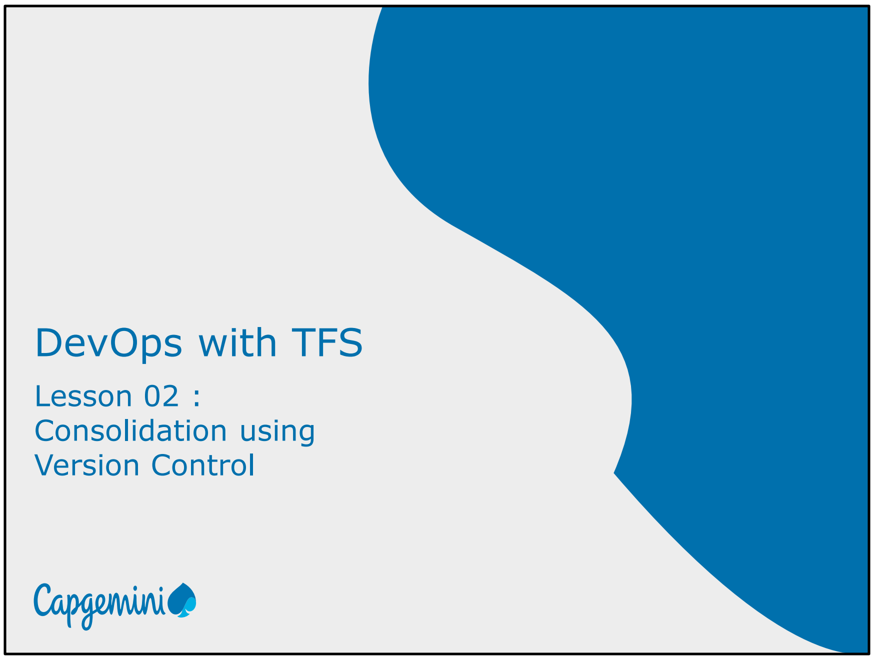


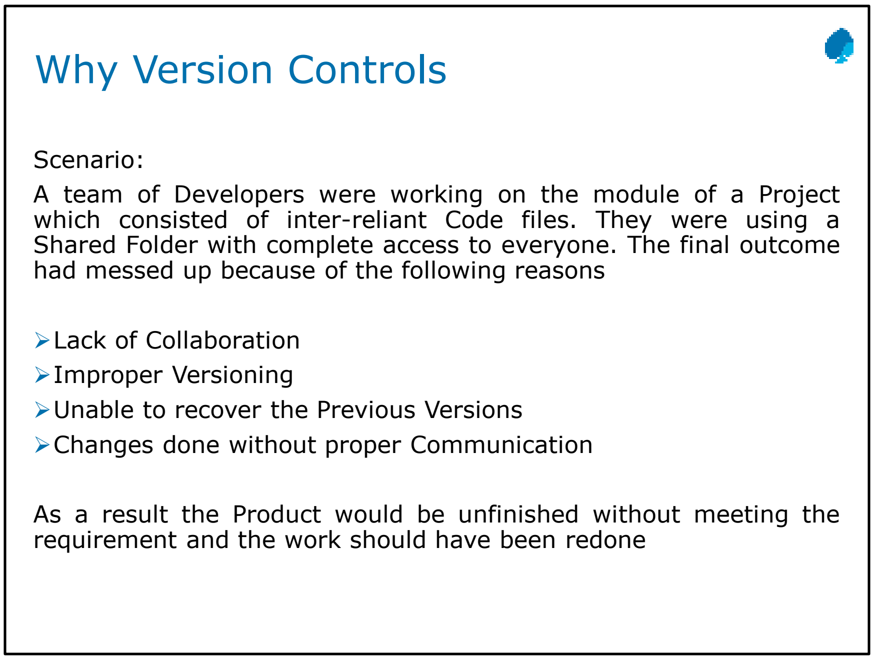
DevOps with TFS

Lesson 02 : Consolidation using Version Control

Capgemini

Table of Contents

- Version Control
  - Why Version Control?
  - What is Version Control?
  - Version Controls Tools
  - Advantages of Version Control
- Version Control in TFS
- Team Projects & Team Project Collections
- TFVC Overview
- Git + TFS Overview
- Git or TFVC



Why Version Controls

Scenario:

