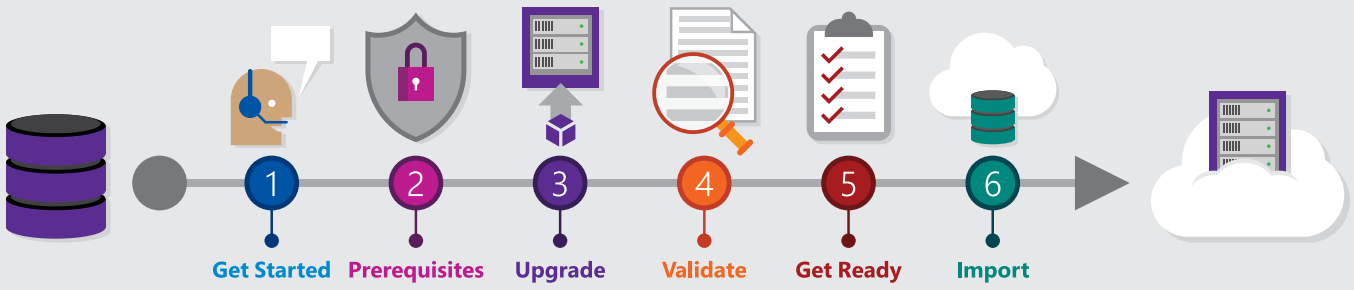
Microsoft

Data Migration Tool for Azure DevOps

Migration Guide



Last updated: April 3, 2019

Download the latest version of this guide.

https://aka.ms/TFSImportData

Published: November 2016

Last updated: April 3, 2019

Change history

Date	Description of changes
September 2018	Updated to reflect that Visual Studio Team Services (VSTS) has been rebranded to be Azure DevOps Services.
November 2017	Updated to reflect the fact that the TFS Database Import Service is now in GA. Additional updates cover the changes to the identity import experience.
October 2017	Updated the guide to reflect that import codes are no longer required for queuing imports. Several other sections were updated to cover common questions.
August 2017	Updated guidance to reflect the new improvements made to TfsMigrator and changes to the import specification file. In addition, content was added to section one to address some common questions.
April 2017	Removed sections that are no longer needed, minor text tweaks to make common questions on scenarios clearer, added additional technical documentation links, and corrected some broken links.
April 2016	Initial version of the TFS to Visual Studio Team Services Migration Guide
November 2016	Initial version of the TFS to Visual Studio Team Services Migration Guide

Introduction

Ever since Azure DevOps Services was released to provide a hosted SaaS service for development teams, Azure DevOps Server (formely Team Foundation Server) customers have been asking Microsoft to be able to import their Azure DevOps Sever databases to take advantage of all the great capabilities of Azure DevOps Services. We are happy to say that the data migration tool for Azure DevOps is now available.

This Migration Guide will walk through the different steps along the way. We have organized the guide into six phases of the migration timeline. The goal of the migration is to move your team from on-premises Azure DevOps Server to Azure DevOps Services in the cloud. After the migration, your team will be able to connect to Azure DevOps Services and keep working like usual with all of the data, permissions, and customizations from your TFS database.

This guide is not meant to replace the technical documentation for the data migration tool, but is more of a way for you to easily gather the tasks you will need to perform and the resources you will need to be successful.

How to give feedback for this guide

If you have any suggestions for how we can make this guide better, please e-mail us at AzureDevOpsImport@microsoft.com

Download the latest version of this guide.

https://aka.ms/TFSImportData

Introduction

How to find a DevOps Partner to help

We highly recommend finding a trained Microsoft Partner with the DevOps Competency, Microsoft Consulting Services, or Microsoft Premier Support to help you with your migration project to Azure DevOps Services. The team at Microsoft has been working very closely to train the DevOps Partner community so that they can understand the different parts of migrating to Azure DevOps Services.

Some Microsoft customers may even have help or funding available as part of their Premier Support Agreement, Enterprise Agreement, or other programs available to assist with getting help with migrating to Azure DevOps Services and adopting Microsoft Azure. You can contact your Azure App SSP or Microsoft Reseller to find out more information and if you qualify.

Your Azure App SSP or Microsoft Reseller can help make recommendations for getting in touch with a trained Partner to help you with your migration to Azure DevOps Services. You can also find a list of trained DevOps parters available at https://aka.ms/FindDevOpsPartner.

☐ **Task:** Find a DevOps Partner

Find a DevOps partner.

https://aka.ms/FindDevOpsPartner

DevOps Partner Name:

Partner Contact Info:

Introduction

How to find your Azure App SSP (Solution Sales Professional)

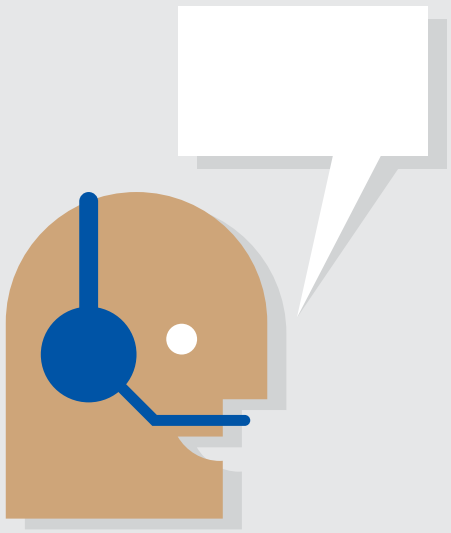
You may have questions about the different options available for getting access to the developer tools and services of the Microsoft Cloud including Azure DevOps Services. Your Azure App SSP or Microsoft Reseller is best equipped to give your team the best guidance for different types of help available to you. Reach out to your Microsoft Reseller if you have one, or if you need help with finding your Azure App SSP, you can reach out to us at FindYourDevSalesRep@microsoft.com.

My Azure App SSP or Microsoft Reseller:	
Contact Info:	

How to ask questions about the migration

During your migration, you may have questions that come up. Our first recommendation is to work with a trained DevOps Partner or Microsoft Consulting Services. If you are not able to get an answer from them, you can e-mail us at AzureDevOpsImport@microsoft.com.

1 Get Started



Microsoft

Get Started 1

☑ Task summary

- ☐ **Choose datacenter:** Choose the datacenter for your Azure DevOps organization.
- ☐ **Download data migration tool:** Download the data migration tool from https://aka.ms/DownloadTFSMigrator.
- ☐ **Reserve Azure DevOps Services organization(s):** Reserve Azure DevOps Services organization(s) for each of the desired final names.

In this first phase, we are going to help you start your Azure DevOps Services migration project and help you understand why you would want to migrate from Azure DevOps Server (formely Team Foundation Server) to Azure DevOps Services.

Note: If you're currently on Team Foundation Server (TFS), please note that the product has been rebranded to be Azure DevOps Server with the 2019 release. This guide is still applicable for migrating to Azure DevOps Server if your organization is using TFS. However, you will still have to upgrade to one of the supported versions mentioned in chapter 3 of this guide.

1 Get Started

Why migrate to Azure DevOps Services?

If you are a on-premises Azure DevOps Server customer, you have already understood how valuable it is to bring your development team together in one place with traceability from each aspect of the development life-cycle. Now that Azure DevOps Services has been launched, many customers who have migrated from Azure DevOps Server have said they did so because:

Upgraded every three weeks

Since Azure DevOps Services is cloud-based, Microsoft automatically upgrades your organization with the latest features as they are released. You can stay up to date on new releases at https://aka.ms/AzureDevOpsFeatureTimeline.

Azure DevOps Features Timeline.

https://aka.ms/AzureDevOpsFeatureTimeline

Simplified administration

The Microsoft team will monitor your Azure DevOps Services organization around the clock to make sure it is available for your team. You no longer need to worry about managing the core TFS infrastructure any longer.

Accessible from anywhere

Your team members will have the flexibility they need to securely access your organization from work, home, or their mobile devices. If you have remote development teams, you will have a great solution to collaborate together from anywhere.

Cloud-first innovation

Microsoft releases features to Azure DevOps Services ahead of making it available in updates from Azure DevOps Server. You'll be able to make your team more productive much sooner.

Included with Visual Studio subscriptions

Visual Studio (formerly MSDN) subscribers have Azure DevOps Services already included as one of their subscription benefits.

Power your Cloud Modernization initiatives

Adopting Azure DevOps Services will help your company with modernizing and driving agility and DevOps practices by making it easier to deploy your apps to the cloud and increase delivery of new business value.

Leverage developer services in the Microsoft Cloud

Using Azure DevOps Services allows you to also take advantage of the many other developer services in the Microsoft Cloud like Azure, on-demand build & deployment servers, the Load Testing Service, Application Insights, HockeyApp, Xamarin Test Cloud, and many others.

Get Started 1

Ready for the Enterprise

Azure DevOps Services is ready for teams of any size including development teams in large enterprises. There are many enterprise customers who have adopted Azure DevOps Services and empowering their development teams to collaborate more effectively. Microsoft is also migrating to Azure DevOps Services as its single engineering system to build its commercial, internal products, and cloud-based services.

Scalable to any team

Supports teams of any size: from tens to thousands. Backed by a 99.9% SLA and monitored by our 24x7 operations teams.

Secure by design

Core Azure services provide a secure foundation. Multi-layered security and governance technologies, operational practices, and compliance policies keep your data locked down.

Compliant

Azure DevOps Services is built on Azure and has the compliance certifications many enterprises need including ISO 27001 and SOC 1 and SOC 2 compliance which demonstrate our commitment. More information is available in Phase 2 of this guide.

Azure Active Directory integration

Integration with Azure AD makes it easy to manage entire organizations. Use a common identity to access both cloud and on-premises resources. Establish and enforce password lifetime and complexity controls. Enable additional security features like multi-factor authentication.

Choice of data location

Teams can choose to host their data in different locations around the world including Europe, Australia, Brazil, India, and the United States.

1 Get Started

Fundamental differences between TFS and Azure DevOps

Authentication

With Azure DevOps Server, you typically connect to a server on your on-premises network and authenticate with Windows Authentication and Active Directory. With Azure DevOps Services, you'll authenticate with Azure Active Directory organization credentials. To provide additional security, you can also require multi-factor authentication, IP address restrictions, conditional access, and more. In Phase 2 of this guide, you will go through the steps of setting up Azure Active Directory if your company has not implemented Azure Active Directory.

Reporting

Both Azure DevOps Server and Azure DevOps Services have a variety of tools to give your teams insight into the progress as well as the quality of your software projects. These include:

- Dashboards and lightweight charts
- Excel reports, SQL Server Reporting Services reports, and SharePoint dashboards are available only in Team Foundation Server and not in Azure DevOps Services. These powerful options have been more complicated to use.
- A Power BI connector is available only in Azure DevOps Services which provides a nice combination of simplicity and power.
- REST APIs are also available for getting live data from Azure DevOps Services programmatically

Note: Process Customization is now possible in Azure DevOps Services. If your team projects in Azure DevOps Server includes process template customizations, you will validate them to make sure existing customizations are supported during Phase 4 of this migration guide. Once validated, the data migration tool will import your database including your process customizations.

