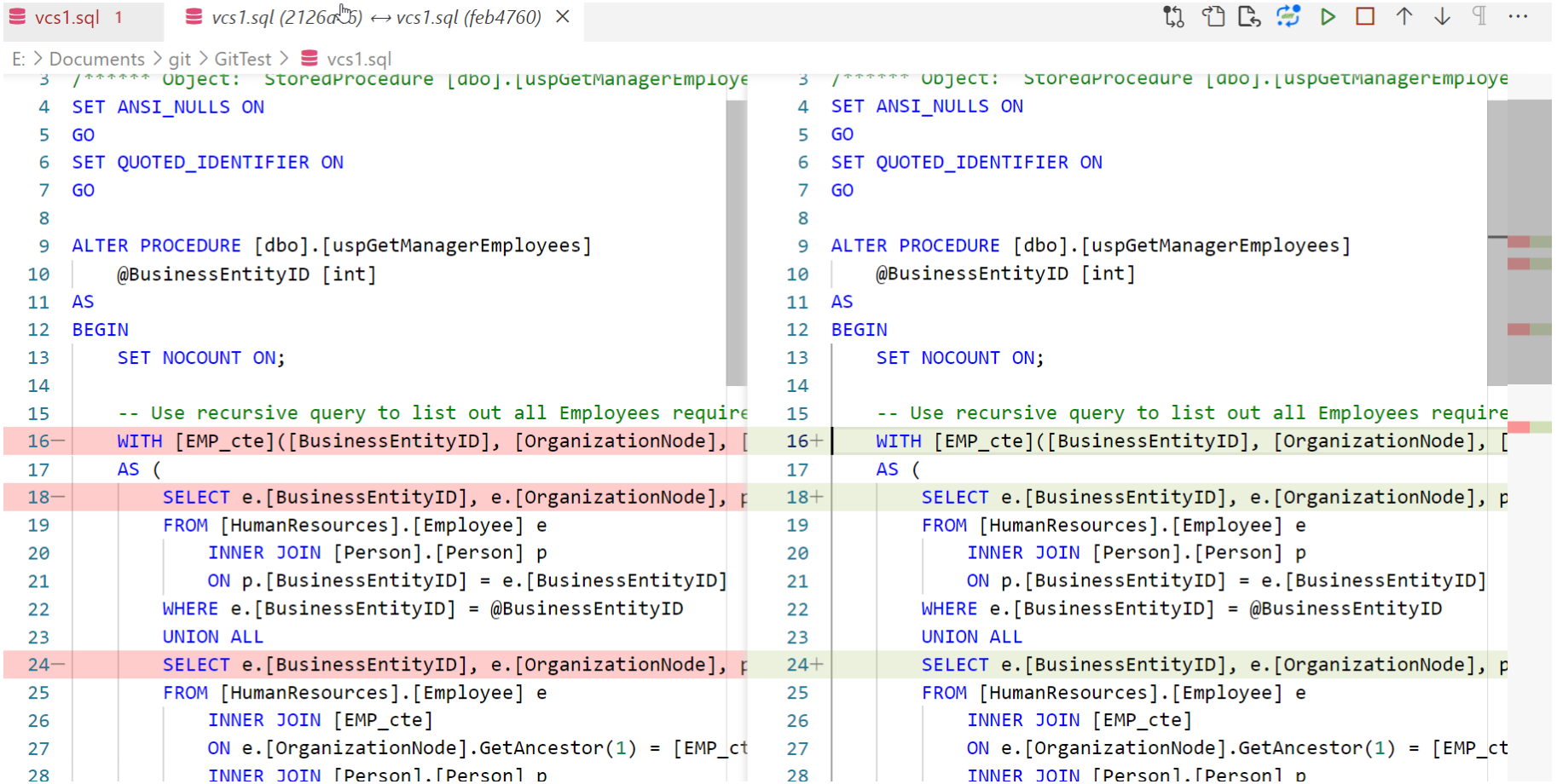
SQLQuery5.sql - ARI...ISTOTLE\Steve (63)) | SQLQuery4.sql - ARI...ISTOTLE\Steve (60))* | SQLQuery3.sql - ARI...ISTOTLE\Steve (60))*

```
USE [AdventureWorks2017]
GO
/****** Object:  StoredProcedure [dbo].[uspGetManagerEmployees]    Script Date: 8
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[uspGetManagerEmployees]
    @BusinessEntityID [int]
AS
BEGIN
    SET NOCOUNT ON;

    -- Use recursive query to list out all Employees required for a particular Ma
    WITH [EMP_cte]([BusinessEntityID], [OrganizationNode], [FirstName], [LastName
    AS (
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName], p.[Last
        FROM [HumanResources].[Employee] e
            INNER JOIN [Person].[Person] p
            ON p.[BusinessEntityID] = e.[BusinessEntityID]
        WHERE e.[BusinessEntityID] = @BusinessEntityID
        UNION ALL
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName], p.[Last
        FROM [HumanResources].[Employee] e
            INNER JOIN [EMP_cte]
            ON e.[OrganizationNode].GetAncestor(1) = [EMP_cte].[OrganizationNode]
            INNER JOIN [Person].[Person] p
            ON p.[BusinessEntityID] = e.[BusinessEntityID]
        )
```