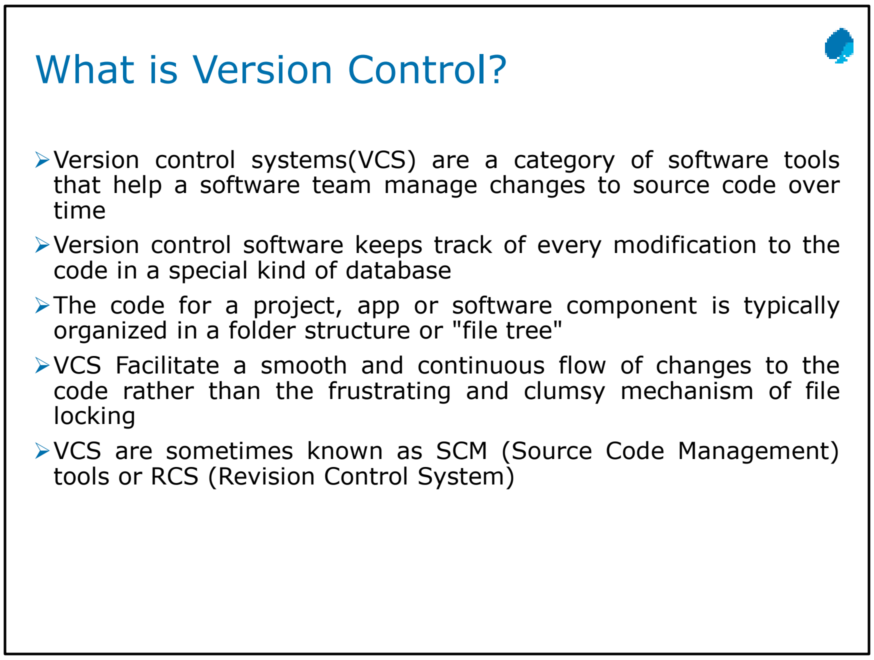
A team of Developers were working on the module of a Project which consisted of inter-reliant Code files. They were using a Shared Folder with complete access to everyone. The final outcome had messed up because of the following reasons

- Lack of Collaboration
- Improper Versioning
- Unable to recover the Previous Versions
- Changes done without proper Communication

As a result the Product would be unfinished without meeting the requirement and the work should have been redone

Have you ever,
Made a change to code and later realised it was a mistake and wanted to revert back?
Lost code or had a backup that was too old?
Had to maintain multiple versions of a product?
Wanted to see the difference between two (or more) versions of your code?
Wanted to prove that a particular change broke or fixed a piece of code?
Wanted to review the history of some code?
Wanted to submit a change to someone else's code?
Wanted to share your code, or let other people work on your code?
Wanted to see how much work is being done, and where, when and by whom?
Wanted to experiment with a new feature without interfering with working code?

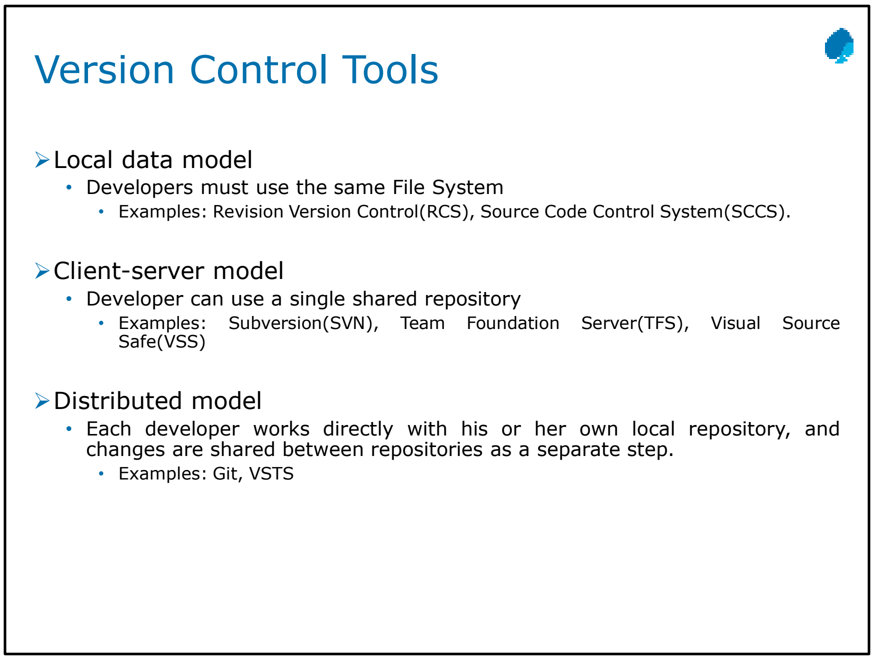
The one stop solution for all these would be a version control system.



What is Version Control?

- Version control systems(VCS) are a category of software tools that help a software team manage changes to source code over time
- Version control software keeps track of every modification to the code in a special kind of database
- The code for a project, app or software component is typically organized in a folder structure or "file tree"
- VCS Facilitate a smooth and continuous flow of changes to the code rather than the frustrating and clumsy mechanism of file locking
- VCS are sometimes known as SCM (Source Code Management) tools or RCS (Revision Control System)

