Understanding TFS migrations from on-premises to Visual Studio Online – Part 1: Concepts

Martin Hinshelwood, Willy-Peter Schaub

Since Team Foundation Server (TFS) 2005, the ALM Rangers and ALM MVPs have had a mission to provide out of band solutions to missing features and guidance. In this article, we discuss guidance for understanding, planning, and implementing migrations from TFS to Visual Studio Online.

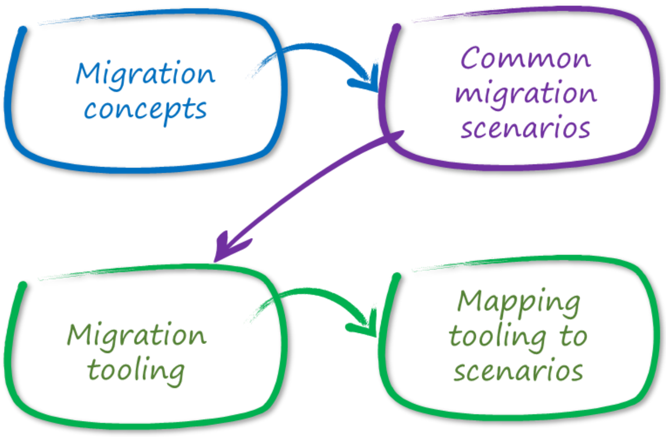
Adoption of cloud infrastructure

"Over the last two years, the industry has seen an incredible adoption of cloud infrastructure to power many of today's IT application workloads. For early adopters the appeal was greater elasticity *and* resiliency *at a lower upfront cost. Today, with the maturity of cloud providers, attractive price plans and better on-premises integration IT teams are readying to on-board the vast majority of their application workloads. A shift of budget amounts from CAPEX (capital expenditures) to OPEX (operational expenditures) as a more attractive financial expenditure strategy is further increasing this momentum."* (Rodriguez, 2014)

With the adoption of Microsoft Azure, many teams are considering a move to Microsoft's own cloud-based version of TFS, Visual Studio Online (VSO). Other organizations might have barriers to using a public service, preferring to remain on a private cloud.

This Part 1: **Concepts** whitepaper provides practical guidance on selecting a suitable migration strategy as well as tooling to migrate to Visual Studio Online without affecting your operational environment or data fidelity.

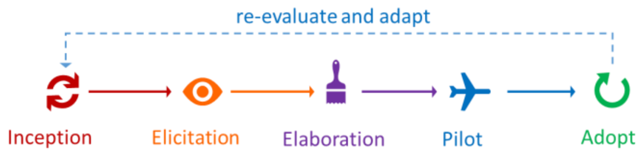
It covers the following core topics:



In Part 2: **Walkthrough**, we will walk you through the migration of a simulated on-premises environment, using Brian Keller's VM[1], to Visual Studio Online. We will use the recommended out-of-box tooling option as discussed herein, emphasising its simplicity and constraints.

Migration concepts

Migrations generally vary in terms of requirements, technology, scale, and complexities making the planning phase the most important phase of any migration. As with any migration, including on-premises to on-premises, our recommended planning process for a successful migration to Visual Studio Online consists of the five steps shown below.



Inception - Create a basic understanding of the migration and synchronization requirements, available solutions, and constraints.

[1] http://aka.ms/ALMVMs

Elicitation - Understand the business requirements and expectations; negotiate the scope, such as the need for history, and associated complexity and costs.

Elaboration - Create a detailed definition of viable migration scenarios, data preparation, and on-going maintenance.

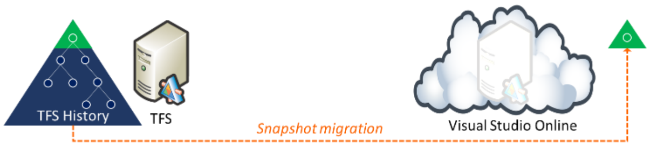
Pilot - Evaluate the proposed migration scenario, products, and configuration in a controlled pilot environment and assess impacts on the environment and the stakeholders.

Adopt - Never underestimate the impact and effort of migrations in a production environment, automation, maintenance, training, and on-going support. Be prepared to continuously re-evaluate, adapt, and improve your migration strategy.

