What is the Team Data Science Process?

Azure Machine Learning

The Team Data Science Process (TDSP) is an agile, iterative data science methodology to deliver predictive analytics solutions and intelligent applications efficiently. TDSP helps improve team collaboration and learning by suggesting how team roles work best together. TDSP includes best practices and structures from Microsoft and other industry leaders to help toward successful implementation of data science initiatives. The goal is to help companies fully realize the benefits of their analytics program.

This article provides an overview of TDSP and its main components. We provide a generic description of the process here that can be implemented with different kinds of tools. A more detailed description of the project tasks and roles involved in the lifecycle of the process is provided in additional linked topics. Guidance on how to implement the TDSP using a specific set of Microsoft tools and infrastructure that we use to implement the TDSP in our teams is also provided.

Key components of the TDSP

TDSP has the following key components:

- A **data science lifecycle** definition
- A **standardized project structure**
- **Infrastructure and resources** recommended for data science projects
- **Tools and utilities** recommended for project execution

Data science lifecycle

The Team Data Science Process (TDSP) provides a lifecycle to structure the development of your data science projects. The lifecycle outlines the full steps that successful projects follow.

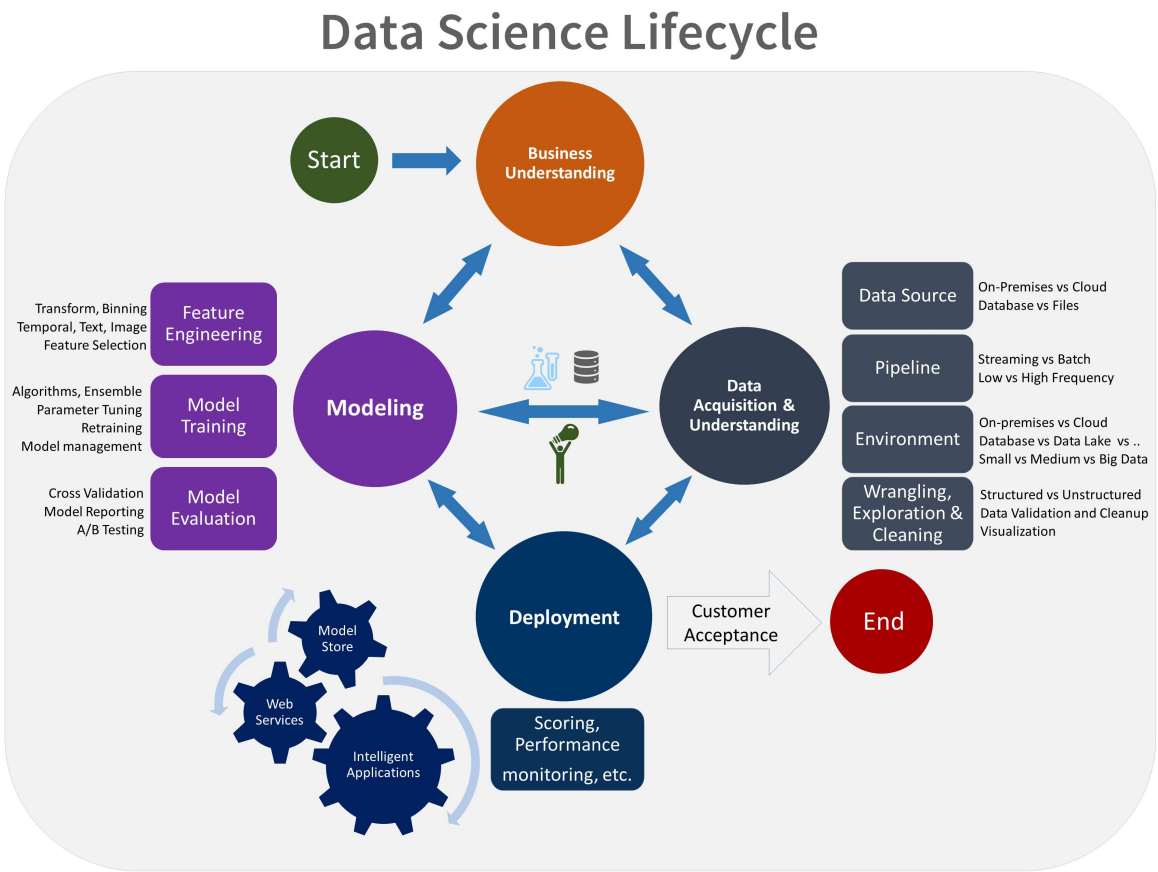
If you are using another data science lifecycle, such as CRISP-DM, KDD, or your organization's own custom process, you can still use the task-based TDSP in the context of those development lifecycles. At a high level, these different methodologies have much in common.

This lifecycle has been designed for data science projects that ship as part of intelligent applications. These applications deploy machine learning or artificial intelligence models for predictive analytics. Exploratory data science projects or improvised analytics projects can also benefit from using this process. But in such cases some of the steps described may not be needed.

The lifecycle outlines the major stages that projects typically execute, often iteratively:

- **Business Understanding**
- **Data Acquisition and Understanding**
- **Modeling**
- **Deployment**

Here is a visual representation of the **Team Data Science Process lifecycle**.

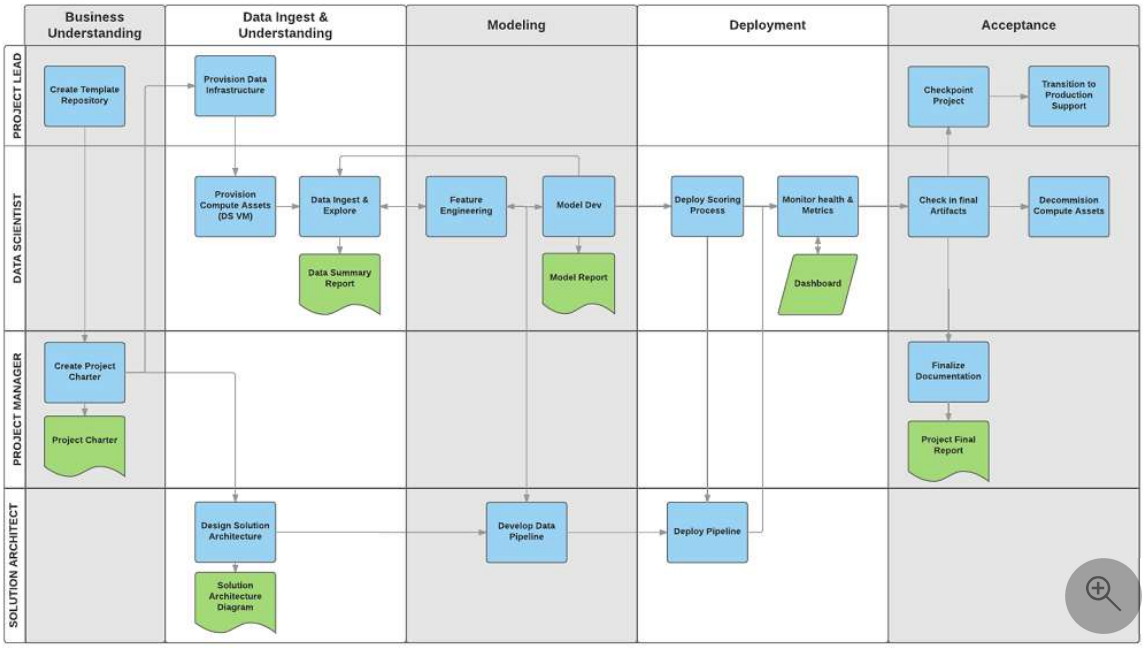


The goals, tasks, and documentation artifacts for each stage of the lifecycle in TDSP are described in the Team Data Science Process lifecycle topic. These tasks and artifacts are associated with project roles:

- Solution architect
- Project manager
- Data engineer
- Data scientist

- Application developer
- Project lead

The following diagram provides a grid view of the tasks (in blue) and artifacts (in green) associated with each stage of the lifecycle (on the horizontal axis) for these roles (on the vertical axis).



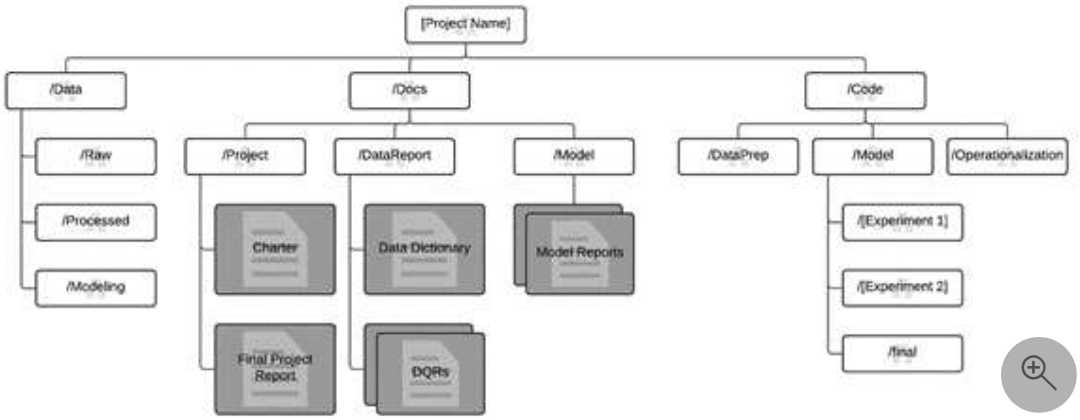
Standardized project structure

Having all projects share a directory structure and use templates for project documents makes it easy for the team members to find information about their projects. All code and documents are stored in a version control system (VCS) like Git, TFS, or Subversion to enable team collaboration. Tracking tasks and features in an agile project tracking system like Jira, Rally, and Azure DevOps allows closer tracking of the code for individual features. Such tracking also enables teams to obtain better cost estimates. TDSP recommends creating a separate repository for each project on the VCS for versioning, information security, and collaboration. The standardized structure for all projects helps build institutional knowledge across the organization.

We provide templates for the folder structure and required documents in standard locations. This folder structure organizes the files that contain code for data exploration and feature extraction, and that record model iterations. These templates make it easier for team members to understand work done by others and to add new members to teams. It is easy to view and update document templates in markdown format. Use templates to provide checklists with key questions for each project to insure that the

problem is well defined and that deliverables meet the quality expected. Examples include:

- a project charter to document the business problem and scope of the project
- data reports to document the structure and statistics of the raw data
- model reports to document the derived features
- model performance metrics such as ROC curves or MSE



The directory structure can be cloned from GitHub.

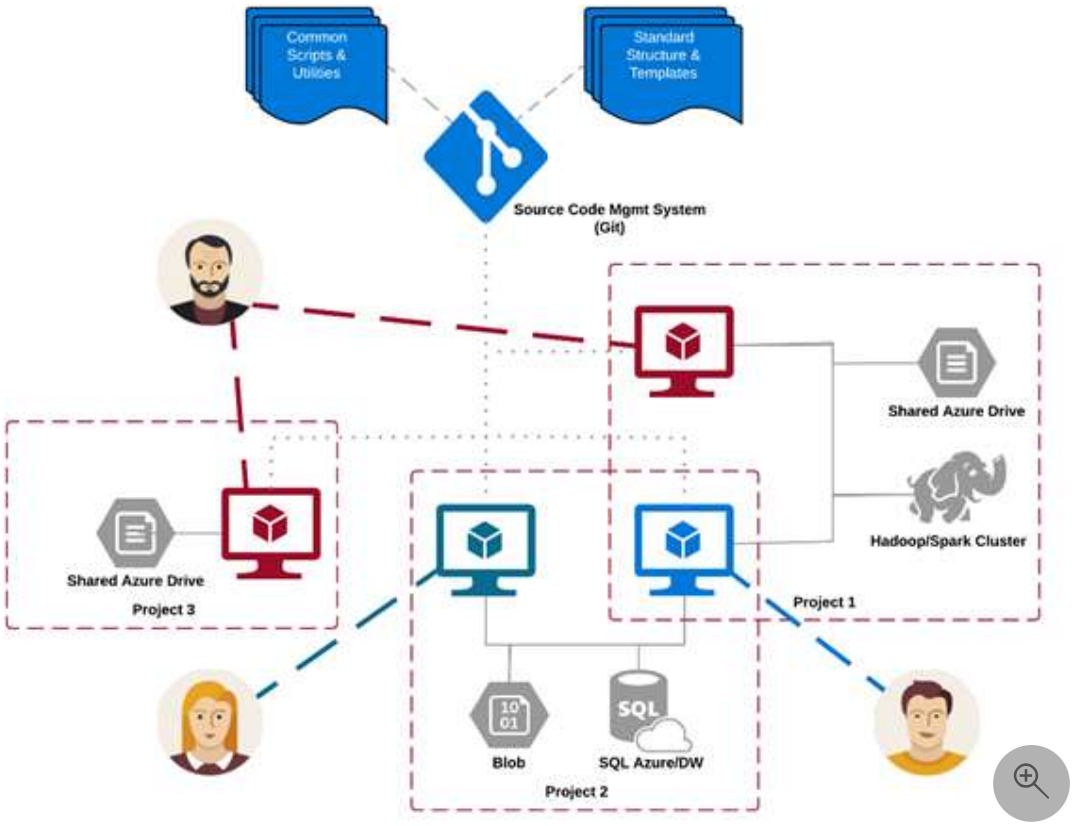
Infrastructure and resources for data science projects

TDSP provides recommendations for managing shared analytics and storage infrastructure such as:

- cloud file systems for storing datasets
- databases
- big data (SQL or Spark) clusters
- machine learning service

The analytics and storage infrastructure, where raw and processed datasets are stored, may be in the cloud or on-premises. This infrastructure enables reproducible analysis. It also avoids duplication, which may lead to inconsistencies and unnecessary infrastructure costs. Tools are provided to provision the shared resources, track them, and allow each team member to connect to those resources securely. It is also a good practice to have project members create a consistent compute environment. Different team members can then replicate and validate experiments.

Here is an example of a team working on multiple projects and sharing various cloud analytics infrastructure components.



Tools and utilities for project execution

Introducing processes in most organizations is challenging. Tools provided to implement the data science process and lifecycle help lower the barriers to and increase the consistency of their adoption. TDSP provides an initial set of tools and scripts to jump-start adoption of TDSP within a team. It also helps automate some of the common tasks in the data science lifecycle such as data exploration and baseline modeling. There is a well-defined structure provided for individuals to contribute shared tools and utilities into their team's shared code repository. These resources can then be leveraged by other projects within the team or the organization. Microsoft provides extensive tooling inside Azure Machine Learning supporting both open-source (Python, R, ONNX, and common deep-learning frameworks) and also Microsoft's own tooling (AutoML).

Next steps

Team Data Science Process: Roles and tasks Outlines the key personnel roles and their associated tasks for a data science team that standardizes on this process.

The Team Data Science Process lifecycle

Article • 11/15/2022

The Team Data Science Process (TDSP) provides a recommended lifecycle that you can use to structure your data-science projects. The lifecycle outlines the complete steps that successful projects follow. If you use another data-science lifecycle, such as the Cross Industry Standard Process for Data Mining (CRISP-DM), Knowledge Discovery in Databases (KDD), or your organization's own custom process, you can still use the task-based TDSP.

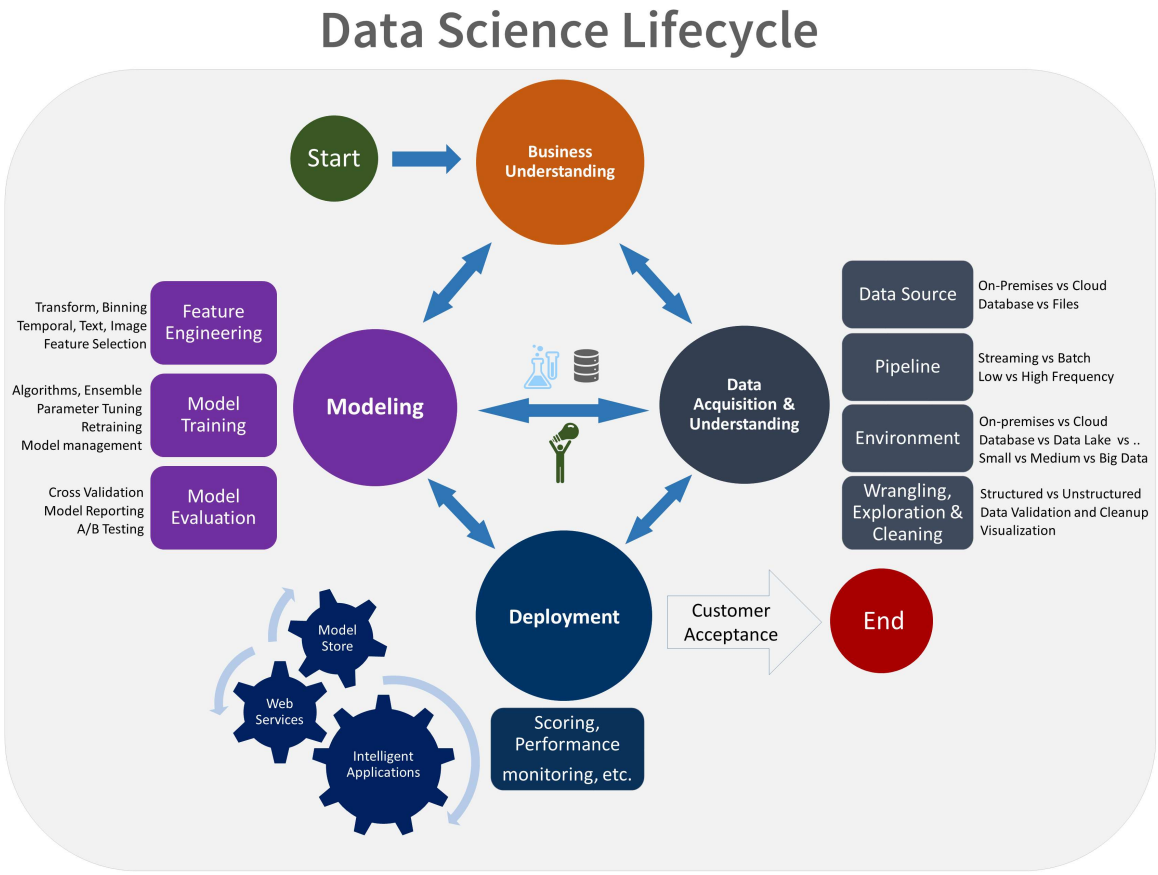
This lifecycle is designed for data-science projects that are intended to ship as part of intelligent applications. These applications deploy machine learning or artificial intelligence models for predictive analytics. Exploratory data-science projects and improvised analytics projects can also benefit from the use of this process. But for those projects, some of the steps described here might not be needed.

Five lifecycle stages

The TDSP lifecycle is composed of five major stages that are executed iteratively. These stages include:

1. Business understanding
2. Data acquisition and understanding
3. Modeling
4. Deployment
5. Customer acceptance

Here is a visual representation of the TDSP lifecycle:



The TDSP lifecycle is modeled as a sequence of iterated steps that provide guidance on the tasks needed to use predictive models. You deploy the predictive models in the production environment that you plan to use to build the intelligent applications. The goal of this process lifecycle is to continue to move a data-science project toward a clear engagement end point. Data science is an exercise in research and discovery. The ability to communicate tasks to your team and your customers by using a well-defined set of artifacts that employ standardized templates helps to avoid misunderstandings. Using these templates also increases the chance of the successful completion of a complex data-science project.

For each stage, we provide the following information:

- **Goals**: The specific objectives.
- **How to do it**: An outline of the specific tasks and guidance on how to complete them.
- **Artifacts**: The deliverables and the support to produce them.

Next steps

For examples of how to execute steps in TDSPs that use Azure Machine Learning, see Use the TDSP with Azure Machine Learning.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Related resources

- What is the Team Data Science Process?
- Compare the machine learning products and technologies from Microsoft
- Machine learning at scale

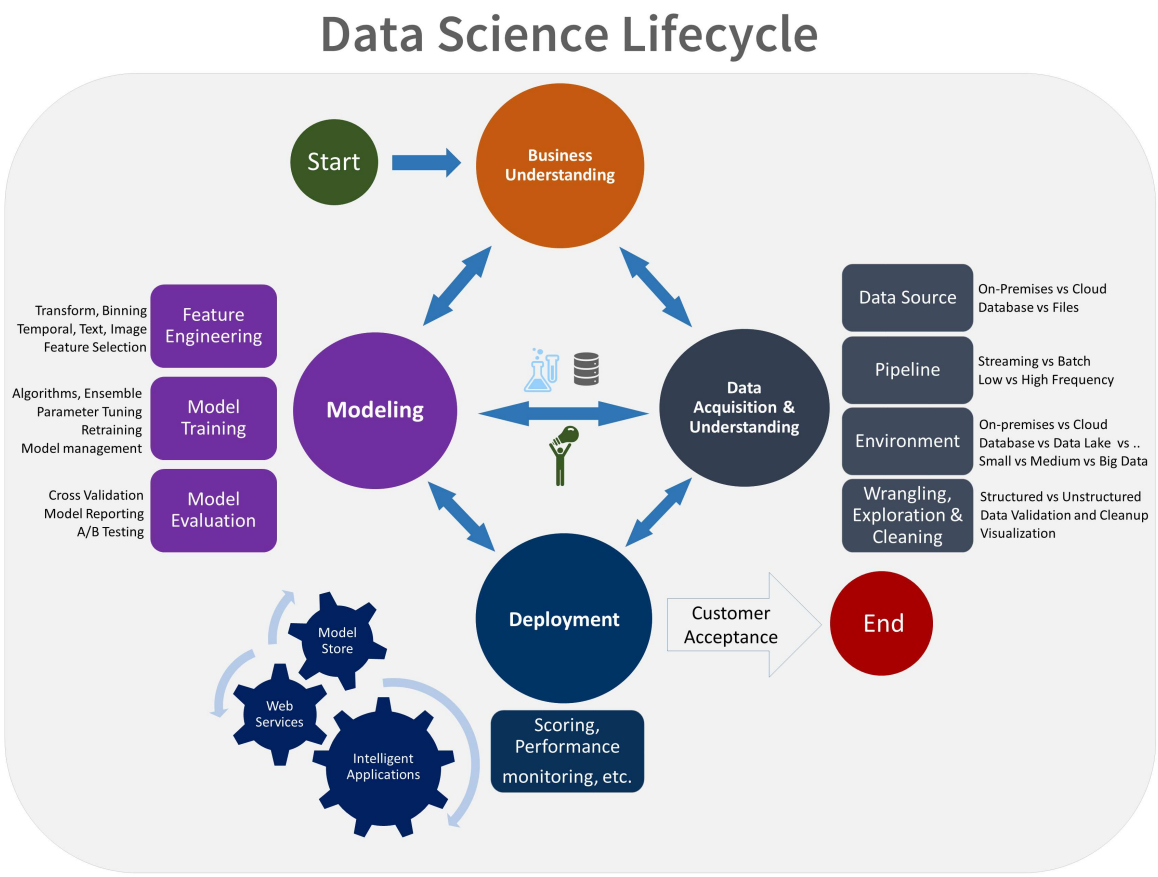
The business understanding stage of the Team Data Science Process lifecycle

Article • 11/15/2022

This article outlines the goals, tasks, and deliverables associated with the business understanding stage of the Team Data Science Process (TDSP). This process provides a recommended lifecycle that you can use to structure your data-science projects. The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. **Business understanding**
2. **Data acquisition and understanding**
3. **Modeling**
4. **Deployment**
5. **Customer acceptance**

Here is a visual representation of the TDSP lifecycle:



Goals

- Specify the key variables that are to serve as the model targets and whose related metrics are used determine the success of the project.
- Identify the relevant data sources that the business has access to or needs to obtain.

How to do it

There are two main tasks addressed in this stage:

- **Define objectives**: Work with your customer and other stakeholders to understand and identify the business problems. Formulate questions that define the business goals that the data science techniques can target.
- **Identify data sources**: Find the relevant data that helps you answer the questions that define the objectives of the project.

Define objectives

1. A central objective of this step is to identify the key business variables that the analysis needs to predict. We refer to these variables as the *model targets*, and we use the metrics associated with them to determine the success of the project. Two examples of such targets are sales forecasts or the probability of an order being fraudulent.

2. Define the project goals by asking and refining "sharp" questions that are relevant, specific, and unambiguous. Data science is a process that uses names and numbers to answer such questions. You typically use data science or machine learning to answer five types of questions:

    - How much or how many? (regression)
    - Which category? (classification)
    - Which group? (clustering)
    - Is this weird? (anomaly detection)
    - Which option should be taken? (recommendation)

    Determine which of these questions you're asking and how answering it achieves your business goals.

3. Define the project team by specifying the roles and responsibilities of its members. Develop a high-level milestone plan that you iterate on as you discover more information.

4. Define the success metrics. For example, you might want to achieve a customer churn prediction. You need an accuracy rate of "x" percent by the end of this three-month project. With this data, you can offer customer promotions to reduce churn. The metrics must be **SMART**:

    - **S**pecific
    - **M**easurable
    - **A**chievable
    - **R**elevant
    - **T**ime-bound

Identify data sources

Identify data sources that contain known examples of answers to your sharp questions. Look for the following data:

- Data that's relevant to the question. Do you have measures of the target and features that are related to the target?
- Data that's an accurate measure of your model target and the features of interest.

For example, you might find that the existing systems need to collect and log additional kinds of data to address the problem and achieve the project goals. In this situation, you might want to look for external data sources or update your systems to collect new data.

Artifacts

Here are the deliverables in this stage:

- Charter document: A standard template is provided in the TDSP project structure definition. The charter document is a living document. You update the template throughout the project as you make new discoveries and as business requirements change. The key is to iterate upon this document, adding more detail, as you progress through the discovery process. Keep the customer and other stakeholders involved in making the changes and clearly communicate the reasons for the changes to them.
- Data sources: The **Raw data sources** section of the **Data definitions** report that's found in the TDSP project **Data report** folder contains the data sources. This section specifies the original and destination locations for the raw data. In later stages, you fill in additional details like the scripts to move the data to your analytic environment.
- Data dictionaries: This document provides descriptions of the data that's provided by the client. These descriptions include information about the schema

(the data types and information on the validation rules, if any) and the entity-relation diagrams, if available.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to each step in the lifecycle of the TDSP:

1. Business understanding
2. Data acquisition and understanding
3. Modeling
4. Deployment
5. Customer acceptance

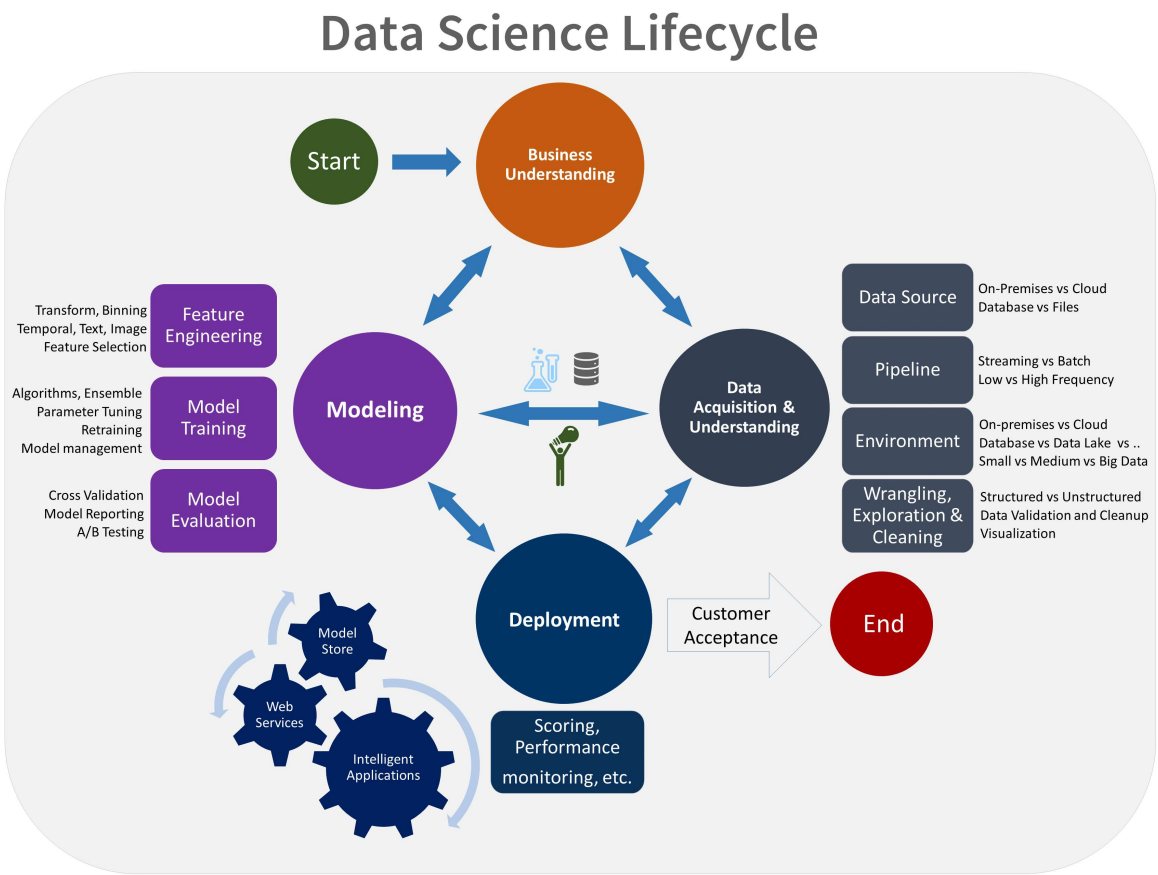
Data acquisition and understanding stage of the Team Data Science Process

Article • 11/15/2022

This article outlines the goals, tasks, and deliverables associated with the data acquisition and understanding stage of the Team Data Science Process (TDSP). This process provides a recommended lifecycle that you can use to structure your data-science projects. The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. **Business understanding**
2. **Data acquisition and understanding**
3. **Modeling**
4. **Deployment**
5. **Customer acceptance**

Here is a visual representation of the TDSP lifecycle:



Goals

- Produce a clean, high-quality data set whose relationship to the target variables is understood. Locate the data set in the appropriate analytics environment so you are ready to model.
- Develop a solution architecture of the data pipeline that refreshes and scores the data regularly.

How to do it

There are three main tasks addressed in this stage:

- **Ingest the data** into the target analytic environment.
- **Explore the data** to determine if the data quality is adequate to answer the question.
- **Set up a data pipeline** to score new or regularly refreshed data.

Ingest the data

Set up the process to move the data from the source locations to the target locations where you run analytics operations, like training and predictions. For technical details and options on how to move the data with various Azure data services, see Load data into storage environments for analytics.

Explore the data

Before you train your models, you need to develop a sound understanding of the data. Real-world data sets are often noisy, are missing values, or have a host of other discrepancies. You can use data summarization and visualization to audit the quality of your data and provide the information you need to process the data before it's ready for modeling. This process is often iterative. For guidance on cleaning the data, see Tasks to prepare data for enhanced machine learning.

After you're satisfied with the quality of the cleansed data, the next step is to better understand the patterns that are inherent in the data. This data analysis helps you choose and develop an appropriate predictive model for your target. Look for evidence for how well connected the data is to the target. Then determine whether there is sufficient data to move forward with the next modeling steps. Again, this process is often iterative. You might need to find new data sources with more accurate or more relevant data to augment the data set initially identified in the previous stage.

Set up a data pipeline

In addition to the initial ingestion and cleaning of the data, you typically need to set up a process to score new data or refresh the data regularly as part of an ongoing learning process. Scoring may be completed with a data pipeline or workflow. The Move data from a SQL Server instance to Azure SQL Database with Azure Data Factory article gives an example of how to set up a pipeline with Azure Data Factory.