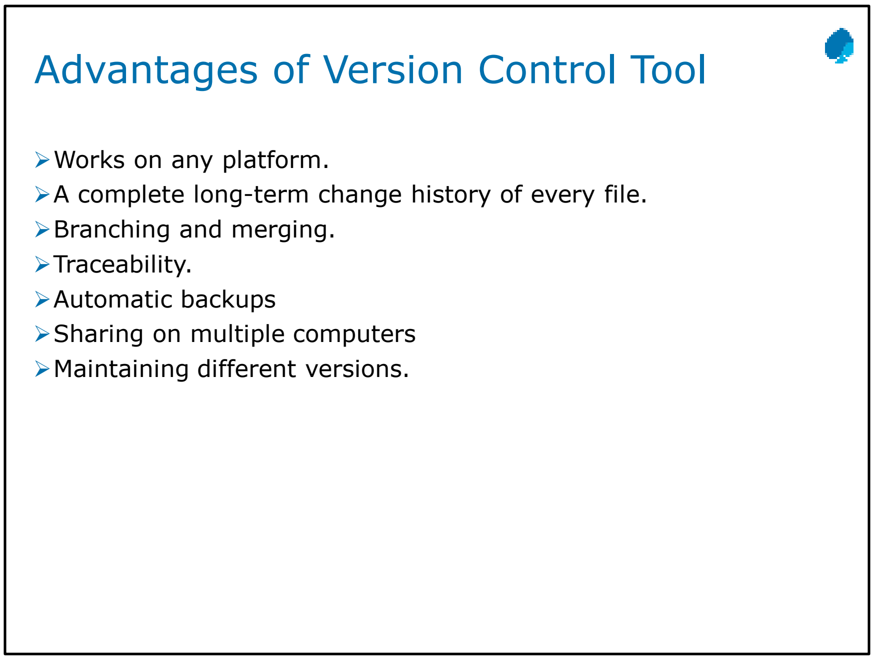
Version control protects source code from both catastrophe and the casual degradation of human error and unintended consequences.
If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.
For almost all software projects, the source code is like the crown jewels - a precious asset whose value must be protected. For most software teams, the source code is a repository of the invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort.



Version Control Tools

- Local data model
  - Developers must use the same File System
    - Examples: Revision Version Control(RCS), Source Code Control System(SCCS).
- Client-server model
  - Developer can use a single shared repository
    - Examples: Subversion(SVN), Team Foundation Server(TFS), Visual Source Safe(VSS)
- Distributed model
  - Each developer works directly with his or her own local repository, and changes are shared between repositories as a separate step.
    - Examples: Git, VSTS

Version control tools are a great way to enable collaboration, maintain versions, and track changes across the team.
The greatest benefit of using version control tools is that you have the capacity to deal with an unlimited number of people, working on the same code base, without having to make sure that files are delivered back and forth.

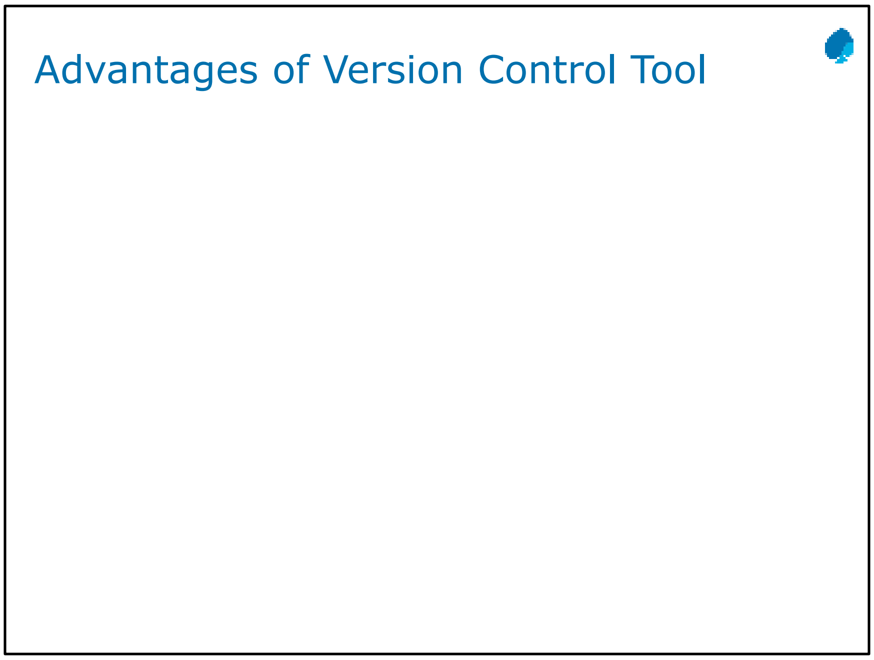


Advantages of Version Control Tool

- ➢Works on any platform.
- ➢A complete long-term change history of every file.
- ➢Branching and merging.
- ➢Traceability.
- ➢Automatic backups
- ➢Sharing on multiple computers
- ➢Maintaining different versions.