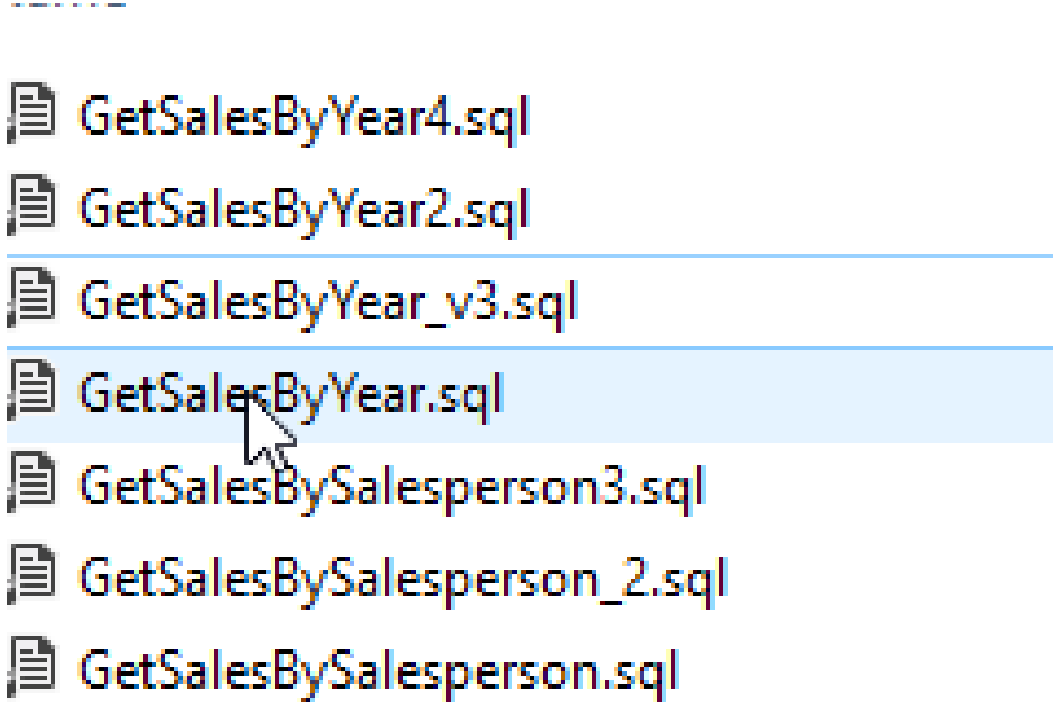
SQLQuery5.sql - ARI...ISTOTLE\Steve (63))* | SQLQuery4.sql - ARI...ISTOTLE\Steve (60))* | SQLQuery3.sql - ARI...ISTOTLE\Steve (60))*

```
USE [AdventureWorks2017]
GO
/****** Object:  StoredProcedure [dbo].[uspGetManagerEmployees]    Script Date: 8
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[uspGetManagerEmployees]
    @BusinessEntityID [int]
AS
BEGIN
    SET NOCOUNT ON;

    -- Use recursive query to list out all Employees required for a particular Ma
    WITH [EMP_cte]([BusinessEntityID], [OrganizationNode], [FullName] [RecursionL
    AS (
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName]+ p.[Last
        FROM [HumanResources].[Employee] e
            INNER JOIN [Person].[Person] p
            ON p.[BusinessEntityID] = e.[BusinessEntityID]
        WHERE e.[BusinessEntityID] = @BusinessEntityID
        UNION ALL
        SELECT e.[BusinessEntityID], e.[OrganizationNode], p.[FirstName] + p.[Las
        FROM [HumanResources].[Employee] e
            INNER JOIN [EMP_cte]
            ON e.[OrganizationNode].GetAncestor(1) = [EMP_cte].[OrganizationNode]
            INNER JOIN [Person].[Person] p
            ON p.[BusinessEntityID] = e.[BusinessEntityID]
        )
```