Data Migrated

Since each collection maps to one database in Azure DevOps Server, and the migration process works by importing an entire collection, all databases in the collection will be brought over to Azure DevOps Services. More specifically, this means that all of your work items, work item history, TFVC changesets, TFVC changeset history, Git data, build definitions, build history, and other data stored in the collection will be migrated over. Furthermore, the work item, TFVC changeset, and Git commit numbers/IDs will be retained and won't change as part of the migration. It's important to note that some data isn't brought along during the migration. Any data that resides in a separate database outside the collection database won't be imported. Prime examples of this scenario include reporting and SharePoint data. There are also a few other instances where data won't be brought over:

- **Extensions** - Extensions will need to be reinstalled post import. Local extensions will need to be published to the Marketplace as private extensions and shared with the account post import to be installed. We're working on adding Extensions to the migrations as soon as possible.
- **Service Hooks -** Service Hooks data currently isn't included in the migration process. Hooks will need to be reconfigured after import.
- **Load Test** - Load test data will not be brought over as part of import. You will have to reconfigure load test after import.
- **Mentions -** Mentions of users in work item discussions will remain but reference the on-premise identity and not the new AAD identity. Hovering on the user name will not display a contact card. Mentions of pull requests and other work items will have an invalid hyper-link.
- **Project Server Integrations -** Project Server Integration does not exist for Azure DevOps Services.

Some Azure DevOps Server features can be in preview for import to Azure DevOps Services. You can elect to include preview features with your import. If a feature isn't listed above or in a preview, then the data will be included in your import every time you run one. You can learn more about what features are in preview by visiting https://aka.ms/AzureDevOpsImportPreviewFeatures.

1 Get Started

Process Customizations

Azure DevOps Services supports two different process models:

- **Inherited**
- **Hosted XML**

By default, Hosted XML is turned off in Azure DevOps Services. Only if you have customized a project in Azure DevOps Server, we will turn on the Hosted XML process model during import. Once your project is on Hosted XML, there is a path to upgrade it to inherited post import: https://aka.ms/XMLtoInherited

When importing into Azure DevOps Services, there are limitations and key principles that we want to make you aware of:

- **Azure DevOps Services is English only** - Azure DevOps Server supports multiple languages, however today, Azure DevOps Services only supports English. If your collection uses the non-English language, you can't use the Import Service. This is also true if your collection has been non-English in the past, and you have converted the language to English during a TFS upgrade.
- **Inheritance -** A project which was created from the Agile, Scrum or CMMI process template and was never customized, will be on the Inheritance process model after the import.
- **Hosted XML -** Any project with customizations will use the Hosted XML process model.
- **Process per customized project** - Although Azure DevOps Services allows projects to share a process, the data migration tool will create a Hosted XML process for each customized team project. For example, if you have 30 customized projects, you will have 30 Hosted XML processes to manage. If you want to further customize your Hosted XML process all your projects, you will need to update each Hosted XML process separately.
- **Consolidate Processes -** Azure DevOps Services currently does not support consolidating projects to use a shared process.
- **Process validation -** The process validation of the data migration tool will detect the target process model for each project. Before you can migrate, you need to fix any process validation errors for the Hosted XML projects. You might want to consider updating the process of your projects to match one of our processes (Agile, Scrum or CMMI) to take benefit of the Inheritance process model. Learn more on the process

validation types in our documentation.

New Work Item Form

Work items in Azure DevOps Services have been given a face lift. This is the same modern experience that appears in TFS 2017 for on-premises customers. If you're already using the new modern work item experience then this section can be skipped. The new user experience provides the building blocks for:

- Improved readability and usability
- Richer, interactive experiences within the work item including discussion, code viewing and more
- Extensibility support

When a collection is imported, the form definitions for all work item types (WITs) defined in your project collection undergo an automated transformation to the new layout. This is a best-effort transformation meant to maintain the field groupings and layout of your customized WIT definitions. You can manually opt-in to the new experience on-premises before migrating to the cloud.

Reviewing and optimizing the transformed web layout will allow you to adjust this new layout to your own needs. Even though these changes can also be done once the import is completed we recommend you to do them beforehand to prepare for the change.

1 Get Started

Relationship between Azure DevOps Server databases and Azure DevOps organizations

Before diving too deeply into planning your migration, it's important to understand at a high-level how the database import process functions. Imports operate on two main concepts:

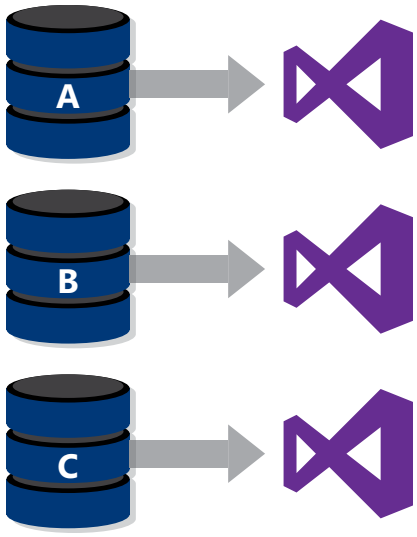
Team Project Collection

Collections in Azure DevOps Server are a physical container for team projects and their artifacts. Each collection equates to a single SQL database and are the source of import for migrations to Azure DevOps Services.

Azure DevOps Services organization

Organizations are the management unit in the cloud-hosted service. Logically they map 1:1 to the concept of a team project collection in Azure DevOps Server. Therefore, organizations are the destination of imports for migrations to Azure DevOps Services. Azure DevOps Services organizations are represented as https://dev.azure.com/contoso where contoso represents the name of the Azure DevOps Services organization.

Each time you import a team project collection SQL database, the data migration tool will create a brand new Azure DevOps organization with a name that you provide. This means that you cannot import a collection database into an existing Azure DevOps Services organization or consolidate multiple collection databases into a single Azure DevOps Services organization. It is a one-to-one mapping between team project collections and Azure DevOps Services organizations.



Team project collection mapping worksheet

Collection name:		→	Azure DevOps organization name:	
Collection name:		→	Azure DevOps organization name:	
Collection name:		→	Azure DevOps organization name:	
Collection name:		→	Azure DevOps organization name:	
Collection name:		→	Azure DevOps organization name:	

Datacenter location

You have options for the location of your Azure DevOps Services organization data. You will need to select a region where your data will be imported and mark down that region's shorthand code. Later on in the import process you will use that shorthand code. You can find supported regions for import at https://aka.ms/AzureDevOpsImportPreviewFeatures.

☐ Task: Choose the datacenter for your Azure DevOps Services organization.

Selected region's shorthand code:

1 Get Started

Purchases needed for Azure DevOps Services

Another question that typically comes up is what type of licensing will a company need to adopt Azure DevOps Services? The good news is that you are likely to have all of the licenses you already need. We have created a simplified worksheet below that you can walk through that should cover most cases. If you have any specific questions about your situation, be sure to reach out to your Developer Solution Sales Specialist or Microsoft Reseller.

User licenses worksheet

1	Number of team members	
2	Number of stakeholders	
3	Subtract Line (2) from Line (1)*	
4	Number of Visual Studio Subscribers** (formerly known as MSDN Subscribers)	
5	Subtract Line (4) from Line (3)	
6	Subtract 5 from Line (5)***	

* *Stakeholders are free.*

** *Visual Studio Subscribers have Azure DevOps Services included as a benefit of the subscription.*

*** *Each Azure DevOps Services organization gets five free users.*

You have no charge for an unlimited number of stakeholders to be able to access your Azure DevOps Services organizations without needing a user license. You also do not have a charge for any Visual Studio (formerly known as MSDN) subscribers since Azure DevOps Services is included as a benefit of the subscription. Each Azure DevOps Services organization includes no charge for access to the core features of Azure DevOps Services for the first five users.

This leaves you with the total number of Azure DevOps Services user licenses that you will need to purchase to cover your team also represented by Line 6 in the worksheet above.

You will ultimately purchase any needed Azure DevOps Services user licenses through the Visual Studio Marketplace or the Azure portal. We will discuss this more in Phase 5 of this guide.

You can find out more about pricing for Azure DevOps Services at https://aka.ms/AzureDevOpsPricing and leveraging the Azure Pricing Calculator. There are additional services that you could take advantage of like hosted load testing services, Test Manager extensions, and more that would have additional costs. If you have any questions about your specific situation, you can reach out to your DevOps Partner, Microsoft Reseller, or your Microsoft Developer Solutions Sales Specialist.

Get Started 1

Download the Data Migration Tool

The bulk of the work throughout the migration to Azure DevOps Services is handled by a new tool available from Microsoft called the data migartion tool for Azure DevOps; or migrator tool for short. The migrator tool will be used throughout this guide with the following high-level steps:

- Validating a team project collection
- Prepare and generate the files used to customize the import
- Queuing an import of an Azure DevOps Server database to Azure DevOps Services

You will want to download the latest version of the migrator tool and then you will run it from a Windows PC. The user who runs this tool must have:

1. The `TFSEXECROLE` role in SQL Server, and
2. Permissions to connect to both the TFS configuration and collection databases

☐ **Task:** Download the TFS Migrator tool from https://aka.ms/DownloadTFSMigrator.

Tip: The migrator tool will be continually updated based on feedback and our learnings from importing many customer databases. Be sure to download the latest version of the migrator tool each week to keep up to date.

1 Get Started

Reserve your Azure DevOps Services organization name(s)

Since the migration project make take some time to complete, you may want to "reserve" the name of your Azure DevOps Services organization so that the name can be available for your final import. For example, if you are from Contoso company and want a Azure DevOps organization that matches your company name like *https://dev.azure.com/contoso*, you can create an organization with that name now.

Download the TFS Migration Tool

https://aka.ms/DownloadTFSMigrator

However, we mentioned above that you can only import into a brand-new Azure DevOps Services organization. That is okay, because you when you are ready to start the final import, you could import into a Azure DevOps Services organization named *https://dev.azure.com/contoso-temporary* and then rename it to the desired name of *https://dev.azure.com/contoso* after deleting the originally reserved organization or changing its name to something else.

Tip: If the desired name is already taken, we can help facilitate passing your contact information to the current owner of the organization. They can then choose to reach out to you if they want to transfer ownership or rename their existing organization so that you can create a new organization with the desired name.

☐ **Task:** Reserve Azure DevOps Services organization(s) for each of the desired final names.

Project Limits

Customers with a large numbers of projects in a collection should note that Azure DevOps Services has a limit of 300 projects per organization. Above 300 projects certain experiences, such as connecting to the organization from Visual Studio, start to degrade. If your collection has more than 300 projects then you will either need to split the collection or delete older projects to get below the limit.