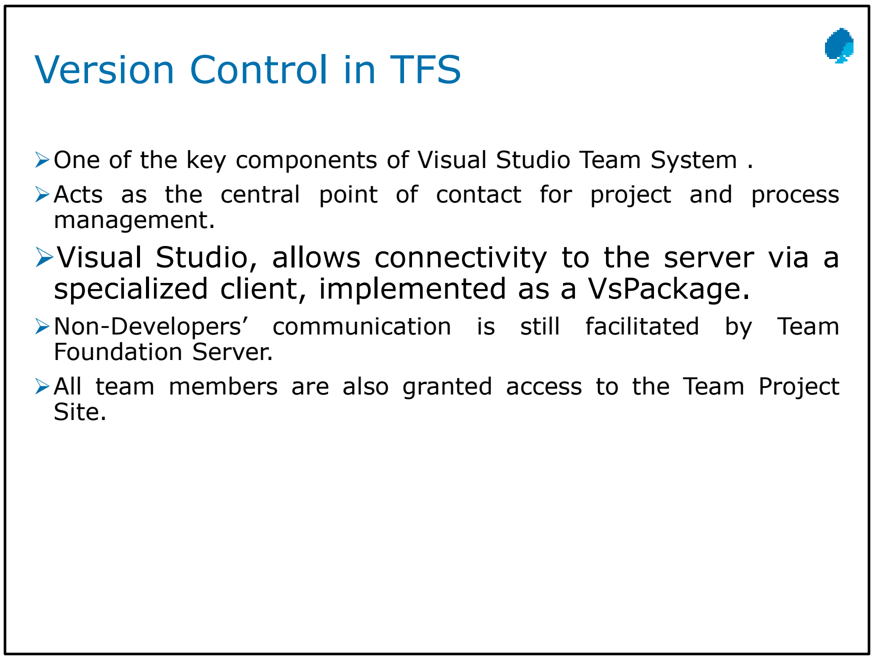
A complete long-term change history of every file:- This means every change made by many individuals over the years. Changes include the creation and deletion of files as well as edits to their contents. Different VCS tools differ on how well they handle renaming and moving of files. This history should also include the author, date and written notes on the purpose of each change.
Branching and merging:-Having team members work concurrently is a no-brainer, but even individuals working on their own can benefit from the ability to work on independent streams of changes. Creating a "branch" in VCS tools keeps multiple streams of work independent from each other while also providing the facility to merge that work back together, enabling developers to verify that the changes on each branch do not conflict.
Traceability:Being able to trace each change made to the software and connect it to project management and bug tracking software such as JIRA, and being able to annotate each change with a message describing the purpose and intent of the change can help not only with root cause analysis and other forensics
This can be especially important for working effectively with legacy code and is crucial in enabling developers to estimate future work with any accuracy.

While it is possible to develop software without using any version control, doing so subjects the project to a huge risk that no professional team would be advised to accept. So the question is not whether to use version control but which version control system to use.



Advantages of Version Control Tool

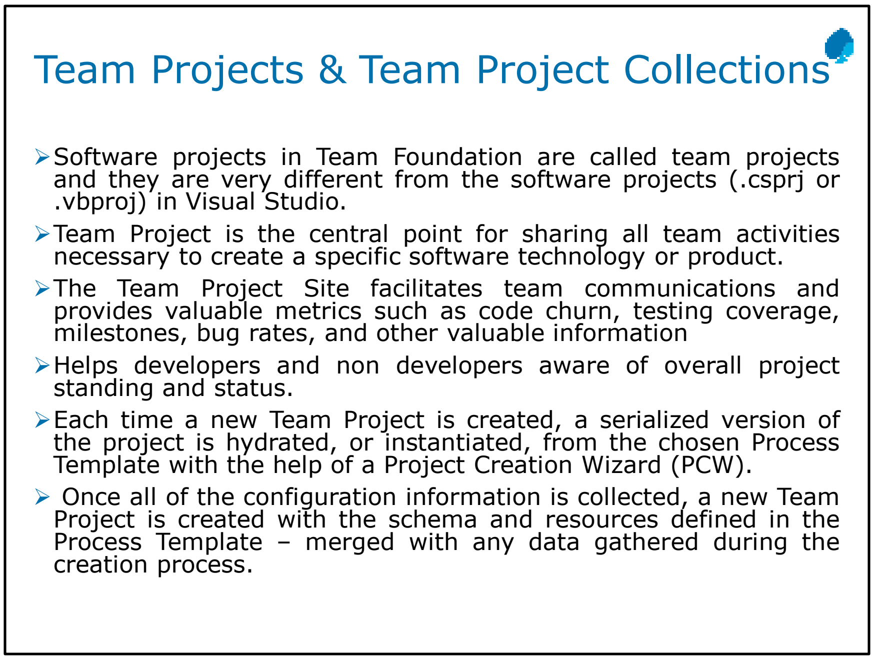
With a VCS you just have one copy of each file, so you have a much cleaner and easier to navigate working directory. The VCS should make it easy for you to find out where changes were made, so that you can easily find things when you want to restore something that you previously edited out. Since the VCS stores everything, you know that every version of the file is available in one place.
Version control systems also let you work on several machines.
Helps for many people to collaborate on a single document/project, keeps track of all the changes to the document as it evolves, you can revert and/or merge edits.



Version Control in TFS

- One of the key components of Visual Studio Team System .
- Acts as the central point of contact for project and process management.
- Visual Studio, allows connectivity to the server via a specialized client, implemented as a VsPackage.
- Non-Developers' communication is still facilitated by Team Foundation Server.
- All team members are also granted access to the Team Project Site.