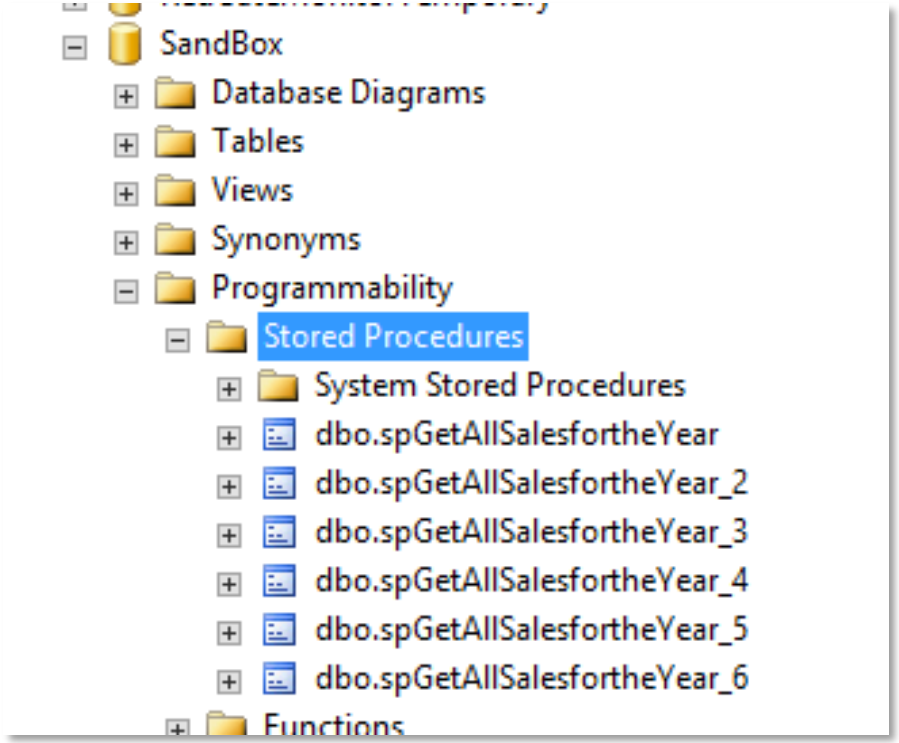
Is this easier?



How many of you do this?