In this stage, you develop a solution architecture of the data pipeline. You develop the pipeline in parallel with the next stage of the data science project. Depending on your business needs and the constraints of your existing systems into which this solution is being integrated, the pipeline can be one of the following options:

- Batch-based
- Streaming or real time
- A hybrid

Artifacts

The following are the deliverables in this stage:

- Data quality report: This report includes data summaries, the relationships between each attribute and target, variable ranking, and more.
- **Solution architecture**: The solution architecture can be a diagram or description of your data pipeline that you use to run scoring or predictions on new data after you have built a model. It also contains the pipeline to retrain your model based on new data. Store the document in the Project directory when you use the TDSP directory structure template.
- **Checkpoint decision**: Before you begin full-feature engineering and model building, you can reevaluate the project to determine whether the value expected is sufficient to continue pursuing it. You might, for example, be ready to proceed, need to collect more data, or abandon the project as the data does not exist to answer the question.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to each step in the lifecycle of the TDSP:

1. Business understanding
2. Data acquisition and understanding
3. Modeling
4. Deployment
5. Customer acceptance

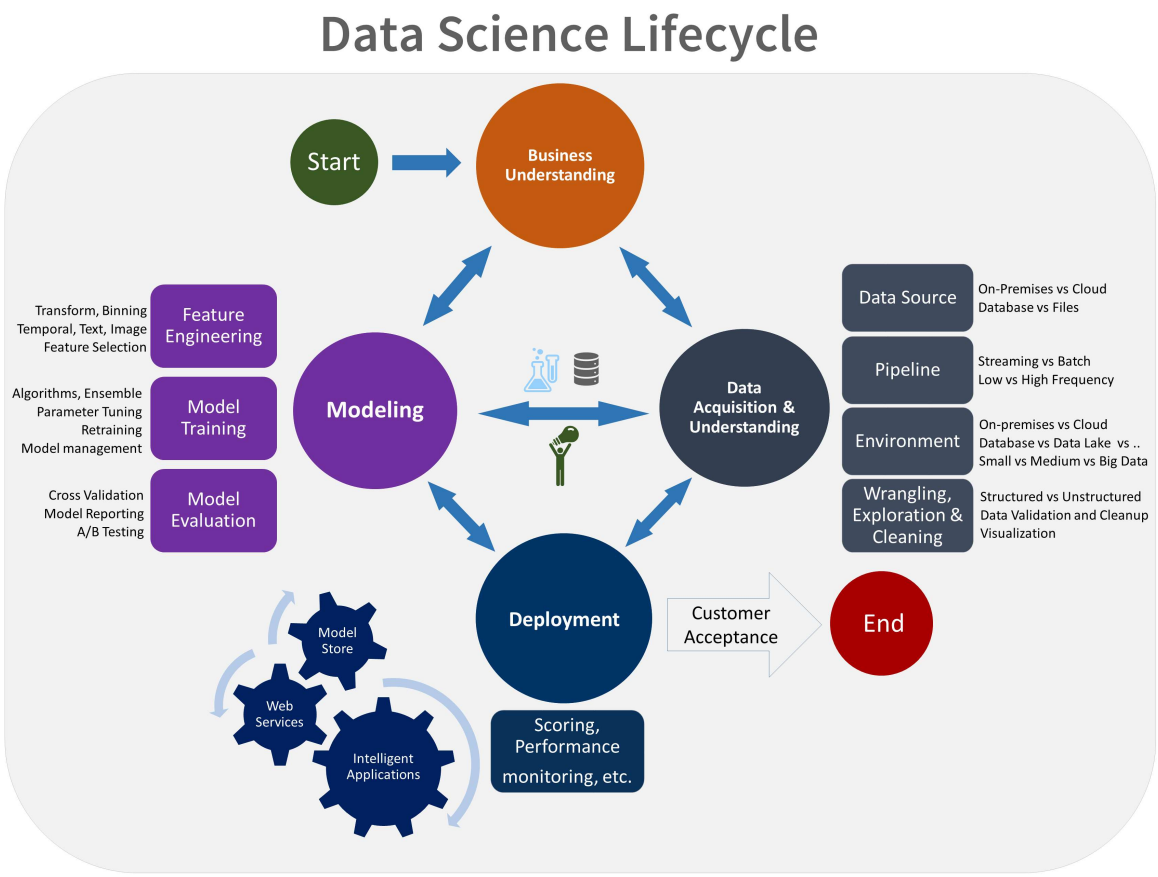
Modeling stage of the Team Data Science Process lifecycle

Article • 11/15/2022

This article outlines the goals, tasks, and deliverables associated with the modeling stage of the Team Data Science Process (TDSP). This process provides a recommended lifecycle that you can use to structure your data-science projects. The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. **Business understanding**
2. **Data acquisition and understanding**
3. **Modeling**
4. **Deployment**
5. **Customer acceptance**

Here is a visual representation of the TDSP lifecycle:



Goals

- Determine the optimal data features for the machine-learning model.

- Create an informative machine-learning model that predicts the target most accurately.
- Create a machine-learning model that's suitable for production.

How to do it

There are three main tasks addressed in this stage:

- **Feature engineering**: Create data features from the raw data to facilitate model training.
- **Model training**: Find the model that answers the question most accurately by comparing their success metrics.
- Determine if your model is **suitable for production.**

Feature engineering

Feature engineering involves the inclusion, aggregation, and transformation of raw variables to create the features used in the analysis. If you want insight into what is driving a model, then you need to understand how the features relate to each other and how the machine-learning algorithms are to use those features.

This step requires a creative combination of domain expertise and the insights obtained from the data exploration step. Feature engineering is a balancing act of finding and including informative variables, but at the same time trying to avoid too many unrelated variables. Informative variables improve your result; unrelated variables introduce unnecessary noise into the model. You also need to generate these features for any new data obtained during scoring. As a result, the generation of these features can only depend on data that's available at the time of scoring.

Model training

Depending on the type of question that you're trying to answer, there are many modeling algorithms available. For guidance on choosing a prebuilt algorithm with designer, see Machine Learning Algorithm Cheat Sheet for Azure Machine Learning designer; other algorithms are available through open-source packages in R or Python. Although this article focuses on Azure Machine Learning, the guidance it provides is useful for any machine-learning projects.

The process for model training includes the following steps:

- **Split the input data** randomly for modeling into a training data set and a test data set.
- **Build the models** by using the training data set.
- **Evaluate** the training and the test data set. Use a series of competing machine-learning algorithms along with the various associated tuning parameters (known as a *parameter sweep*) that are geared toward answering the question of interest with the current data.
- **Determine the "best" solution** to answer the question by comparing the success metrics between alternative methods.

See Train models with Azure Machine Learning for options on training models in Azure Machine Learning.

> ⓘ Note
>
> **Avoid leakage**: You can cause data leakage if you include data from outside the training data set that allows a model or machine-learning algorithm to make unrealistically good predictions. Leakage is a common reason why data scientists get nervous when they get predictive results that seem too good to be true. These dependencies can be hard to detect. To avoid leakage often requires iterating between building an analysis data set, creating a model, and evaluating the accuracy of the results.

Model Evaluation

After training, the data scientist focuses next on model evaluation.

- **Checkpoint decision**: Evaluate whether the model performs sufficiently for production. Some key questions to ask are:
  - Does the model answer the question with sufficient confidence given the test data?
  - Should you try any alternative approaches?
  - Should you collect additional data, do more feature engineering, or experiment with other algorithms?
- **Interpreting the Model**: Use the Azure Machine Learning Python SDK to perform the following tasks:
  - Explain the entire model behavior or individual predictions on your personal machine locally.
  - Enable interpretability techniques for engineered features.
  - Explain the behavior for the entire model and individual predictions in Azure.

    - Upload explanations to Azure Machine Learning Run History.
    - Use a visualization dashboard to interact with your model explanations, both in a Jupyter notebook and in the Azure Machine Learning workspace.
    - Deploy a scoring explainer alongside your model to observe explanations during inferencing.
- **Assessing Fairness**: The Fairlearn open-source Python package with Azure Machine Learning performs the following tasks:
    - Assess the fairness of your model predictions. This process will help you learn more about fairness in machine learning.
    - Upload, list, and download fairness assessment insights to/from Azure Machine Learning studio.
    - See the fairness assessment dashboard in Azure Machine Learning studio to interact with your model(s)' fairness insights.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to each step in the lifecycle of the TDSP:

1. Business understanding
2. Data acquisition and understanding
3. Modeling
4. Deployment
5. Customer acceptance

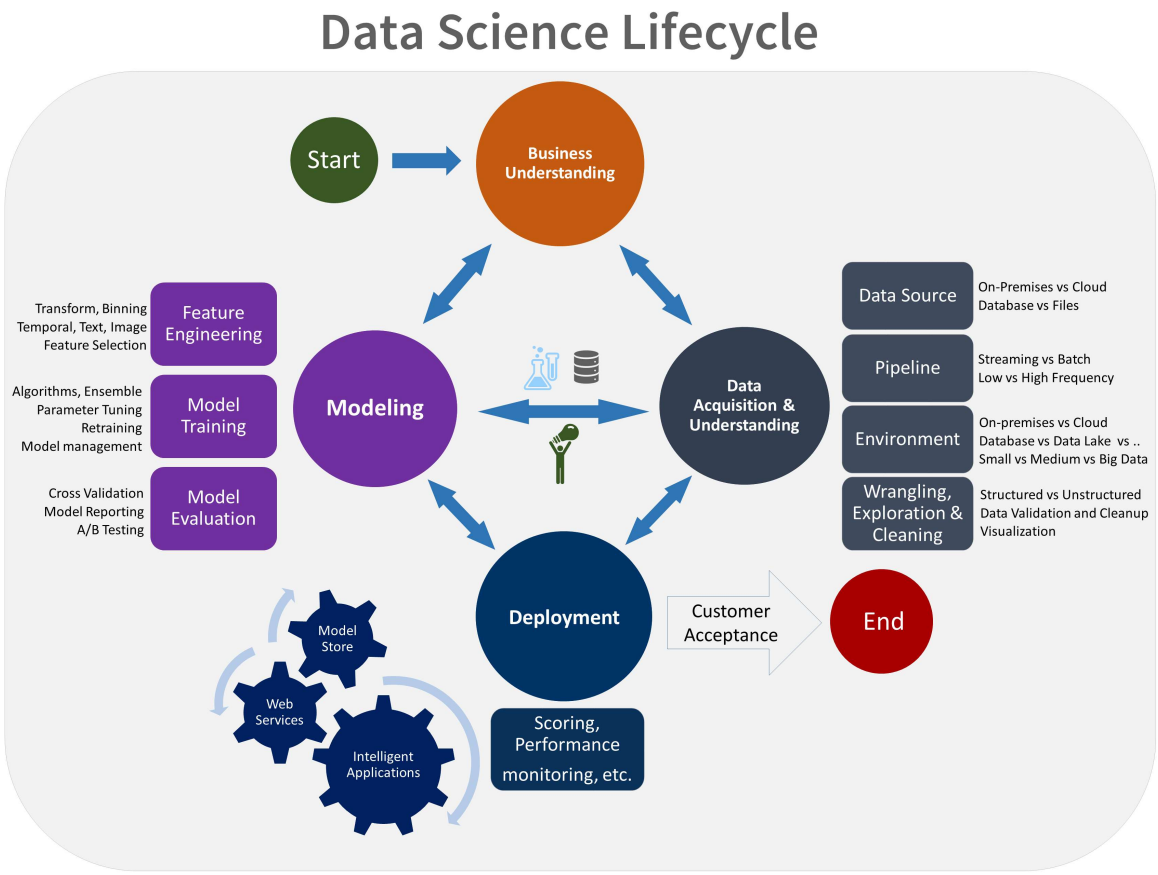
Deployment stage of the Team Data Science Process lifecycle

Article • 11/15/2022

This article outlines the goals, tasks, and deliverables associated with the deployment of the Team Data Science Process (TDSP). This process provides a recommended lifecycle that you can use to structure your data-science projects. The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. **Business understanding**
2. **Data acquisition and understanding**
3. **Modeling**
4. **Deployment**
5. **Customer acceptance**

Here is a visual representation of the TDSP lifecycle:



Goal

Deploy models with a data pipeline to a production or production-like environment for final user acceptance.

How to do it

The main task addressed in this stage:

Operationalize the model: Deploy the model and pipeline to a production or production-like environment for application consumption.

Operationalize a model

After you have a set of models that perform well, you can operationalize them for other applications to consume. Depending on the business requirements, predictions are made either in real time or on a batch basis. To deploy models, you expose them with an open API interface. The interface enables the model to be easily consumed from various applications, such as:

- Online websites
- Spreadsheets
- Dashboards
- Line-of-business applications
- Back-end applications

For examples of model operationalization with Azure Machine Learning, see Deploy machine learning models to Azure. It is a best practice to build telemetry and monitoring into the production model and the data pipeline that you deploy. This practice helps with subsequent system status reporting and troubleshooting.

Artifacts

- A status dashboard that displays the system health and key metrics
- A final modeling report with deployment details
- A final solution architecture document

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to each step in the lifecycle of the TDSP:

1. Business understanding
2. Data Acquisition and understanding
3. Modeling
4. Deployment
5. Customer acceptance

For Azure, we recommend applying TDSP using Azure Machine Learning: for an overview of Azure Machine Learning see What is Azure Machine Learning?.

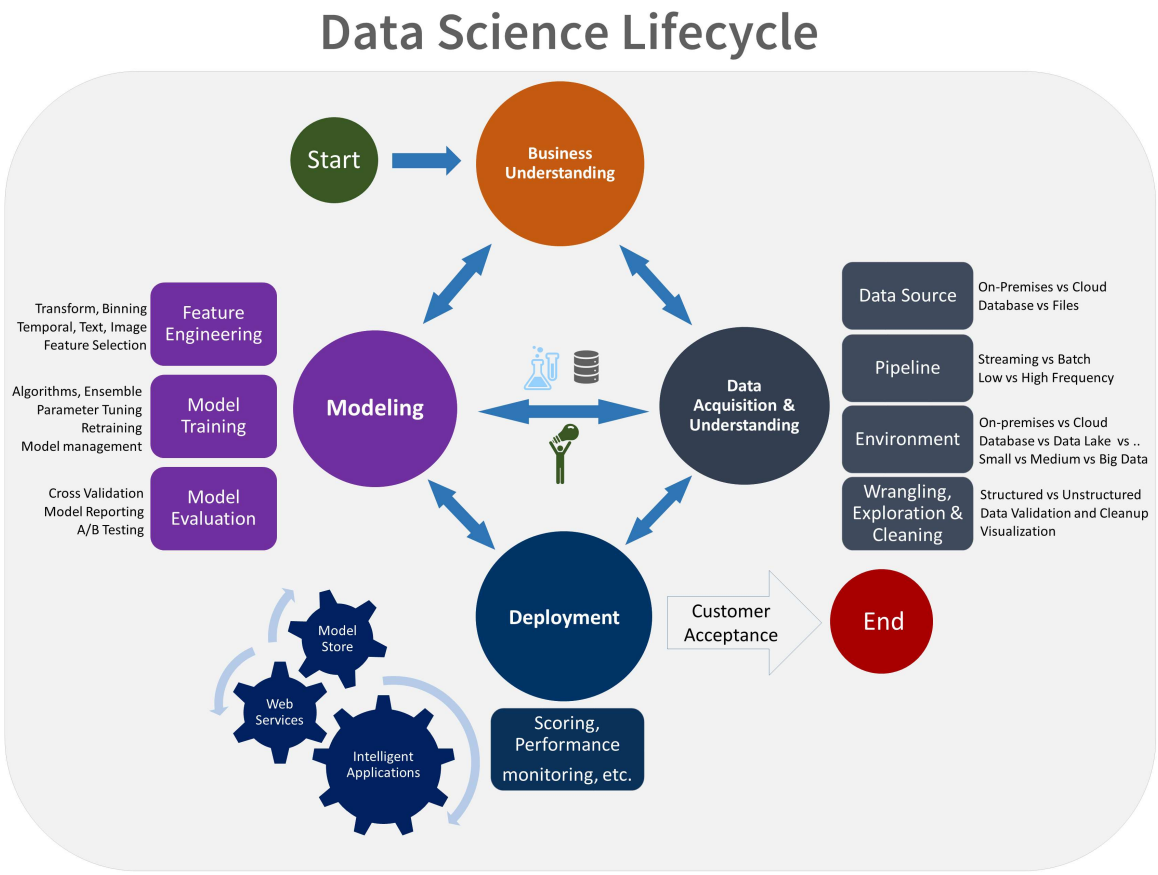
Customer acceptance stage of the Team Data Science Process lifecycle

Article • 11/15/2022

This article outlines the goals, tasks, and deliverables associated with the customer acceptance stage of the Team Data Science Process (TDSP). This process provides a recommended lifecycle that you can use to structure your data-science projects. The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. **Business understanding**
2. **Data acquisition and understanding**
3. **Modeling**
4. **Deployment**
5. **Customer acceptance**

Here is a visual representation of the TDSP lifecycle:



Goal

Finalize project deliverables: Confirm that the pipeline, the model, and their deployment in a production environment satisfy the customer's objectives.

How to do it

There are two main tasks addressed in this stage:

- **System validation**: Confirm that the deployed model and pipeline meet the customer's needs.
- **Project hand-off**: Hand the project off to the entity that's going to run the system in production.

The customer should validate that the system meets their business needs and that it answers the questions with acceptable accuracy to deploy the system to production for use by their client's application. All the documentation is finalized and reviewed. The project is handed-off to the entity responsible for operations. This entity might be, for example, an IT or customer data-science team or an agent of the customer that's responsible for running the system in production.

Artifacts

The main artifact produced in this final stage is the **Exit report of the project for the customer**. This technical report contains all the details of the project that are useful for learning about how to operate the system. TDSP provides an Exit report template. You can use the template as is, or you can customize it for specific client needs.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to each step in the lifecycle of the TDSP:

1. Business understanding
2. Data acquisition and understanding
3. Modeling
4. Deployment
5. Customer acceptance

For Azure, we recommend applying TDSP using Azure Machine Learning: for an overview of Azure Machine Learning see What is Azure Machine Learning?.

Team Data Science Process roles and tasks

Article • 12/06/2022

The Team Data Science Process (TDSP) is a framework developed by Microsoft that provides a structured methodology to efficiently build predictive analytics solutions and intelligent applications. This article outlines the key personnel roles and associated tasks for a data science team standardizing on this process.

This introductory article links to tutorials on how to set up the TDSP environment. The tutorials provide detailed guidance for using Azure DevOps Projects, Azure Repos repositories, and Azure Boards. The motivating goal is moving from concept through modeling and into deployment.

The tutorials use Azure DevOps because that is how to implement TDSP at Microsoft. Azure DevOps facilitates collaboration by integrating role-based security, work item management and tracking, and code hosting, sharing, and source control. The tutorials also use an Azure Data Science Virtual Machine (DSVM) as the analytics desktop, which has several popular data science tools pre-configured and integrated with Microsoft software and Azure services.

You can use the tutorials to implement TDSP using other code-hosting, agile planning, and development tools and environments, but some features may not be available.

Structure of data science groups and teams

Data science functions in enterprises are often organized in the following hierarchy:

- Data science group
  - Data science team/s within the group

In such a structure, there are group leads and team leads. Typically, a data science project is done by a data science team. Data science teams have project leads for project management and governance tasks, and individual data scientists and engineers to perform the data science and data engineering parts of the project. The initial project setup and governance is done by the group, team, or project leads.

Definition and tasks for the four TDSP roles

With the assumption that the data science unit consists of teams within a group, there are four distinct roles for TDSP personnel:

1. **Group Manager**: Manages the entire data science unit in an enterprise. A data science unit might have multiple teams, each of which is working on multiple data science projects in distinct business verticals. A Group Manager might delegate their tasks to a surrogate, but the tasks associated with the role do not change.

2. **Team Lead**: Manages a team in the data science unit of an enterprise. A team consists of multiple data scientists. For a small data science unit, the Group Manager and the Team Lead might be the same person.

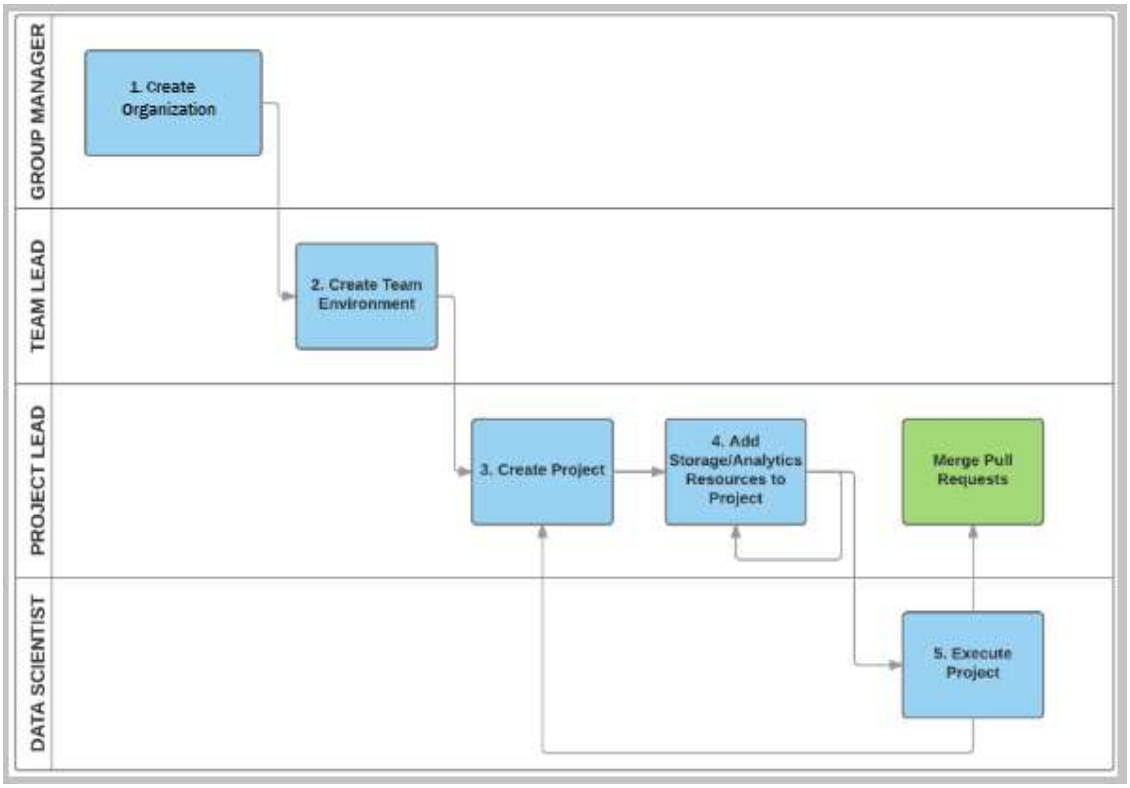
3. **Project Lead**: Manages the daily activities of individual data scientists on a specific data science project.

4. **Project Individual Contributors**: Data Scientists, Business Analysts, Data Engineers, Architects, and others who execute a data science project.

> ⓘ Note
>
> Depending on the structure and size of an enterprise, a single person may play more than one role, or more than one person may fill a role.

Tasks to be completed by the four roles

The following diagram shows the top-level tasks for each Team Data Science Process role. This schema and the following, more detailed outline of tasks for each TDSP role can help you choose the tutorial you need based on your responsibilities.



Group Manager tasks

The Group Manager or a designated TDSP system administrator completes the following tasks to adopt the TDSP:

- Creates an Azure DevOps **organization** and a group project within the organization.
- Creates a **project template repository** in the Azure DevOps group project, and seeds it from the project template repository developed by the Microsoft TDSP team. The Microsoft TDSP project template repository provides:
  - A **standardized directory structure**, including directories for data, code, and documents.
  - A set of **standardized document templates** to guide an efficient data science process.
- Creates a **utility repository**, and seeds it from the utility repository developed by the Microsoft TDSP team. The TDSP utility repository from Microsoft provides a set of useful utilities to make the work of a data scientist more efficient. The Microsoft utility repository includes utilities for interactive data exploration, analysis, reporting, and baseline modeling and reporting.
- Sets up the **security control policy** for the organization account.

For detailed instructions, see Group Manager tasks for a data science team.

Team Lead tasks

The Team Lead or a designated project administrator completes the following tasks to adopt the TDSP:

- Creates a team **project** in the group's Azure DevOps organization.
- Creates the **project template repository** in the project, and seeds it from the group project template repository set up by the Group Manager or delegate.
- Creates the **team utility repository**, seeds it from the group utility repository, and adds team-specific utilities to the repository.
- Optionally creates Azure file storage to store useful data assets for the team. Other team members can mount this shared cloud file store on their analytics desktops.
- Optionally mounts the Azure file storage on the team's **DSVM** and adds team data assets to it.
- Sets up **security control** by adding team members and configuring their permissions.

For detailed instructions, see Team Lead tasks for a data science team.

Project Lead tasks

The Project Lead completes the following tasks to adopt the TDSP:

- Creates a **project repository** in the team project, and seeds it from the project template repository.
- Optionally creates **Azure file storage** to store the project's data assets.
- Optionally mounts the Azure file storage to the **DSVM** and adds project data assets to it.
- Sets up **security control** by adding project members and configuring their permissions.

For detailed instructions, see Project Lead tasks for a data science team.

Project Individual Contributor tasks

The Project Individual Contributor, usually a Data Scientist, conducts the following tasks using the TDSP:

- Clones the **project repository** set up by the project lead.
- Optionally mounts the shared team and project **Azure file storage** on their **Data Science Virtual Machine** (DSVM).

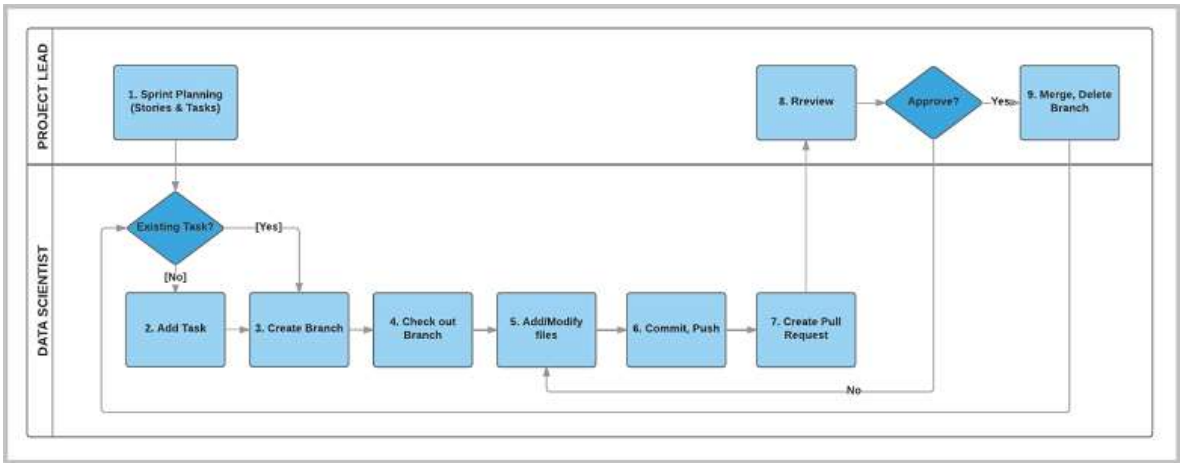
- Executes the project.

For detailed instructions for onboarding onto a project, see Project Individual Contributor tasks for a data science team.

Data science project execution workflow

By following the relevant tutorials, data scientists, project leads, and team leads can create work items to track all tasks and stages for project from beginning to end. Using Azure Repos promotes collaboration among data scientists and ensures that the artifacts generated during project execution are version controlled and shared by all project members. Azure DevOps lets you link your Azure Boards work items with your Azure Repos repository branches and easily track what has been done for a work item.

The following figure outlines the TDSP workflow for project execution:



The workflow steps can be grouped into three activities:

- Project Leads conduct sprint planning
- Data Scientists develop artifacts on `git` branches to address work items
- Project Leads or other team members do code reviews and merge working branches to the primary branch

For detailed instructions on project execution workflow, see Agile development of data science projects.

TDSP project template repository

Use the Microsoft TDSP team's project template repository to support efficient project execution and collaboration. The repository gives you a standardized directory structure and document templates you can use for your own TDSP projects.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Explore more detailed descriptions of the roles and tasks defined by the Team Data Science Process:

- Group Manager tasks for a data science team
- Team Lead tasks for a data science team
- Project Lead tasks for a data science team
- Project Individual Contributor tasks for a data science team

Related resources

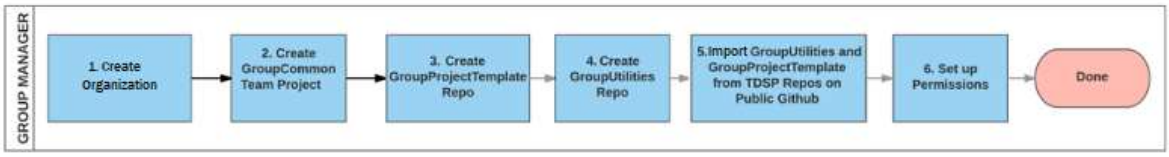
- Team Data Science Process group manager tasks
- Tasks for the team lead on a Team Data Science Process team
- Project lead tasks in the Team Data Science Process
- Tasks for an individual contributor in the Team Data Science Process

Team Data Science Process group manager tasks

Article • 11/15/2022

This article describes the tasks that a *group manager* completes for a data science organization. The group manager manages the entire data science unit in an enterprise. A data science unit may have several teams, each of which is working on many data science projects in distinct business verticals. The group manager's objective is to establish a collaborative group environment that standardizes on the Team Data Science Process (TDSP). For an outline of all the personnel roles and associated tasks handled by a data science team standardizing on the TDSP, see Team Data Science Process roles and tasks.

The following diagram shows the six main group manager setup tasks. Group managers may delegate their tasks to surrogates, but the tasks associated with the role don't change.



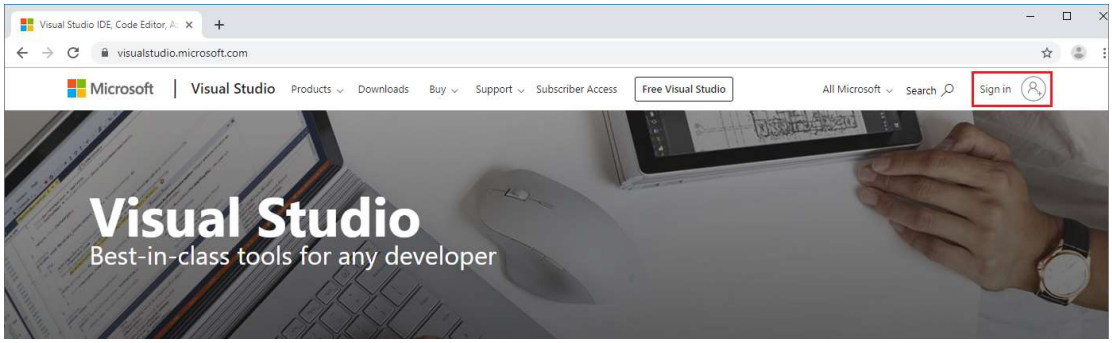
1. Set up an **Azure DevOps organization** for the group.
2. Create the default **GroupCommon project** in the Azure DevOps organization.
3. Create the **GroupProjectTemplate** repository in Azure Repos.
4. Create the **GroupUtilities** repository in Azure Repos.
5. Import the contents of the Microsoft TDSP team's **ProjectTemplate** and **Utilities** repositories into the group common repositories.
6. Set up **membership** and **permissions** for team members to access the group.

The following tutorial walks through the steps in detail.

> **Note**
>
> This article uses Azure DevOps to set up a TDSP group environment, because that is how to implement TDSP at Microsoft. If your group uses other code hosting or development platforms, the Group Manager's tasks are the same, but the way to complete them may be different.

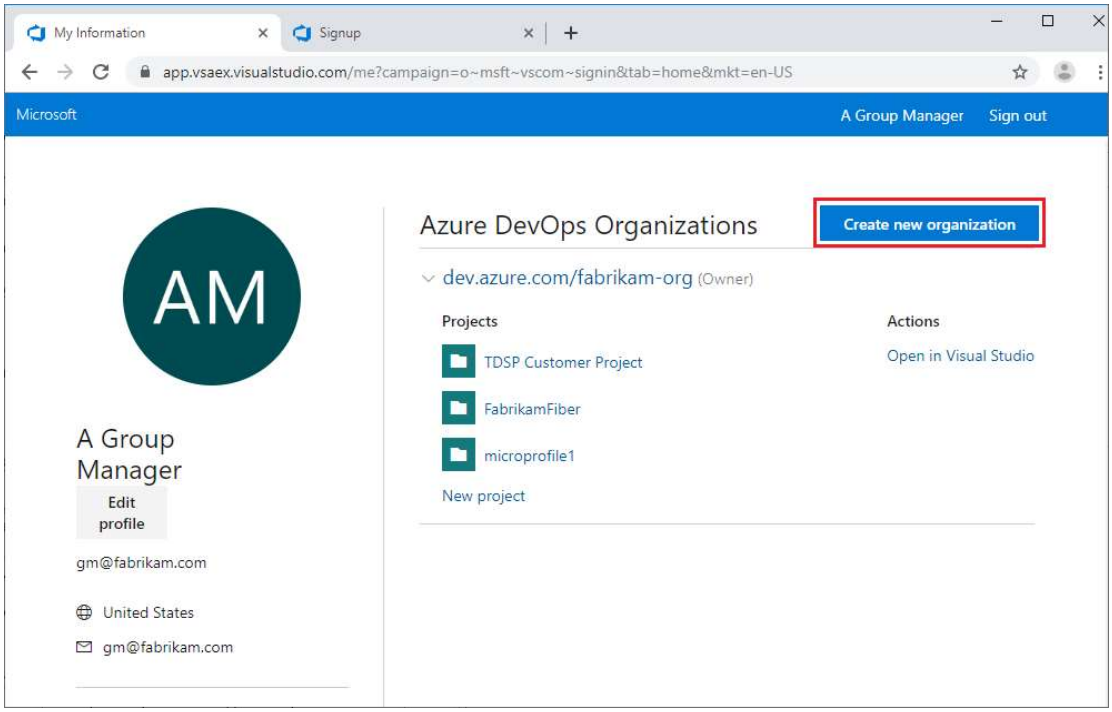
Create an organization and project in Azure DevOps

1. Go to visualstudio.microsoft.com, select **Sign in** at upper right, and sign into your Microsoft account.



   If you don't have a Microsoft account, select **Sign up now**, create a Microsoft account, and sign in using this account. If your organization has a Visual Studio subscription, sign in with the credentials for that subscription.