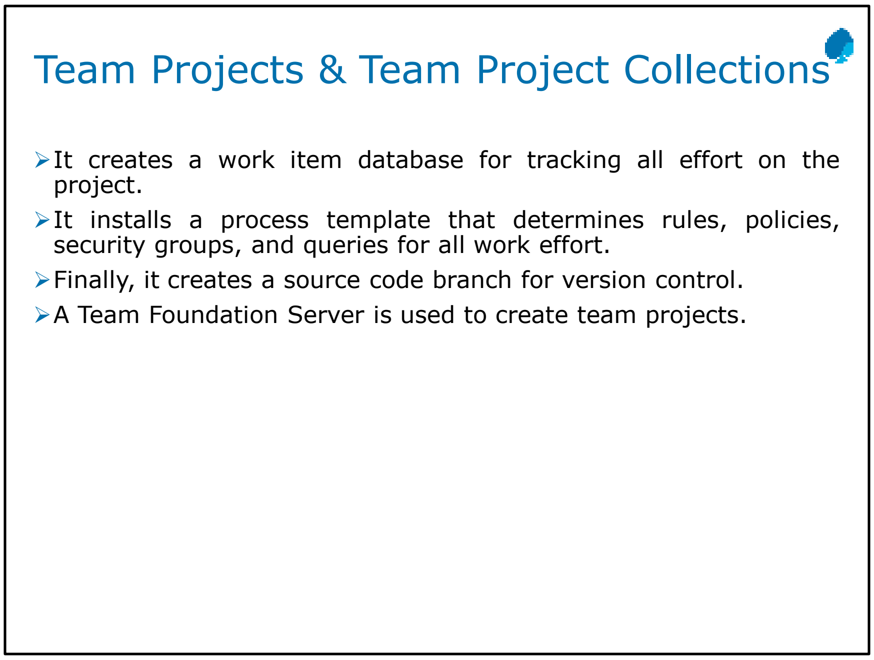
Team Foundation is built upon a SQL Server 2005 backend with a middle tier comprised of ASP.NET and Windows SharePoint Services that both provide complex, yet flexible, interactivity and serve a variety of clients including Visual Studio, Microsoft Office, browsers, and a number of command line tools. Visual Studio, the most important client of Team Foundation Server actually allows connectivity to the server via a specialized client, implemented as a VsPackage, that allows for deeply integrated communication with the server.
Managers, analysts, and other share holders in various roles will also need access to team resources. TFS allows for complete interaction with work items through applications like Microsoft Project, Excel, and even third party applications. This allows project managers and other non-developers to interact with developers to help improve communication.



Team Projects & Team Project Collections

- Software projects in Team Foundation are called team projects and they are very different from the software projects (.csprj or .vbproj) in Visual Studio.
- Team Project is the central point for sharing all team activities necessary to create a specific software technology or product.
- The Team Project Site facilitates team communications and provides valuable metrics such as code churn, testing coverage, milestones, bug rates, and other valuable information
- Helps developers and non developers aware of overall project standing and status.
- Each time a new Team Project is created, a serialized version of the project is hydrated, or instantiated, from the chosen Process Template with the help of a Project Creation Wizard (PCW).
- Once all of the configuration information is collected, a new Team Project is created with the schema and resources defined in the Process Template – merged with any data gathered during the creation process.

Team Explorer provides a tightly integrated mechanism for interacting with team resources stored on the Team Foundation Server for a given project. Within each Team Project, key resources, such as process documentation, work items, reports, build settings, and source control repositories are organized as defined by the Team Project's Process Template.
Information gathered during setup can then be easily communicated to customized Plug-ins, ensuring strong integration with core Team System components.



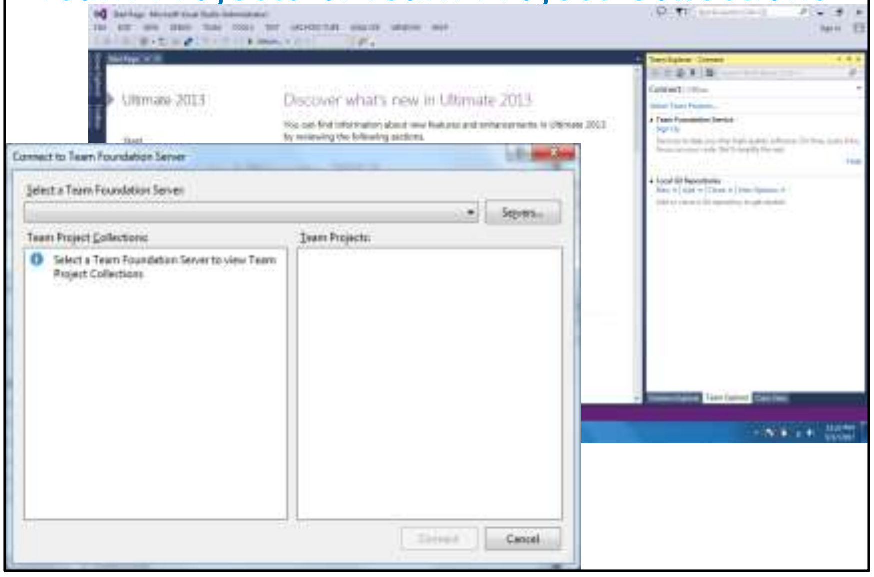
Team Projects & Team Project Collections

