Version control system, Azure Devops, Git

What is Version Control System?

A version control system allows users to keep track of the changes in software development projects, and enable them to collaborate on those projects. Using it, the developers can work together on code and separate their tasks through branches.

There can be several branches in a version control system, according to the number of collaborators. The branches maintain individuality as the code changes remain in a specified branchs.

Developers can combine the code changes when required. Further, they can view the history of changes, go back to the previous versions and use/manage code in the desired fashion. [1]

Benefits of Using a Version Control System

The main advantages of using a version control system include streamlining the development process, management of code for multiple projects and keeping a history of all changes within a code.

A version control software saves all the changes in a repository. Hence, if the developers make a mistake, they can undo it. At the same time, they can compare the new code with a previous version(s) to resolve their grievance. This can reduce human errors and unintended consequences to a great extent. A great fit for any web development company around the globe.

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.

What is Git?

Git is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source. Developers who have worked with Git are well represented in the pool of available software development talent and it works well on a wide range of operating systems and IDEs (Integrated Development Environments). [2]

Having a distributed architecture, Git is an example of a DVCS (hence Distributed Version Control System). Rather than have only one single place for the full version history of the software as is common in once-popular version control systems like CVS or Subversion (also known as SVN), in Git, every developer's working copy of the code is also a repository that can contain the full history of all changes.

In addition to being distributed, Git has been designed with performance, security and flexibility in mind. [4]

Azure Devops

Azure DevOps Server (formerly Team Foundation Server (TFS) and Visual Studio Team System (VSTS)) is a Microsoft product that provides version control (either with Team Foundation Version Control (TFVC) or Git), reporting, requirements management, project management (for both agile software development and waterfall teams), automated builds, testing and release management capabilities. [3]

At the heart of Azure DevOps is the "work item". A work item represents a thing – it can be work that needs to be accomplished, a risk to track, a test case, a bug or virtually anything else a user can imagine. Work items are defined through the XML documents and are highly extensible. Work items are combined into a Process Template that contains these and other pieces of information to provide a development framework. Azure DevOps includes Process Templates for the Microsoft Solutions Framework for Agile, Scrum and CMMI. Teams can choose to use a built-in template or one of the many templates available for use created by third parties. Process templates can be customized using the Process Template Editor, which is part of the Power Tools.

Work items can be linked to each other using different relationships to create a hierarchical tree of work items or a flat relationship between work items. Work items can also be linked to external artifacts such as web pages, documents on a file share or documents stored in another repository such as SharePoint. Work items can also be linked to source code, build results, test results and specific versions of items in source control.

The flexibility in the work item system allows Azure DevOps to play many roles from requirements management to bug tracking, risk and issue tracking, as well as recording the results of reviews. The extensible linking capabilities ensure that traceability from requirements to source code to test cases and results can be accomplished and reported on for auditing purposes as well as historical understanding of changes.

Azure DevOps supports two different types of source control - its original source control engine called Team Foundation Version Control (TFVC) and with the release of TFS 2013, it supports Git as a core source control repository. With the release of TFS 2013, Microsoft added native support for Git. This is not a Microsoft specific implementation but a standard implementation based on the libgit2 library. This is the same library that powers the popular GitHub and the code is freely available from GitHub. Because Microsoft took the approach of using a standard library, any Git client can now be used natively with Azure DevOps (in other words, developers can use their favorite tools and never install the standard Azure DevOps clients). This allows tools on any platform and any IDE that support Git to connect to Azure DevOps. For example, both Xcode and Android Studio support Git plug-ins. In addition, if developers do not want to use Microsoft's Team Explorer Everywhere plug-in for Eclipse, they can choose to use eGit to connect to Azure DevOps.

Using Git does not preclude the benefit of using Azure DevOps work item or build system. When checking code in with Git, referencing the work item ID in the check-in comment will associate the check-in with the given work item. Likewise, Team Build will also build Git projects.

One of the major reasons to use Azure DevOps as a Git repository is that it is backed by SQL Server and is afforded the same protection as Team Foundation

Version Control (TFVC). This gives developers some choices when choosing the type of project and work style that works best for them.

References

[1] What is version control? https://about.gitlab.com/topics/version-control/.

[2] Scott Chacon and Ben Straub. Getting started - what is git? https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F.

[3] Jamie Cool. Azure devops. https://azure.microsoft.com/en-us/blog/introducing-azure-devops/.

[4] Jacob Stopak. The evolution of version control system. https://initialcommit.com/blog/Technical-Guide-VCS-Internals.