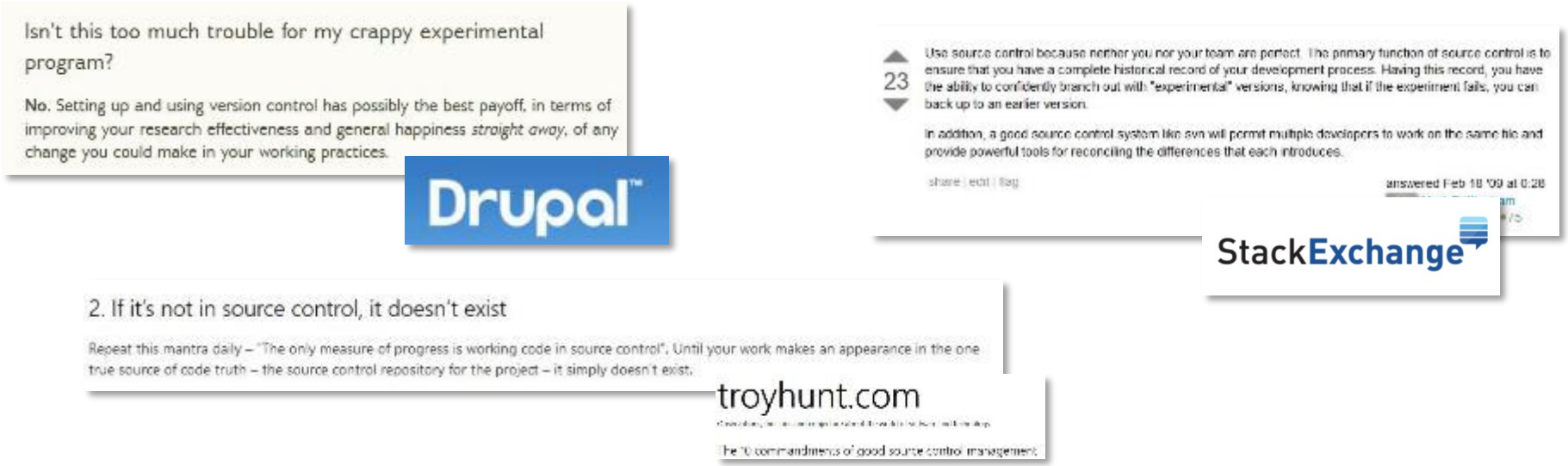
Or this?

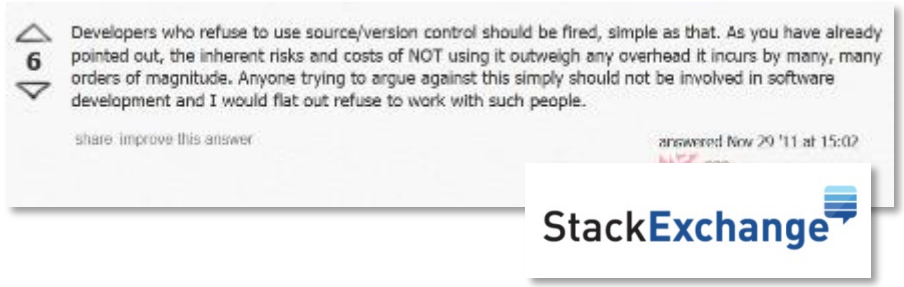

SandBox
Database Diagrams
Tables
Views
Synonyms
Programmability
Stored Procedures
System Stored Procedures
dbo.spGetAllSalesfortheYear
dbo.spGetAllSalesfortheYear_2
dbo.spGetAllSalesfortheYear_3
dbo.spGetAllSalesfortheYear_4
dbo.spGetAllSalesfortheYear_5
dbo.spGetAllSalesfortheYear_6
Functions

If you do this, you are using Version Control

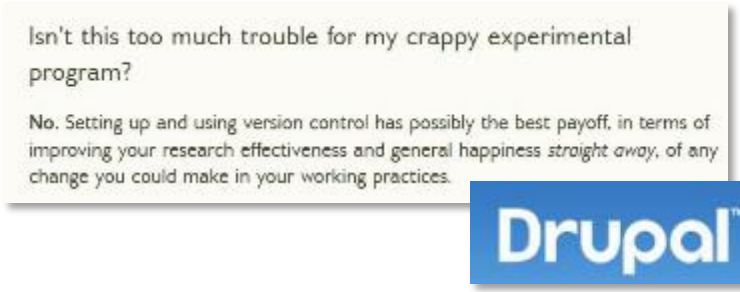
- It’s not good version control
- Tracking changes is hard for yourself
- It’s worse in a team environment
- Plus, version control is important for your career.

Developers who refuse to use source/version control should be fired, simple as that.