2. After you sign in, at upper right on the Azure DevOps page, select **Create new organization**.

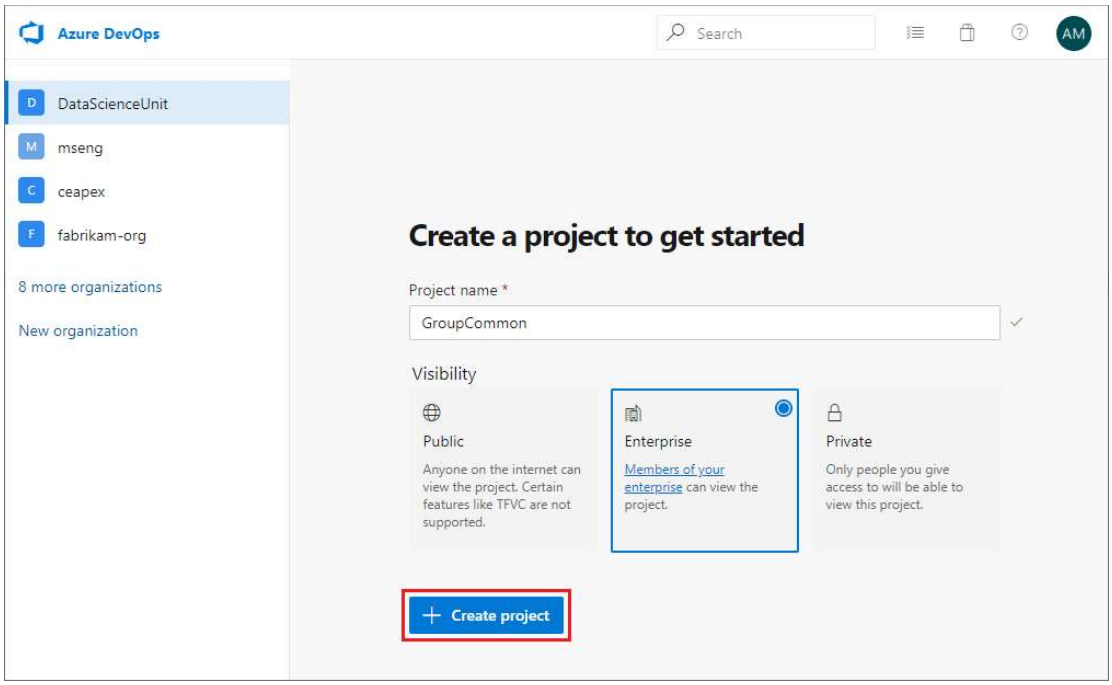


3. If you're prompted to agree to the Terms of Service, Privacy Statement, and Code of Conduct, select **Continue**.

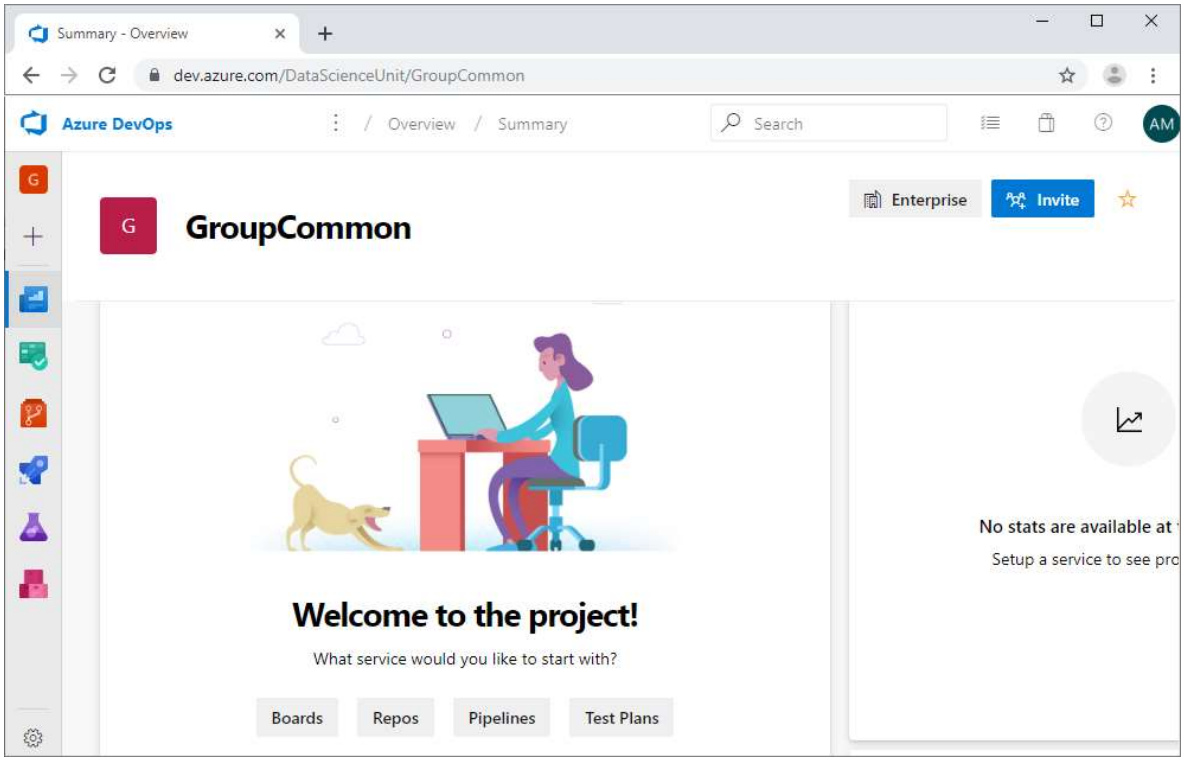
4. In the signup dialog, name your Azure DevOps organization and accept the host region assignment, or drop down and select a different region. Then select

Continue.

5. Under **Create a project to get started**, enter *GroupCommon*, and then select **Create project**.



The **GroupCommon** project **Summary** page opens. The page URL is *https://<servername>/<organization-name>/GroupCommon*.



Set up the group common repositories

Azure Repos hosts the following types of repositories for your group:

- **Group common repositories**: General-purpose repositories that multiple teams within a data science unit can adopt for many data science projects.
- **Team repositories**: Repositories for specific teams within a data science unit. These repositories are specific for a team's needs, and may be used for multiple projects within that team, but are not general enough to be used across multiple teams within a data science unit.
- **Project repositories**: Repositories for specific projects. Such repositories may not be general enough for multiple projects within a team, or for other teams in a data science unit.

To set up the group common repositories in your project, you:

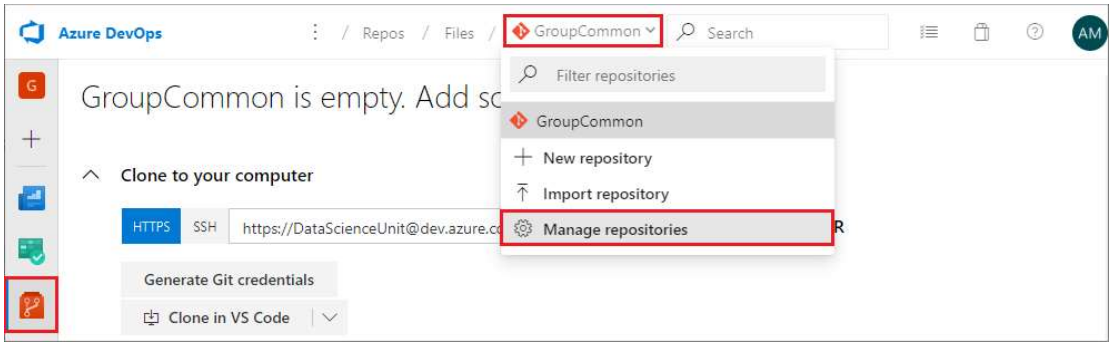
- Rename the default **GroupCommon** repository to **GroupProjectTemplate**
- Create a new **GroupUtilities** repository

Rename the default project repository to GroupProjectTemplate

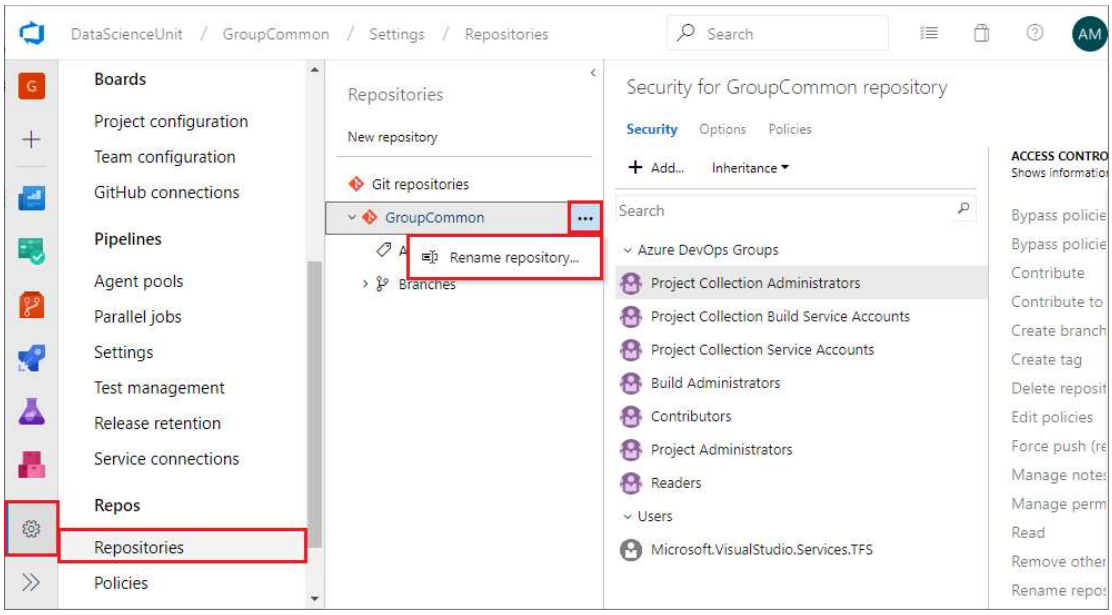
To rename the default **GroupCommon** project repository to **GroupProjectTemplate**:

1. On the **GroupCommon** project **Summary** page, select **Repos**. This action takes you to the default **GroupCommon** repository of the GroupCommon project, which is currently empty.

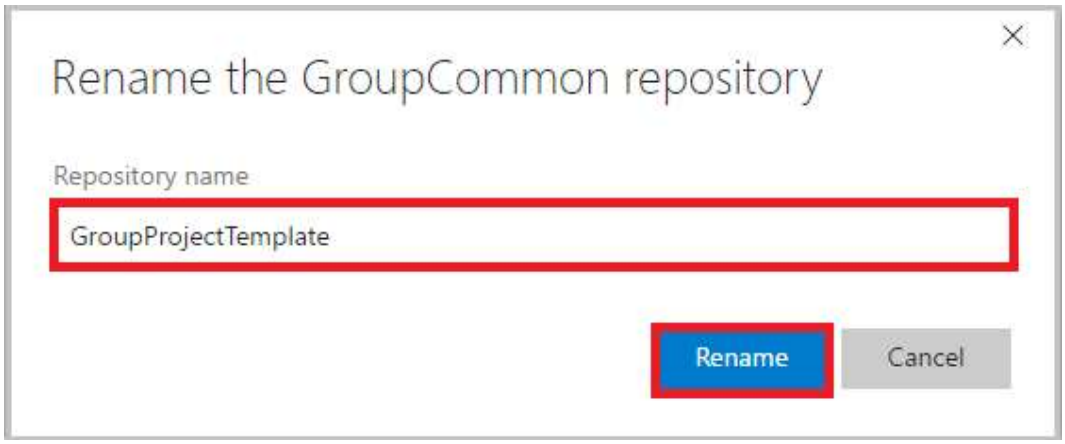
2. At the top of the page, drop down the arrow next to **GroupCommon** and select **Manage repositories**.



3. On the **Project Settings** page, select the **...** next to **GroupCommon**, and then select **Rename repository**.



4. In the **Rename the GroupCommon repository** popup, enter *GroupProjectTemplate*, and then select **Rename**.

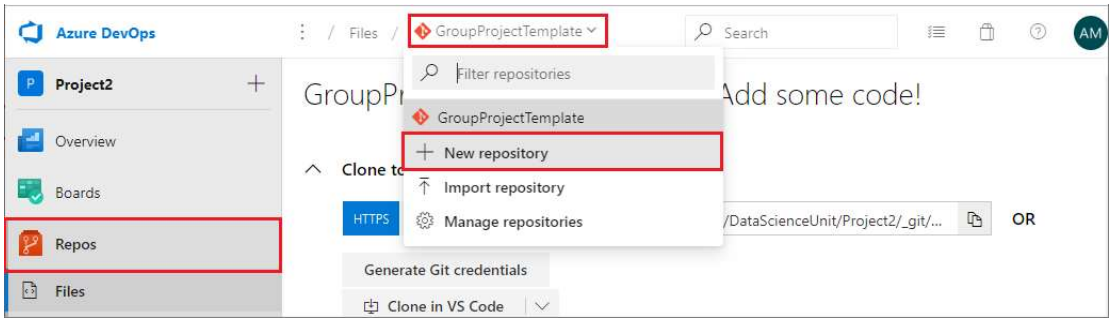


Create the GroupUtilities repository

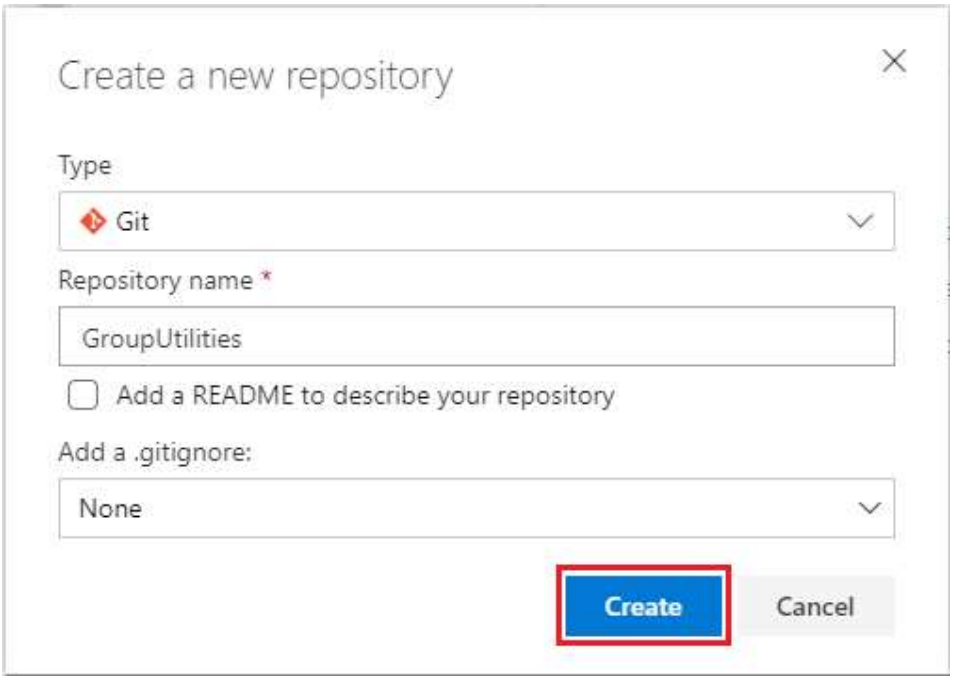
To create the **GroupUtilities** repository:

1. On the **GroupCommon** project **Summary** page, select **Repos**.

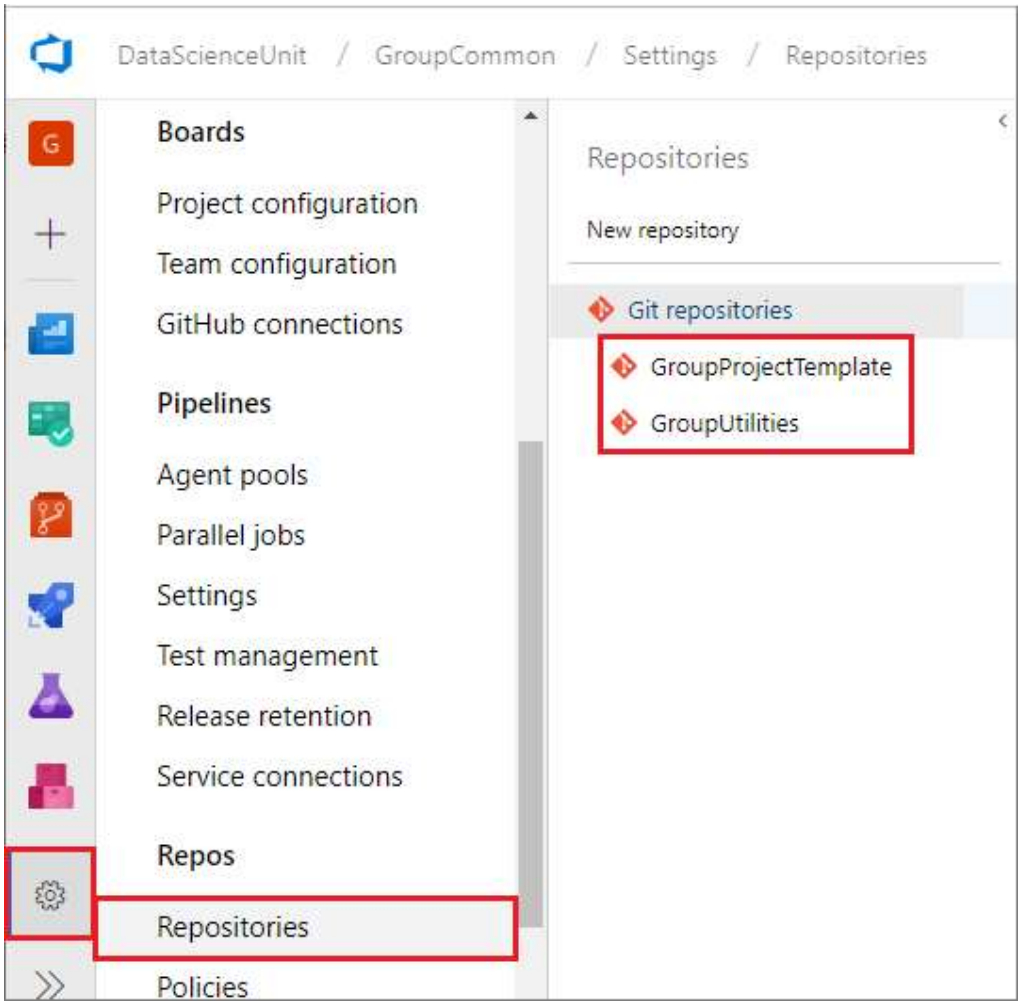
2. At the top of the page, drop down the arrow next to **GroupProjectTemplate** and select **New repository**.



3. In the **Create a new repository** dialog, select **Git** as the **Type**, enter *GroupUtilities* as the **Repository name**, and then select **Create**.



4. On the **Project Settings** page, select **Repositories** under **Repos** in the left navigation to see the two group repositories: **GroupProjectTemplate** and **GroupUtilities**.



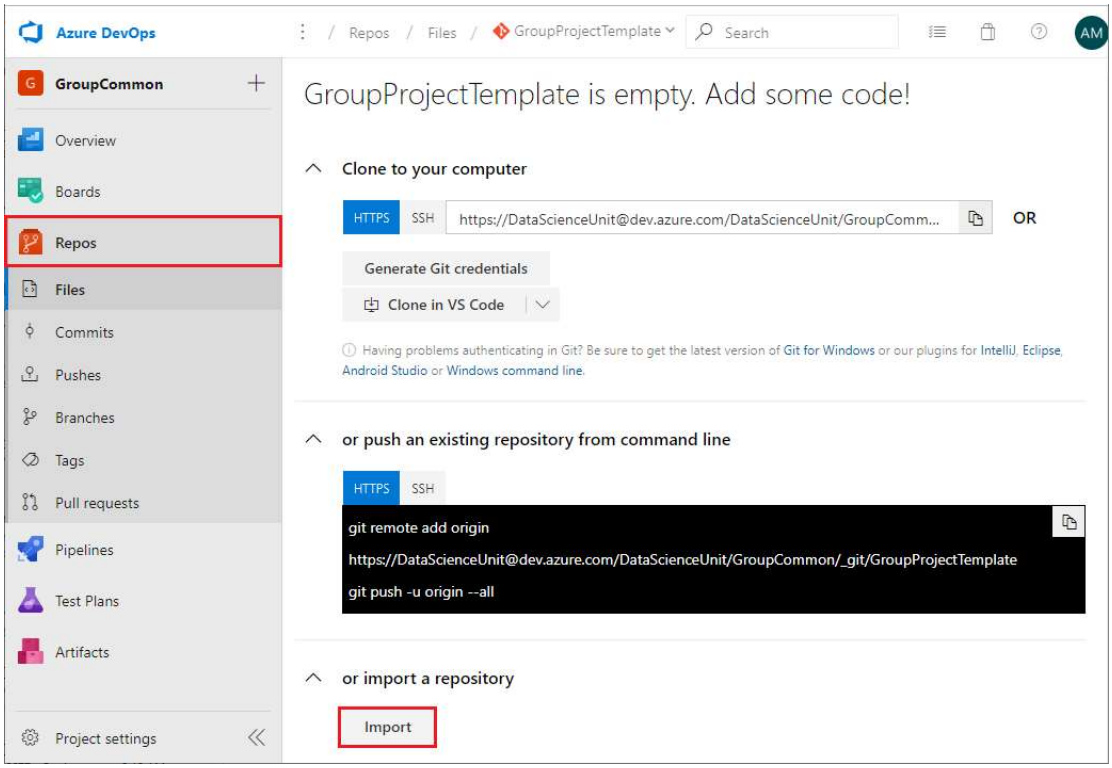
Import the Microsoft TDSP team repositories

In this part of the tutorial, you import the contents of the **ProjectTemplate** and **Utilities** repositories managed by the Microsoft TDSP team into your **GroupProjectTemplate** and **GroupUtilities** repositories.

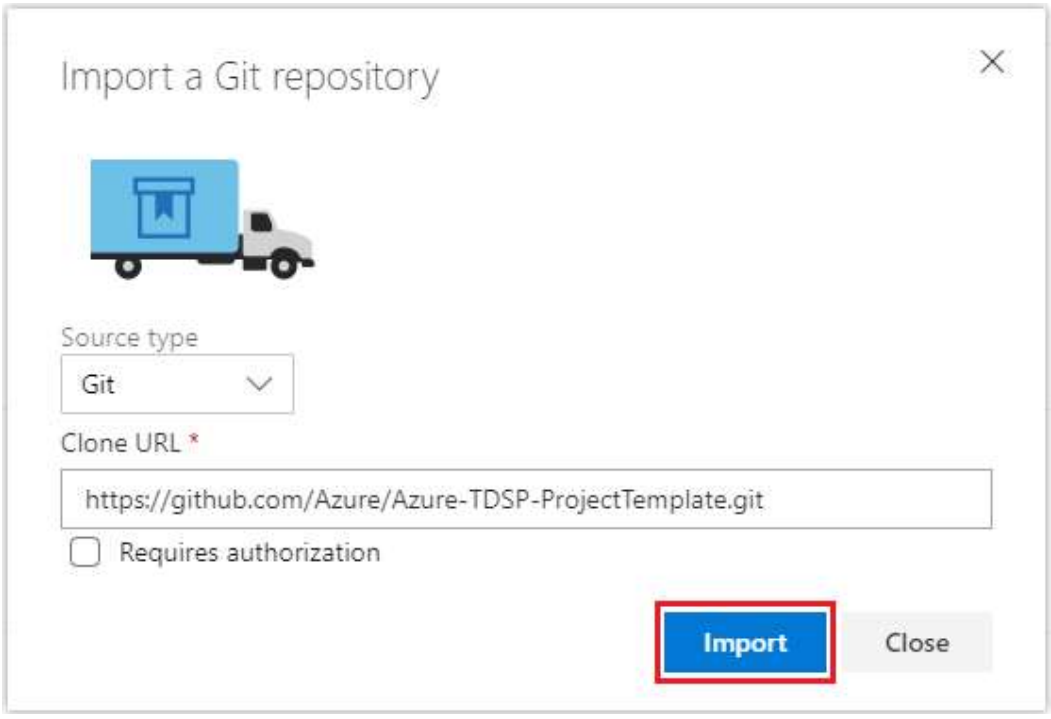
To import the TDSP team repositories:

1. From the **GroupCommon** project home page, select **Repos** in the left navigation. The default **GroupProjectTemplate** repo opens.

2. On the **GroupProjectTemplate is empty** page, select **Import**.



3. In the **Import a Git repository** dialog, select **Git** as the **Source type**, and enter *https://github.com/Azure/Azure-TDSP-ProjectTemplate.git* for the **Clone URL**. Then select **Import**. The contents of the Microsoft TDSP team ProjectTemplate repository are imported into your GroupProjectTemplate repository.



4. At the top of the **Repos** page, drop down and select the **GroupUtilities** repository.

Each of your two group repositories now contains all the files, except those in the *.git* directory, from the Microsoft TDSP team's corresponding repository.

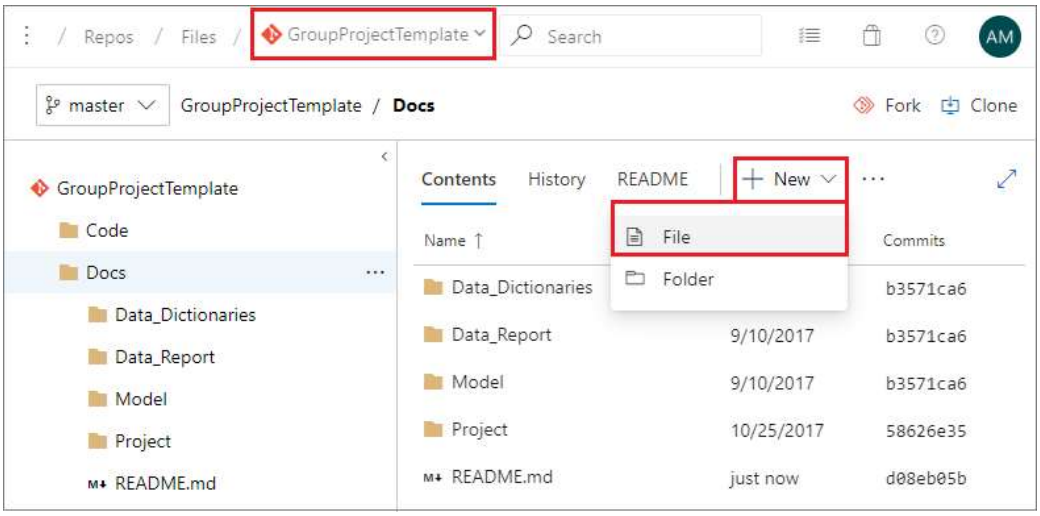
Customize the contents of the group repositories

If you want to customize the contents of your group repositories to meet the specific needs of your group, you can do that now. You can modify the files, change the directory structure, or add files that your group has developed or that are helpful for your group.

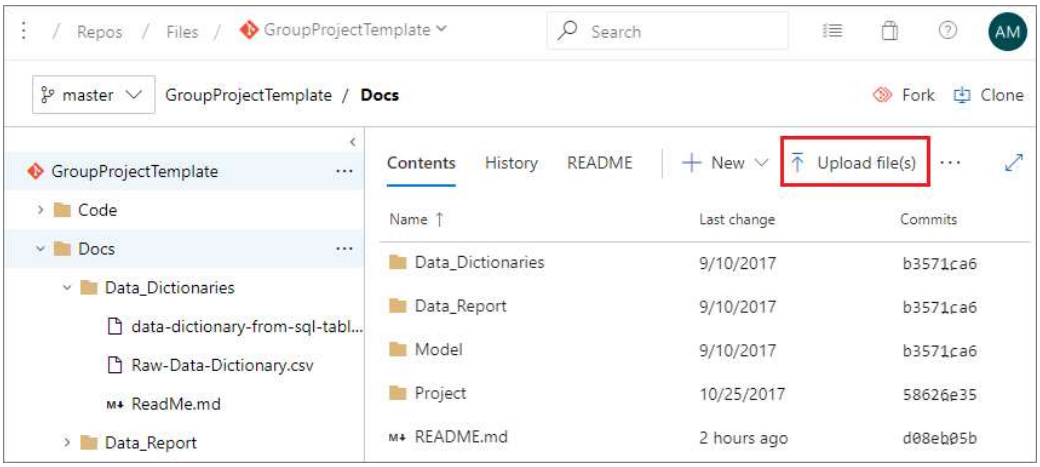
Make changes in Azure Repos

To customize repository contents:

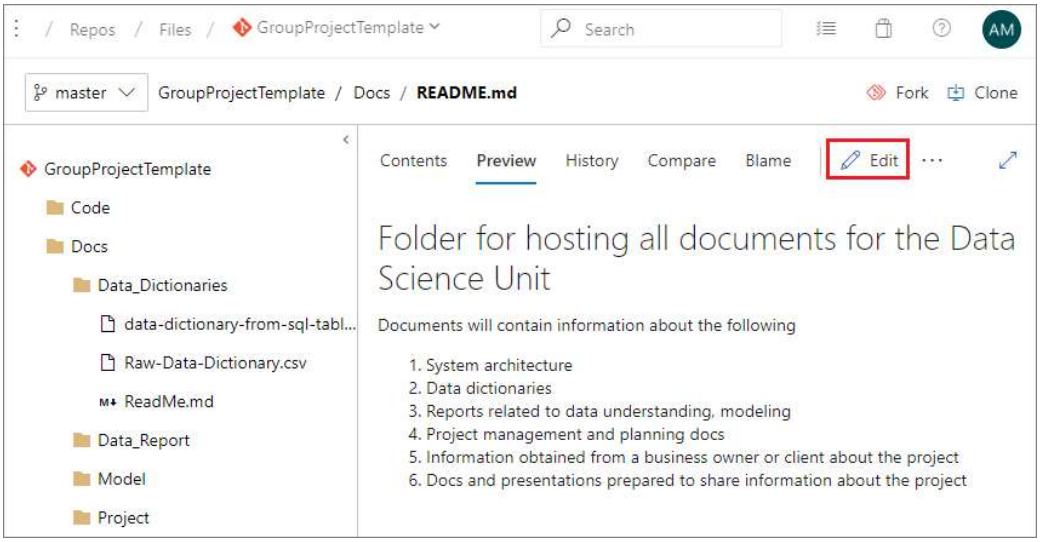
1. On the **GroupCommon** project **Summary** page, select **Repos**.
2. At the top of the page, select the repository you want to customize.
3. In the repo directory structure, navigate to the folder or file you want to change.
    - To create new folders or files, select the arrow next to **New**.



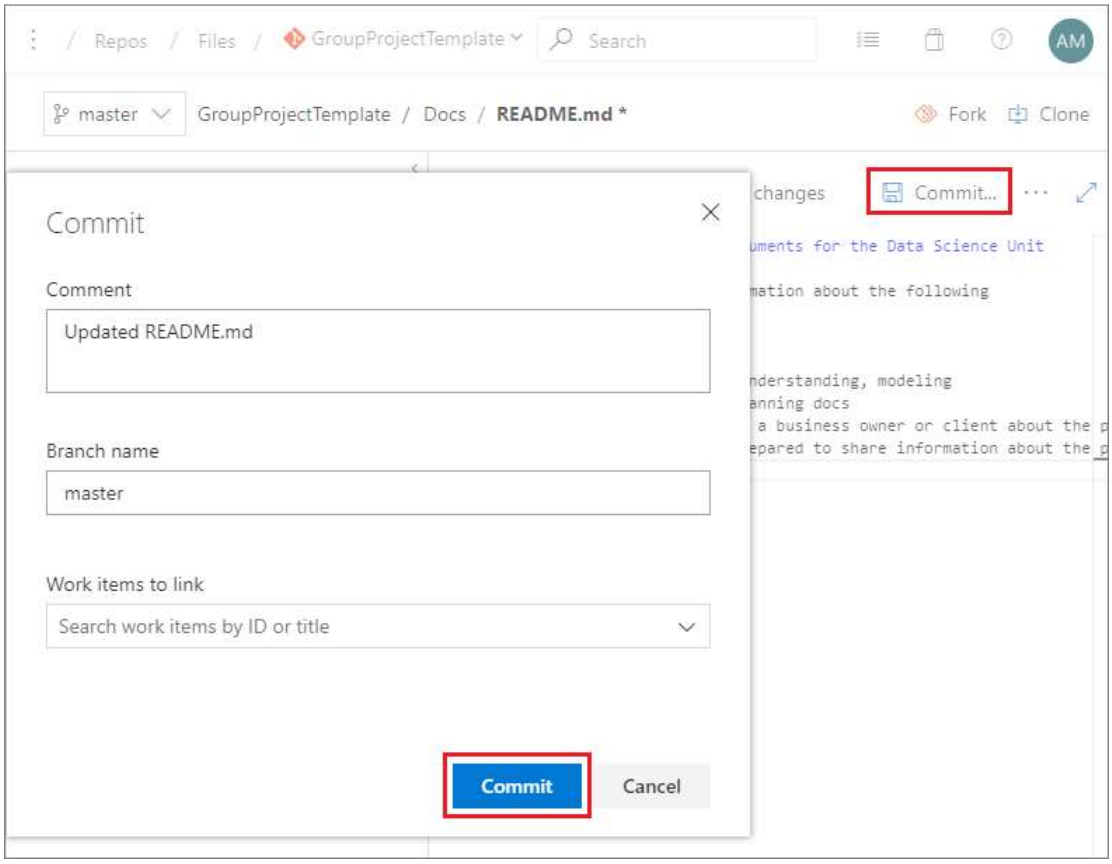
    - To upload files, select **Upload file(s)**.