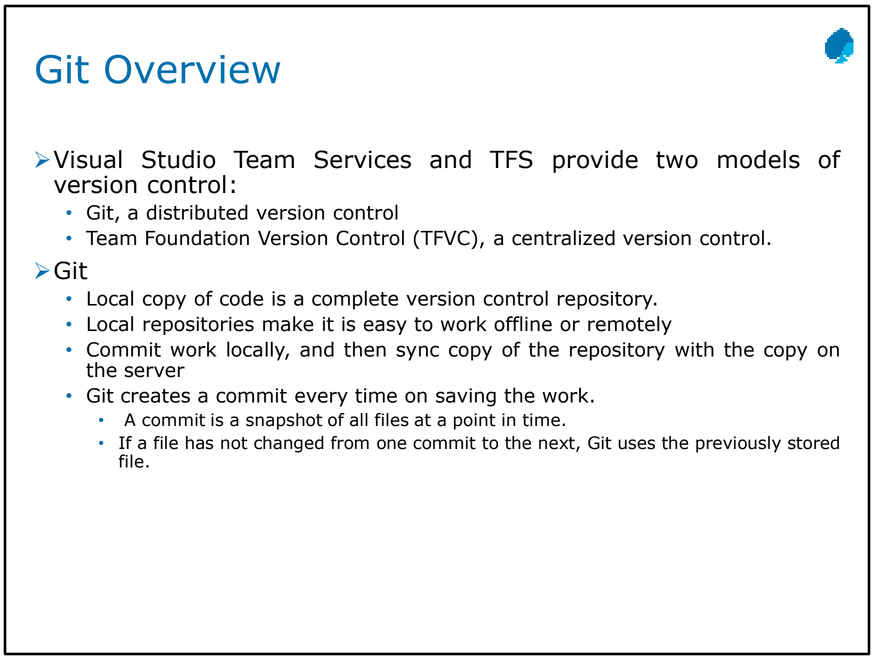
- It creates a work item database for tracking all effort on the project.
- It installs a process template that determines rules, policies, security groups, and queries for all work effort.
- Finally, it creates a source code branch for version control.
- A Team Foundation Server is used to create team projects.

All team projects are stored and managed on a Team Foundation Server. To start working on a team project, you must first connect to the appropriate Team Foundation Server.
To connect to a Team Foundation Server you must be a member of the **Team Foundation Valid Users** security group.
Team Explorer displays team projects from one Team Foundation Server.
To add a team project to Team Explorer
On the **File** menu, point to **Open**, and then click **Team Project**.
In the **Connect to Team Foundation Server** dialog box, use the drop-down list to select a valid Team Foundation Server.
Under **Team projects**, select the team projects from the list that you want to add to Team Explorer and then click **OK**.
The team project you selected appears under the Team Foundation Server node.

Adding and removing team projects in Team Explorer does not affect or delete team projects on the Team Foundation Server.

Team Projects & Team Project Collections


Ultimate 2013
Discover what's new in Ultimate 2013
Connect to Team Foundation Server
Select a Team Foundation Server:
Servers...
Team Project Collections:
Team Projects:
Select a Team Foundation Server to view Team Project Collections
Connect
Cancel



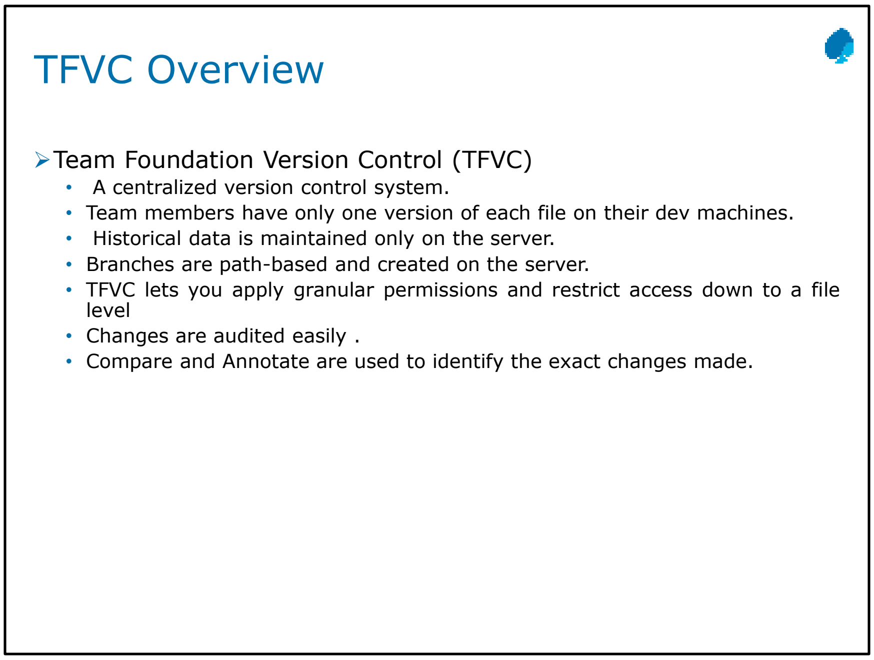
Git Overview

- Visual Studio Team Services and TFS provide two models of version control:
  - Git, a distributed version control
  - Team Foundation Version Control (TFVC), a centralized version control.
- Git
  - Local copy of code is a complete version control repository.
  - Local repositories make it is easy to work offline or remotely
  - Commit work locally, and then sync copy of the repository with the copy on the server
  - Git creates a commit every time on saving the work.
    - A commit is a snapshot of all files at a point in time.
    - If a file has not changed from one commit to the next, Git uses the previously stored file.