2 Cloud Prerequisites

Microsoft

2 Cloud Prerequisites

☑ Task summary

- ☐ **Implement Azure Active Directory:** Make sure your team has a working Azure Active Directory tenant by implementing Azure Active Directory to synchronize with your on-premises Active Directory environment.

In the second phase of your migration to Azure DevOps Services, we want to help you focus on some of the prerequisites for migrating their data to the cloud that many organizations have faced. Some of these may pertain to you and you will find the resources in the section helpful to your organization. There are other organizations who already have each of these prerequisites in place and will be able to bypass the second phase completely.

Note: Team Foundation Server (TFS) became Azure DevOps Server with the 2019 release of the on-premises product. If you're on an older version with the name TFS then you can still import using the data migration tool. You will just need to upgrade to a supported version of Azure DevOps Server. Therefore references to Azure DevOps Server can be freely interchanged with TFS if you're still using a version named TFS.

Cloud Prerequisites 2

Compliance

Azure DevOps Services is built on all the great reliability, scalability, and standards compliance that Microsoft Azure has come to be known for. In addition to the bedrock foundation that Azure provides, Azure DevOps Services has taken the extra steps of getting certification for individual compliance standards that customers need for their cloud-based software development services. To date, Azure DevOps Services has the following compliance certifications:

1. ISO 27001:2013
2. SOC 1 Type 2
3. SOC 2 Type 2
4. HIPAA BAA (Business Associate Agreement)
5. EU Model Clauses

The SOC audit for Azure DevOps Services covers controls for data security, availability, processing integrity, and confidentiality.

Some of our customers who have migrated from Team Foundation Server to Azure DevOps Services have needed to go through internal security reviews before adopting Azure DevOps Services. We have found that the following resources were important in helping internal security teams feel comfortable with their company adopting Azure DevOps Services.

Our internal data protection and security whitepaper

Microsoft strives for transparency about how we protect your data through multi-layered security and governance technologies, operational practices, and compliance policies. The team has documented its data protection and security practices in a whitepaper. You can learn more by reading that whitepaper at https://aka.ms/AzureDevOpsSecurity.

Data protection and security compliance policies whitepaper.

https://aka.ms/AzureDevOpsSecurity

Compliance audit report requests

The compliance audit reports are available upon request for companies who have Non-Disclosure Agreements in place with Microsoft. These audit reports were performed by our auditor, Deloitte. If you would like a copy of these reports, you can send us an e-mail at AzureDevOpsImport@microsoft.com or you can reach out to your Developer Solution Specialist.

2 Cloud Prerequisites

Azure Active Directory

The main task for Phase 2 is to make sure your team has a working Azure Active Directory tenant that will be used for authenticating your team members in your Azure DevOps Services organization.

☐ Task: Implement Azure Active Directory to synchronize with your on-premises Active Directory environment.

User authentication in Azure DevOps Server is handled on-premises by using Active Directory. With Azure DevOps Services, users are authenticated through an Azure Active Directory tenant which works very similarly to Active Directory on-premises. In Phase 5, you will be verifying an identity map log file that will show how your on-premises Active Directory organizations will match to your Azure Active Directory organizations. This will allow each of your team members to see their individual history, preserve security permissions, and make sure they have access to all of their personal settings including favorites, personal queries, etc.

Microsoft Organizations (or MSAs) are available to use for authentication with Azure DevOps Services but not available for mapping when you are importing a Team Foundation Server database using the Database Import Service.

Many companies who leverage services from the Microsoft Cloud including Office 365 and Azure, already have an Azure Active Directory tenant setup that is synchronizing user organizations and groups from Active Directory on-premises. Our recommendation is to not setup a separate Azure AD tenant for your Azure DevOps Services implementation. You will want to use the same Azure Active Directory tenant as other Microsoft Cloud services at your company.

If your company already has Azure Active Directory available, then you can skip this step in this guide and move forward.

Cloud Prerequisites 2

Synchronizing identities and groups with Azure AD Connect

By synchronizing your on-premises Active Directory with Azure Active Directory, your team members will be able to use the same credentials to authenticate and your Azure DevOps Services administrators will be able to leverage your Active Directory groups for setting permissions within your Azure DevOps Services organization.

To setup the synchronization, you will want to use the Azure AD Connect technology. You will likely want to work with your IT department, your DevOps Partner, Microsoft Premier Support, or Microsoft Consulting Services to help set up Azure AD Connect with your on-premises environment.

The documentation for setting up Azure AD Connect is available at https://aka.ms/AzureADConnect.

Note: DirSync was a predecessor technology to Azure AD Connect. You will want to upgrade to Azure AD Connect if you are using DirSync.

To read more about how Azure DevOps Services can be set up to use Azure Active Directory, you can visit: https://aka.ms/AzureDevOpsAADOrganization. Since you will be importing your TFS database, you will not be following the steps exactly in that article but it is good reference information for how it works. The data migration tool will set up the link to your Azure Active Directory tenant when your Azure DevOps Services organization is created as part of the beginning of the import process.

2 Cloud Prerequisites

Additional security for Cloud authentication

Once you have Azure Active Directory setup, we have seen many customers take additional steps that are available natively with Azure Active Directory to provide additional security measures for access to development team data as well as other data in Microsoft Cloud services like Office 365 and Azure. The next sections cover optional additional steps you can take to further secure your Azure DevOps Services organization.

Multi-Factor Authentication

One of the main additional security mechanisms that our customers have added is taking advantage of Multi-Factor Authentication (MFA) requirements as part of getting access to the data stored in a Azure DevOps Services organizations. Two-step verification is a method of authentication that requires more than one verification method and adds a critical second layer of security to user sign-ins and transactions. It works by requiring any two or more of the following verification methods:

- Something you know (typically a password)
- Something you have (a trusted device that is not easily duplicated, like a phone)
- Something you are (biometrics)

You can learn more about setting up Multi-Factor Authentication requirements with Azure Active Directory here: https://aka.ms/AzureADMFA

Conditional Access

The other common security practice we see with teams adopting Azure DevOps Services is to set conditional access rules in Azure Active Directory that provide for additional security mechanisms based on which applications they are signing into and from what location they are signing-in from. For example, you may want to specify that accessing Azure DevOps Services always requires MFA or that MFA is only required if your team member is accessing Azure DevOps Services from outside of the office (i.e., when they are at home, from their phone, or traveling).

Conditional Access capabilities allow for powerful combinations of security policies based on your organization's needs. You can find more information about setting up Azure Conditional Access here: https://aka.ms/AzureConditionalAccess

3 Upgrade

Microsoft

3 Upgrade TFS

☑ Task summary

- ☐ **Upgrade your Azure DevOps Server or Team Foundation Server:** Upgrade your Team Foundation Server or Azure DevOps Server to one of the supported versions.
- ☐ **Run "Configuration Features":** Run the "Configure Features" wizard on every team project in each of your team project collections.

One of the major prerequisites for migrating your collection database is to get your database schema version as close as possible to what is currently deployed in Azure DevOps Services. In Phase 3 of your migration project, you will work on upgrading your Team Foundation Server or Azure DevOps Server to one of the supported versions for the data migration tool for Azure DevOps.

Note: Team Foundation Server (TFS) became Azure DevOps Server with the 2019 release of the on-premises product. If you're on an older version with the name TFS then you can still import using the data migration tool. You will just need to upgrade to a supported version of Azure DevOps Server. Therefore references to Azure DevOps Server can be freely interchanged with TFS if you're still using a version named TFS.

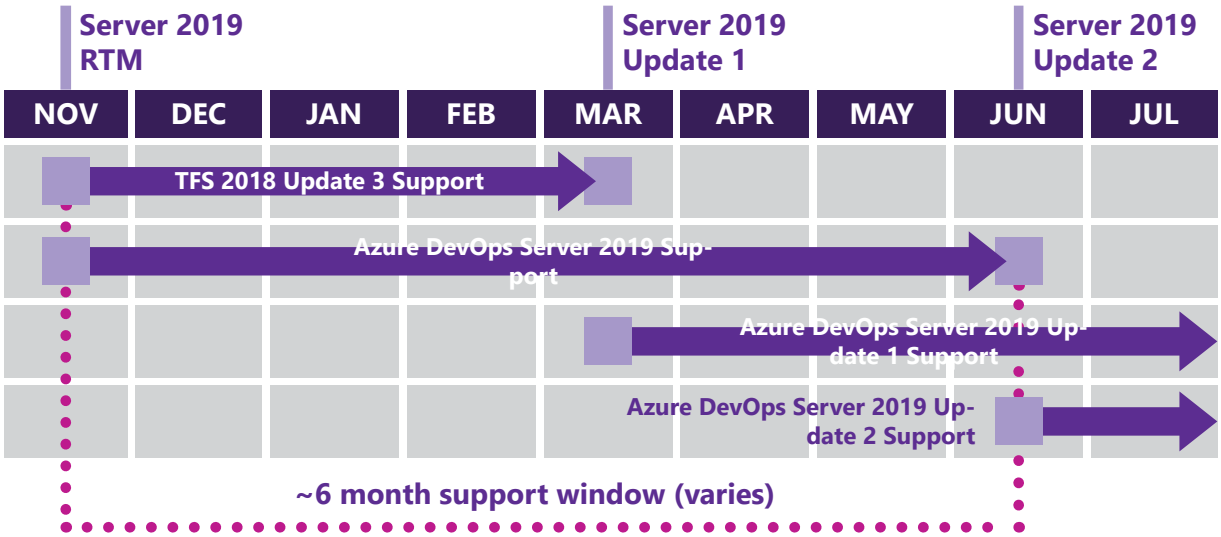
Timeline of support for Azure DevOps Server versions

It is important to note that the data migration tool for Azure DevOps does not support all versions of Azure DevOps Server/TFS databases. At any given time, the data migraton tool will support the current version of Azure DevOps Server and the previous version. Updates are included in the timeline for supported versions.

We have included a visual representation of how long individual releases will be supported as soon as new versions come out. We do not currently have release dates for future Azure DevOps server updates so we have added sample release dates as examples to help you understand the impact on the supported database versions timeline.

See which versions of Azure DevOps Server are currently supported here: https://aka.ms/AzureDevOpsImportSupportedVersions

Sample release schedule



Sample release schedule—dates are examples and not actual.

Note: It is important to look ahead in the entire migration process. If you need to upgrade, we suggest that you upgrade to the latest version of Azure DevOps Server at any given time since you may have a longer migration project cycle than other customers.