- To edit existing files, navigate to the file and then select **Edit**.



4. After adding or editing files, select **Commit**.



Make changes using your local machine or DSVM

If you want to make changes using your local machine or DSVM and push the changes up to the group repositories, make sure you have the prerequisites for working with Git and DSVMs:

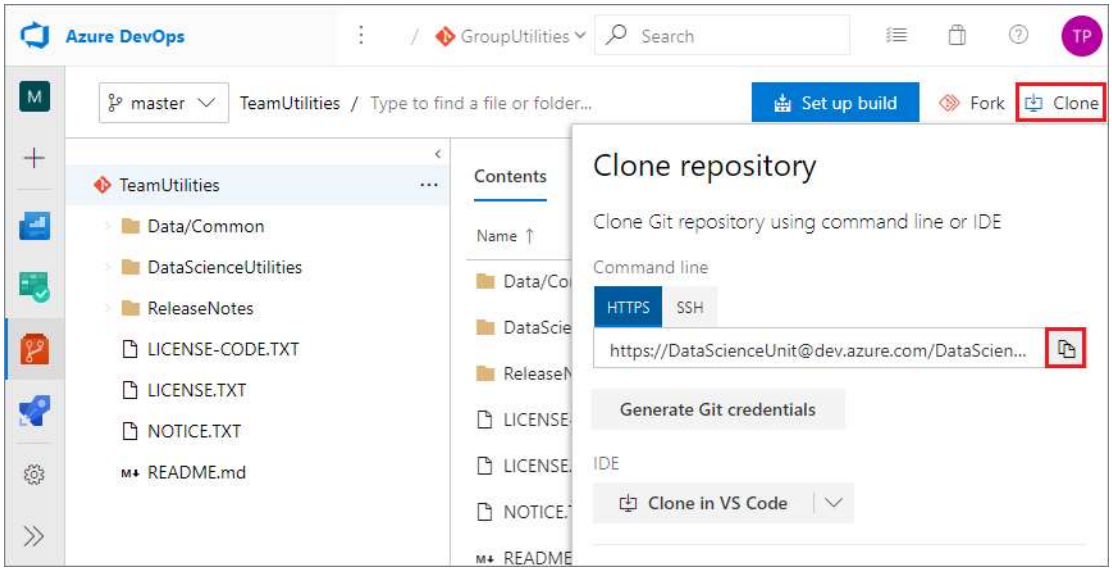
- An Azure subscription, if you want to create a DSVM.
- Git installed on your machine. If you're using a DSVM, Git is pre-installed. Otherwise, see the Platforms and tools appendix.
- If you want to use a DSVM, the Windows or Linux DSVM created and configured in Azure. For more information and instructions, see the Data Science Virtual Machine Documentation.
- For a Windows DSVM, Git Credential Manager (GCM) installed on your machine. In the *README.md* file, scroll down to the **Download and Install** section and select the **latest installer**. Download the *.exe* installer from the installer page and run it.
- For a Linux DSVM, an SSH public key set up on your DSVM and added in Azure DevOps. For more information and instructions, see the **Create SSH public key** section in the Platforms and tools appendix.

First, copy or *clone* the repository to your local machine.

1. On the **GroupCommon** project **Summary** page, select **Repos**, and at the top of the page, select the repository you want to clone.

2. On the repo page, select **Clone** at upper right.

3. In the **Clone repository** dialog, select **HTTPS** for an HTTP connection, or **SSH** for an SSH connection, and copy the clone URL under **Command line** to your clipboard.



4. On your local machine, create the following directories:

   - For Windows: **C:\GitRepos\GroupCommon**
   - For Linux, **$/GitRepos/GroupCommon** on your home directory

5. Change to the directory you created.

6. In Git Bash, run the command `git clone <clone URL>.`

   For example, either of the following commands clones the **GroupUtilities** repository to the *GroupCommon* directory on your local machine.

   HTTPS connection:

   ```
   Bash

   git clone
   https://DataScienceUnit@dev.azure.com/DataScienceUnit/GroupCommon/_git/
   GroupUtilities
   ```

   SSH connection:

   ```
   Bash

   git clone
   git@ssh.dev.azure.com:v3/DataScienceUnit/GroupCommon/GroupUtilities
   ```

After making whatever changes you want in the local clone of your repository, you can push the changes to the shared group common repositories.

Run the following Git Bash commands from your local **GroupProjectTemplate** or **GroupUtilities** directory.

```
Bash

git add .
git commit -m "push from local"
git push
```

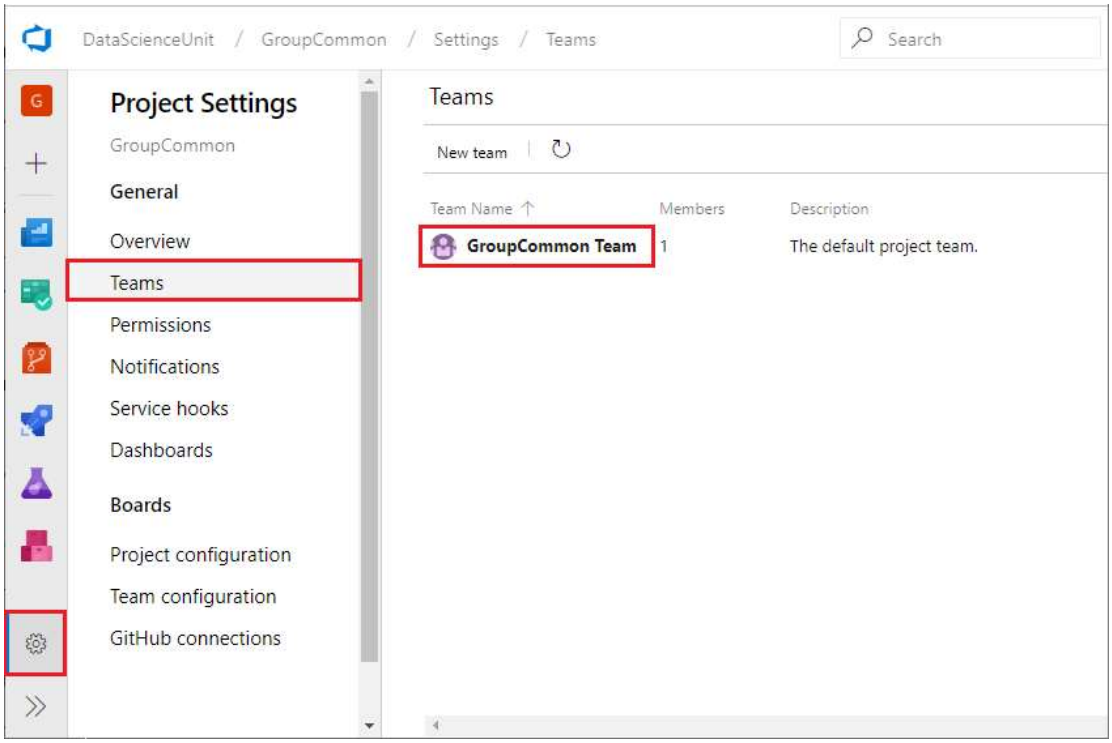
> **Note**
>
> If this is the first time you commit to a Git repository, you may need to configure global parameters *user.name* and *user.email* before you run the `git commit` command. Run the following two commands:
>
> `git config --global user.name <your name>`
>
> `git config --global user.email <your email address>`
>
> If you're committing to several Git repositories, use the same name and email address for all of them. Using the same name and email address is convenient when building Power BI dashboards to track your Git activities in multiple repositories.

Add group members and configure permissions

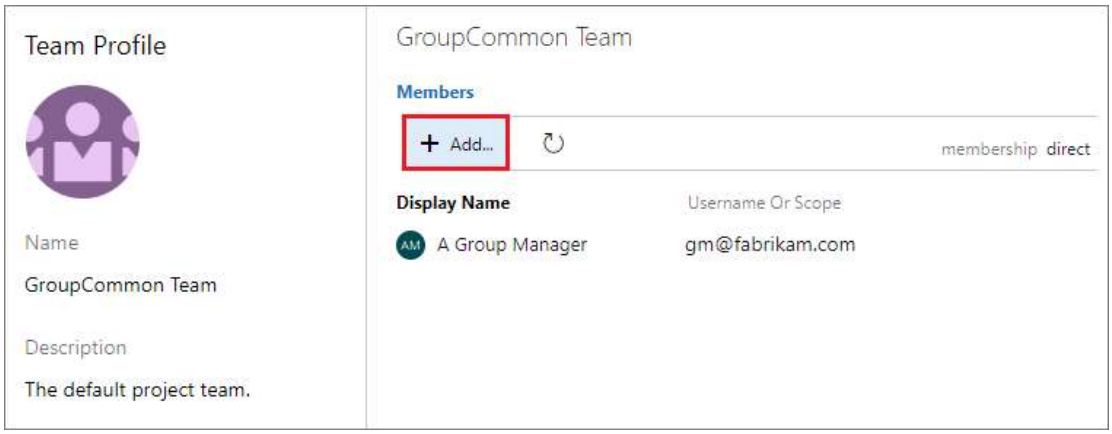
To add members to the group:

1. In Azure DevOps, from the **GroupCommon** project home page, select **Project settings** from the left navigation.

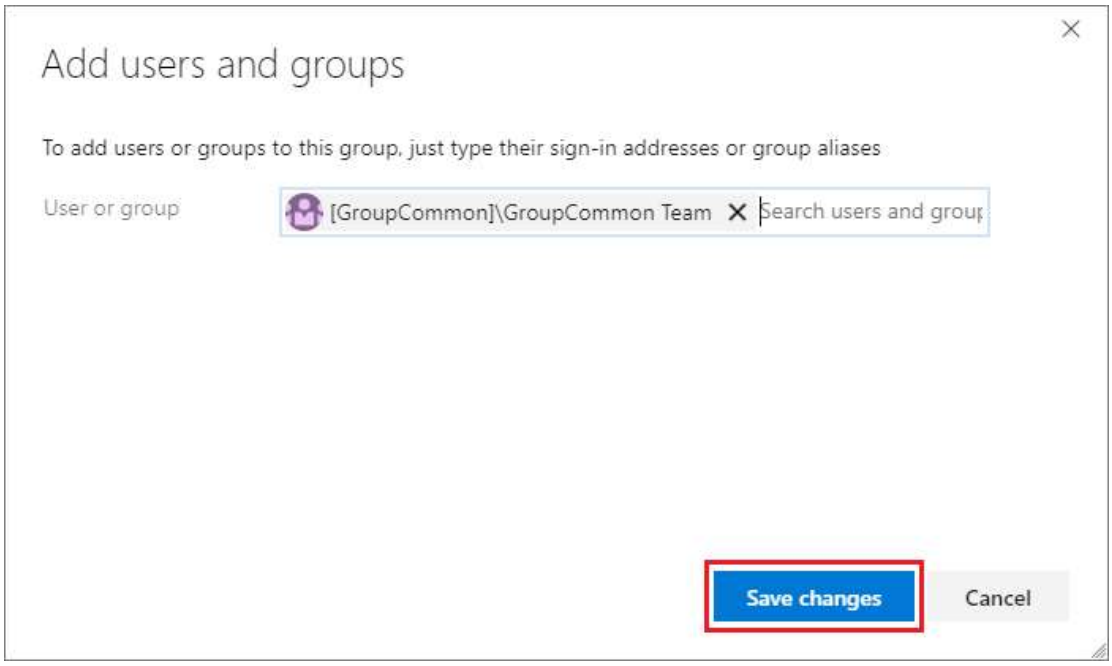
2. From the **Project Settings** left navigation, select **Teams**, then on the **Teams** page, select the **GroupCommon Team**.



3. On the **Team Profile** page, select **Add**.

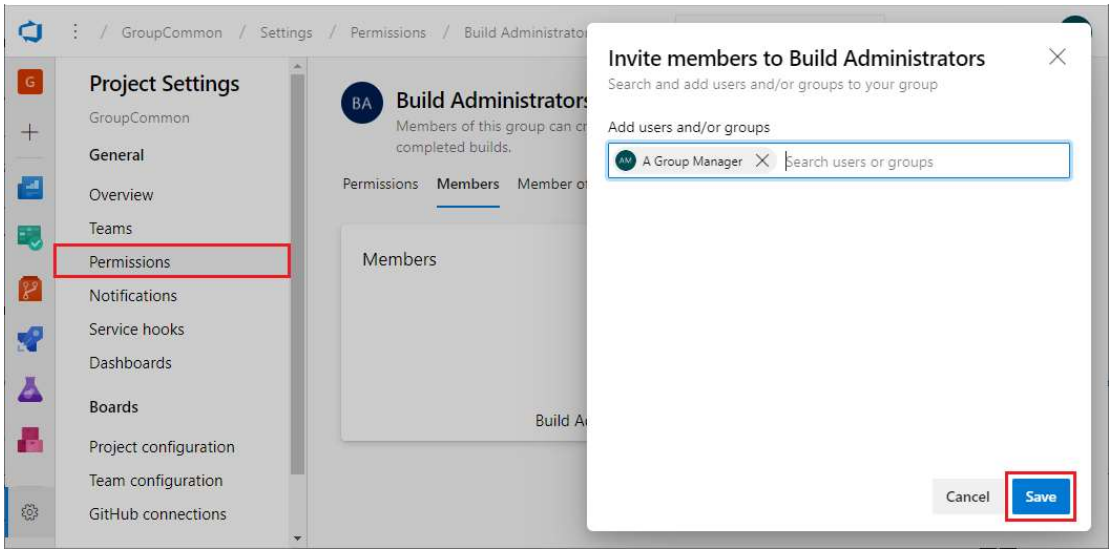


4. In the **Add users and groups** dialog, search for and select members to add to the group, and then select **Save changes**.



To configure permissions for members:

1. From the **Project Settings** left navigation, select **Permissions**.
2. On the **Permissions** page, select the group you want to add members to.
3. On the page for that group, select **Members**, and then select **Add**.
4. In the **Invite members** popup, search for and select members to add to the group, and then select **Save**.



Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to detailed descriptions of the other roles and tasks in the Team Data Science Process:

- Team Lead tasks for a data science team
- Project Lead tasks for a data science team
- Project Individual Contributor tasks for a data science team

Tasks for the team lead on a Team Data Science Process team

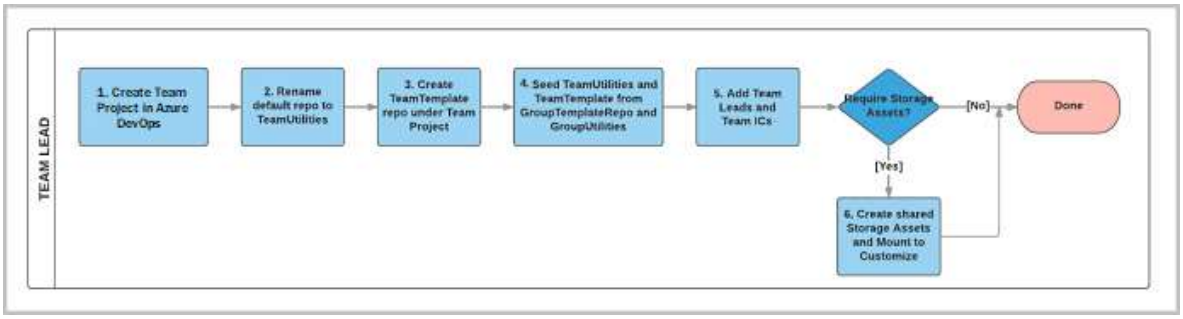
Article • 11/15/2022

This article describes the tasks that a *team lead* completes for their data science team. The team lead's objective is to establish a collaborative team environment that standardizes on the Team Data Science Process (TDSP). The TDSP is designed to help improve collaboration and team learning.

The TDSP is an agile, iterative data science methodology to efficiently deliver predictive analytics solutions and intelligent applications. The process distills the best practices and structures from Microsoft and the industry. The goal is successful implementation of data science initiatives and fully realizing the benefits of their analytics programs. For an outline of the personnel roles and associated tasks for a data science team standardizing on the TDSP, see Team Data Science Process roles and tasks.

A team lead manages a team consisting of several data scientists in the data science unit of an enterprise. Depending on the data science unit's size and structure, the group manager and the team lead might be the same person, or they could delegate their tasks to surrogates. But the tasks themselves do not change.

The following diagram shows the workflow for the tasks the team lead completes to set up a team environment:



1. Create a **team project** in the group's organization in Azure DevOps.
2. Rename the default team repository to **TeamUtilities**.
3. Create a new **TeamTemplate** repository in the team project.
4. Import the contents of the group's **GroupUtilities** and **GroupProjectTemplate** repositories into the **TeamUtilities** and **TeamTemplate** repositories.
5. Set up **security control** by adding team members and configuring their permissions.

6. If required, create team data and analytics resources:

    - Add team-specific utilities to the **TeamUtilities** repository.
    - Create **Azure file storage** to store data assets that can be useful for the entire team.
    - Mount the Azure file storage to the team lead's **Data Science Virtual Machine** (DSVM) and add data assets to it.

The following tutorial walks through the steps in detail.

> ⓘ Note
>
> This article uses Azure DevOps and a DSVM to set up a TDSP team environment, because that is how to implement TDSP at Microsoft. If your team uses other code hosting or development platforms, the team lead tasks are the same, but the way to complete them may be different.

Prerequisites

This tutorial assumes that the following resources and permissions have been set up by your group manager:

- The Azure DevOps **organization** for your data unit
- **GroupProjectTemplate** and **GroupUtilities** repositories, populated with the contents of the Microsoft TDSP team's **ProjectTemplate** and **Utilities** repositories
- Permissions on your organization account for you to create projects and repositories for your team

To be able to clone repositories and modify their content on your local machine or DSVM, or set up Azure file storage and mount it to your DSVM, you need the following:

- An Azure subscription.
- Git installed on your machine. If you're using a DSVM, Git is pre-installed. Otherwise, see the Platforms and tools appendix.
- If you want to use a DSVM, the Windows or Linux DSVM created and configured in Azure. For more information and instructions, see the Data Science Virtual Machine Documentation.
- For a Windows DSVM, Git Credential Manager (GCM) installed on your machine. In the *README.md* file, scroll down to the **Download and Install** section and select the **latest installer**. Download the *.exe* installer from the installer page and run it.

- For a Linux DSVM, an SSH public key set up on your DSVM and added in Azure DevOps. For more information and instructions, see the **Create SSH public key** section in the Platforms and tools appendix.

Create a team project and repositories

In this section, you create the following resources in your group's Azure DevOps organization:

- The **MyTeam** project in Azure DevOps
- The **TeamTemplate** repository
- The **TeamUtilities** repository

The names specified for the repositories and directories in this tutorial assume that you want to establish a separate project for your own team within your larger data science organization. However, the entire group can choose to work under a single project created by the group manager or organization administrator. Then, all the data science teams create repositories under this single project. This scenario might be valid for:

- A small data science group that doesn't have multiple data science teams.
- A larger data science group with multiple data science teams that nevertheless wants to optimize inter-team collaboration with activities such as group-level sprint planning.

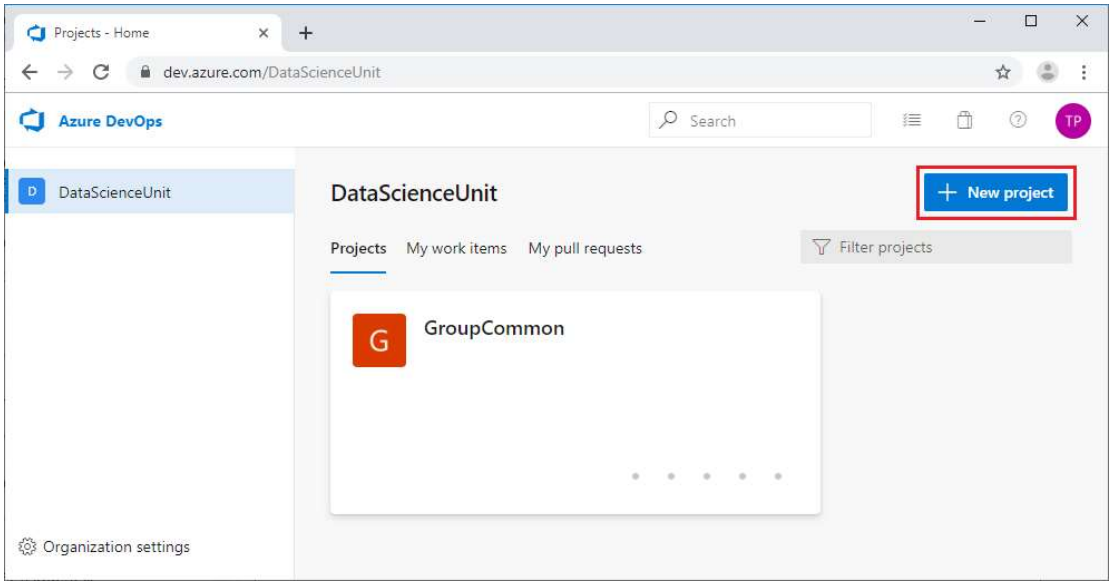
If teams choose to have their team-specific repositories under a single group project, the team leads should create the repositories with names like *<TeamName>Template* and *<TeamName>Utilities*. For instance: *TeamATemplate* and *TeamAUtilities*.

In any case, team leads need to let their team members know which template and utilities repositories to set up and clone. Project leads should follow the project lead tasks for a data science team to create project repositories, whether under separate projects or a single project.

Create the MyTeam project

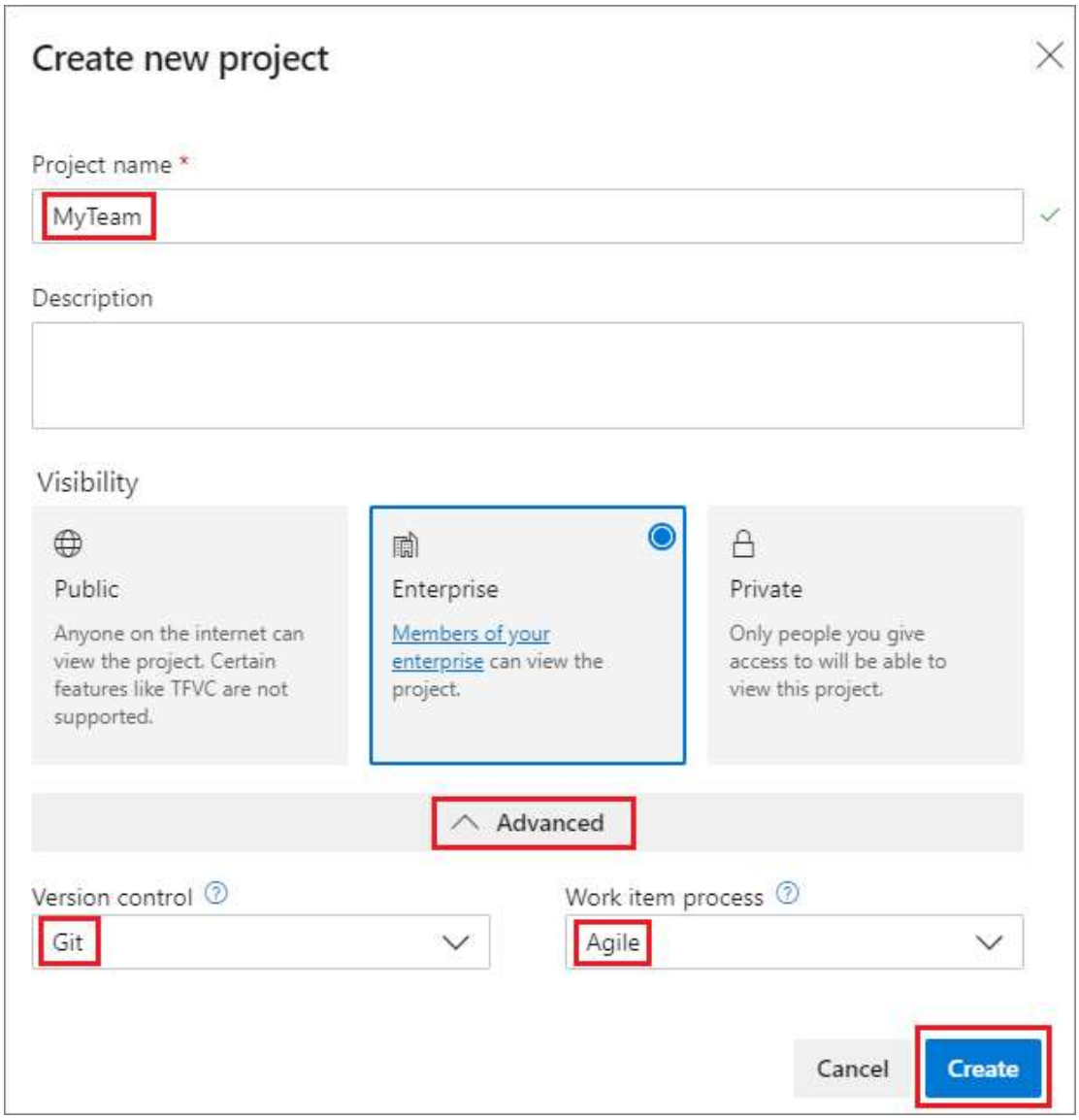
To create a separate project for your team:

1. In your web browser, go to your group's Azure DevOps organization home page at URL *https://<server name>/<organization name>*, and select **New project**.



2. In the **Create project** dialog, enter your team name, such as *MyTeam*, under **Project name**, and then select **Advanced**.

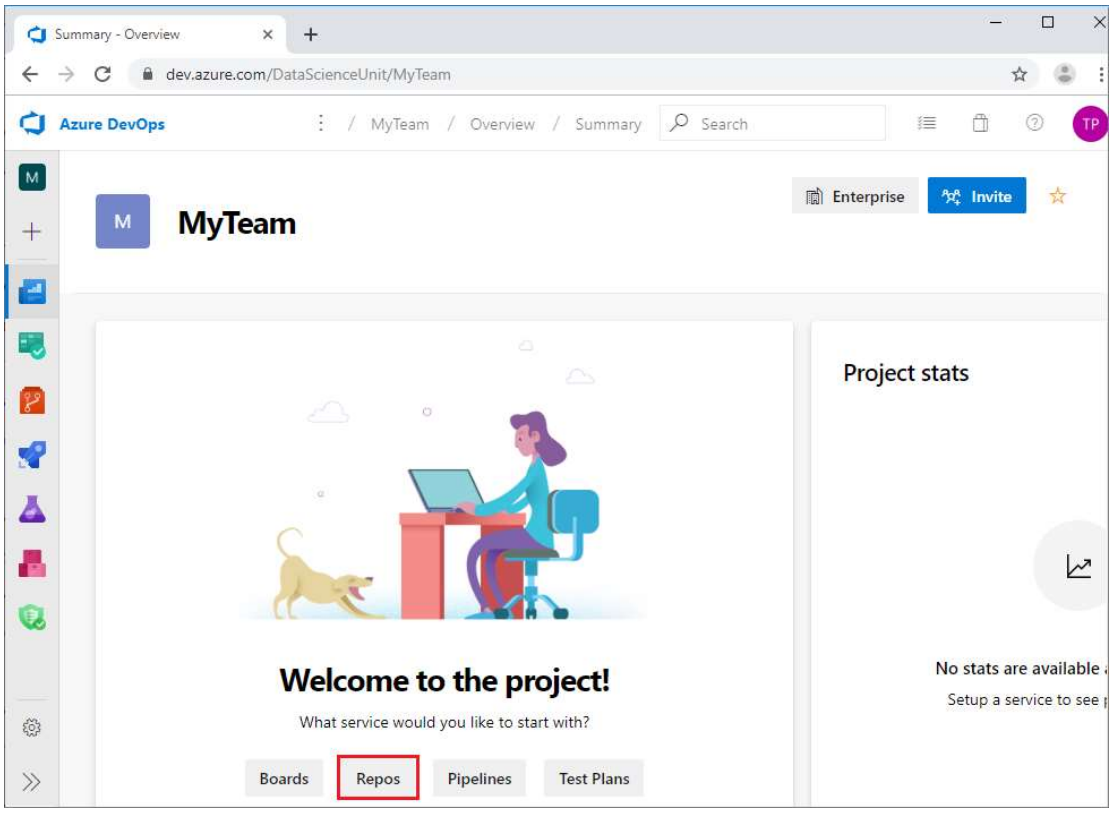
3. Under **Version control**, select **Git**, and under **Work item process**, select **Agile**. Then select **Create**.



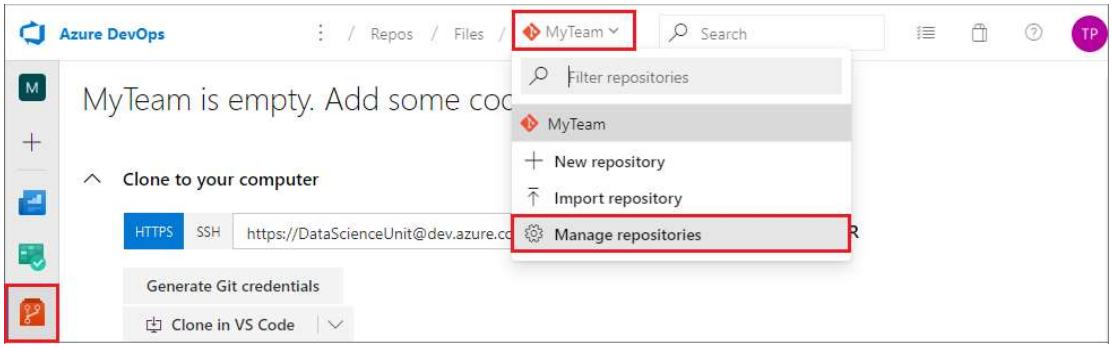
The team project **Summary** page opens, with page URL *https://<server name>/<organization name>/<team name>*.

Rename the MyTeam default repository to TeamUtilities

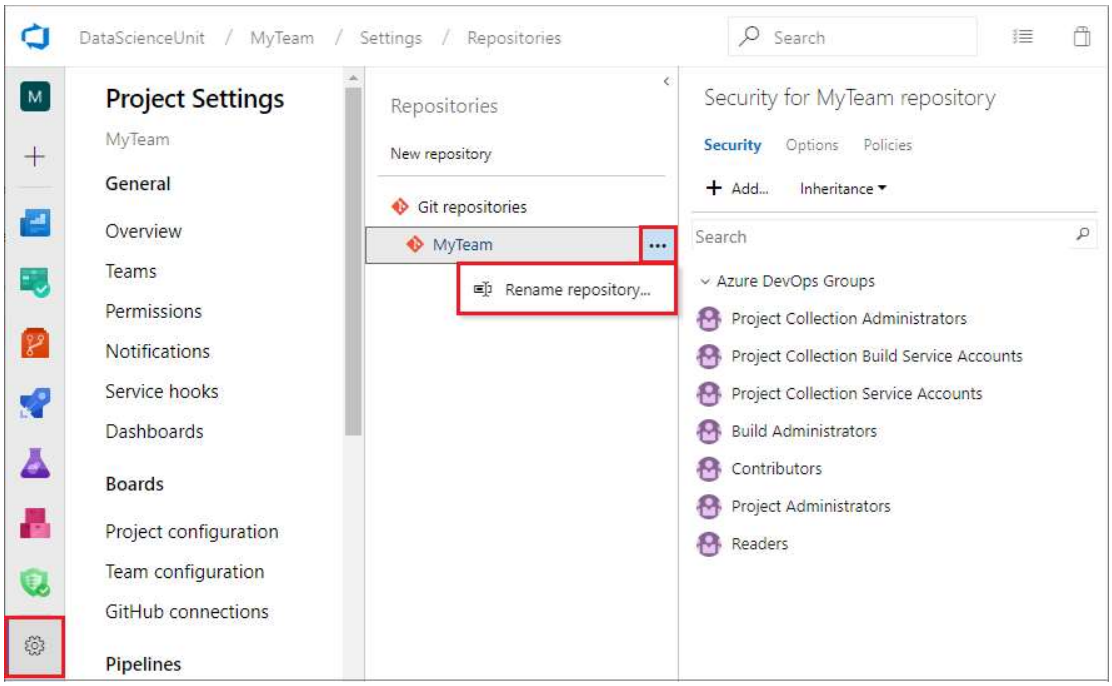
1. On the **MyTeam** project **Summary** page, under **What service would you like to start with?**, select **Repos**.