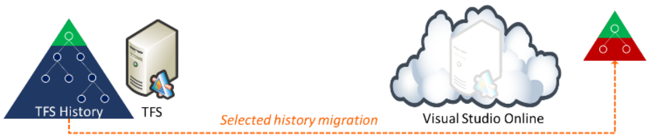
Understand and determine the need for history when planning and evaluating your migration strategy. In the context of this whitepaper, history refers to the changes made over time. History may seem important at first, but history is seldom required, and is incredibly expensive and complex to migrate. Options for migrating history from on-premises to Visual Studio Online range from a recommended **snapshot**, **selected history**, **archive**, to continuous **synchronization**.

For on-premises TFS to Visual Studio Online we strongly recommend backing up the history, and migrating only the "tip of the iceberg," also known as a **snapshot** migration.

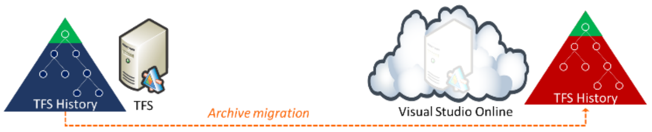
Based on a manual get and check-in latest, you can use out-of-the-box features of Visual Studio by maintaining the source system for history look-ups.



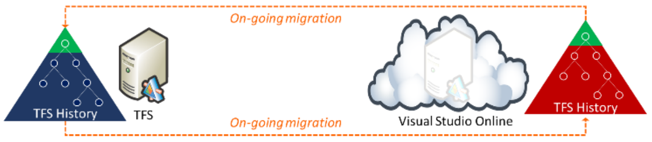
When migrating history is required, consider a **selected history** migration. Aggressively scope your history, using the snapshot mode, to migrate only what is relevant. Avoid migrating years of outdated data.



Reconsider any decision to perform an **archive** migration of the entire history, which will be complex and may potentially take a long time,



Synchronization can help a soft start or gradual migration, such as a phased migration from on-premises to Visual Studio Online. Synchronization is complex, introduces delta computation latency, support and maintenance complexity, and cost.



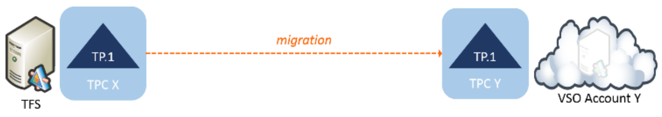
If you can move your environment all at once, you pay an upfront price and need not worry about ongoing synchronization and associated maintenance. If the risk and cost of running an ongoing synchronization solution is less than the risk and cost of a one-off migration you could consider synchronization.

Now that we have an understanding of history and the basic options for migrating history, let us explore the most common migration scenarios for migrating from on-premises to Visual Studio Online.

Common migration scenarios

Migrating Team Project to Team Project is often considered the simplest scenario. This is the default migration scenario requested by customers.

It involves migrating a TFS Team Project from an on-premises Team Project Collection, to a new Team Project, in the Default Team Project Collection of a Visual Studio Online account.



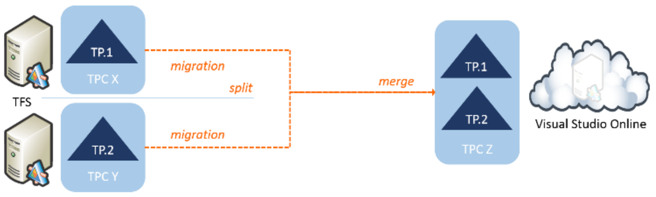
Consolidating Team Projects is another common migration scenario requested or required by customers. The push for a Single Team Project scenario became popular and is a priority for customers with TFS 2012, which introduced the concept of multiple teams within a Team Project. When planning a consolidation, it is important to understand that there are artifacts that TFS scopes to the Team Project, for example:

- **Queries** - While work item queries may contain results from multiple Team Projects, many tools (Excel & Project) that hook into these queries will only display results from the connected Team Project.
- **Area / Iteration** - Even if your teams worked in the same cadence you would be unable to have their

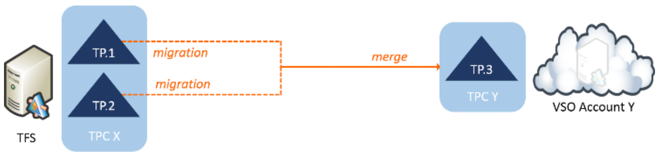
data in the same node hierarchy. While you can work around this in reporting, it can make this infinitely more complicated.