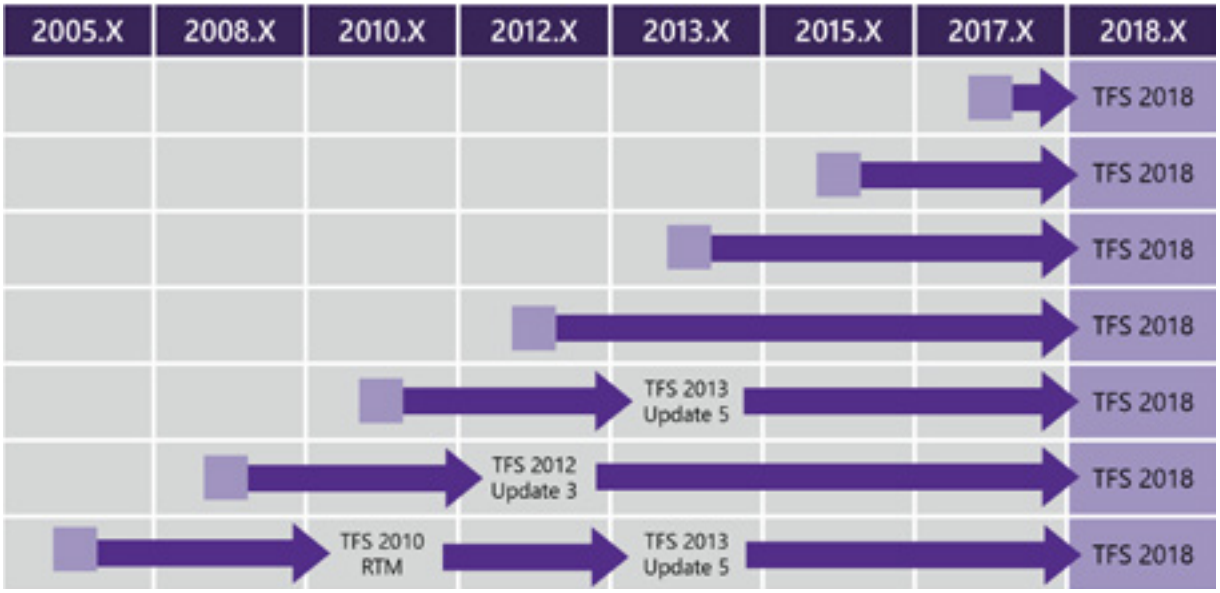
3 Upgrade TFS

Tip: It's required that you complete at least one "dry run" import per database you want to import. You should include padding in your migration timeline for at least one "dry run" database import attempt to test out the dry run imported Azure DevOps Services organization. We will cover that in Phase 5 and Phase 6 of this guide but it is good to understand this now so that you can plan accordingly for upgrading your TFS server and then migrating to Azure DevOps Services before you will need to upgrade your server again.

Upgrade paths

Microsoft has been releasing Azure DevOps Server and Team Foundation Server for over 10 years now so many customers are on various versions. Your goal will be to get to the latest version of Azure DevOps Server as mentioned in the previous section. Depending on which version of TFS you currently have in production, you have a few different paths to get you to the latest version. We have summarized each of the upgrade paths in the diagram below from which TFS version you currently have installed.

See the illustration above for our recommended steps.



Upgrade TFS 3

Tip: Azure DevOps Server updates are self-contained as of TFS 2012. As such, there is no need to upgrade to an RTM version of TFS and then apply the update – just upgrade to the update version directly. See Azure DevOps Server System Requirements for Dependencies.

One thing to remember as you plan for upgrades for your Azure DevOps Server environment are the underlying system requirements of the dependencies of Azure DevOps Server at different Azure DevOps Server versions. Azure DevOps Server has several dependencies that you will need to verify are still supported along your upgrade path:

- Operating System
- SQL Server
- SharePoint
- Project Server
- Visual Studio IDE
- Office
- Build Agent

There is a full list of system requirements for every version of Azure DevOps Server available for your reference at https://aka.ms/TFSSystemRequirements.

Upgrading Team Foundation Server or Azure DevOps Server

Now that you know what your upgrade path looks like for your Azure DevOps Server environment, you can start the steps of upgrading. This is when you will likely want to work with your selected DevOps Partner, Microsoft Consulting Services, or Microsoft Premier Support to help you with planning out the upgrade portion of your Azure DevOps Services migration project. Each of these partners are well trained in the steps necessary to upgrade your Team Foundation Server environment.

☐ **Task:** Upgrade your Team Foundation Server or Azure DevOps Server.

Upgrade resources

For your convenience, we are including each of the Upgrade Guides for the different Team Foundation Server and Azure DevOps Server upgrades you may need to perform given the upgrade paths above.

- Azure DevOps Server Upgrade Guide: https://aka.ms/AzureDevOpsServer2019Upgrade
- TFS 2018 Upgrade Guide: https://aka.ms/TFS2018Upgrade
- TFS 2017 Upgrade Guide: https://aka.ms/TFS2017Upgrade
- TFS 2013 Update 5 Upgrade Guide: https://aka.ms/TFS2013Upgrade
- TFS 2012 Update 3 Upgrade Guide: https://aka.ms/TFS2012Upgrade
- TFS 2010 Upgrade Guide: https://aka.ms/TFS2010Upgrade

3 Upgrade TFS

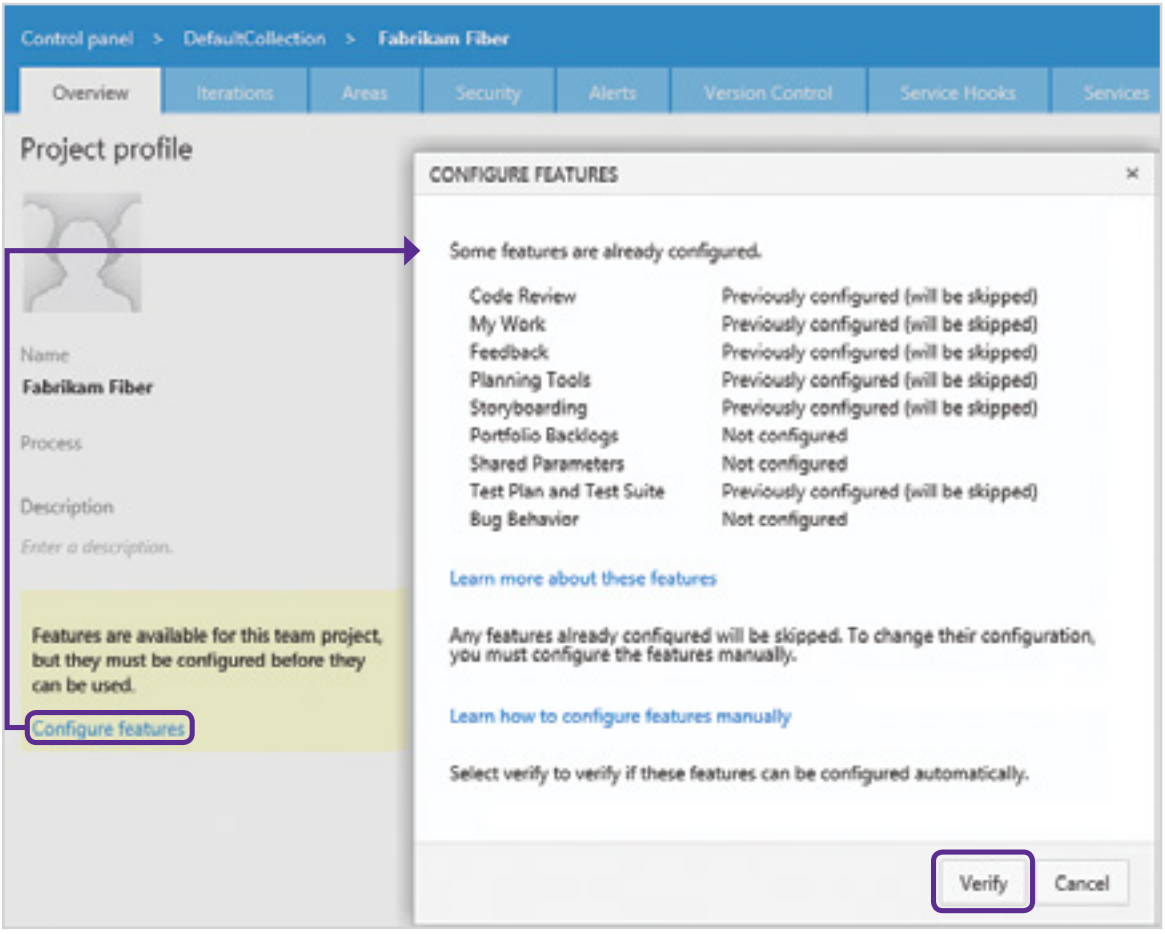
Post-upgrade steps

Now that you have upgraded your Team Foundation Server to Azure DevOps Server 2019, there are some additional post-upgrade tasks that we highly suggest taking before you move to the next phase of your Azure DevOps Services migration project.

Configure Features wizard

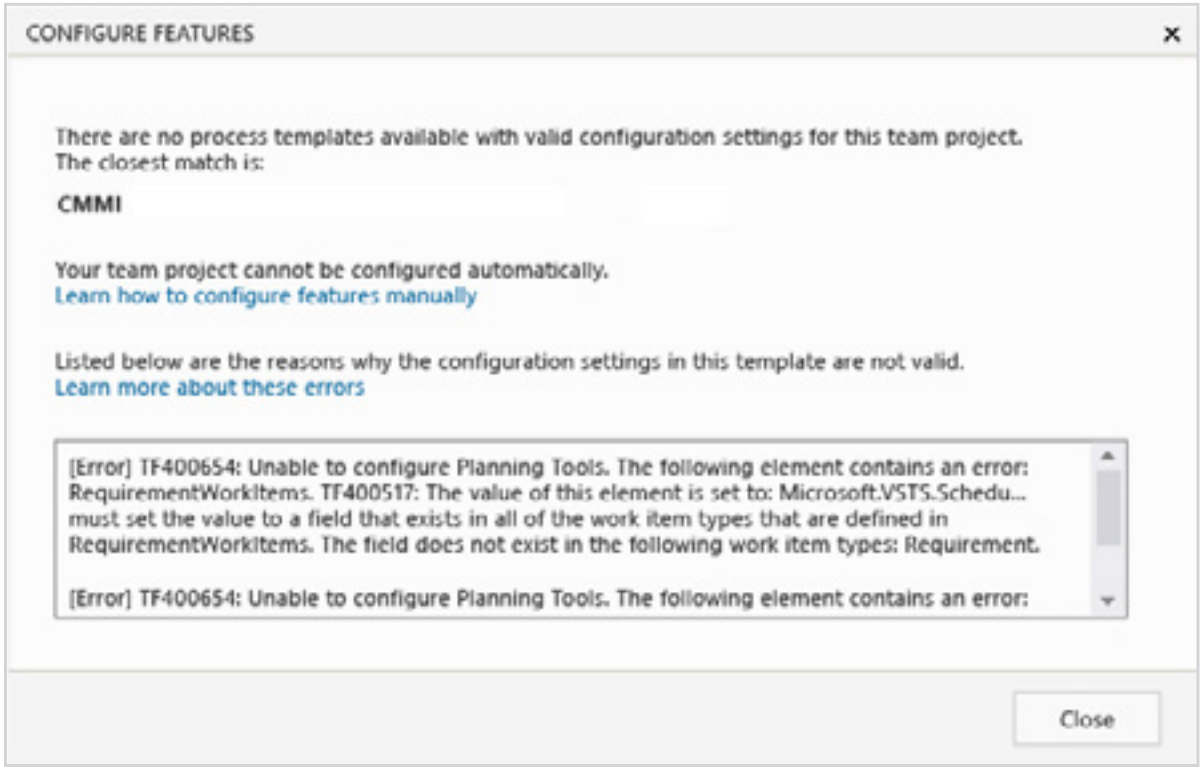
☐ Task: Run the "Configure Features" wizard on every team project in each of your team project collections.