2. On the **MyTeam** repo page, select the **MyTeam** repository at the top of the page, and then select **Manage repositories** from the dropdown.



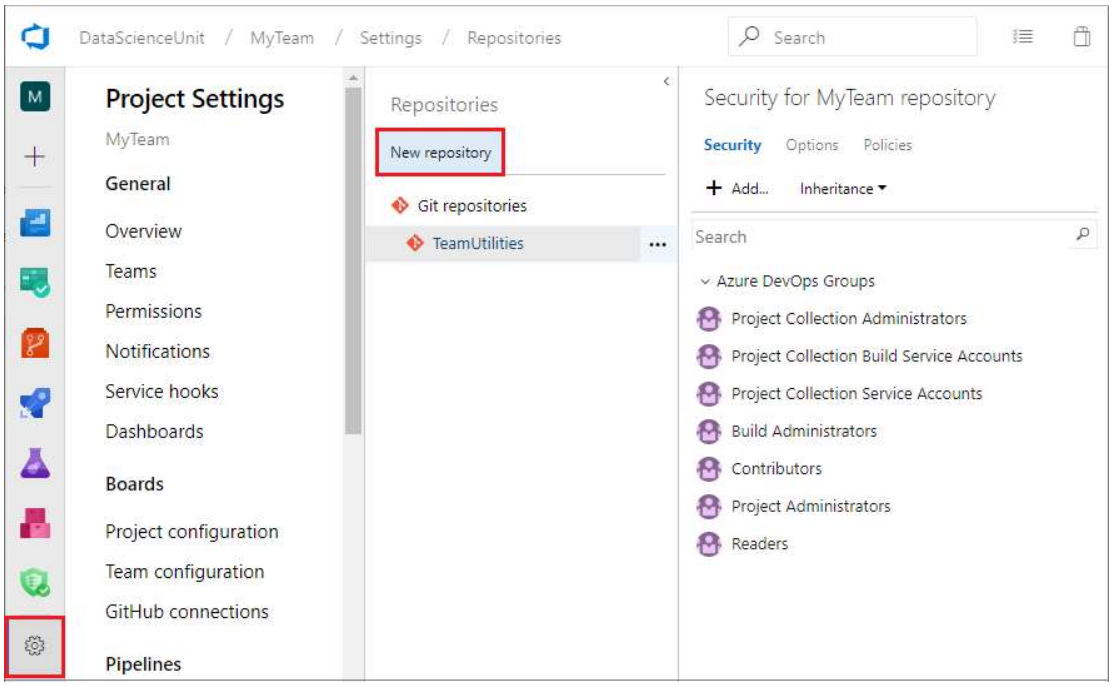
3. On the **Project Settings** page, select the **...** next to the **MyTeam** repository, and then select **Rename repository**.



4. In the **Rename the MyTeam repository** popup, enter *TeamUtilities*, and then select **Rename**.

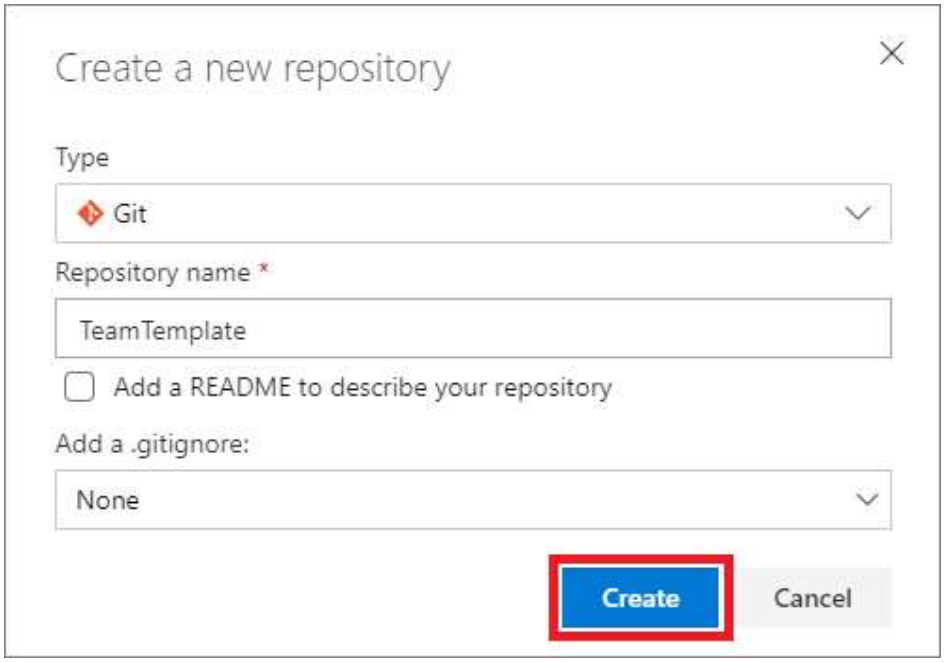
Create the TeamTemplate repository

1. On the **Project Settings** page, select **New repository.**

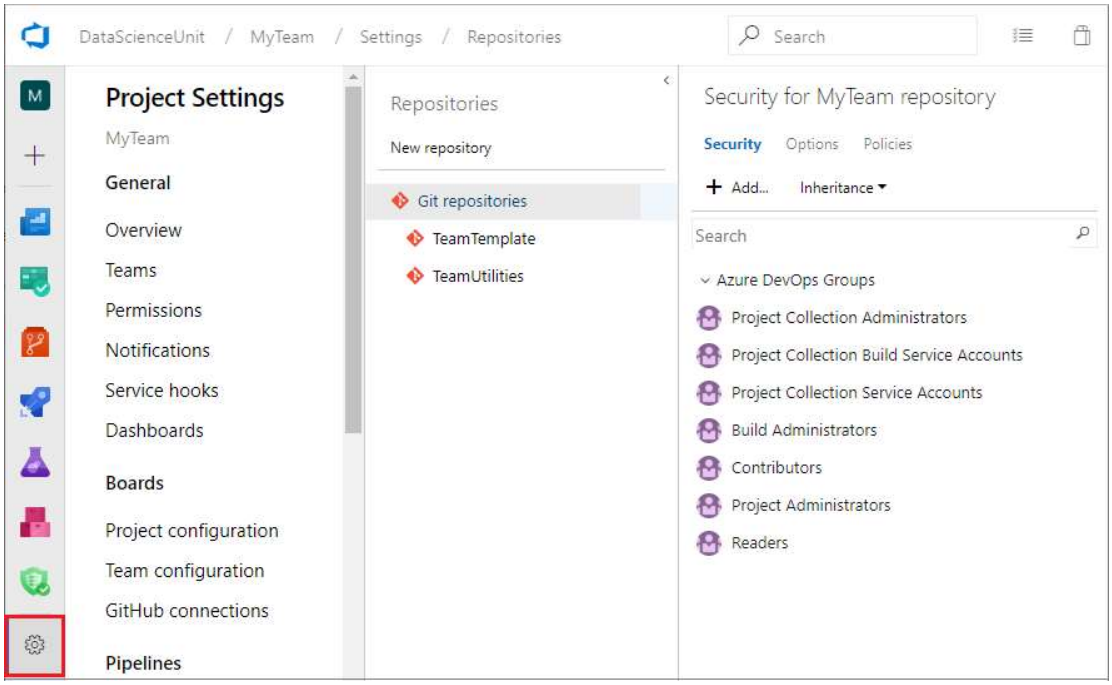


Or, select **Repos** from the left navigation of the **MyTeam** project **Summary** page, select a repository at the top of the page, and then select **New repository** from the dropdown.

2. In the **Create a new repository** dialog, make sure **Git** is selected under **Type**. Enter *TeamTemplate* under **Repository name**, and then select **Create**.



3. Confirm that you can see the two repositories **TeamUtilities** and **TeamTemplate** on your project settings page.



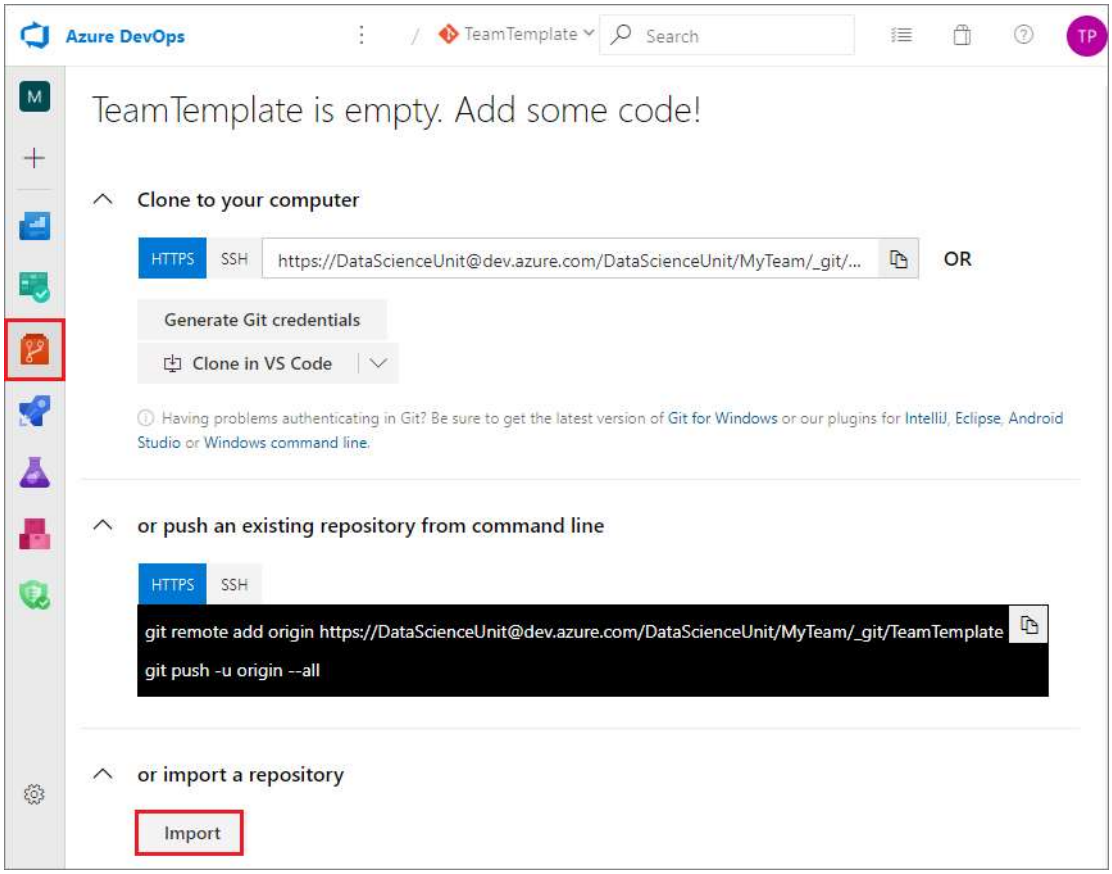
Import the contents of the group common repositories

To populate your team repositories with the contents of the group common repositories set up by your group manager:

1. From your **MyTeam** project home page, select **Repos** in the left navigation. If you get a message that the **MyTeam** template is not found, select the link in **Otherwise, navigate to your default TeamTemplate repository.**

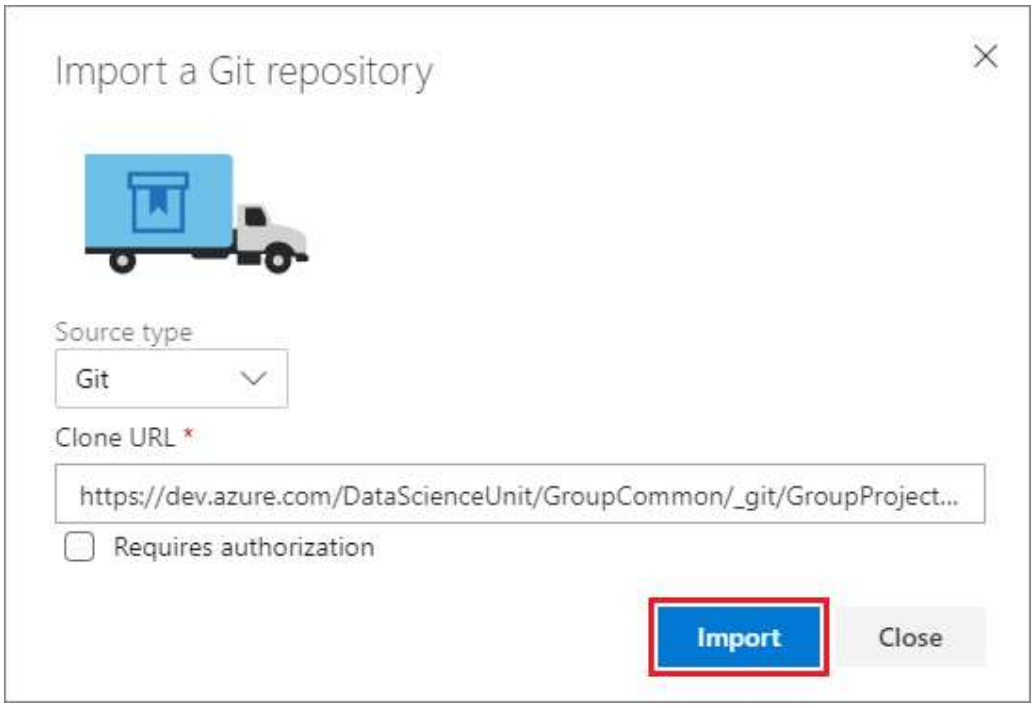
   The default **TeamTemplate** repository opens.

2. On the **TeamTemplate is empty** page, select **Import**.



3. In the **Import a Git repository** dialog, select **Git** as the **Source type**, and enter the URL for your group common template repository under **Clone URL**. The URL is *https://<server name>/<organization name>/_git/<repository name>*. For example: *https://dev.azure.com/DataScienceUnit/GroupCommon/_git/GroupProjectTemplate*.

4. Select **Import**. The contents of your group template repository are imported into your team template repository.



5. At the top of your project's **Repos** page, drop down and select the **TeamUtilities** repository.

6. Repeat the import process to import the contents of your group common utilities repository, for example *GroupUtilities*, into your **TeamUtilities** repository.

Each of your two team repositories now contains the files from the corresponding group common repository.

Customize the contents of the team repositories

If you want to customize the contents of your team repositories to meet your team's specific needs, you can do that now. You can modify files, change the directory structure, or add files and folders.

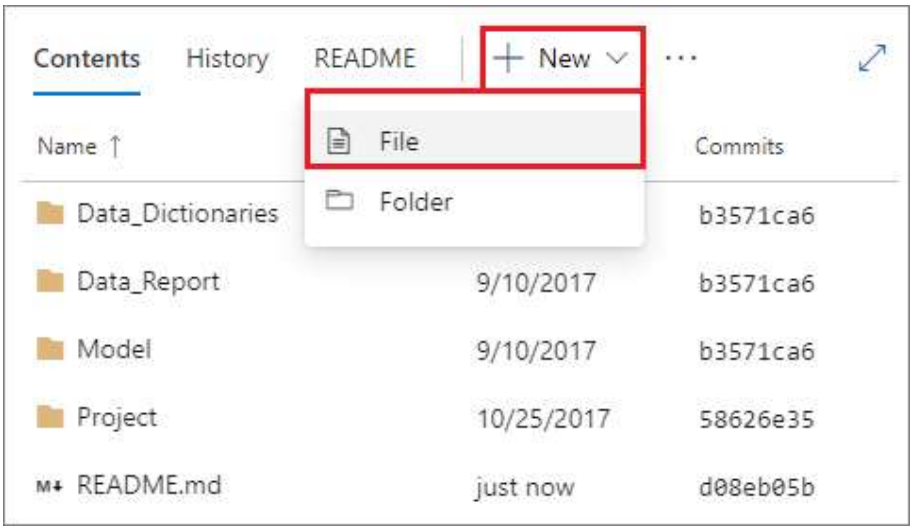
To modify, upload, or create files or folders directly in Azure DevOps:

1. On the **MyTeam** project **Summary** page, select **Repos**.

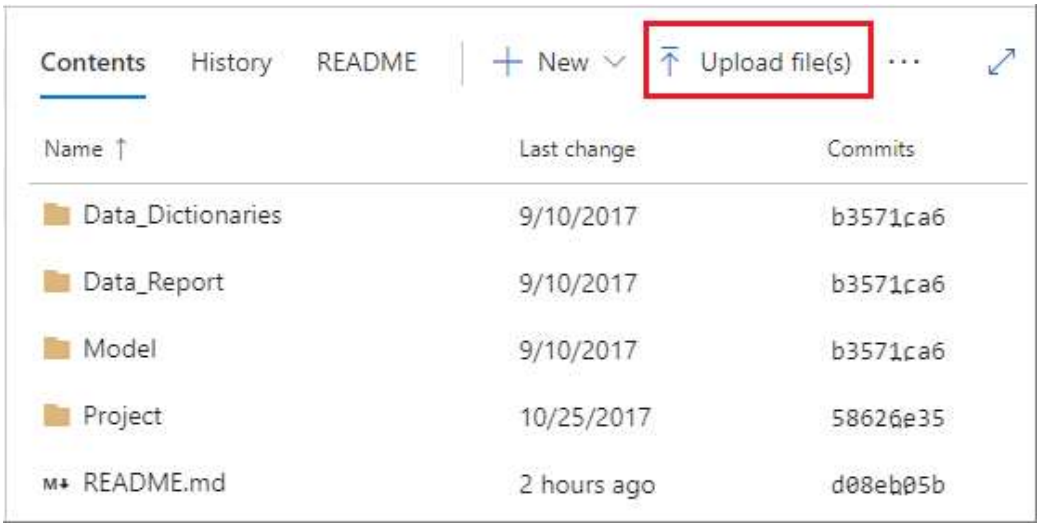
2. At the top of the page, select the repository you want to customize.

3. In the repo directory structure, navigate to the folder or file you want to change.

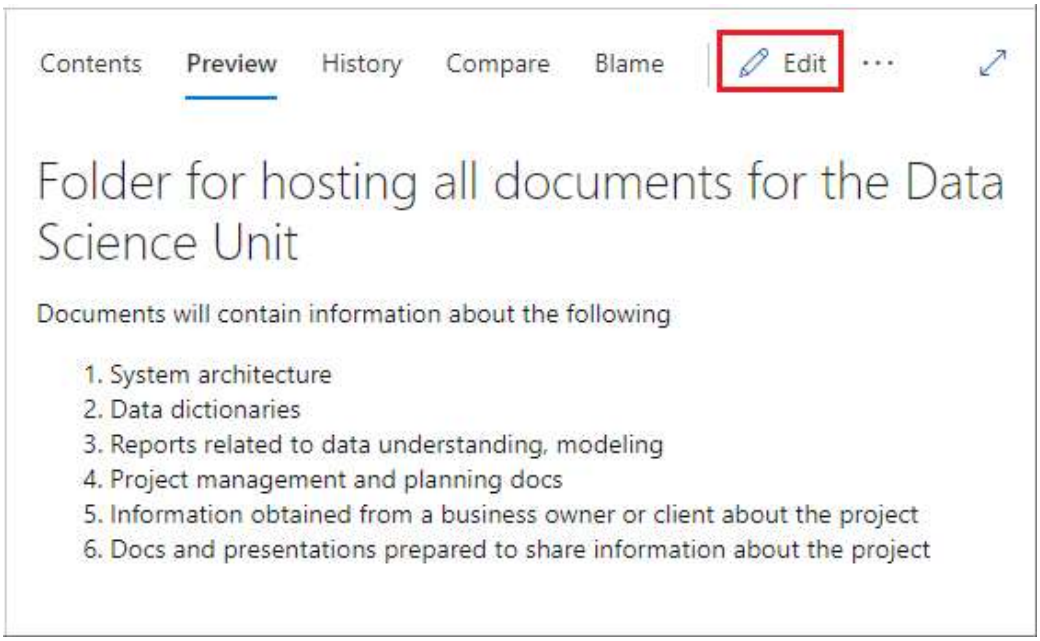
    - To create new folders or files, select the arrow next to **New**.



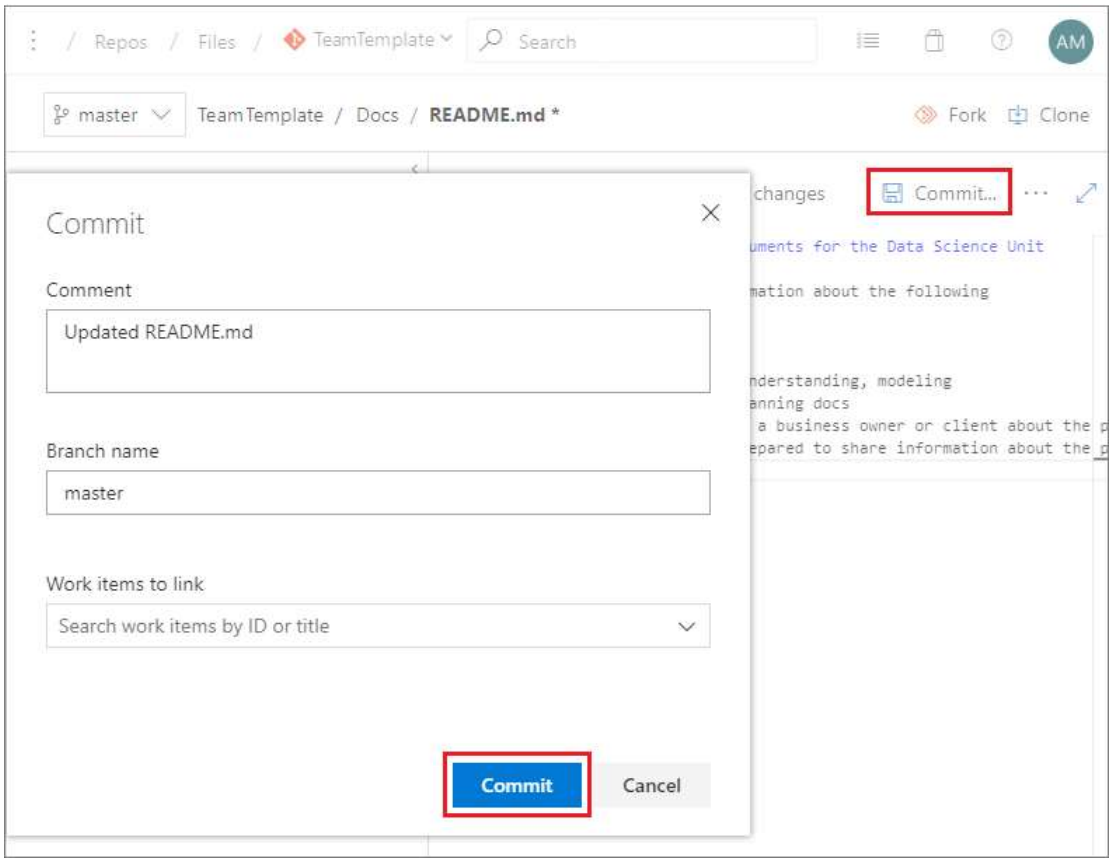
- To upload files, select **Upload file(s)**.



- To edit existing files, navigate to the file and then select **Edit**.



4. After adding or editing files, select **Commit**.



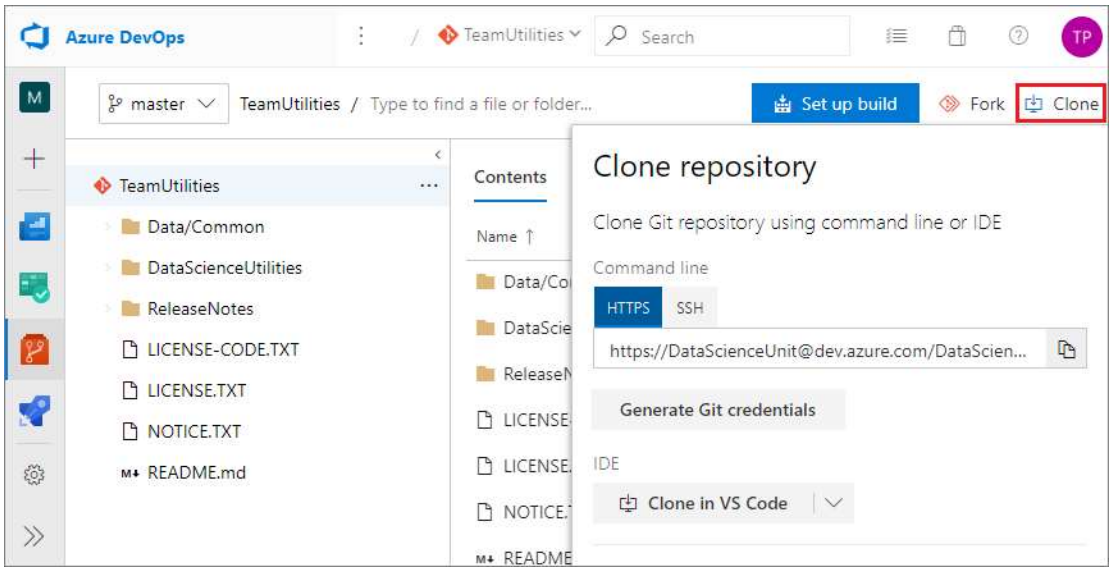
To work with repositories on your local machine or DSVM, you first copy or *clone* the repositories to your local machine, and then commit and push your changes up to the shared team repositories,

To clone repositories:

1. On the **MyTeam** project **Summary** page, select **Repos**, and at the top of the page, select the repository you want to clone.

2. On the repo page, select **Clone** at upper right.

3. In the **Clone repository** dialog, under **Command line**, select **HTTPS** for an HTTP connection or **SSH** for an SSH connection, and copy the clone URL to your clipboard.



4. On your local machine, create the following directories:

   - For Windows: **C:\GitRepos\MyTeam**
   - For Linux, **$home/GitRepos/MyTeam**

5. Change to the directory you created.

6. In Git Bash, run the command `git clone <clone URL>`, where <clone URL> is the URL you copied from the **Clone** dialog.

   For example, use one of the following commands to clone the **TeamUtilities** repository to the *MyTeam* directory on your local machine.

   HTTPS connection:

   ```
   Bash

   git clone
   https://DataScienceUnit@dev.azure.com/DataScienceUnit/MyTeam/_git/TeamU
   tilities
   ```

   SSH connection:

   ```
   Bash

   git clone git@ssh.dev.azure.com:v3/DataScienceUnit/MyTeam/TeamUtilities
   ```

After making whatever changes you want in the local clone of your repository, commit and push the changes to the shared team repositories.

Run the following Git Bash commands from your local **GitRepos\MyTeam\TeamTemplate** or **GitRepos\MyTeam\TeamUtilities** directory.

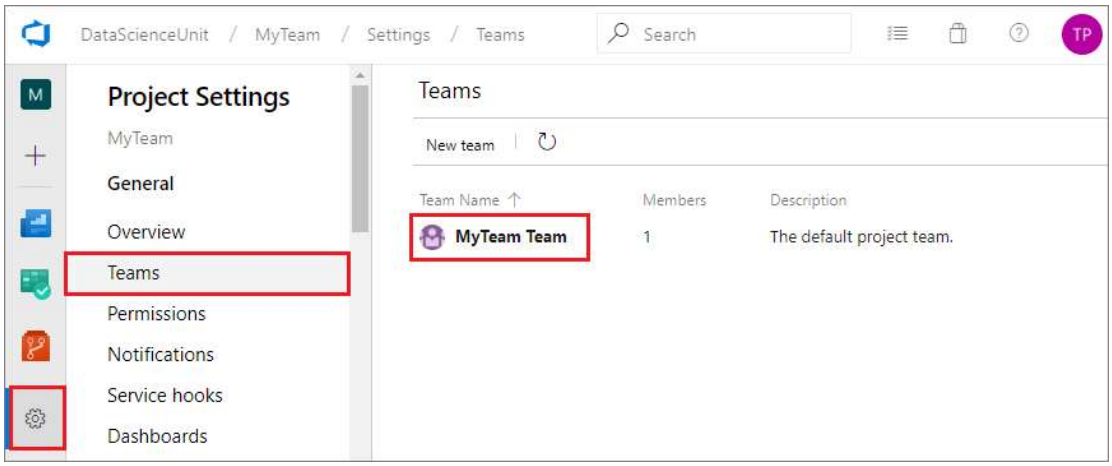
```
Bash

git add .
git commit -m "push from local"
git push
```

> ⓘ Note
>
> If this is the first time you commit to a Git repository, you may need to configure global parameters *user.name* and *user.email* before you run the `git commit` command. Run the following two commands:
>
> `git config --global user.name <your name>`
>
> `git config --global user.email <your email address>`
>
> If you're committing to several Git repositories, use the same name and email address for all of them. Using the same name and email address is convenient when building Power BI dashboards to track your Git activities in multiple repositories.

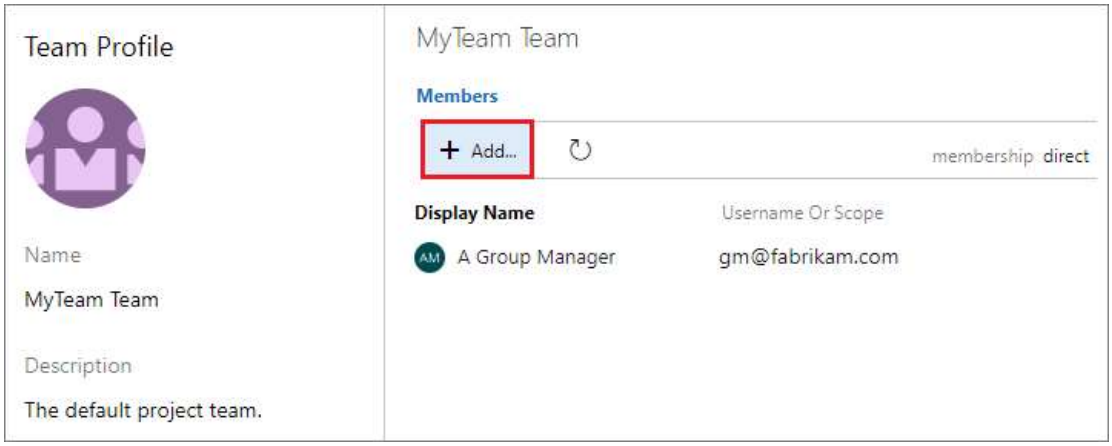
Add team members and configure permissions

To add members to the team:

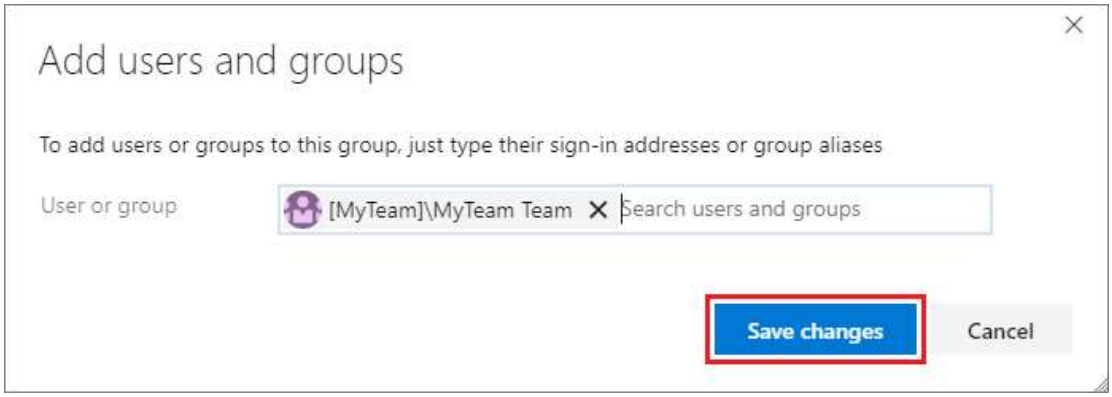
1. In Azure DevOps, from the **MyTeam** project home page, select **Project settings** from the left navigation.
2. From the **Project Settings** left navigation, select **Teams**, then on the **Teams** page, select the **MyTeam Team**.



3. On the **Team Profile** page, select **Add**.



4. In the **Add users and groups** dialog, search for and select members to add to the group, and then select **Save changes**.