- **Builds** - Even though source code is not restricted to a Team Project, the boundary includes Build Definitions.

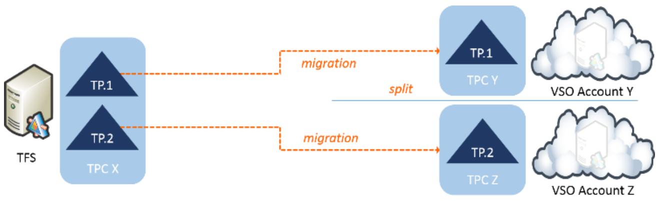
A consolidation can involve migrating Team Projects from different on-premises Team Project Collections to one Team Project Collection hosted in Visual Studio Online.



A more complex migration can involve migrating Team Projects from one or more on-premises Team Project Collections to one Team Project in Visual Studio Online.



Splitting up a Team Project Collection involves the process of splitting up on-premises Team Project Collections into two or more Team Projects hosted in separate Visual Studio Online **accounts**, each with one Team Project Collection.



Companies merge and companies split, creating a business need to either split a Collection into multiple Collections, or even split a Team Project from one to many.

Consolidate Platforms on Visual Studio Online includes specialised scenarios for companies and large enterprises consolidating their TFS and other on-premises TFS infrastructure on Visual Studio Online. There are many source control, work item tracking, build management systems, and process frameworks, making this the most complex scenario and beyond the scope of this whitepaper.

Next, we will take a brief look at some of the top migration tools and conclude by sharing “real-world” application of the tools for these scenarios.

Migration tooling

Migrations are complex and hard to plan and implement effectively. As such, it is important to work with experienced partners, who can deliver real-world experience and migration services.

NOTE

Our experience shows that it is important to realize that no solution will move your data with full fidelity. You may lose Build history and configuration, Test results, and your test suites among other things.

- We strongly recommend using out-of-the-box tooling. Using Visual Studio, Microsoft Excel (Import Excel data into TFS with History[2]), and the TFS Power Tools, you can perform simple migrations effectively. If the out-of-the-box tools are not an option, you can evaluate the following solutions as part of the elaboration and pilot phase.
- TFS Integration Platform[3] is an open source and customizable solution to help teams migrating from third party ALM tools to TFS. It is a long-term production service, designed to move data from selected competitor’s products to TFS on a one-time, partial fidelity basis. People have used it for many other purposes with varying degrees of success. Microsoft uses it for synchronization internally, but it definitely is not a problem-free process and is no longer supported
- TaskTop Sync[4] provides enterprise-scale work-item synchronization services among disparate ALM tools such as TFS, **Visual Studio Online**, Bugzilla, HP HC, Serene ABM, VersionOne, and ZenDesk.

See Team Foundation Server Migration and Integration Solutions[5] for more partner solutions and services.

Mapping tooling to scenarios

It is important to realize that migration scenarios are plagued with edge cases. We have yet to meet a migration scenario that went according to plan. Often, switching tactics halfway through, or hedging your bets, can get you there more gracefully.

Simple ‘tip’ migrations using out-of-the-box tools are the recommended way to move your data into Visual

[2] http://nakedalm.com/import-excel-data-into-tfs-with-history/

[3] http://tfsintegration.codeplex.com/

[4] http://tasktop.com/sync

[5] http://msdn.microsoft.com/en-us/vstudio/bb840033

Studio Online. This avoids the complexity of understanding and migrating history, as well as exceptions caused by missing or corrupt history. You can move your source code by doing a checkout in Visual Studio, then copying your files to the new location and checking-in. This also works for moving to Git in Visual Studio Online as well as from almost any other source control system to Visual Studio Online.

Work Items can be migrated by using the Excel plugin for Visual Studio Online to simply paste the data in and publish it. However, this does result in all of your Work Items being in the initial state for each type. This model works well when you have only a small number of work items.