The process used by your team projects does not get upgraded along with your collection databases. Instead, you'll need to run the *Configure Features* wizard to incorporate process changes that enable new functionality like agile planning tools and code reviews. This step is an important part of migrating to Azure DevOps Services, since it helps to ensure that the processes used in your team projects conform to the requirements of the Database Import Service. To find out more about how to use the "Configure Features" wizard, you can find the documentation article at https://aka.ms/TFSConfigureFeatures.



Applying process template updates manually

If you have heavily customized your process or have used third party process templates, the "Configure Features" wizard may not be able to automatically configure features for your team projects. In these cases, you will need to configure features manually. See the section in the documentation article titled "Apply updates manually" in https://aka.ms/TFSConfigureFeatures for more information. If you find yourself in this situation, we highly recommend working with a trained DevOps Partner, Microsoft Consulting Services, or Microsoft Premier Support.



4 Validate Your Server

Microsoft

Validate Your TFS Server 4

☑ Task summary

- ☐ **Run validations with migration tool:** Run the validation of each team project collection database with the migrator tool.
- ☐ **Review logs and fix errors:** Review the logs and fix any errors that were found.
- ☐ **Repeat validation checks:** Repeat the validation and error fixing process until there are no more errors remaining in the logs.

Now that you have your Azure DevOps Server environment upgraded to the latest version, you will start the work of ensuring that it is ready to import. The focus of Phase 4 is using the migrator tool to run verification steps to discover any errors. This section of the guide will also help you troubleshoot some common errors that may come up as well as what to do to fix them.

If you have not downloaded the latest version of the migrator tool, refer to Phase 1 of this guide for where to download it.

4 Validate Your TFS Server

Validating a team project collection

Since each team project collection is its own SQL database, you will run the validation process on each team project collection in your environment. Validation will examine a variety of aspects of your collection, including:

- Size of your collection database
- Collation of the SQL database
- Identities of users in the collection
- Analyze process template customizations

☐ Task: Run the validation of each team project collection database with the migrator tool.

You begin a validation by using the migrator tool. We recommend that you run the migrator tool from one of the application tier (AT) servers for your Azure DevOps Server environment. You can find out more about the specific command-line options by requesting the help text with the command below.

```
Migrator validate /help
```

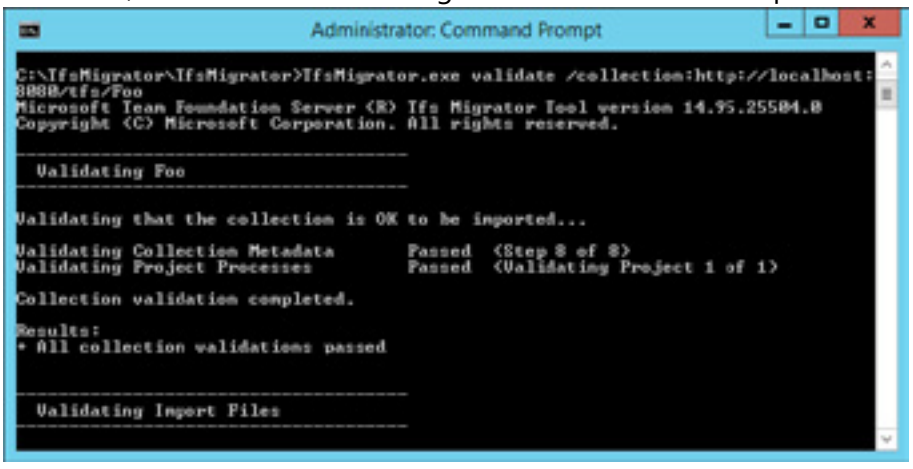
The most common way to start a validation is to specify the URL of the team project collection with the command below.

```
Migrator validate /collection:http://localhost:8080/tfs/DefaultCollection
```

There is additional technical documentation available for the validation phase available at https://aka.ms/AzureDevOpsImportValidate.

Review validation warnings and errors

Once the migrator tool is finished, there will be a set of log files and a set of results printed to the command prompt screen.



Validate Your TFS Server 4

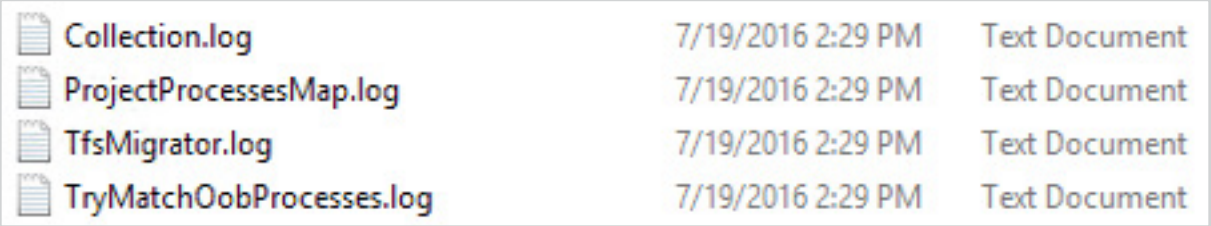
If there were no errors and all the validation checks have passed, then your team project collection is ready and you can move on to the next phase. If it does not say that all of the validation checks have passed, then you will need to look through the log files to find any errors and fix them.

☐ Task: Review the logs and fix any errors that were found.

There are a set of logs that are generated during the validation phase. The main log that you will want to focus on is the `Migrator.log` file which contains the main details on the validation checks that were run. The other files exist to contain only the errors in the section of the validation checks that match their file name.

The `TryMatchOobProcesses.log` contains the list of errors that would block your projects from landing in the inherited process model post import. TfsMigrator will look at your projects and determine if a project is using an Out-of-Box (OOB) process such as Agile, Scrum, or CMMI. If it is, and it does not contain any customizations, that project will be brought into the inherited model. Errors in this file will not prevent you from doing an import. If you have customizations that you want to keep during import then this log can be ignored. Instead you should focus on the errors in the `Migrator.log` file.

Learn more about how processes are imported at https://aka.ms/ImportProcesses



There are several types of errors that could show up in the logs from the validation checks. Solutions for many of the errors are documented in our troubleshooting guide at https://aka.ms/AzureDevOpsImportTroubleshooting. You should also leverage your trained DevOps Partner, Microsoft Consulting Services, or Microsoft Premier Support to help you with solutions for each of the errors you encounter.

4 Validate Your TFS Server

Process template errors

The most common types of errors that we have seen have been process template errors that are either because the latest features of Azure DevOps Server have not been added to older team projects or there are customizations that Azure DevOps Services does not support now. There are many customizations that Azure DevOps Services does support so the validation checks only look for customizations that need to be fixed before migrating to Azure DevOps Services.

A list of supported process customizations is available at https://aka.ms/SupportedProcessCustomizations

At the end of Phase 3, you ran the "Configure Features" wizard on each of the team projects in your collections so you should not have any errors related to missing process template items from newer features of TFS.

For the remaining types of process errors, you will use the `witadmin.exe` command-line tool that is included with installations of Visual Studio. There is deeper technical documentation for addressing many of the process errors that show up in the validation logs at https://aka.ms/ImportProcesses.

There are a few tips for tools you can use to help you with addressing process errors in addition to `witadmin.exe`.

To help with troubleshooting process template errors, you may want to automate exporting the process templates for each of the team projects in your team project collection. There is an undocumented command for the migrator tool that will help you out. You can add this option at the end of the `validate` command to generate zip files of each of the process templates used by each of the team projects.

```
Migrator validate /collection:http://localhost:8080/tfs/DefaultCollection
-SaveProcessZips
```

Another tool that many TFS administrators find helpful in this scenario is the TFS Team Project Manager available on GitHub at https://aka.ms/TeamProjectManager. One of the most useful features of this tool is the ability to compare each team project with known process templates (like the out of the box process templates). You can then look at the comparison details for the work item types and project process configuration settings to see what is different.

Validate Your TFS Server 4

Collection size

The data migration tool for Azure DevOps can import very large databases, but once a database reaches a certain size it will have to be imported using a method other than the DACPAC method discussed heavily in the latter part of this guide. The migrator tool will let you know if you need to use the alternative method of setting up a SQL Azure VM to complete an import. If you don't see any collection size warnings then you can continue with the DACPAC method. Details on proceeding with both methods are discussed in Phase 6. This includes links to deeper technical documentation for configuring a SQL Azure VM should you need to go that route.

SQL Database collation

There are only two collations natively supported by the TFS Database Import Service and Azure DevOps Services. Those two collations are:

- SQL_Latin1_General_CP1_CI_AS
- Latin1_General_CI_AS

If you have a different collation, it's still possible to import. See the following page for more details: https://aka.ms/AzureDevOpsImportCollations

Repeating the validation checks

There will be a few iterations where you will resolve some errors and then repeat running the validation checks to see if the error is no longer detected in the validation log files. You will want to repeat this process until there are no more errors and you see the success confirmation that all collection validation checks have passed.

☐ **Task:** Repeat this validation and error fixing process until there are no more errors remaining in the logs.

5 Get Ready for Import

Microsoft

Get Ready for Import 5

☑ Task summary

- ☐ **Assign, activate, and map Azure DevOps Services subscriptions:** Ensure that each of the Visual Studio (formerly MSDN) subscriptions are assigned, activated, and mapped to each subscriber's Azure Active Directory organization.
- ☐ **Generate import settings:** Generate import settings and related files using the `Migrator prepare` command.
- ☐ **Provide the configurable settings:** Provide the configurable settings in the Import Specification file.
- ☐ **Review** the Identity Map log file
- ☐ **Task:** Create an Azure Storage Container in the same datacenter as the final Azure DevOps Services organization.