Git is a distributed version control system, meaning your local copy of code is a complete version control repository. These fully-functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server. This paradigm differs from centralized version control where clients must synchronize code with a server before creating new versions of code.

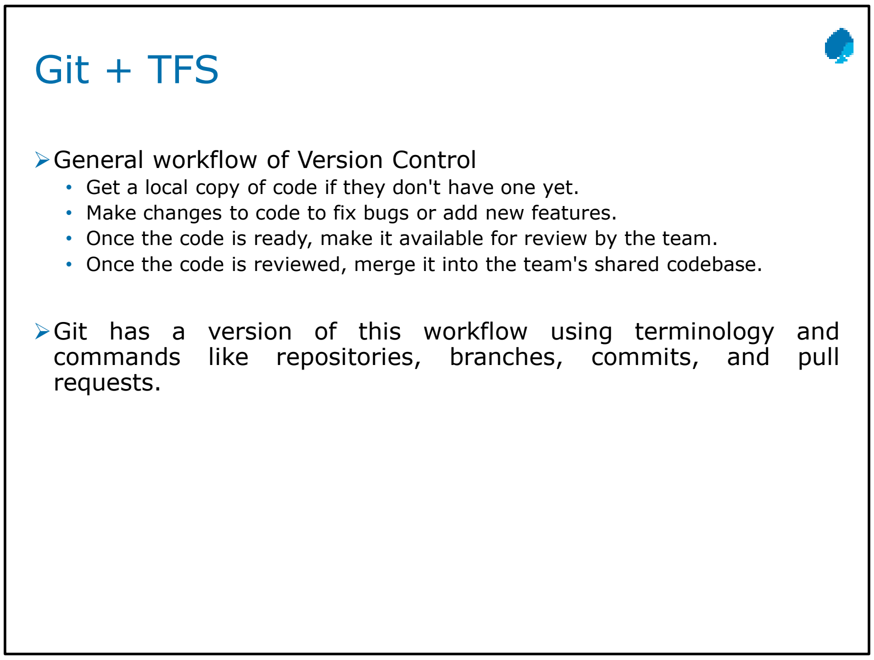
Git's flexibility and popularity make it a great choice for any team.

Every time you save your work, Git creates a commit. Commits create links to other commits, forming a graph of your development history . You can revert your code to a previous commit, inspect how files changed from one commit to the next, and review information such as where and when changes were made. Commits are identified in Git by a unique cryptographic hash of the contents of the commit. Because everything is hashed, it is impossible to make changes, lose information, or corrupt files without Git detecting it.



TFVC Overview

- Team Foundation Version Control (TFVC)
  - A centralized version control system.
  - Team members have only one version of each file on their dev machines.
  - Historical data is maintained only on the server.
  - Branches are path-based and created on the server.
  - TFVC lets you apply granular permissions and restrict access down to a file level
  - Changes are audited easily .
  - Compare and Annotate are used to identify the exact changes made.


Git + TFS

- General workflow of Version Control
  - Get a local copy of code if they don't have one yet.
  - Make changes to code to fix bugs or add new features.
  - Once the code is ready, make it available for review by the team.
  - Once the code is reviewed, merge it into the team's shared codebase.

- Git has a version of this workflow using terminology and commands like repositories, branches, commits, and pull requests.

Create a branch for the changes you plan to make and give it a name, such as fix-bug-123
Commit changes to your branch. People often have multiple commits for a bug fix or feature.
Push your branch to the remote repository.
Create a pull request so other people can review your changes. To incorporate feedback, you might need to make more commits and push more changes.
Complete your pull request and resolve any merge conflicts from changes other people made after you created your branch.

Git or TFVC

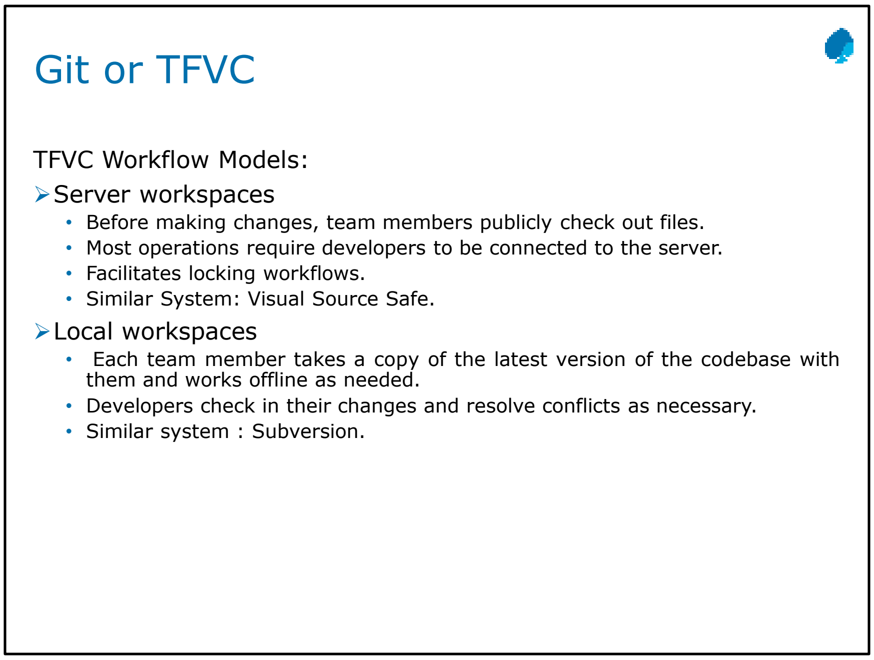
Git (distributed)