Simple 'history' migration of Team Projects to Visual Studio Online is achievable using the OpsHub Visual Studio Online Migration Utility. This tool allows you migrate one or more Team Projects to **identically** named Visual Studio Online Team Projects that have the same Process Template as you do locally. You will not be able to:

- **Rename your Team Projects** – The names of your Team Projects in Visual Studio Online must exactly match the local name.
- **Change Process Template** – Process Templates must be identical. If you customize your Process Templates for an on-premises TFS and VSO, you will be updating these as you make changes on the service, and you will create conflicts.
- **Source branch Scope** – You cannot migrate code that has been branched outside of the Team Projects in Scope. You must migrate all Team Projects that have related code.
- **Move to Git** – The Visual Studio Online version control format must be the same as the local one.

Consolidating Team Projects with TFS Integration Platform tools have been the solution of choice since the release of TFS 2012. It supports a fully customizable migration model. You can change your Team Project name, move source code around, limit scope on the fly, and morph work items from one type to another.

The TFS Integration Platform tools also support continuous migrations. This allows your team to continue using the old system while you migrate and test the code. You can then re-run the migration and continue from where you left off. This can be invaluable with large teams or long-running migrations.

However, the current TFS Integration Tools do not support:

- **Visual Studio 2013** – You need to install the TFS 2012 Object Model and that will require installing Visual Studio 2012 on the machine hosting the TFS Integration Platform tools,
- **Allow partial history** – Although you can achieve this in a limited fashion for version control, there is no support for work items.
- **Move to Git** – The 2012 API does not have support for Git.

Although the TFS product team supports the TFS Integration Platform tools, they are in maintenance mode and there is no work underway for new features.

All work items that you save into Visual Studio Online must comply with the rules. The TFS Integration Platform tools provide the ability to map data, fields, and types to the new system. This time consuming activity is fraught with error. To alleviate this, you can put the TFS Integration tools into 'Bypass Mode' that will switch it from using the APIs to the Web Services. This mode allows you to write whatever data you like into TFS. While it is dangerous, it provides a way to simplify the migration and reduce errors.

Since we no longer fully support the TFS Integration Platform tools, the only viable solution for consolidating Team Projects beyond 2013 is to create your own migration tools.

Conclusion

What should be evident after reading this paper is that every migration scenario is customer and environment specific and plagued with edge cases.

There is **no silver bullet** or one-size-fits-all solution!

It is imperative that you engage an integration subject matter expert and invest in the planning and piloting phases to determine the most effective strategy for **your** environment.

Thanks for taking the time to read this and keep a look out for more articles from the ALM Rangers[6].

[6] http://aka.ms/vsarunderstand

Reference Information

- Migration and Integration Solutions[7]
- TFS Integration Tools Blogs and Reference Sites[8]
- TFS Planning, Disaster Avoidance, and Recovery Guide[9]
- Import Excel data into TFS with History[10]
- Migrating from an On-premises Team Foundation Server to Team Foundation Service Preview Using the TFS Integration Tools[11]

Martin Hinshelwood is an independent consultant with over 14 years of software development experience. He currently specializes in ALM from Scrum and EBMgt to TFS and Visual Studio. He is a Microsoft ALM MVP and ALM Ranger. He has extensive migration experience and a number of custom tools to help you with migrations.

You can reach Martin via his blog at nakedalm.com/blog. You can also follow him on Twitter at twitter.com/MrHinsh.

Willy-Peter Schaub is a senior program manager with the Visual Studio ALM Rangers at the Microsoft Canada Development Center. Since the mid-'80s, he has been striving for simplicity and maintainability in software engineering. Read his blog at blogs.msdn.com/b/willy-peter_schaub and follow him on Twitter at twitter.com/wpschaub.

THANKS to the following technical experts for reviewing this article: Aaron Hallberg, Bandar Alsharfi, Bijan Javidi, Bill Essary, Bill Heys, Ed Blankenship, Mario Rodriguez, Peter Antal, and Wendell Phillips.

[7] http://msdn.microsoft.com/en-us/vstudio/bb840033

[8] http://aka.ms/vsartoctip

[9] http://aka.ms/treasure5

[10] http://nakedalm.com/import-excel-data-into-tfs-with-history/

[11] http://msdn.microsoft.com/en-us/magazine/jj130558.aspx