Now that you have confirmed that your Azure DevOps Server collection database is validated, your team can start to prepare for your dry run and final imports. This section of the guide is dedicated to preparing your team and generating the files that are needed by the data migration tool in Azure DevOps Services.

5 Get Ready for Import

Subscriptions

You may remember from Phase 1 of this guide that Visual Studio (formerly MSDN) subscribers include access to Azure DevOps Services as a benefit of their subscription. Taking advantage of that benefit requires that each subscription is assigned, activated, and mapped to the Azure Active Directory organization for the subscriber if the subscription is not assigned to the Azure Active Directory organization from the beginning. This will likely take some time since each of your team members will potentially have action items and you will want to work with your company's designated Visual Studio Subscriptions Administrator (formerly MSDN Administrator).

☐ **Task:** Ensure that each of the Visual Studio (formerly MSDN) subscriptions are assigned, activated, and mapped to each subscriber's Azure Active Directory organization.

The high-level set of steps for each subscription your team owns are:

1. The Visual Studio Subscriptions Administrator logs into the Administrator's Portal and assigns a subscription to each of the team members. The recommended approach for this step is to assign the subscription to the Azure Active Directory organization of the subscriber.
2. The subscriber then goes to the subscriber portal and logs in with the same e-mail address to activate the subscription.
3. If the subscription was activated using a Microsoft Organization (MSA), then the subscriber will need to link their Azure Active Directory organization to their subscription so that Azure DevOps Services will recognize the subscriber's benefit when they login to Azure DevOps Services with their Azure Active Directory organization.

Assign subscription

In the first step, the Subscriptions Administrator for your company will log in to the Administrator's Portal (https://aka.ms/VSSubscriptionAdminPortal) and assign each of the available subscriptions to the relevant team members.

The new portal supports assigning a subscription to an Azure Active Directory organization, so we highly recommend that the administrator assigns the subscription to the Azure Active Directory organization for the subscriber (i.e., johndoe@contoso.com) instead of assigning to a personal Microsoft Organization (johndoe@outlook.com).

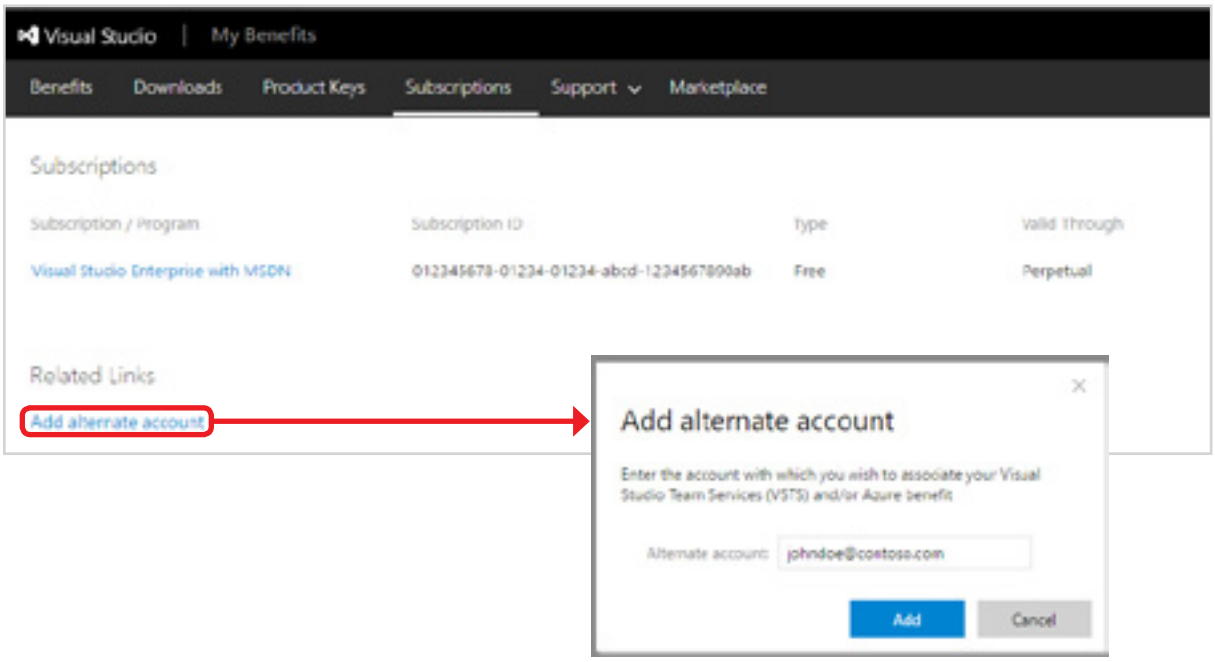
Activate subscription

Each subscriber will then login to the Visual Studio Subscriptions Portal at https://my.visualstudio.com with the organization that was assigned. If the administrator assigned the subscription to the subscriber's Azure Active Directory organization, then they need to sign-in with their Azure Active Directory organization.

Link subscription to Azure Active Directory organization

Many legacy subscribers will find that their subscription was assigned to and activated with a Microsoft Organization. The last step for each subscriber to take is to link the Visual Studio Subscription to their Azure Active Directory organization. There are a few different methods for doing this step which are documented at https://aka.ms/LinkVSSubscriptionToAADOrganization.

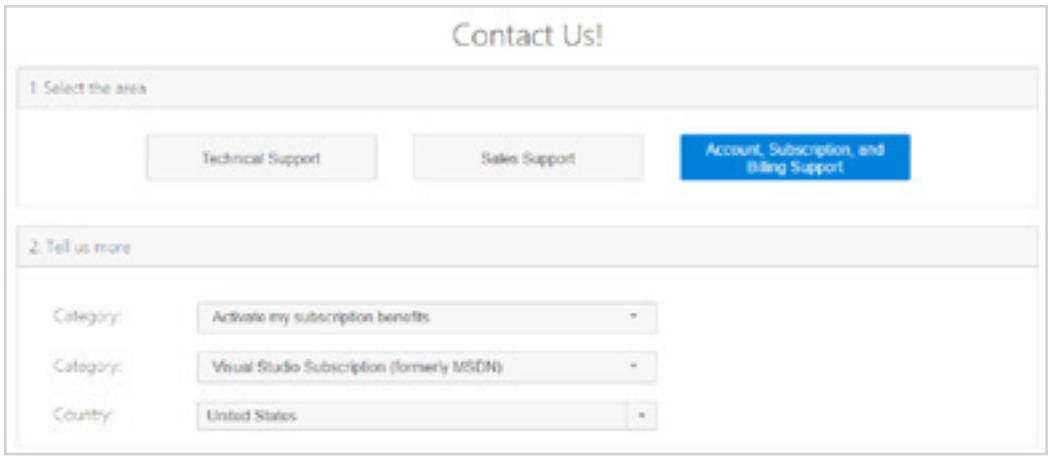
Note: This is a very important step for each subscriber to complete before the final production import of your TFS database.



5 Get Ready for Import

Help with subscriptions

If your team needs any help with activating the benefits of your subscriptions, you can reach out to the Support team at https://aka.ms/VSSubscriptionHelp. When creating a new support case, choose "Organization, Subscription, and Billing Support" and then "Activate my subscription benefits" to get started.



Generate import files with `prepare` step in `TfsMigrator`

You are ready to generate the import specification and related files you will need to queue an import of your collection database. This section will cover the different files that are produced but first you will need to run the `prepare` command to generate them.

```
Migrator prepare /collection:http://localhost:8080/tfs/DefaultCollection/
tenantDomainName:contoso.com /Region:CUS
```

The tenant domain name option is the name of your company's Azure Active Directory tenant. The `prepare` command will contact your Azure Active Directory tenant so it will prompt you to login with a user from the tenant with permissions to read information about all of the users in the Azure Active Directory tenant. It is important to understand that the `prepare` command needs to have access to the Internet for this step. If your Azure DevOps Server does not have access to the Internet, then you will need to run this command from a different computer.

Organization region refers to the location you plan to import your TFS collection into Azure DevOps Services. In Phase 1 you selected a region and recorded its shorthand code to be used in the prepare command. Just in case, a list of supported regions can be discovered on the following page https://aka.ms/ImportSupportedRegion.

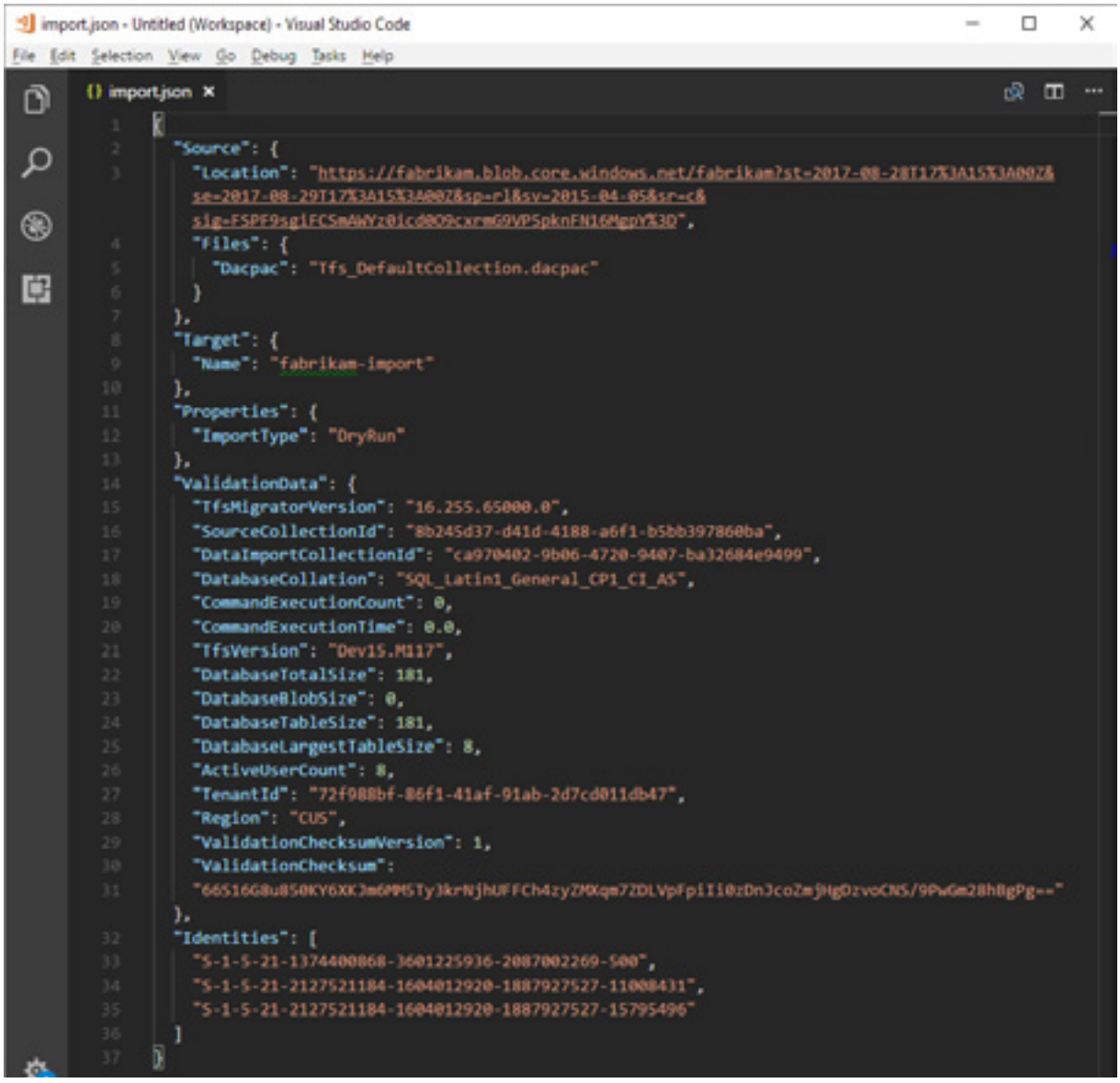
More information about the `prepare` command is available at https://aka.ms/TfsMigratorPrepare.