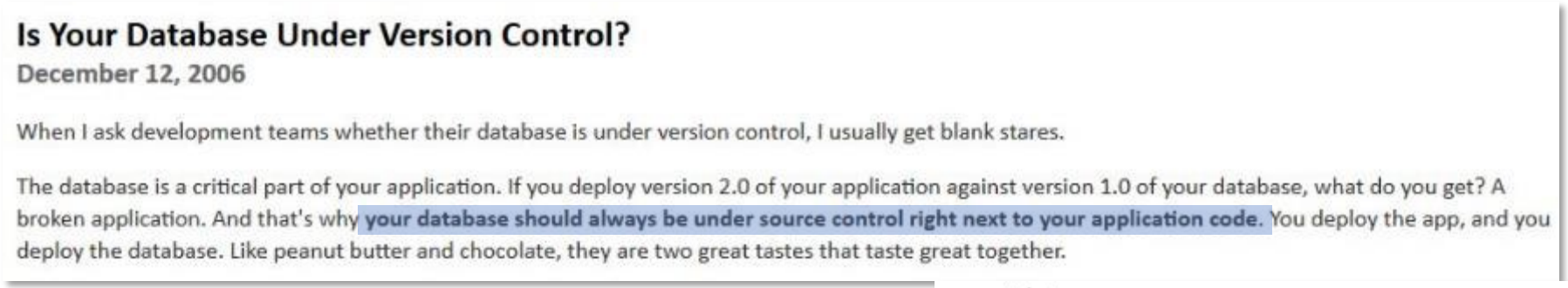




Use source control because neither you nor your developers are perfect.



“…your database should always be under source control right next to your application code.”



Is Your Database Under Version Control?

December 12, 2006

When I ask development teams whether their database is under version control, I usually get blank stares.

The database is a critical part of your application. If you deploy version 2.0 of your application against version 1.0 of your database, what do you get? A broken application. And that's why **your database should always be under source control right next to your application code**. You deploy the app, and you deploy the database. Like peanut butter and chocolate, they are two great tastes that taste great together.

DevOps Requires a Foundation

- For many companies, the foundation of building better software is version control
- Knowing version control is important for your career prospects
- While some employers will train you, think about who you hire?
  - I’m great at writing fast queries, but I don’t know anything about git
  - I’m great at writing fast queries and sharing code with others through git

It’s not just code

- Documentation
  - https://github.com/MicrosoftDocs/sql-docs
  - https://github.com/dbafromthecold/SqlServerAndContainersGuide/wiki/Building-a-custom-image
- Books
  - https://github.com/EbookFoundation/free-programming-books
  - https://github.com/aosabook/aosabook
- Blogging
  - https://developmentseed.org/blog
- Sites
  - https://sqlsaturday.com/
  - https://datasaturdays.com/

The Basics of Version Control

A primer for DBAs

All code is text

- For almost all software systems, we use code to tell the computer what to do
- We author, experiment, and finally save code for our team to see
- A “version” is simply a text file that we’ve decided is working (ish)
- We use a common system to save these versions of text files
- This is called a:
  - Version control system (VCS)
  - Source control system (SC)
  - Revision control system (RCS)

Two Major Types of VCS

- Centralized (TFS, Subversion, Vault)
  - Uses a “server” to store all code (client/server)
  - Users all connect to the server to store and retrieve code
  - Always online
  - Allow file locking
  - Contains a canonical copy
  - Needs backup
- Distributed (Git, Mercurial)
  - Each node is essentially its own server (peer to peer)
  - Often one node is chosen as a server (i.e. GitHub)
  - Each copy of code is a working copy, no canonical copies
  - Every copy is a backup (albeit, potentially at different points of time)

Common Concepts

- Repository (database) – where files are stored
- Branch (copy) – A set of files that were all copied at the same time. This is also a verb to make a copy
- Checkout (file|open) – Retrieve a file(s)
- Commit (file|save) – Make a save in the repository
- Merge – move changes from one branch into another
- Tag – A label at a point in time for a set of files
- Trunk/main/master – Often the main copy of development code that isn’t a branch (ish)