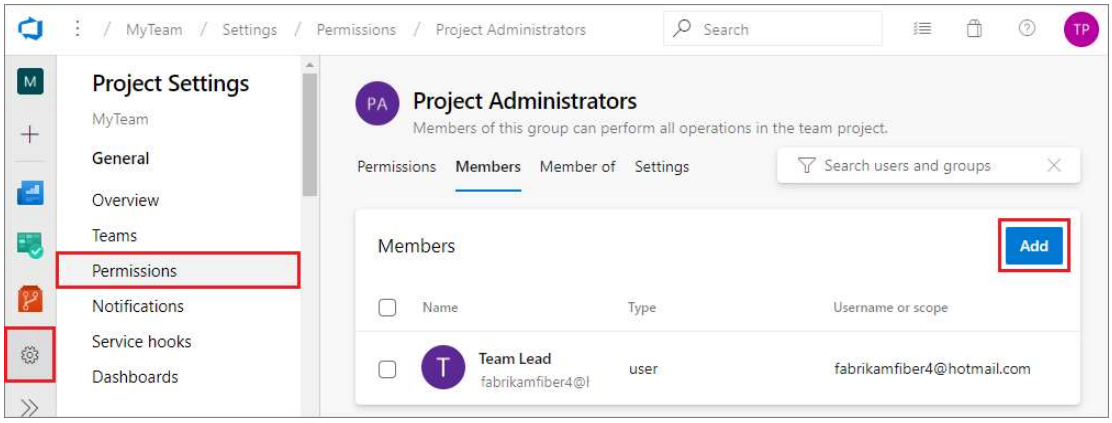
To configure permissions for team members:

1. From the **Project Settings** left navigation, select **Permissions**.

2. On the **Permissions** page, select the group you want to add members to.

3. On the page for that group, select **Members**, and then select **Add**.

4. In the **Invite members** popup, search for and select members to add to the group, and then select **Save**.



Create team data and analytics resources

This step is optional, but sharing data and analytics resources with your entire team has performance and cost benefits. Team members can execute their projects on the shared resources, save on budgets, and collaborate more efficiently. You can create Azure file storage and mount it on your DSVM to share with team members.

For information about sharing other resources with your team, such as Azure HDInsight Spark clusters, see Platforms and tools. That topic provides guidance from a data science perspective on selecting resources that are appropriate for your needs, and links to product pages and other relevant and useful tutorials.

> **Note**
>
> To avoid transmitting data across data centers, which might be slow and costly, make sure that your Azure resource group, storage account, and DSVM are all hosted in the same Azure region.

Create Azure file storage

1. Run the following script to create Azure file storage for data assets that are useful for your entire team. The script prompts you for your Azure subscription information, so have that ready to enter.

    - On a Windows machine, run the script from the PowerShell command prompt:

        ```
        PowerShell
        ```

```
wget "https://raw.githubusercontent.com/Azure/Azure-MachineLearning-DataScience/master/Misc/TDSP/CreateFileShare.ps1" -outfile "CreateFileShare.ps1"
.\CreateFileShare.ps1
```

- On a Linux machine, run the script from the Linux shell:

```shell
wget "https://raw.githubusercontent.com/Azure/Azure-MachineLearning-DataScience/master/Misc/TDSP/CreateFileShare.sh"
bash CreateFileShare.sh
```

2. Log in to your Microsoft Azure account when prompted, and select the subscription you want to use.

3. Select the storage account to use, or create a new one under your selected subscription. You can use lowercase characters, numbers, and hyphens for the Azure file storage name.

4. To facilitate mounting and sharing the storage, press Enter or enter *Y* to save the Azure file storage information into a text file in your current directory. You can check in this text file to your **TeamTemplate** repository, ideally under **Docs\DataDictionaries**, so all projects in your team can access it. You also need the file information to mount your Azure file storage to your Azure DSVM in the next section.

Mount Azure file storage on your local machine or DSVM

1. To mount your Azure file storage to your local machine or DSVM, use the following script.

   - On a Windows machine, run the script from the PowerShell command prompt:

```PowerShell
wget "https://raw.githubusercontent.com/Azure/Azure-MachineLearning-DataScience/master/Misc/TDSP/AttachFileShare.ps1" -outfile "AttachFileShare.ps1"
.\AttachFileShare.ps1
```

   - On a Linux machine, run the script from the Linux shell:

```
```

```shell
wget "https://raw.githubusercontent.com/Azure/Azure-MachineLearning-DataScience/master/Misc/TDSP/AttachFileShare.sh"
bash AttachFileShare.sh
```

2. Press Enter or enter *Y* to continue, if you saved an Azure file storage information file in the previous step. Enter the complete path and name of the file you created.

   If you don't have an Azure file storage information file, enter *n*, and follow the instructions to enter your subscription, Azure storage account, and Azure file storage information.

3. Enter the name of a local or TDSP drive to mount the file share on. The screen displays a list of existing drive names. Provide a drive name that doesn't already exist.

4. Confirm that the new drive and storage is successfully mounted on your machine.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to detailed descriptions of the other roles and tasks defined by the Team Data Science Process:

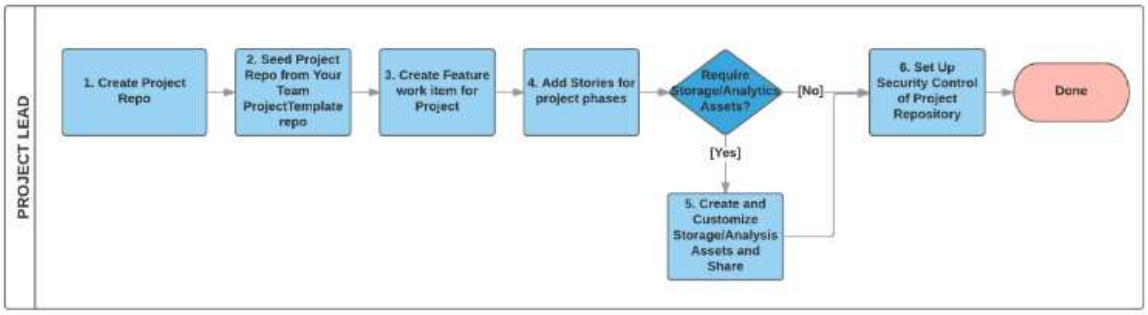
- Group Manager tasks for a data science team
- Project Lead tasks for a data science team
- Project Individual Contributor tasks for a data science team

Project lead tasks in the Team Data Science Process

Article • 11/15/2022

This article describes tasks that a *project lead* completes to set up a repository for their project team in the Team Data Science Process (TDSP). The TDSP is a framework developed by Microsoft that provides a structured sequence of activities to efficiently execute cloud-based, predictive analytics solutions. The TDSP is designed to help improve collaboration and team learning. For an outline of the personnel roles and associated tasks for a data science team standardizing on the TDSP, see Team Data Science Process roles and tasks.

A project lead manages the daily activities of individual data scientists on a specific data science project in the TDSP. The following diagram shows the workflow for project lead tasks:



This tutorial covers Step 1: Create project repository, and Step 2: Seed project repository from your team ProjectTemplate repository.

For Step 3: Create Feature work item for project, and Step 4: Add Stories for project phases, see Agile development of data science projects.

For Step 5: Create and customize storage/analysis assets and share, if necessary, see Create team data and analytics resources.

For Step 6: Set up security control of project repository, see Add team members and configure permissions.

> **Note**
>
> This article uses Azure Repos to set up a TDSP project, because that is how to implement TDSP at Microsoft. If your team uses another code hosting platform, the project lead tasks are the same, but the way to complete them may be different.

Prerequisites

This tutorial assumes that your group manager and team lead have set up the following resources and permissions:

- The Azure DevOps **organization** for your data unit
- A team **project** for your data science team
- Team template and utilities **repositories**
- **Permissions** on your organization account for you to create and edit repositories for your project

To clone repositories and modify content on your local machine or Data Science Virtual Machine (DSVM), or set up Azure file storage and mount it to your DSVM, you also need to consider this checklist:

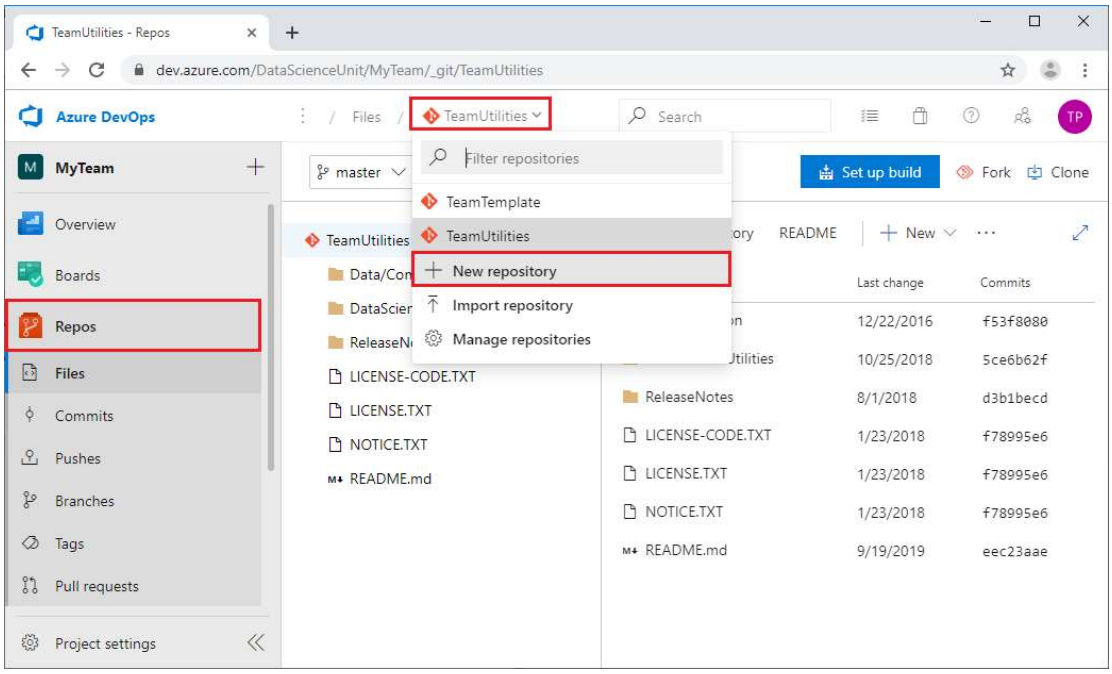
- An Azure subscription.
- Git installed on your machine. If you're using a DSVM, Git is pre-installed. Otherwise, see the Platforms and tools appendix.
- If you want to use a DSVM, the Windows or Linux DSVM created and configured in Azure. For more information and instructions, see the Data Science Virtual Machine Documentation.
- For a Windows DSVM, Git Credential Manager (GCM) installed on your machine. In the *README.md* file, scroll down to the **Download and Install** section and select the **latest installer**. Download the *.exe* installer from the installer page and run it.
- For a Linux DSVM, an SSH public key set up on your DSVM and added in Azure DevOps. For more information and instructions, see the **Create SSH public key** section in the Platforms and tools appendix.

Create a project repository in your team project

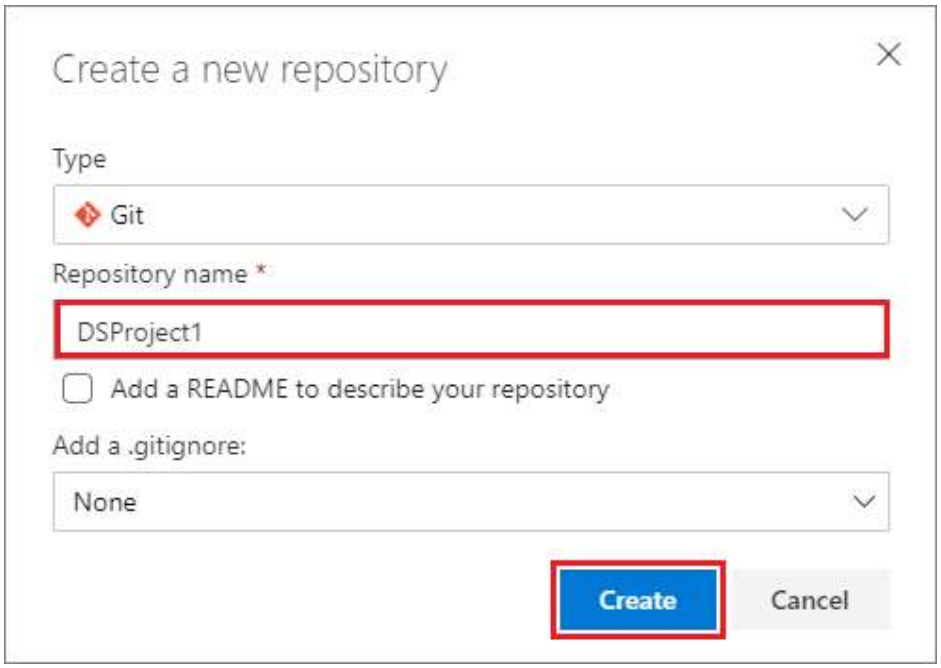
To create a project repository in your team's **MyTeam** project:

1. Go to your team's project **Summary** page at *https://<server name>/<organization name>/<team name>*, for example, **https://dev.azure.com/DataScienceUnit/MyTeam**, and select **Repos** from the left navigation.

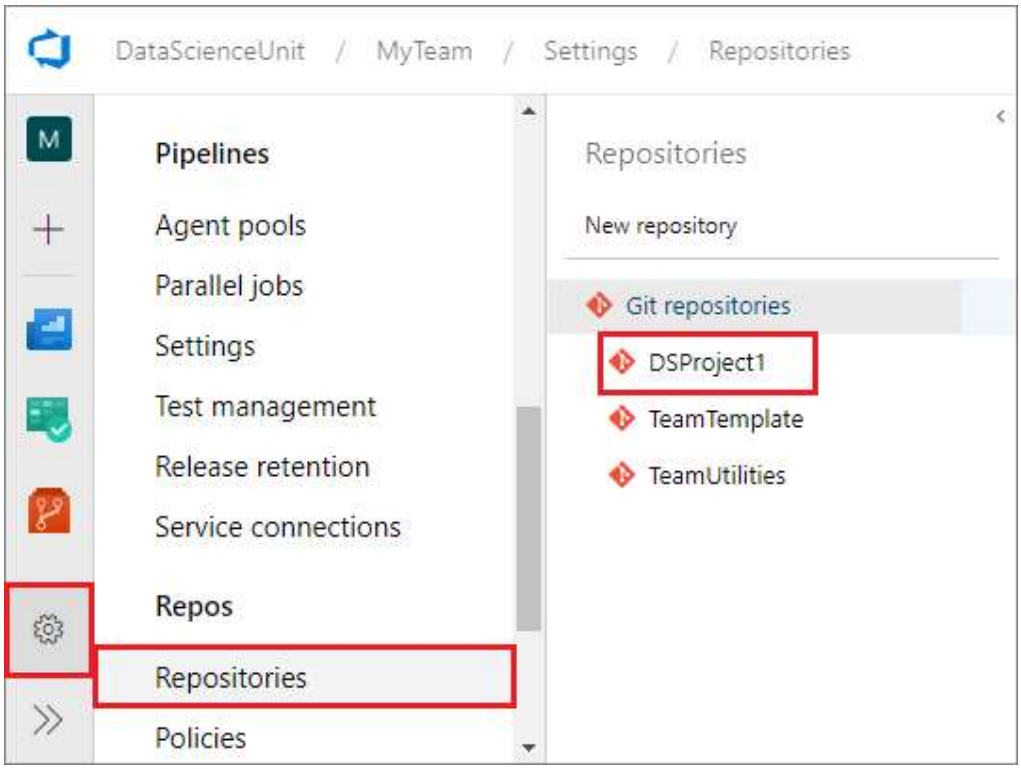
2. Select the repository name at the top of the page, and then select **New repository** from the dropdown.



3. In the **Create a new repository** dialog, make sure **Git** is selected under **Type**. Enter *DSProject1* under **Repository name**, and then select **Create**.



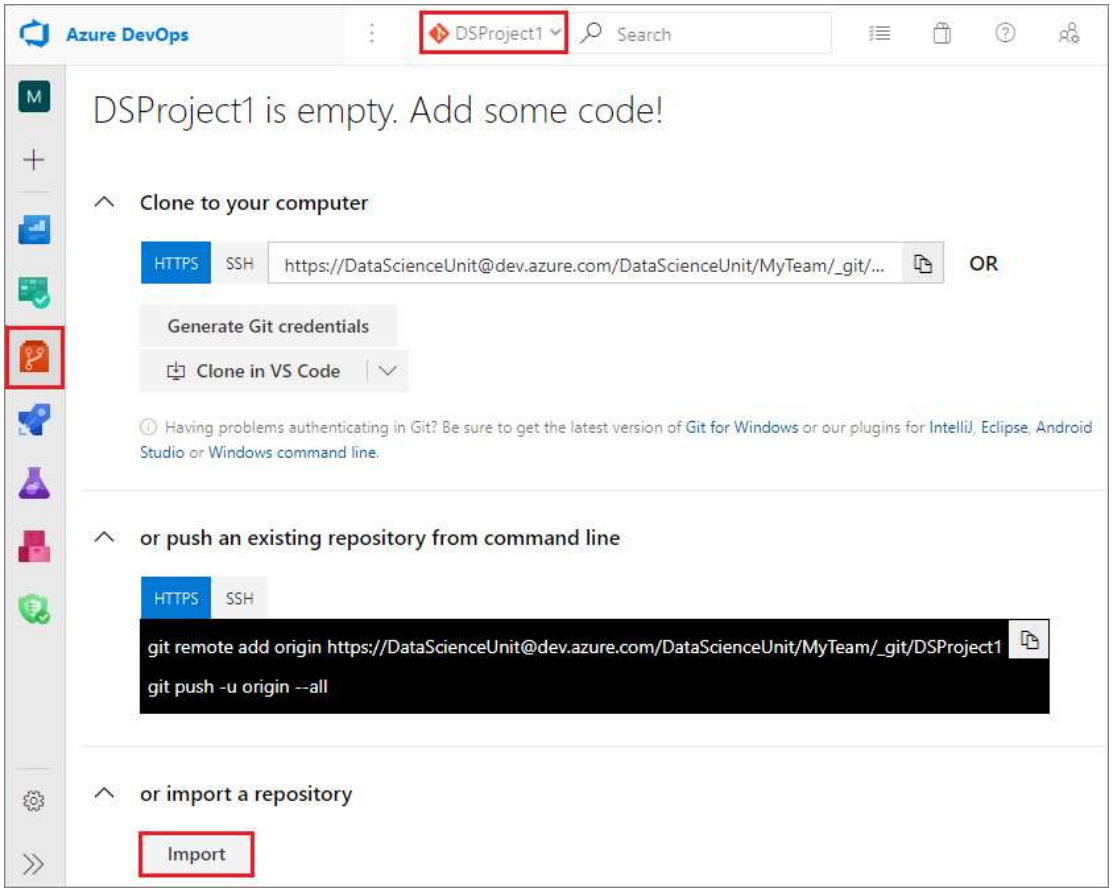
4. Confirm that you can see the new **DSProject1** repository on your project settings page.



Import the team template into your project repository

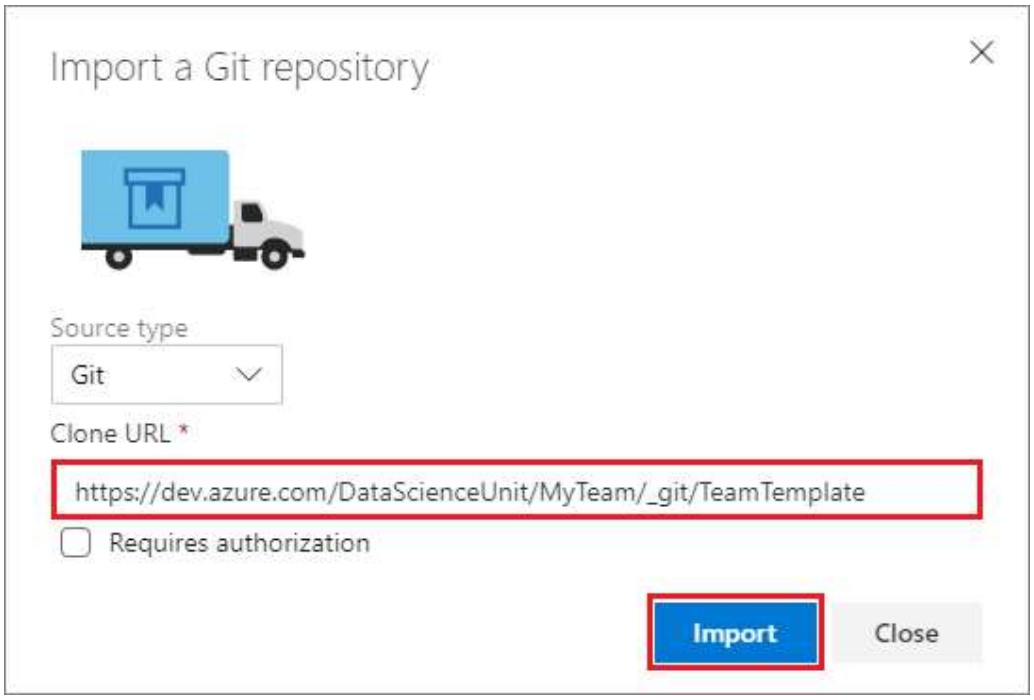
To populate your project repository with the contents of your team template repository:

1. From your team's project **Summary** page, select **Repos** in the left navigation.
2. Select the repository name at the top of the page, and select **DSProject1** from the dropdown.
3. On the **DSProject1 is empty** page, select **Import**.



4. In the **Import a Git repository** dialog, select **Git** as the **Source type**, and enter the URL for your **TeamTemplate** repository under **Clone URL**. The URL is *https://<server name>/<organization name>/<team name>/_git/<team template repository name>*. For example: **https://dev.azure.com/DataScienceUnit/MyTeam/_git/TeamTemplate**.

5. Select **Import**. The contents of your team template repository are imported into your project repository.



If you need to customize the contents of your project repository to meet your project's specific needs, you can add, delete, or modify repository files and folders. You can work directly in Azure Repos, or clone the repository to your local machine or DSVM, make changes, and commit and push your updates to the shared project repository. Follow the instructions at Customize the contents of the team repositories.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to detailed descriptions of the other roles and tasks defined by the Team Data Science Process:

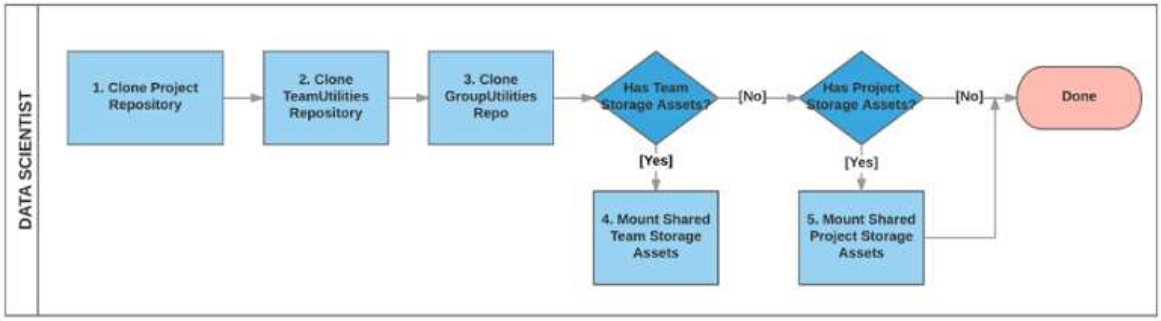
- Group Manager tasks for a data science team
- Team Lead tasks for a data science team
- Individual Contributor tasks for a data science team

Tasks for an individual contributor in the Team Data Science Process

Article • 11/15/2022

This topic outlines the tasks that an *individual contributor* completes to set up a project in the Team Data Science Process (TDSP). The objective is to work in a collaborative team environment that standardizes on the TDSP. The TDSP is designed to help improve collaboration and team learning. For an outline of the personnel roles and their associated tasks that are handled by a data science team standardizing on the TDSP, see Team Data Science Process roles and tasks.

The following diagram shows the tasks that project individual contributors (data scientists) complete to set up their team environment. For instructions on how to execute a data science project under the TDSP, see Execution of data science projects.



- **ProjectRepository** is the repository your project team maintains to share project templates and assets.
- **TeamUtilities** is the utilities repository your team maintains specifically for your team.
- **GroupUtilities** is the repository your group maintains to share useful utilities across the entire group.

> ⓘ Note
>
> This article uses Azure Repos and a Data Science Virtual Machine (DSVM) to set up a TDSP environment, because that is how to implement TDSP at Microsoft. If your team uses other code hosting or development platforms, the individual contributor tasks are the same, but the way to complete them may be different.

Prerequisites

This tutorial assumes that the following resources and permissions have been set up by your group manager, team lead, and project lead:

- The Azure DevOps **organization** for your data science unit
- A **project repository** set up by your project lead to share project templates and assets
- **GroupUtilities** and **TeamUtilities** repositories set up by the group manager and team lead, if applicable
- Azure **file storage** set up for shared assets for your team or project, if applicable
- **Permissions** for you to clone from and push back to your project repository

To clone repositories and modify content on your local machine or DSVM, or mount Azure file storage to your DSVM, you need to consider this checklist:

- An Azure subscription.
- Git installed on your machine. If you're using a DSVM, Git is pre-installed. Otherwise, see the Platforms and tools appendix.
- If you want to use a DSVM, the Windows or Linux DSVM created and configured in Azure. For more information and instructions, see the Data Science Virtual Machine Documentation.
- For a Windows DSVM, Git Credential Manager (GCM) installed on your machine. In the *README.md* file, scroll down to the **Download and Install** section and select the **latest installer**. Download the *.exe* installer from the installer page and run it.
- For a Linux DSVM, an SSH public key set up on your DSVM and added in Azure DevOps. For more information and instructions, see the **Create SSH public key** section in the Platforms and tools appendix.
- The Azure file storage information for any Azure file storage you need to mount to your DSVM.

Clone repositories

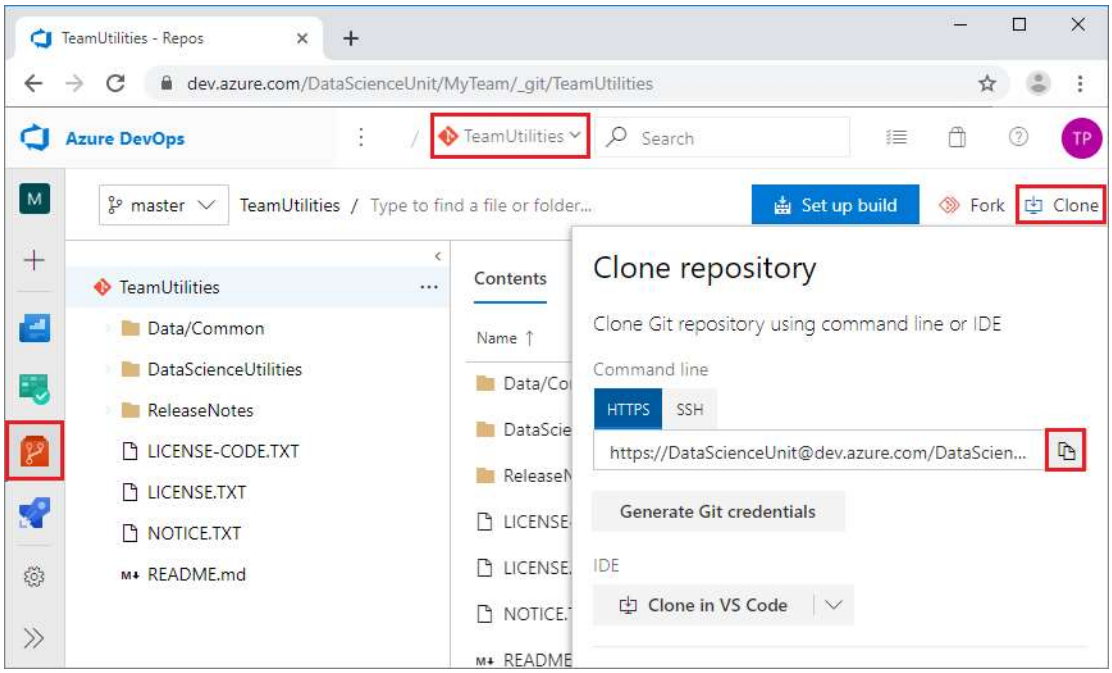
To work with repositories locally and push your changes up to the shared team and project repositories, you first copy or *clone* the repositories to your local machine.

1. In Azure DevOps, go to your team's project Summary page at *https://<server name>/<organization name>/<team name>*, for example, **https://dev.azure.com/DataScienceUnit/MyTeam**.

2. Select **Repos** in the left navigation, and at the top of the page, select the repository you want to clone.

3. On the repo page, select **Clone** at upper right.

4. In the **Clone repository** dialog, select **HTTPS** for an HTTP connection, or **SSH** for an SSH connection, and copy the clone URL under **Command line** to your clipboard.



5. On your local machine or DSVM, create the following directories:

   - For Windows: **C:\GitRepos**
   - For Linux: **$home/GitRepos**

6. Change to the directory you created.

7. In Git Bash, run the command `git clone <clone URL>` for each repository you want to clone.

   For example, the following command clones the **TeamUtilities** repository to the *MyTeam* directory on your local machine.

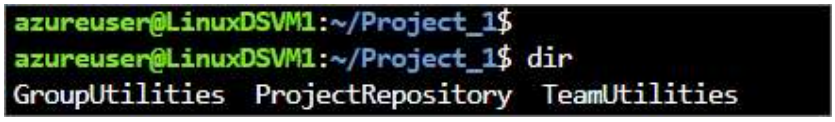
   HTTPS connection:

   ```
   Bash

   git clone
   https://DataScienceUnit@dev.azure.com/DataScienceUnit/MyTeam/_git/TeamU
   tilities
   ```

   SSH connection:

   ```
   Bash

   git clone git@ssh.dev.azure.com:v3/DataScienceUnit/MyTeam/TeamUtilities
   ```

8. Confirm that you can see the folders for the cloned repositories in your local project directory.



Mount Azure file storage to your DSVM

If your team or project has shared assets in Azure file storage, mount the file storage to your local machine or DSVM. Follow the instructions at Mount Azure file storage on your local machine or DSVM.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Here are links to detailed descriptions of the other roles and tasks defined by the Team Data Science Process:

- Group Manager tasks for a data science team
- Team Lead tasks for a data science team
- Project Lead tasks for a data science team

Team Data Science Process project planning

Article • 08/31/2023

The Team Data Science Process (TDSP) provides a lifecycle to structure the development of your data science projects.

The lifecycle outlines the major stages that projects typically execute, often iteratively:

- Business Understanding
- Data Acquisition and Understanding
- Modeling
- Deployment
- Customer Acceptance

For descriptions of each of these stages, see The Team Data Science Process lifecycle.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

See Agile development of data science projects. This document describes a data science project in a systematic, version controlled, and collaborative way by using the Team Data Science Process.

Related resources

- What is the Team Data Science Process?
- Compare the machine learning products and technologies from Microsoft

Agile development of data science projects

Article • 11/15/2022

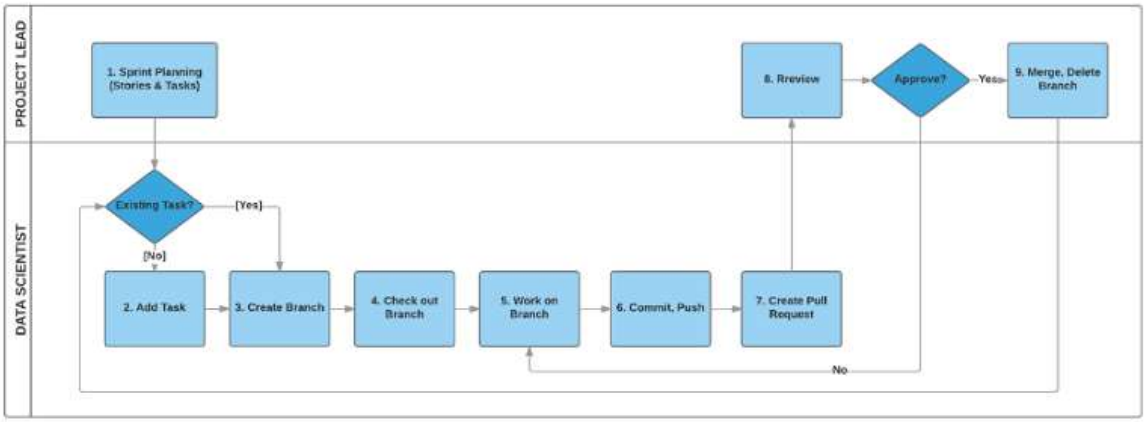
This document describes how developers can execute a data science project in a systematic, version controlled, and collaborative way within a project team by using the Team Data Science Process (TDSP). The TDSP is a framework developed by Microsoft that provides a structured sequence of activities to efficiently execute cloud-based, predictive analytics solutions. For an outline of the roles and tasks that are handled by a data science team standardizing on the TDSP, see Team Data Science Process roles and tasks.

This article includes instructions on how to:

- Do *sprint planning* for work items involved in a project.
- Add *work items* to sprints.
- Create and use an *agile-derived work item template* that specifically aligns with TDSP lifecycle stages.

The following instructions outline the steps needed to set up a TDSP team environment using Azure Boards and Azure Repos in Azure DevOps. The instructions use Azure DevOps because that is how to implement TDSP at Microsoft. If your group uses a different code hosting platform, the team lead tasks generally don't change, but the way to complete the tasks is different. For example, linking a work item with a Git branch might not be the same with GitHub as it is with Azure Repos.