☐ **Task:** Generate import settings and related files using the `Migrator prepare` command.

Import specification file

The import specification file is a JSON file that will instruct the data migration tool how to configure your imported Azure DevOps Services organization, specify the source file locations, and customize the import.

☐ **Task:** Provide the configurable settings in the Import Specification file.



5 Get Ready for Import

Several of the fields are auto-populated during the prepare step but some will need to be configured by you. The fields that you will need to provide are:

- **Organization Name:** the name of the Azure DevOps Services organization that you want to be created for importing your data.
- **Location:** a backup of your database and import files will be uploaded to an Azure storage container. This field specifies the SAS key that will be used by the TFS Database Import Service to securely connect to and read the source files from the Azure storage container. Creating the storage container will be covered later in Phase 5 and generating a SAS key will be covered in Phase 6 before you queue a new import.
- **Dacpac:** a file that packages up your collection's SQL database.
- **Import Type:** The type of import: DryRun or ProductionRun

More information about the import specification file can be found at https://aka.ms/AzureDevOpsImportSpecification.

Note: Each import specification file is meant for a single collection. If you try to use an import specification file generated for another collection the import won't start. You will need to run prepare for each collection you wish to import and use the generated import specification file to queue the import.

Identity Map Log

Arguably the identity map log is of equal importance to the actual data that you will be migrating to Azure DevOps Services. When reviewing the log file it's important to understand how identity import operates and what the potential results could entail. When importing an identity, they could either end up becoming active or historical. The difference between active and historical identities is that active identities can log into Azure DevOps Services whereas historical identities cannot. It's important to note that once imported as a historical identity, there is no way to move that identity to become active.

For a detailed explanation of the identity map log file see the following page: https://aka.ms/AzureDevOpsIdentityMapLog

☐ **Task:** Review the Identity Map log file

Historical vs. active identities

When importing an identity, the Import service will decide whether the identity will become *active* or *historical*.

Note: It is important to note that once an identity is imported as a historical identity, there is no way to transition that identity to become active again in the future.

Active identities

Identities that will be active users in Azure DevOps Services after imported. Active identities will have a license and show up as a user in the organization after migration. Active identities are included in the mapping file. If you want an identity to be active post-migration, make sure to include a correct mapping in the identity mapping file.

Historical identities

Identities that are not specified in the identity mapping file or the identity mapping line is not completely filled out. Historical identities do not have access to the Azure DevOps Services organization post-migration, do not have licenses, and do not show up as a user in the organization. They will be included in the history of data like version control, work items, builds, and other places where they have contributed in the past. Historical identities are recommended for users that are no longer at the company or will not ever be needing access to the Azure DevOps Services organization again. Historical identities **cannot** be converted to an active identity in the future.

5 Get Ready for Import

Licenses

During import, licenses are assigned automatically for all users displayed as 'active' in the "Expected Import Status" column of identity map log. If automatic license assignment is incorrect, don't worry all license assignments can be changed via the Azure DevOps Services User Hub by editing the "access level" of the user(s) after import is complete. We realize that our assignment may not always be perfect, so you will have until the 1st of the following month to reassign licenses as needed. If by the 1st of the next month you haven't linked a subscription to your organization and purchased the correct number of licenses, all of your grace period licenses will be revoked. Alternatively, if auto assignment assigned more licenses than you purchased for next month, then you will not be charged for the extra licenses, but all unpaid licenses will be revoked. To avoid losing access, we recommend you link a subscription and purchase needed licenses before the 1st of the month. For all dry runs, licenses are free for as long as the organization is active.

Azure DevOps Subscriptions

☐ Task: Verify and update licenses in the identity map

Visual Studio Subscriptions (formerly known as MSDN benefits) aren't assigned by default for imports. Instead, users with Visual Studio Subscriptions will be automatically upgraded to use that license. As long as a user's work organization is linked correctly, Azure DevOps Services will automatically apply their Visual Studio subscription benefits on their first login post import.

You don't need to repeat a dry run import if users don't automatically get upgraded to use their Visual Studio Subscription in Azure DevOps Services. Visual Studio Subscription linking is something that happens outside of the scope of an import. As long as the work organization gets linked correctly before or after the import then the user will automatically have their license upgraded on the next sign in. Once they've been upgraded successfully, next time you import the user will be upgraded automatically on the first sign in to the organization. See earlier in Phase 5 for details on setting up this link.

Create an Azure Storage Container in chosen datacenter

Using the data migration tool for Azure DevOps requires having an Azure Storage container in the same Azure datacenter as the final Azure DevOps Services organization. For example, if you intend for your Azure DevOps Services organization to be created in the Central United States datacenter, then you will want to create the Azure Storage container in that same datacenter. This will drastically speed up the time that it takes to import the SQL database since the transfer will occur within the same datacenter.

☐ **Task:** Create an Azure Storage Container in the same datacenter as the final Azure DevOps Services organization.

You can find out more about creating storage containers at https://aka.ms/CreateAzureStorageContainer.

Worksheet

Azure Storage Container Name:	
Datacenter Location:	

Set up Azure subscription for billing

A grace period is placed on the newly imported Azure DevOps Services organization to allow your team to finish any steps it needs and correct license assignments. If you anticipate needing to purchase any additional user plans, build/deployment pipelines, hosted build services, hosted load test services, or other developer services, we highly recommend making sure that you have an Azure Subscription ready for linking to your imported Azure DevOps Services organization once the import has completed. The grace period ends on the first day of the following month after you have completed your import.

We will remind you again in Phase 6 for when you will need to do the linking. This preparation step is more about making sure that you know which Azure Subscription you will use in that later step. You can find out more about setting up an Azure Subscription for Azure DevOps Services billing at https://aka.ms/AzureDevOpsSetupBilling.

6 Import

Microsoft

Import 6

☑ Task summary

- ☐ **Dry run of end-to-end import:** Complete a dry run of the end-to-end import before scheduling your production import.
- ☐ **Detach the team project collection:** Detach the team project collection in the Administration Console.
- ☐ **Create portable backup:** Create portable backup of the Team Project Collection SQL database.
- ☐ **Upload SQL database backup:** Upload SQL database backup and identity map to Azure Storage Container.
- ☐ **Generate SAS key:** Generate a SAS key for the Azure Storage container and modify your import settings file to include the SAS Key.
- ☐ **Delete previous dry run organizations:** Delete any previous dry run Azure DevOps Services organizations.
- ☐ **Rename imported organization:** Rename the imported Azure DevOps Services organization to the desired name that was reserved in Phase 1.
- ☐ **Set up billing:** Set up the billing for the Azure DevOps Services Organization with the Azure subscription identified in Phase 5.
- ☐ **Reconnect to new organization:** Reconnect on-premises build servers to the newly-imported Azure DevOps Services organization.

6 Import

The great news is that your team is now ready to begin the process of starting a dry run of your import and then finally a full production import run. In this phase of the guide, we will walk through the final steps to queue an import as well as discuss the other topics that typically come up at the end of the migration project to get you prepared.

☐ **Task:** Complete a dry run of the end to end import before scheduling your production import.

Considerations for roll back planning

A common concern that teams have for the final production run is to think through what the rollback plan will be if the import failures. This is also why we highly recommend doing a dry run to make sure you are able to import into Azure DevOps Services successfully and that everything is as you expected it to be post import.

Rollback for the final production run is fairly simple. Before you queue the import, you will be detaching the team project collection from Team Foundation Server which will make it unavailable to your team members. If for any reason, you need to roll back the production run and have Team Foundation Server come back online for your team members, you can simply attach the team project collection on-premises again and inform your team that they will continue to work as normal while your team regroups to understand any potential failures.

You can then reach out to Azure DevOps Services customer support for assistance with understanding the cause of the failure. Customer support tickets can be opened from the following page https://aka.ms/AzureDevOpsImportSupport. It's important to note that if the issue requires product group engineers to engage that those cases will be handled during regular business hours.

Timing worksheet

One thing that is very helpful is to keep track of the timing for each of the steps during the dry run import so that you can start to plan out your final production import timeframe. The worksheet below will help you keep track of the different steps to time for your dry run(s) and production import.

Time for each step

	Dry Run Import #1	Dry Run Import #2	Production Import
Detach Collection			
Generate Backup of SQL Database			
Upload Backup and Identity Map to Azure Storage			
Queue Import			
Final UAT Verification of Imported Azure DevOps Services Organization			

6 Import

Detach your team project collection from Azure DevOps Server to prepare it for import

Before generating a backup of your SQL database, the data migration tool requires the collection to be completely detached from Azure DevOps Server (not SQL). The detach process in Azure DevOps Server transfers user identity information that is stored outside of the collection database and makes it portable to move to a new TFS server or in this case, to Azure DevOps Services.

☐ Task: Detach the team project collection in Administration Console.

Detaching a collection is easily done from the Azure DevOps Server Administration Console on your Azure DevOps Server instance. There is a walk-through for detaching the team project collection at https://aka.ms/DetachTFSCollection.

Generate database backup

If Migrator didn't produce any warnings about your collection's size, the data migration tool is able to import from a specific SQL backup format: DACPAC. You can find the command-line tool necessary for generating DACPAC files in the SQL Server Data Tools. Here is a sample command-line entry for generating a DACPAC backup file.

```
SqlPackage.exe /sourceconnectionstring:"Data Source=localhost;Initial
Catalog=Tfs_Foo;Integrated Security=True" /targetFile:C:\DACPAC\
Tfs_Foo.dacpac /action:extract /p:ExtractAllTableData=true
/p:IgnoreUserLoginMappings=true /p:IgnorePermissions=true
/p:Storage=Memory
```

There is more information about generating DACPAC backup files and where to find SQL Server Data Tools available at https://aka.ms/CreateTFSBackupDACPAC.