VCS Brands

- CVS/PVCS
- Vault
- Perforce
- Rational ClearCare
- Visual SourceSafe
- Team Foundation Version Control
- Subversion
- Mercurial
- Git

An Introduction to Git

The VCS winner

Git has won

Git Features

- Is distributed
- Scales (Windows is 3.5mm files, 300GB)
- Easy to get and share code
- Lots of online support for code:
  - GitHub
  - Azure DevOps
  - BitBucket
  - GitLab
- Allows offline work
- Many clients

Getting Git

- Git is free/open source
- https://git-scm.com/
- Download for your platform
- Quick, simple install
- The default is a command line

Git Basics

- Init – create a repository
- Tracked – A file(s) that git versions
- Ignore – A file(s) that git doesn’t track or version
- Add – add a file to be tracked
- Commit – save a version
- Clone – get a copy from another peer/server
- Config – change your setup
- Status – the state of your repository
- Log – a history of changes

Demo

Getting Started with Git

Making Git Fit You

- You need to personalize your install for others
- There are global and repo specific settings
- Lots of setup tutorials
  - https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup
  - https://docs.github.com/en/get-started/getting-started-with-git/setting-your-username-in-git
  - https://careerkarma.com/blog/git-config/

Demo

Configuring Git

Git Clients

- Lots of options
  - CLI
  - Visual Studio
  - VS Code / ADS
  - GitKrakken
  - SourceTree
  - Github Desktop
  - More
- I'll show a few

Working with real code

- Let’s download a repo and write code
- I’ll write code in
  - SSMS
  - ADS (Azure Data Studio)

Demo

Let’s write code

https://github.com/way0utwest/VCS_Primer

Push and Pull

Sharing Code

Remotes

- A remote is a copy of the git repository that is not on your machine
- Each remote is a peer
- In practice, most often we have one main remote acting as a server
- Push is a way of sending your changes to the remote
- Pull is a way of getting changes from the remote
- Usually you want to pull first, just in case there are collisions

Commands

- Push – Send code to a remote repo (usually a central one)
- Pull – Get code from a remote repo
- Fetch – pulls branches and tags from a repo

Demo

Sharing code

Branches

Copies of Code

Git Branches

- A branch is a copy of code, started at a point in time.
- Each copy is independent, and git manages tracking this
- Only one branch in a repository is active
- Changing branches changes the file system
- Create a branch with:
  - Git branch <branch name>
  - Git checkout –b <branch name>
- Delete with the –d option

Demo

Creating a branch

Merges

Getting branches in sync

Merge

- Merging is where we combine changes from two copies
- This could be across branches or remotes
- If the same file is changed in two places, then there is often a merge conflict
- Merging application code is often simpler than merging database code
- There are tools to help with resolving conflicts (Beyond Compare, kdiff3)

Demo

Merging Branches

Pull Requests

A second set of eyes

Pull Request

- A pull request is a request for someone to review your code
  - And potentially merge it into their branch
- The idea is that you are requesting someone else to “pull” your changes
- Usually this is done in a GUI fashion with a server repository

Demo

Asking for Review

Summary

- Version Control is the basis by which we keep track of all our code
- This is useful for teams to be able to share, review, and collaborate on code.
- Git has won and almost everyone is moving to git for new work
- There isn’t great tooling for DBAs
- Git isn’t too hard, but does require some practice
- There are lots of resources available

The End


www.voiceofthedba.com


sjones@sqlservercentral.com


@way0utwest

/in/way0utwest

Git Resources

- Atlassian Tutorials
- https://learngitbranching.js.org/
- Git series at SQLServerCentral
- Git Anatomy
- Git source site
- How to Set up Git Using Git Config
- Where to find system, global and local Git config files on Windows and Ubuntu Linux

References

- http://programmers.stackexchange.com/questions/122150/how-can-i-convince-cowboy-programmers-to-use-source-control
- http://www.makeuseof.com/tag/git-version-control-youre-developer/
- http://www.mactech.com/articles/mactech/Vol.14/14.06/VersionControlAndTheDeveloper/index.html
- http://drupal.org/node/299067