- Git is a distributed version control system.
- Each developer has a copy of the source repository on their dev machine.
- Developers can commit each set of changes on their dev machine and perform version control operations such as history and compare without a network connection.
- Branches are lightweight.
- When you need to switch contexts, you can create a private local branch.
- You can quickly switch from one branch to another to pivot among different variations of your codebase. Later, you can merge, publish, or dispose of the branch.

Git or TFVC

TFVC (centralized)

- Team Foundation Version Control (TFVC) is a centralized version control system.
- Typically, team members have only one version of each file on their dev machines. Historical data is maintained only on the server.
- Branches are path-based and created on the server.

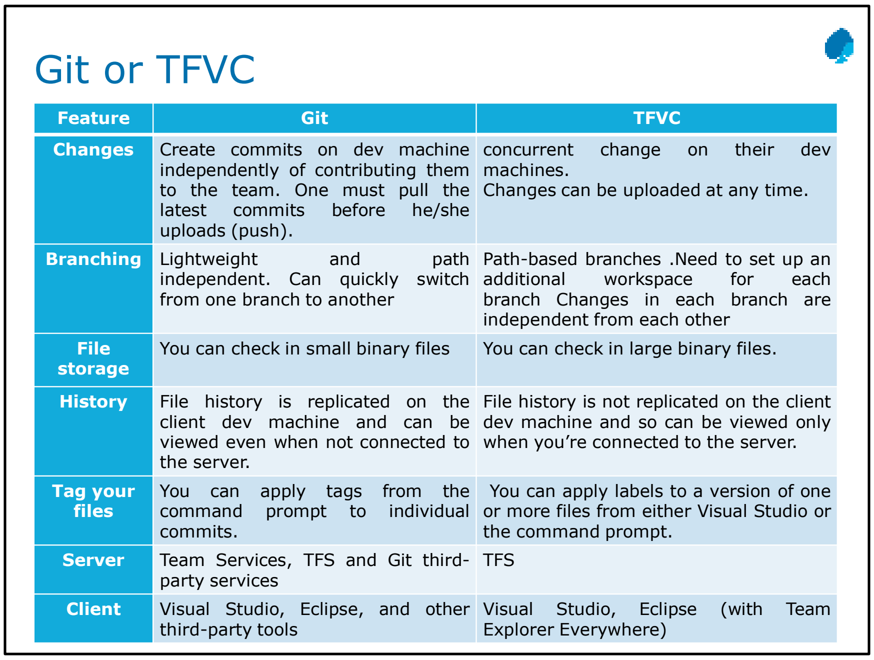


Git or TFVC

TFVC Workflow Models:

- ➢ Server workspaces
  - Before making changes, team members publicly check out files.
  - Most operations require developers to be connected to the server.
  - Facilitates locking workflows.
  - Similar System: Visual Source Safe.
- ➢ Local workspaces
  - Each team member takes a copy of the latest version of the codebase with them and works offline as needed.
  - Developers check in their changes and resolve conflicts as necessary.
  - Similar system : Subversion.

With server workspaces, you can scale up to very large codebases with millions of files per branch and large binary files.



Git or TFVC

Feature	Git	TFVC
Changes	Create commits on dev machine independently of contributing them to the team. One must pull the latest commits before he/she uploads (push).	concurrent change on their dev machines. Changes can be uploaded at any time.
Branching	Lightweight and path independent. Can quickly switch from one branch to another	Path-based branches .Need to set up an additional workspace for each branch Changes in each branch are independent from each other
File storage	You can check in small binary files	You can check in large binary files.
History	File history is replicated on the client dev machine and can be viewed even when not connected to the server.	File history is not replicated on the client dev machine and so can be viewed only when you're connected to the server.
Tag your files	You can apply tags from the command prompt to individual commits.	You can apply labels to a version of one or more files from either Visual Studio or the command prompt.
Server	Team Services, TFS and Git third-party services	TFS
Client	Visual Studio, Eclipse, and other third-party tools	Visual Studio, Eclipse (with Team Explorer Everywhere)

You should use Git for version control in your projects unless you have a specific need for centralized version control features in TFVC.
You can use TFVC repos with Git in the same Team Project so it's easy to add TFVC later if you need centralized version control.

Moving from TFVC to Git
If you have existing TFVC repos, you can migrate them to Git repos using the git-tfs tool. The tool allows you to migrate a TFVC repo to a Git repo in just a couple of commands.

Summary

- ➢ Benefits of Version Control
- ➢ Types of Version Control Tools
- ➢ Version Controls supporting VSTS
  - Git
  - TFVC
- ➢ Workflow of Version Controls
- ➢ Choice between Git & TFVC