☐ Task: Create portable backup of the team project collection SQL database.

Alternate method: importing large collection databases

If Migrator warned that your collection was too big, you will not want to generate a DACPAC backup of your SQL database. There is an alternate method that you will need to take which is setting up your own SQL Server in the same Azure datacenter, restoring the database there, and updating your import settings with a connection string to your database for the data migration tool to use to create a direct connection for importing your database.

You can find out more about the alternate method of importing if you have a large collection at https://aka.ms/AzureDevOpsImportLargeCollection.

Dry run only: attach team project collection again

If this is your **dry run import,** once the SQL database backup of the fully detached team project collection has fully completed, you can attach the team project collection again to make it available to your team members while you continue the rest of the import steps.

If this is your production import, we **do not** recommend attaching your collection to Azure DevOps Server again unless you need to rollback your final import attempt and have TFS available for your team members to continue working.

Upload backup to Azure Storage Container

Once you have your DACPAC backup file ready, you can upload it to the Azure Storage container that you created in Phase 5 of this guide. The time to copy can vary depending on your Internet speed and the size of your backup file. If you're providing your collection data via the alternate SQL Azure VM method then you don't need to upload anything to a storage container - it's not required.

One of the best methods for copying to an Azure Storage container is by using the AzCopy tool. You can find out more how to use it at https://aka.ms/StorageAzCopy.

☐ Task: Upload SQL database backup and identity map to Azure Storage Container

6 Import

Generate SAS key for the Azure Storage Container

The last setting in the import settings file that you will need to update is the SAS key for the Azure Storage Container so that the data migration tool can securely connect to the storage container to give the Import Service the minimal set of permissions needed to access your team's data. The SAS key can even be time limited to cut off access after a desired time period. It is strongly recommended that you time limit the key to be enabled for at least a minimum of seven days.

Note: It is important to treat the SAS key as a secret. Do not leave the key in an insecure location as it grants read and list access to any data that you have stored in the container.

You can find out how to generate a SAS key at https://aka.ms/GenerateSASKey.

☐ **Task:** Generate a SAS key for the Azure Storage container and modify your import settings file to include the SAS key.

Delete previous dry run import Azure DevOps Services organizations

Before you can run a second dry run import or the final production import, you will need to make sure you delete any previous Azure DevOps Services organizations that were created in a previous dry run. You can follow the steps at https://aka.ms/AzureDevOpsDeleteOrganization.

☐ **Task:** Delete any previous dry run Azure DevOps Services organizations.

Queue the import

You are now ready to queue the import with the data migration tool. Ensure that the import specification file has been completed. Then you can simply queue the import with the following command.

```
Migrator import /importFile:C:\TFSDataImportFiles\import.json
```

This will begin the import and the user that queued the import will own the imported organization. They will also receive an e-mail whenever the import has failed or succeeded. If you receive an email that your import failed please reach out to Azure DevOps Services customer support for assistance at https://aka.ms/AzureDevOpsImportSupport.

Post-import steps

A success e-mail will be sent to the organization owner as soon as the import has successfully completed. At this point, anyone with access will be able to login to the newly imported Azure DevOps Services organization. There are a few remaining items that you may want to perform but for the most part, the Azure DevOps Services organization is ready for your team members to use.

We have captured some of the common steps here but you can find an updated list of post-import topics at https://aka.ms/AzureDevOpsPostImport.

Rename final imported organization to desired name

In Phase 1 of this guide, you may have preemptively created organizations with the final Azure DevOps Services organization names that you want to use. If this is your final import, you can rename your newly imported Azure DevOps Services organization to that desired name. The steps to take at a high-level are:

1. Rename placeholder organization to a different name. For example, from dev.azure.com/contoso to dev.azure.com/contoso-old
2. Wait for approximately 1 hour for the desired name to become available again
3. Rename the newly imported organization to the desired name. For example, from dev.azure.com/contoso-import to dev.azure.com/contoso

You can rename a Azure DevOps Services organization by following the directions at https://aka.ms/AzureDevOpsRenameOrganization.

☐ **Task:** Rename the imported Azure DevOps Services organization to the desired name that was reserved in Phase 1.

6 Import

Set up billing

Now that the Azure DevOps Services Organization is created, you can complete the final steps of linking the Azure subscription identified in Phase 5 of this guide so that billing is now linked correctly.

> **Note:** If you are not needing to purchase any additional licenses or needing any additional services like builds, deployments, load testing, marketplace extensions, etc. then you can skip this step.

As a reminder, the final steps of setting up billing with an Azure subscription are detailed at https://aka.ms/AzureDevOpsSetupBilling.

☐ **Task:** Setup the billing for the Azure DevOps Services Organization with the Azure subscription identified in Phase 5.

Configure build agents

If you were using automated build or deployment servers in your Azure DevOps Server environment, you can now connect them to your Azure DevOps Services organization. As part of the import, all of your build definitions have been brought over, but agents and pools need to be reconfigured against the new Azure DevOps Services organization.

You can find the additional steps needed at https://aka.ms/AzureDevOpsPostImportBuilds.

☐ **Task:** Reconnect on-premises build servers to the newly imported Azure DevOps Services organization.

Hosted build and deployment pipelines

Your team can also explore taking advantage of the hosted build servers available now that your team has adopted Azure DevOps Services. Many customers who have transitioned from Azure DevOps Server to Azure DevOps Services have told us that they were able to retire some of their custom build hardware by leveraging the hosted build & deployment services available in Azure DevOps Services.

Summary

Congratulations! Your team is now in Azure DevOps Services and you do not need to worry about upgrading your instance again. Now that you are in Azure DevOps Services you will receive new updates frequently and have many opportunities to implement tools and processes that will help your team to be more effective in building quality software.

Roadmap and release notes for Azure DevOps Services

New releases are deployed approximately every three weeks and we want to make sure you and your team have the information you need to stay up to date with everything that is new. At the same time, we want to make sure you know where we are investing in helping your development teams.

You can find both our roadmap and a detailed set of release notes for each deployment at https://aka.ms/AzureDevOpsFeatureTimeline.

Deployment roadmap and release notes.

https://aka.ms/AzureDevOpsFeatureTimeline

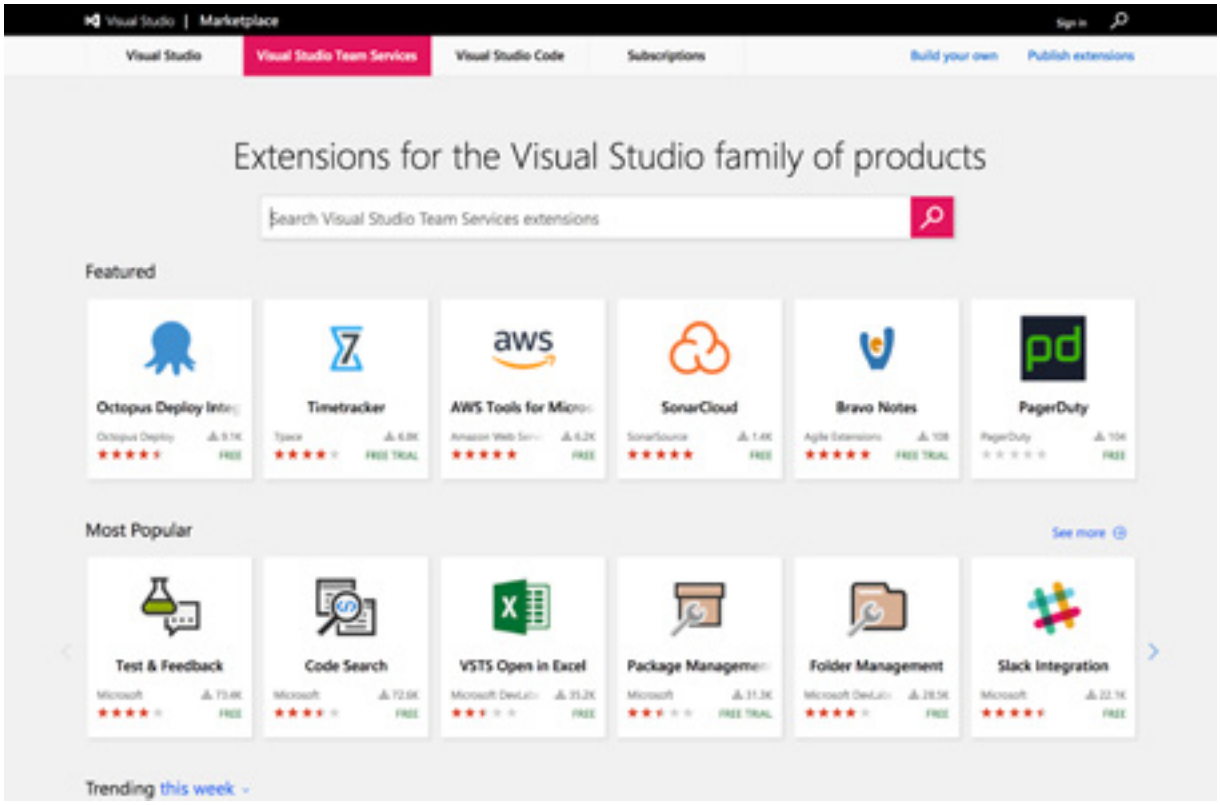
Summary

Find new extensions in the Azure DevOps Services Marketplace

The Azure DevOps Services Marketplace is one of the major benefits of adopting Azure DevOps Services because each of the extensions are extremely easy to install and configure with your Azure DevOps Services Organization. There are many extensions that will add new build/deployment tasks, integrations with many third-party services, and add incredible value for your teams to get more out of the information & functionality in Azure DevOps Services.

Some of the first extensions you should immediately install are:

- Code Search
- Package Management
- HockeyApp
- Microsoft Teams Integration
- Test & Feedback
- Application Insights



Summary

You can find the Azure DevOps Services Marketplace at https://aka.ms/Azure DevOps ServicesMarketplace.

Move forward with other developer services from the Microsoft Cloud

Working in Azure DevOps Services means that it is much easier to take advantage of the many other services available for development teams. Be sure to look at these additional services in the Microsoft Cloud.

- Microsoft Azure
- SQL Server and SQL Azure
- Application Insights
- HockeyApp
- Xamarin Test Cloud
- Development/Test Labs in Microsoft Azure
- Office 365
- Dynamics

Stay connected and involved

As your teams start to leverage more of the many solutions available in Azure DevOps Services, please be sure to stay connected and involved with helping us know what you think we should be building to make Azure DevOps Services even better. We have a User Voice site where you can suggest new features and vote on other developer's feature requests to help us prioritize our future investments. You can find that User Voice site at https://aka.ms/AzureDevOpsUserVoice.