The following figure illustrates a typical sprint planning, coding, and source-control workflow for a data science project:



Work item types

In the TDSP sprint planning framework, there are four frequently used *work item* types: *Features*, *User Stories*, *Tasks*, and *Bugs*. The backlog for all work items is at the project level, not the Git repository level.

Here are the definitions for the work item types:

- **Feature**: A Feature corresponds to a project engagement. Different engagements with a client are different Features, and it's best to consider different phases of a project as different Features. If you choose a schema such as *<ClientName>-<EngagementName>* to name your Features, you can easily recognize the context of the project and engagement from the names themselves.

- **User Story**: User Stories are work items needed to complete a Feature end-to-end. Examples of User Stories include:
  - Get data
  - Explore data
  - Generate features
  - Build models
  - Operationalize models
  - Retrain models

- **Task**: Tasks are assignable work items that need to be done to complete a specific User Story. For example, Tasks in the User Story *Get data* could be:
  - Get SQL Server credentials
  - Upload data to Azure Synapse Analytics

- **Bug**: Bugs are issues in existing code or documents that must be fixed to complete a Task. If Bugs are caused by missing work items, they can escalate to be User Stories or Tasks.

Data scientists may feel more comfortable using an agile template that replaces Features, User Stories, and Tasks with TDSP lifecycle stages and substages. To create an agile-derived template that specifically aligns with the TDSP lifecycle stages, see Use an agile TDSP work template.

> **Note**
>
> TDSP borrows the concepts of Features, User Stories, Tasks, and Bugs from software code management (SCM). The TDSP concepts might differ slightly from their conventional SCM definitions.

Plan sprints

Many data scientists are engaged with multiple projects, which can take months to complete and proceed at different paces. Sprint planning is useful for project prioritization, and resource planning and allocation. In Azure Boards, you can easily create, manage, and track work items for your projects, and conduct sprint planning to ensure projects are moving forward as expected.

For more information about sprint planning, see Scrum sprints.

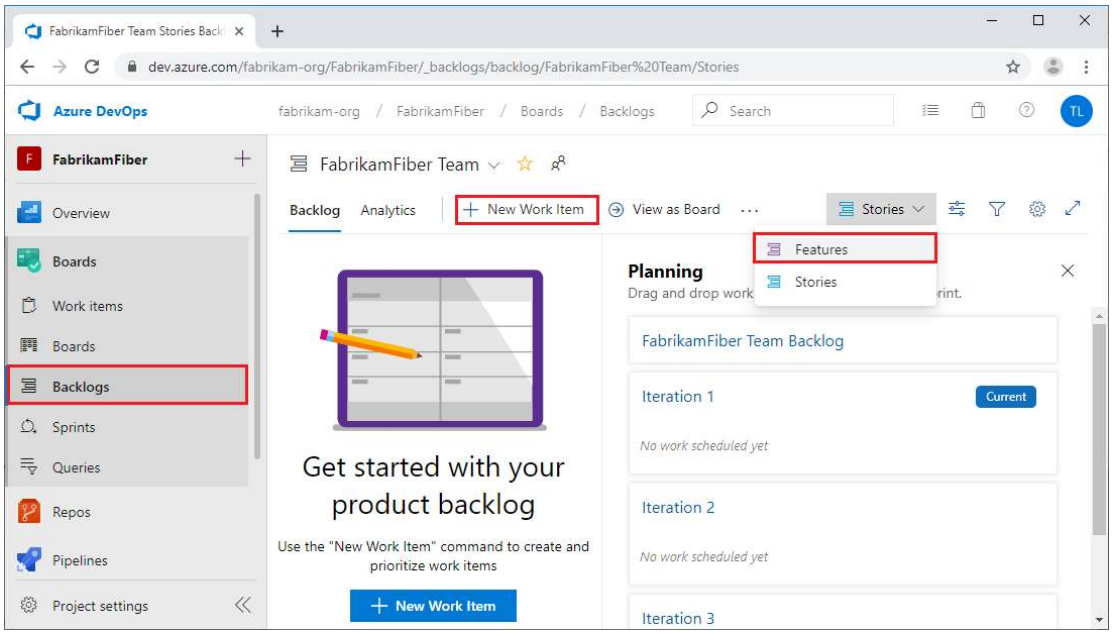
For more information about sprint planning in Azure Boards, see Assign backlog items to a sprint.

Add a Feature to the backlog

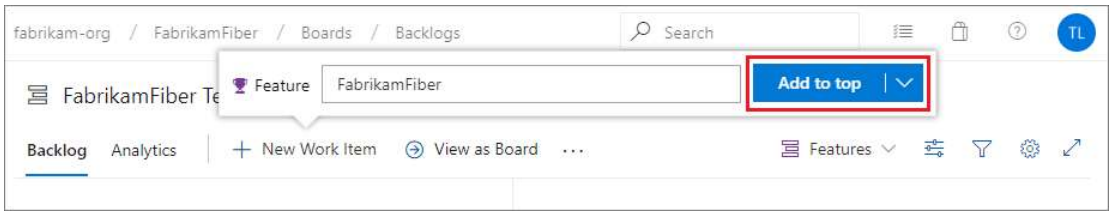
After your project and project code repository are created, you can add a Feature to the backlog to represent the work for your project.

1. From your project page, select **Boards** > **Backlogs** in the left navigation.

2. On the **Backlog** tab, if the work item type in the top bar is **Stories**, drop down and select **Features**. Then select **New Work Item.**



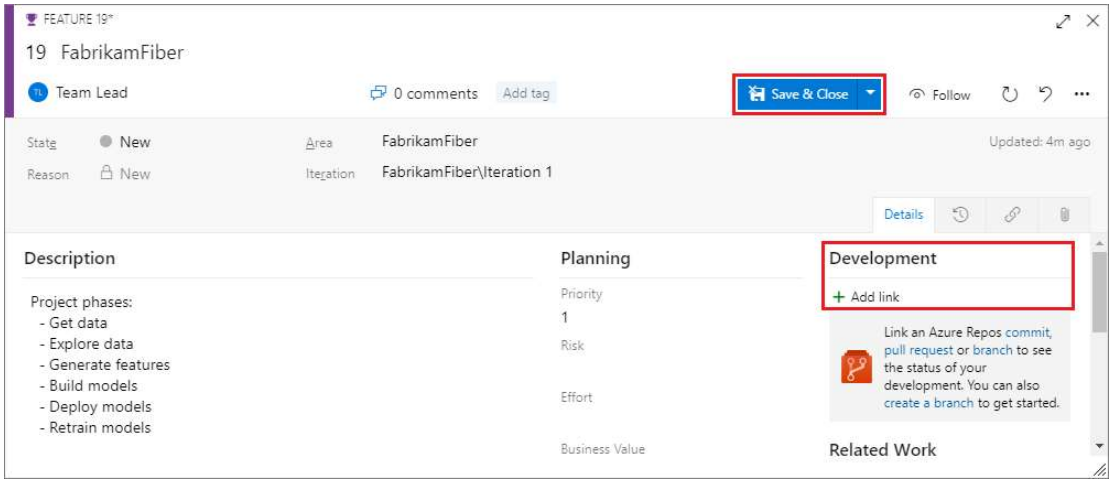
3. Enter a title for the Feature, usually your project name, and then select **Add to top**.



4. From the **Backlog** list, select and open the new Feature. Fill in the description, assign a team member, and set planning parameters.

   You can also link the Feature to the project's Azure Repos code repository by selecting **Add link** under the **Development** section.

   After you edit the Feature, select **Save & Close**.

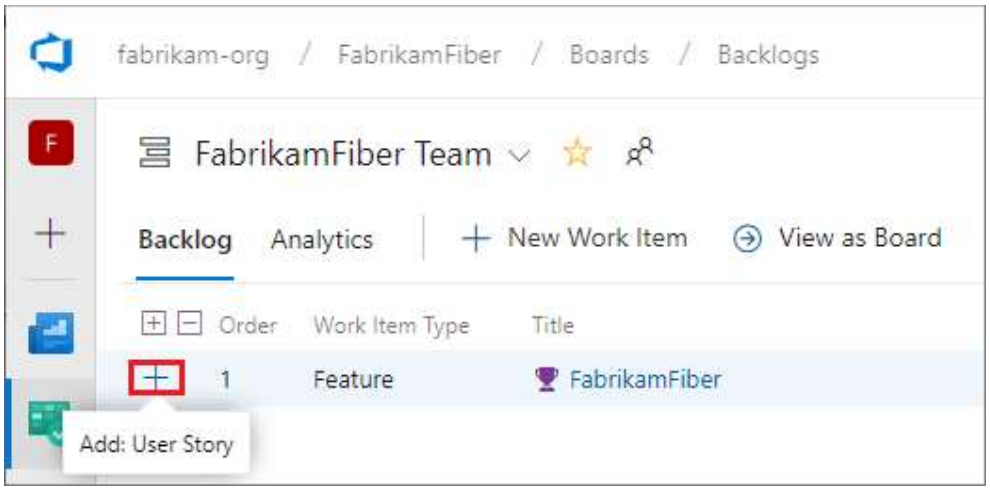


Add a User Story to the Feature

Under the Feature, you can add User Stories to describe major steps needed to complete the project.

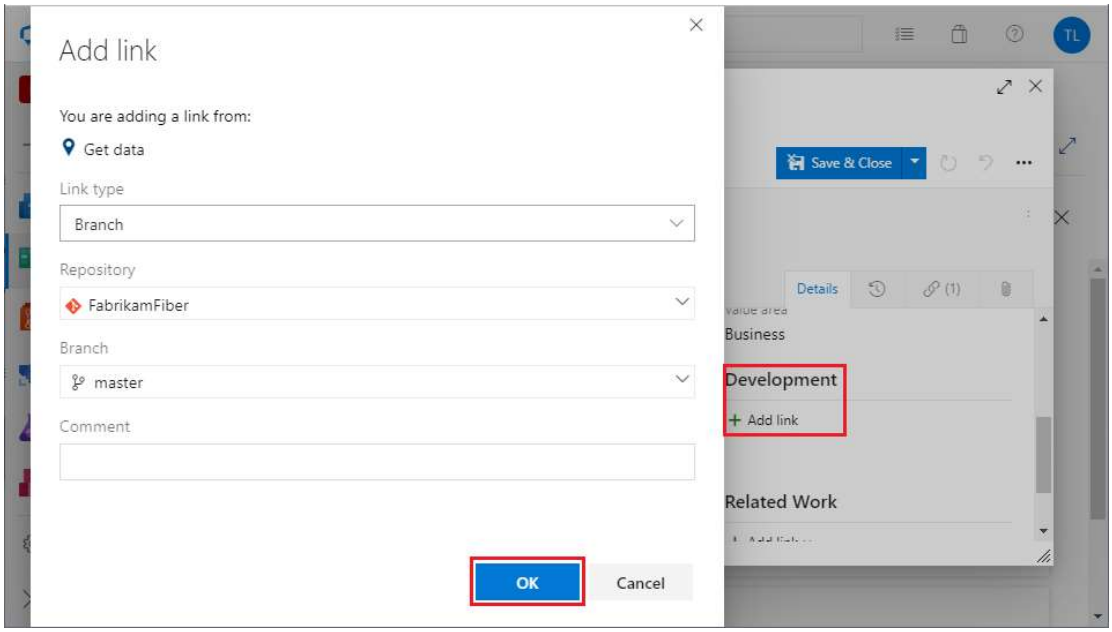
To add a new User Story to a Feature:

1. On the **Backlog** tab, select the + to the left of the Feature.



2. Give the User Story a title, and edit details such as assignment, status, description, comments, planning, and priority.

   You can also link the User Story to a branch of the project's Azure Repos code repository by selecting **Add link** under the **Development** section. Select the repository and branch you want to link the work item to, and then select **OK**.

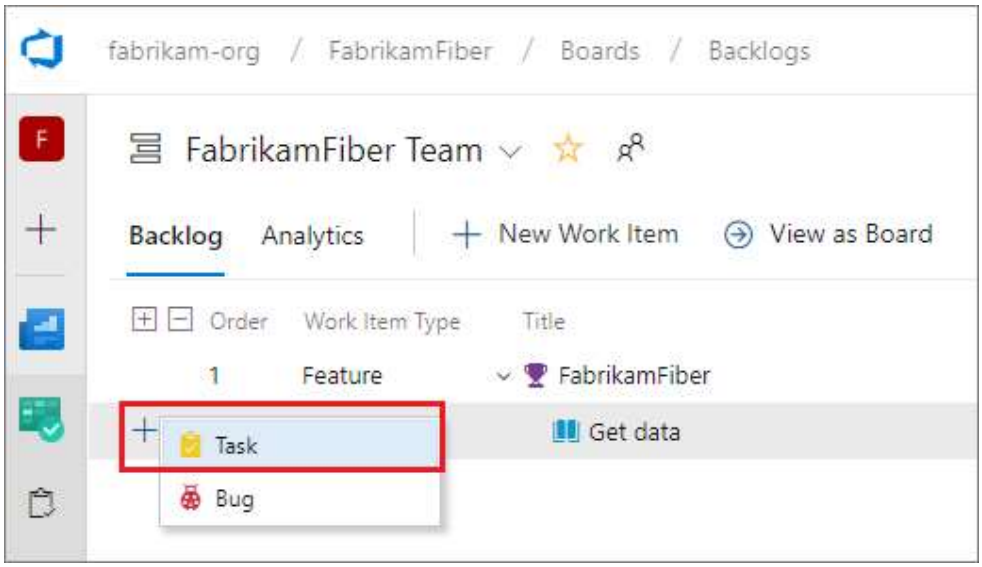


3. When you're finished editing the User Story, select **Save & Close**.

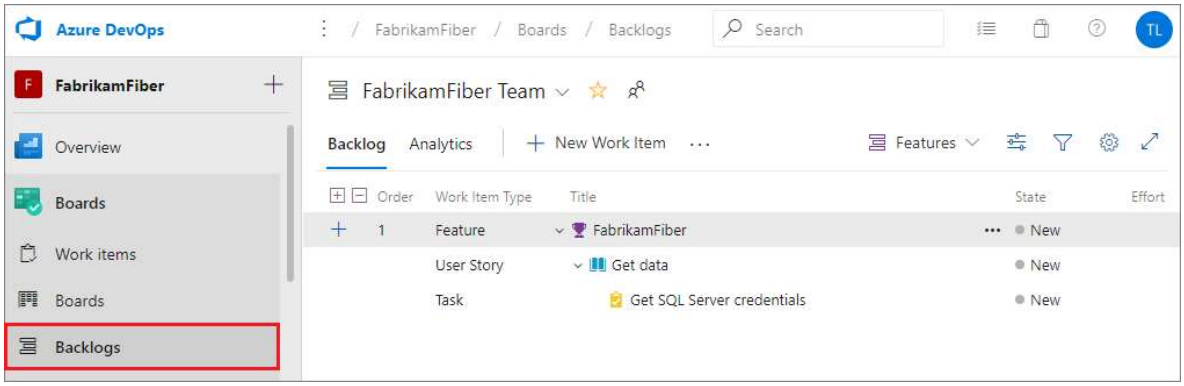
Add a Task to a User Story

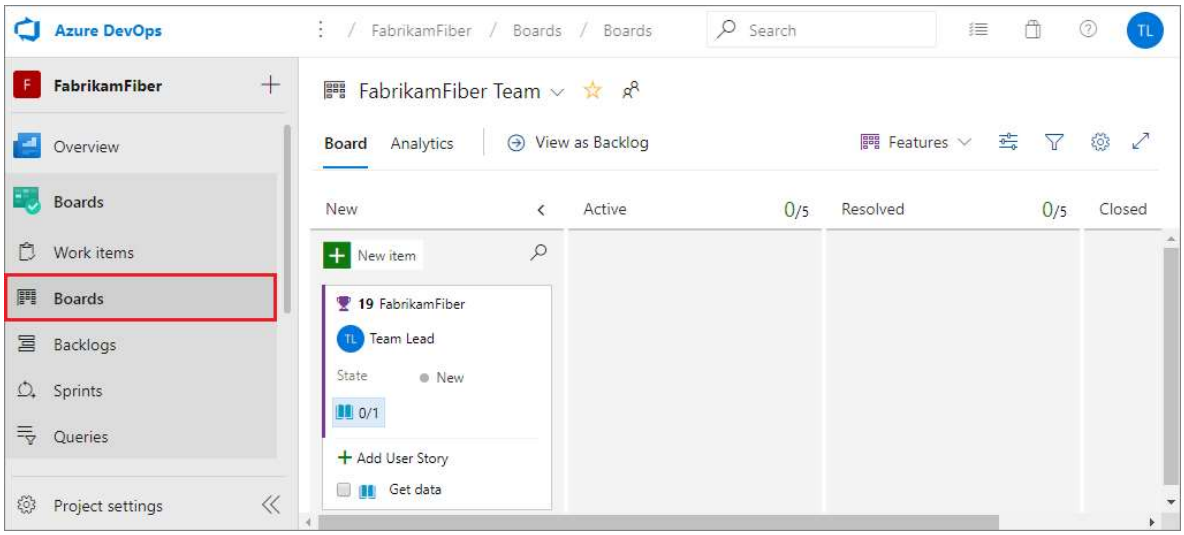
Tasks are specific detailed steps that are needed to complete each User Story. After all Tasks of a User Story are completed, the User Story should be completed too.

To add a Task to a User Story, select the + next to the User Story item, and select **Task**. Fill in the title and other information in the Task.



After you create Features, User Stories, and Tasks, you can view them in the **Backlogs** or **Boards** views to track their status.





Use an agile TDSP work template

Data scientists may feel more comfortable using an agile template that replaces Features, User Stories, and Tasks with TDSP lifecycle stages and substages. In Azure

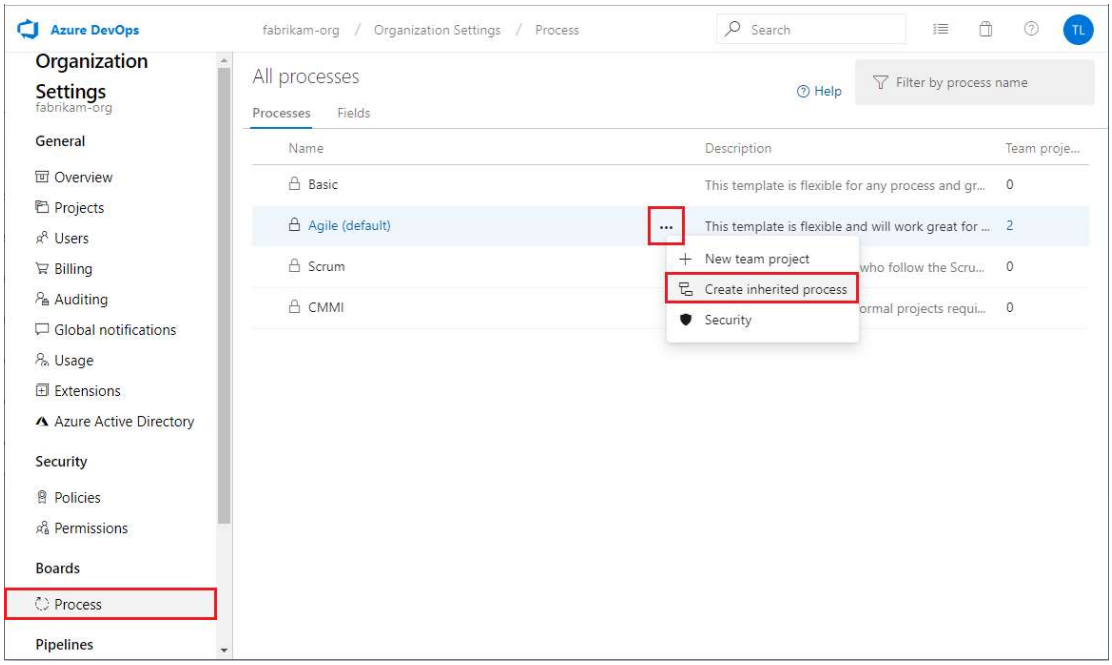
Boards, you can create an agile-derived template that uses TDSP lifecycle stages to create and track work items. The following steps walk through setting up a data science-specific agile process template and creating data science work items based on the template.

Set up an Agile Data Science Process template

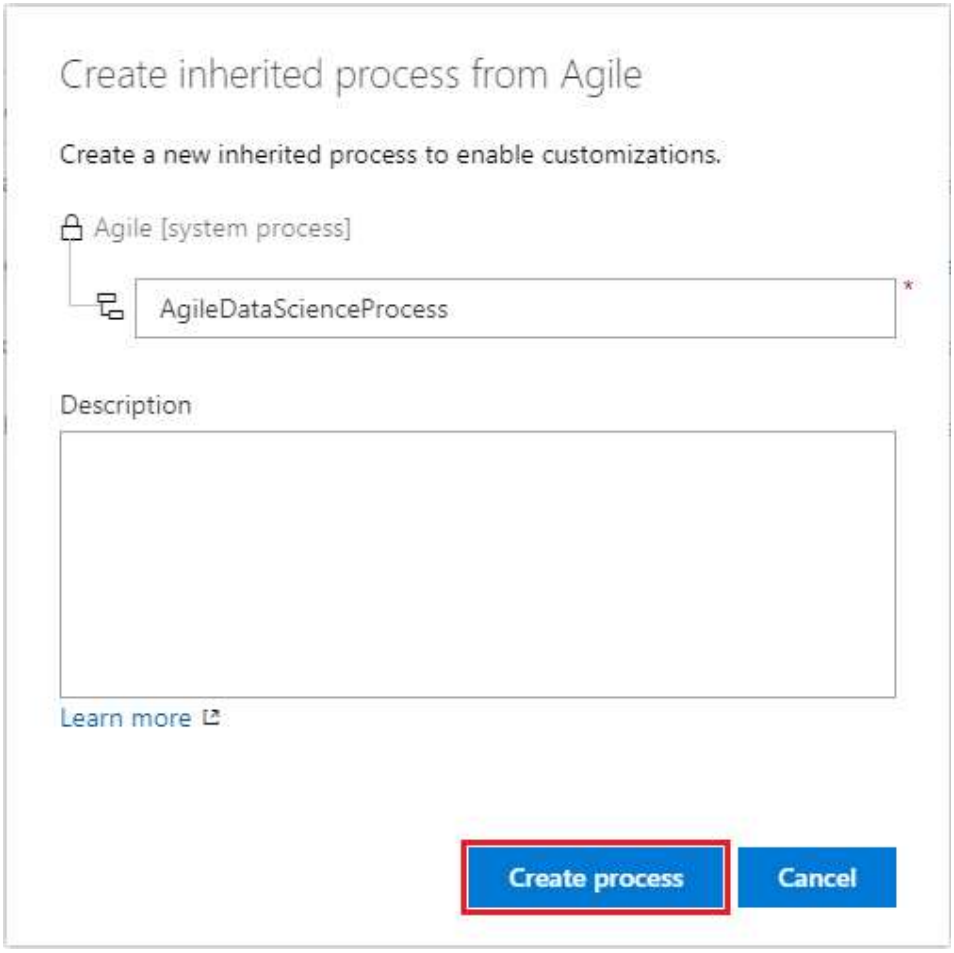
1. From your Azure DevOps organization main page, select **Organization settings** from the left navigation.

2. In the **Organization Settings** left navigation, under **Boards**, select **Process**.

3. In the **All processes** pane, select the **...** next to **Agile**, and then select **Create inherited process**.

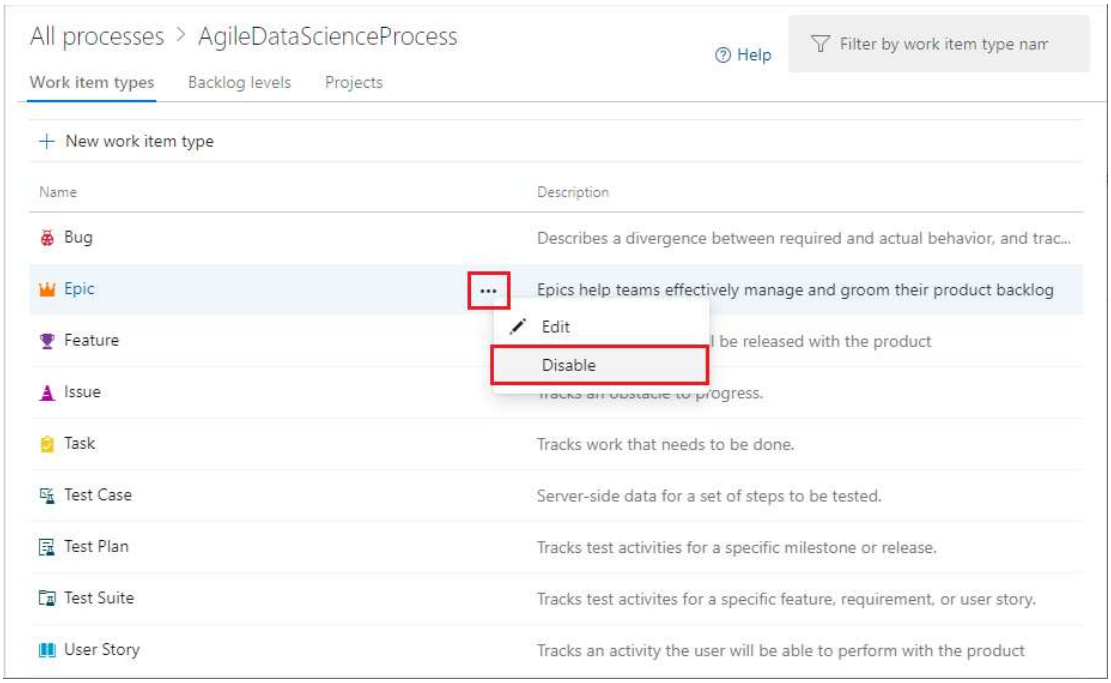


4. In the **Create inherited process from Agile** dialog, enter the name *AgileDataScienceProcess*, and select **Create process**.



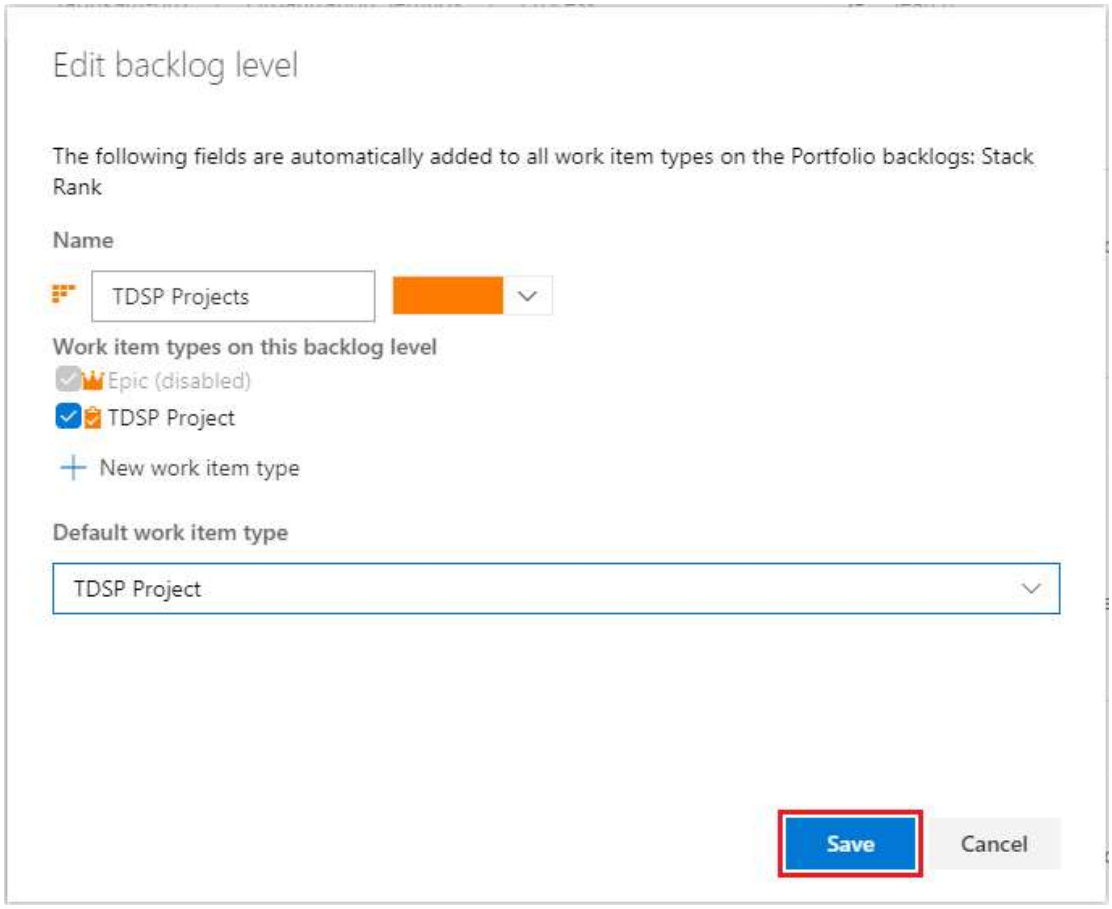
5. In **All processes**, select the new **AgileDataScienceProcess**.

6. On the **Work item types** tab, disable **Epic**, **Feature**, **User Story**, and **Task** by selecting the **...** next to each item and then selecting **Disable**.



7. In **All processes**, select the **Backlog levels** tab. Under **Portfolios backlogs**, select the **...** next to **Epic (disabled)**, and then select **Edit/Rename**.

8. In the **Edit backlog level** dialog box:
   a. Under **Name**, replace **Epic** with *TDSP Projects*.
   b. Under **Work item types on this backlog level**, select **New work item type**, enter *TDSP Project*, and select **Add**.
   c. Under **Default work item type**, drop down and select **TDSP Project**.
   d. Select **Save**.



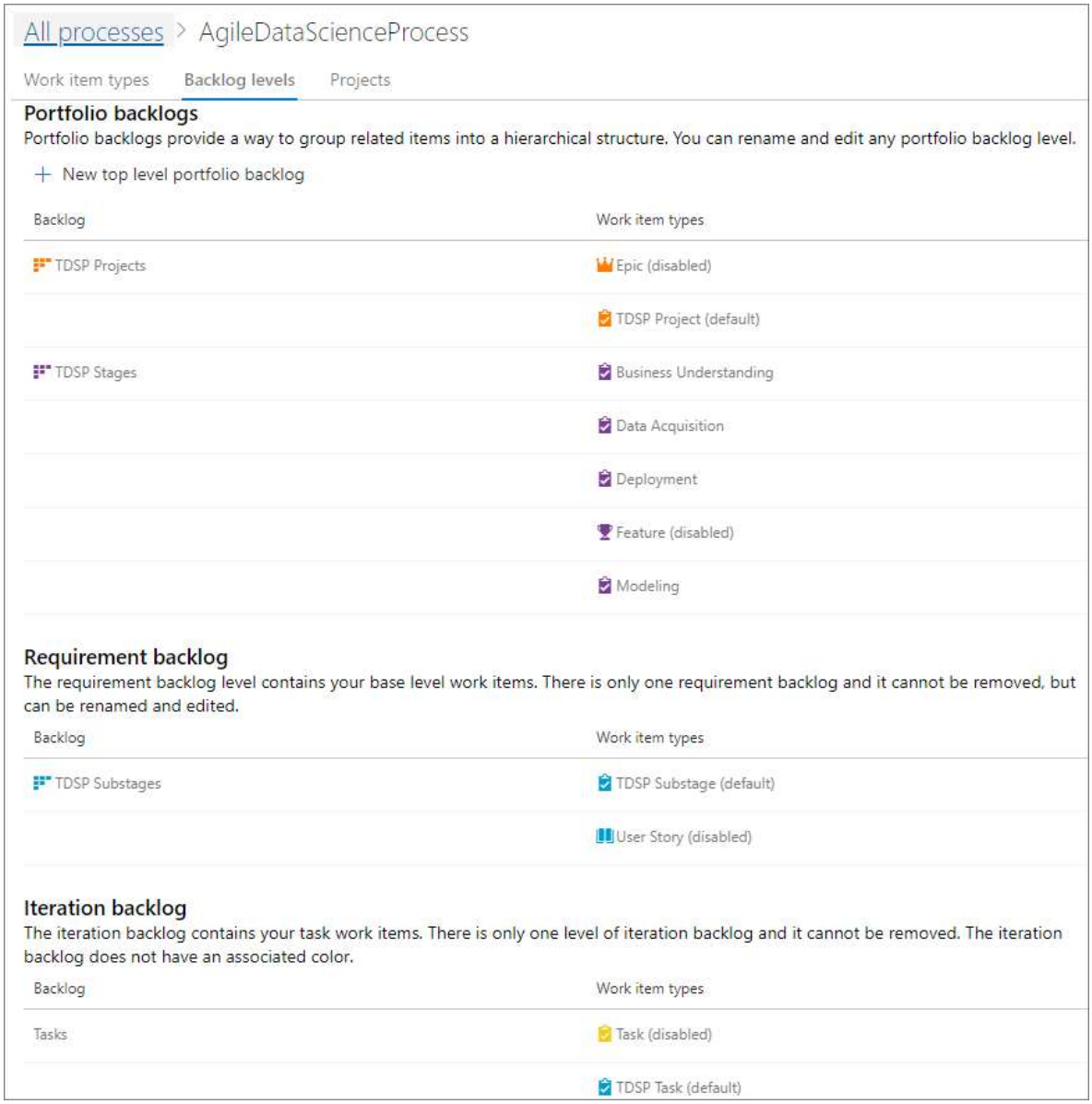
9. Follow the same steps to rename **Features** to *TDSP Stages*, and add the following new work item types:

   - *Business Understanding*
   - *Data Acquisition*
   - *Modeling*
   - *Deployment*

10. Under **Requirement backlog**, rename **Stories** to *TDSP Substages*, add the new work item type *TDSP Substage*, and set the default work item type to **TDSP Substage**.

11. Under **Iteration backlog**, add a new work item type *TDSP Task*, and set it to be the default work item type.

After you complete the steps, the backlog levels should look like this:



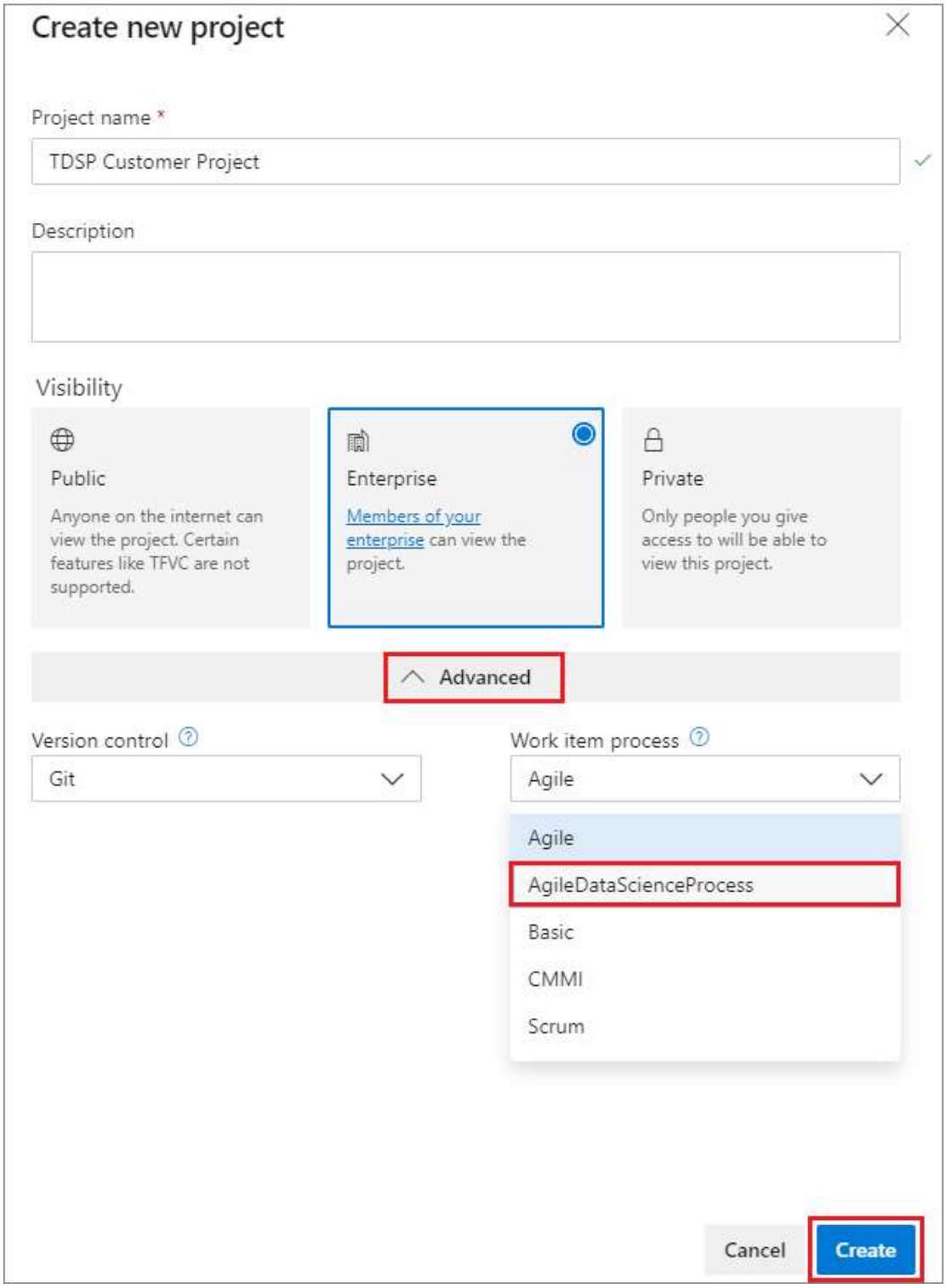
Create Agile Data Science Process work items

You can use the data science process template to create TDSP projects and track work items that correspond to TDSP lifecycle stages.

1. From your Azure DevOps organization main page, select **New project**.

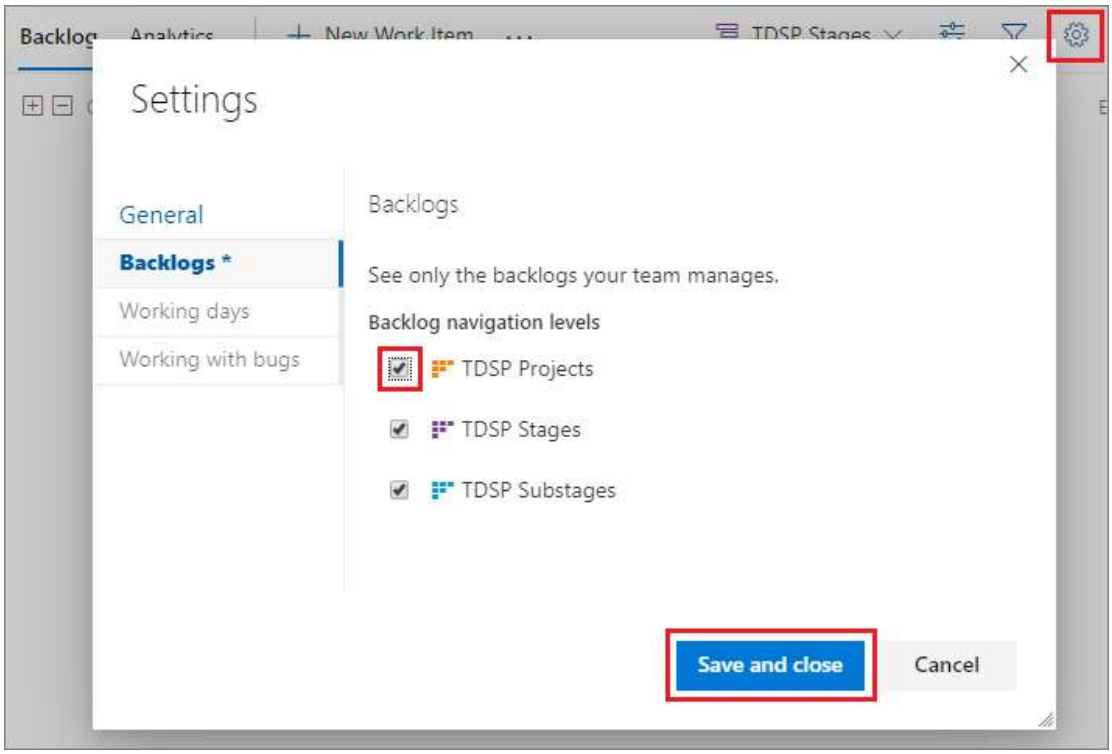
2. In the **Create new project** dialog, give your project a name, and then select **Advanced**.

3. Under **Work item process**, drop down and select **AgileDataScienceProcess**, and then select **Create**.



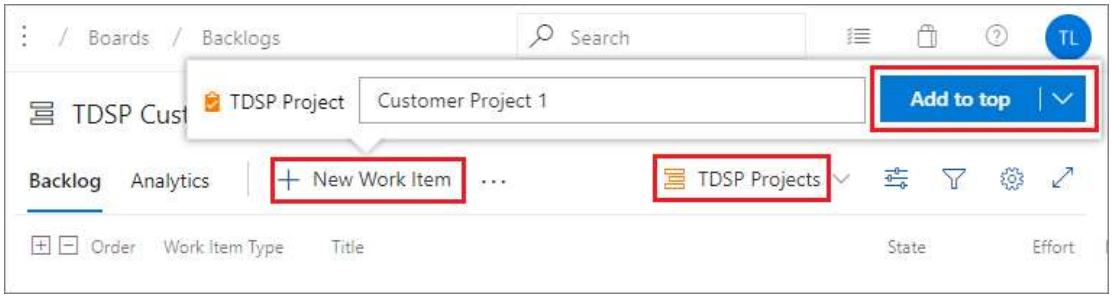
4. In the newly created project, select **Boards** > **Backlogs** in the left navigation.

5. To make TDSP Projects visible, select the **Configure team settings** icon. In the **Settings** screen, select the **TDSP Projects** check box, and then select **Save and close**.

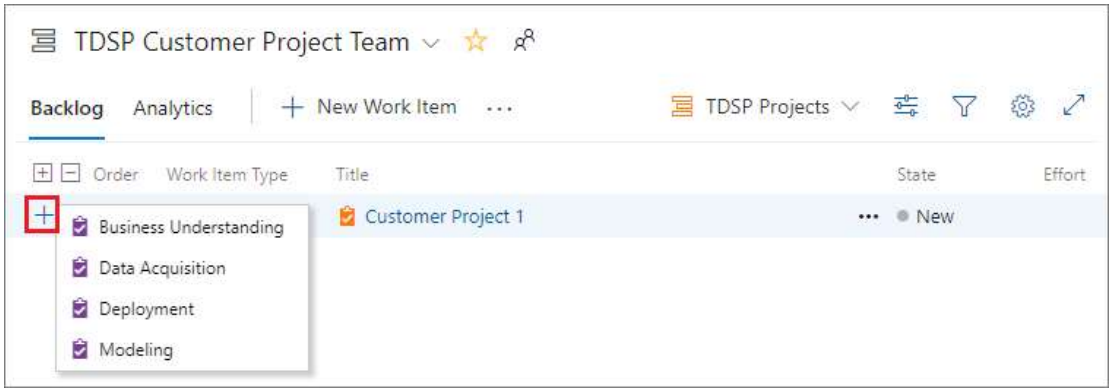


6. To create a data science-specific TDSP Project, select **TDSP Projects** in the top bar, and then select **New work item**.

7. In the popup, give the TDSP Project work item a name, and select **Add to top**.



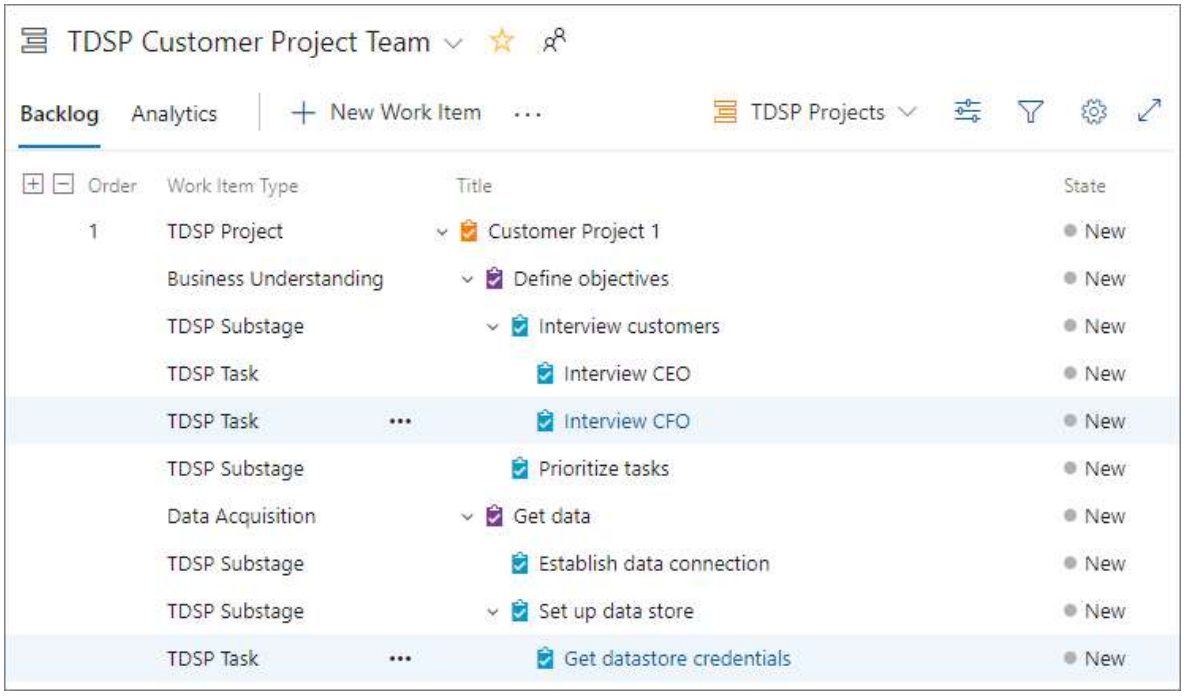
8. To add a work item under the TDSP Project, select the + next to the project, and then select the type of work item to create.



9. Fill in the details in the new work item, and select **Save & Close**.

10. Continue to select the + symbols next to work items to add new TDSP Stages, Substages, and Tasks.

Here is an example of how the data science project work items should appear in **Backlogs** view:



Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- Collaborative coding with Git describes how to do collaborative code development for data science projects using Git as the shared code development framework, and how to link these coding activities to the work planned with the agile process.

Additional resources on agile processes:

- Agile process

- Agile process work item types and workflow

Related resources

- What is the Team Data Science Process?
- Compare the machine learning products and technologies from Microsoft
- Machine learning at scale

Collaborative coding with Git

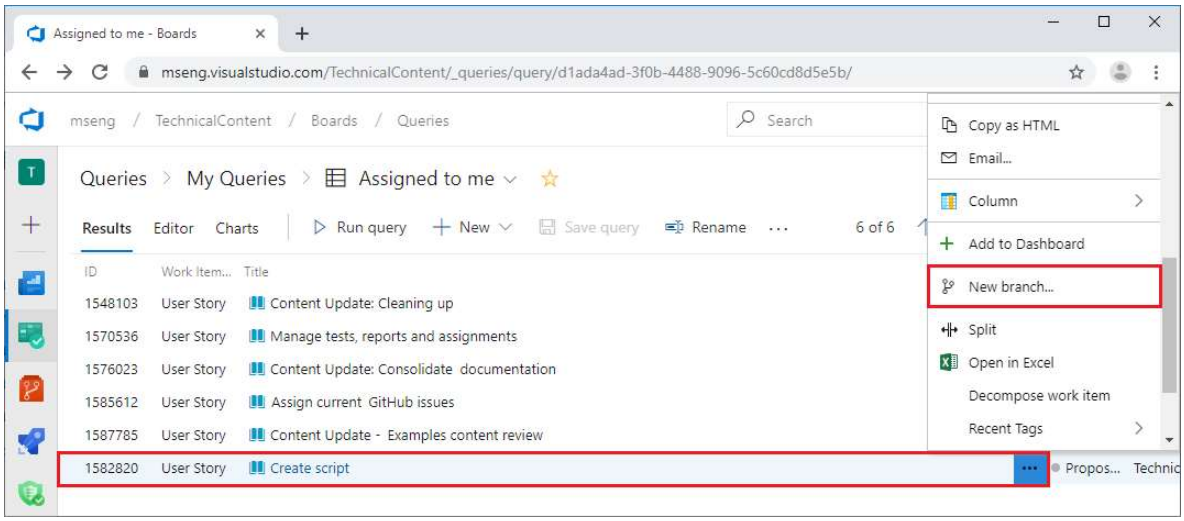
Article • 11/15/2022

This article describes how to use Git as the collaborative code development framework for data science projects. The article covers how to link code in Azure Repos to agile development work items in Azure Boards, how to do code reviews, and how to create and merge pull requests for changes.

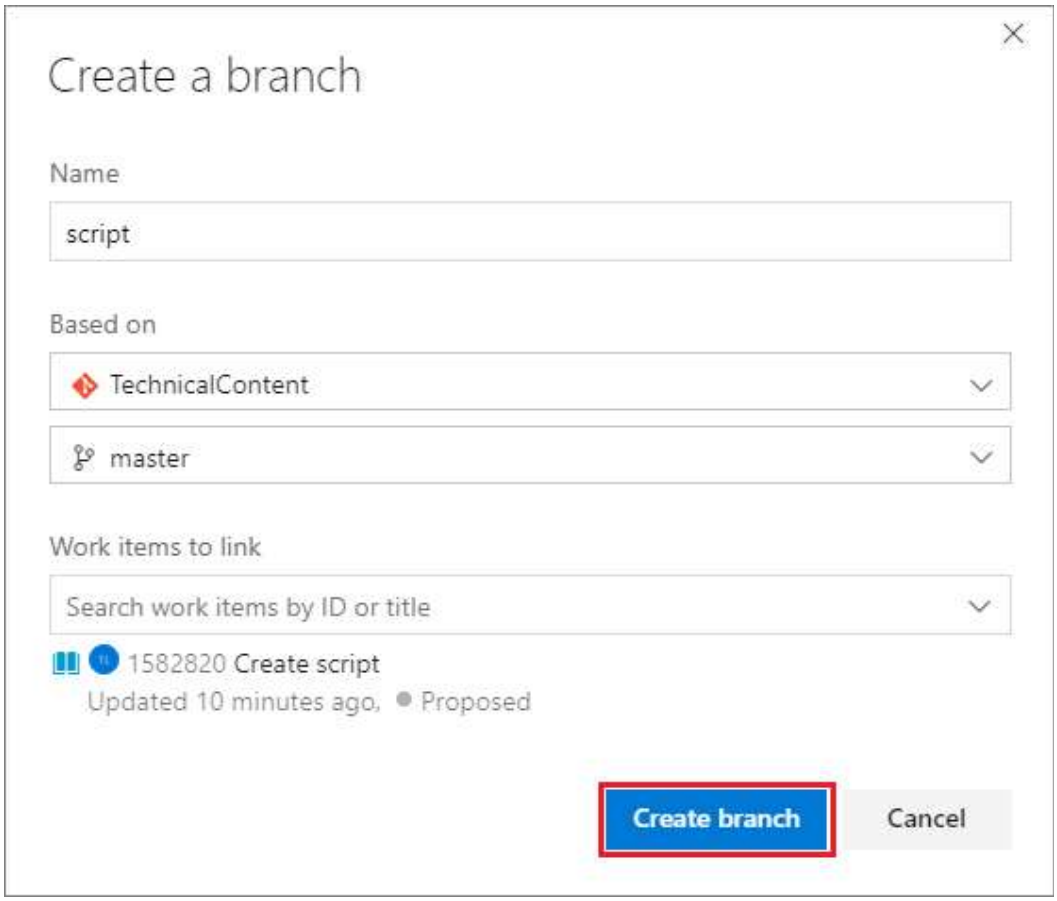
Link a work item to an Azure Repos branch

Azure DevOps provides a convenient way to connect an Azure Boards User Story or Task work item with an Azure Repos Git repository branch. You can link your User Story or Task directly to the code associated with it.

To connect a work item to a new branch, select the **Actions** ellipsis (...) next to the work item, and on the context menu, scroll to and select **New branch**.



In the **Create a branch** dialog, provide the new branch name and the base Azure Repos Git repository and branch. The base repository must be in the same Azure DevOps project as the work item. The base branch can be any existing branch. Select **Create branch**.



You can also create a new branch using the following Git bash command in Windows or Linux:

```
Bash

git checkout -b <new branch name> <base branch name>
```

If you don't specify a <base branch name>, the new branch is based on `main`.

To switch to your working branch, run the following command:

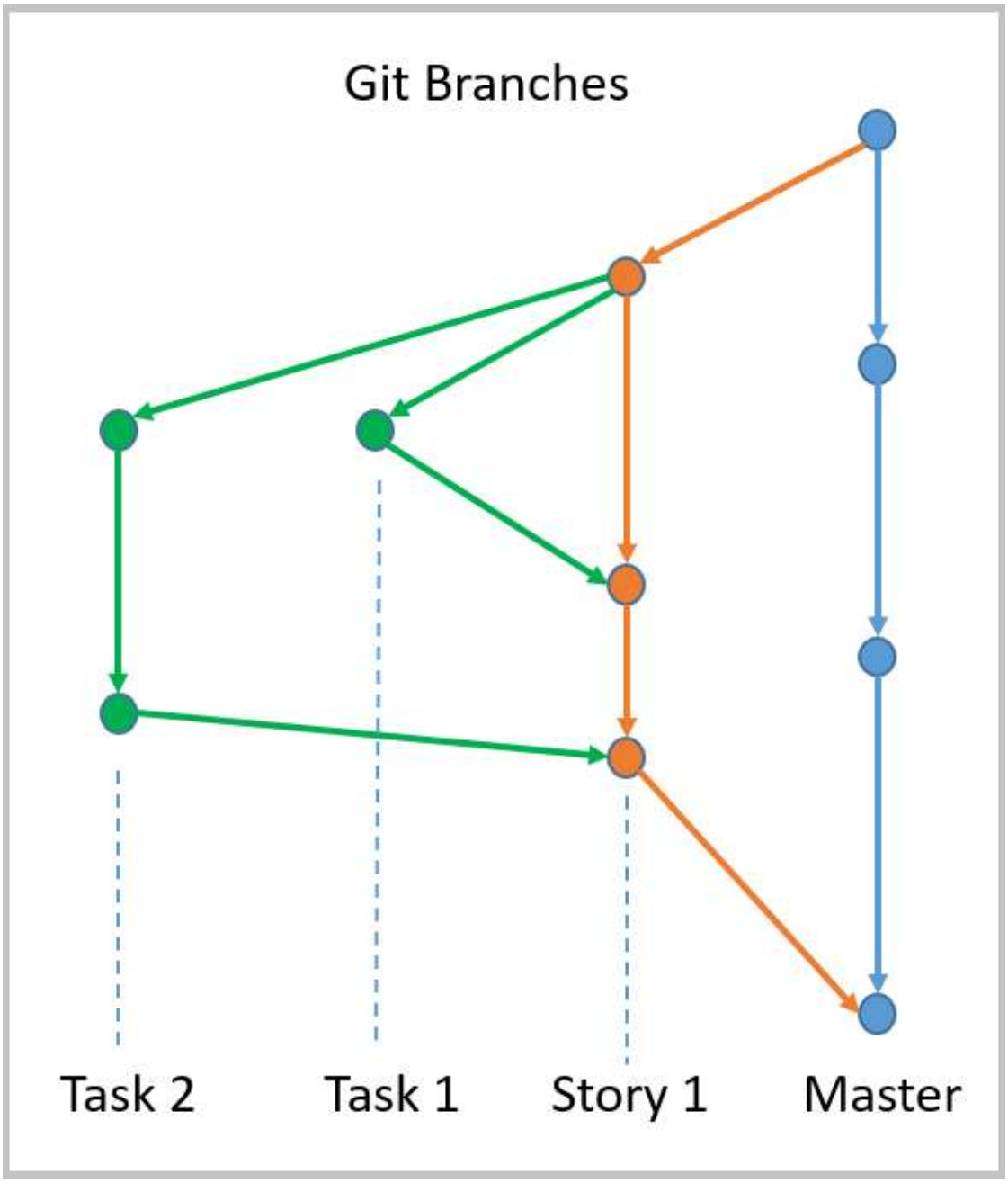
```
Bash

git checkout <working branch name>
```

After you switch to the working branch, you can start developing code or documentation artifacts to complete the work item. Running `git checkout main` switches you back to the `main` branch.

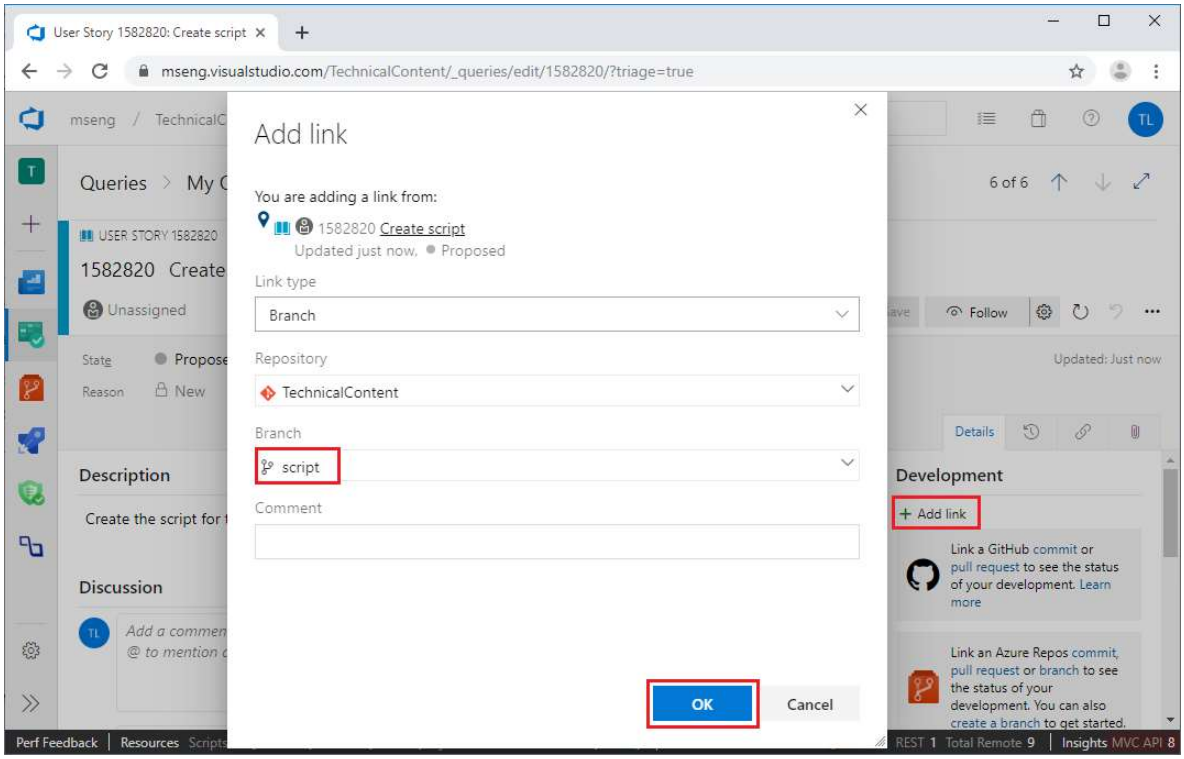
It's a good practice to create a Git branch for each User Story work item. Then, for each Task work item, you can create a branch based on the User Story branch. Organize the branches in a hierarchy that corresponds to the User Story-Task relationship when you

have multiple people working on different User Stories for the same project, or on different Tasks for the same User Story. You can minimize conflicts by having each team member work on a different branch, or on different code or other artifacts when sharing a branch.

The following diagram shows the recommended branching strategy for TDSP. You might not need as many branches as shown here, especially when only one or two people work on a project, or only one person works on all Tasks of a User Story. But separating the development branch from the primary branch is always a good practice, and can help prevent the release branch from being interrupted by development activities. For a complete description of the Git branch model, see A Successful Git Branching Model.



You can also link a work item to an existing branch. On the **Detail** page of a work item, select **Add link**. Then select an existing branch to link the work item to, and select **OK**.

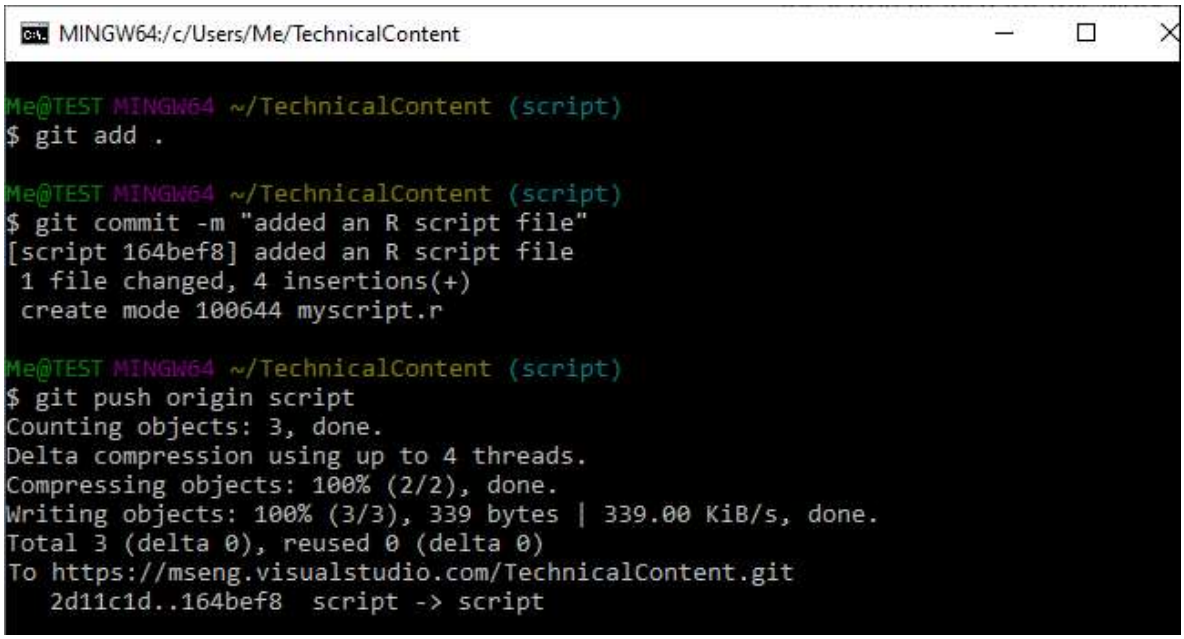


Work on the branch and commit changes

After you make a change for your work item, such as adding an R script file to your local machine's `script` branch, you can commit the change from your local branch to the upstream working branch by using the following Git bash commands:

Bash

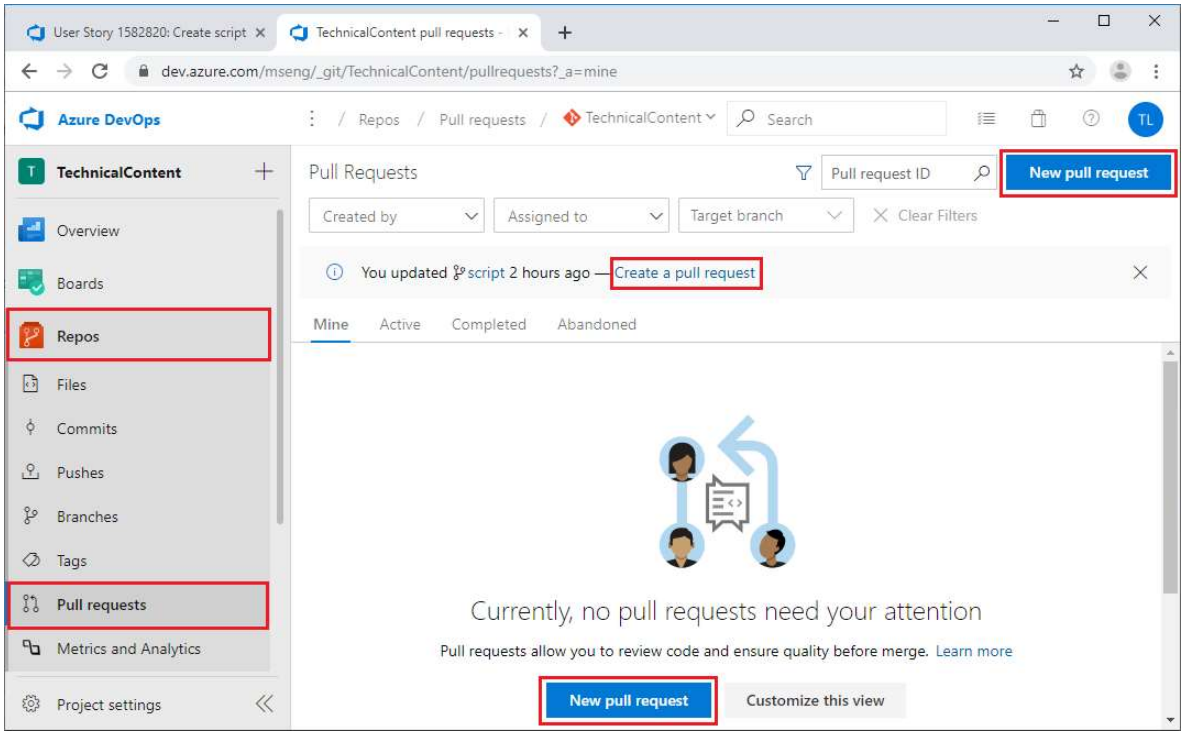
```
git status
git add .
git commit -m "added an R script file"
git push origin script
```



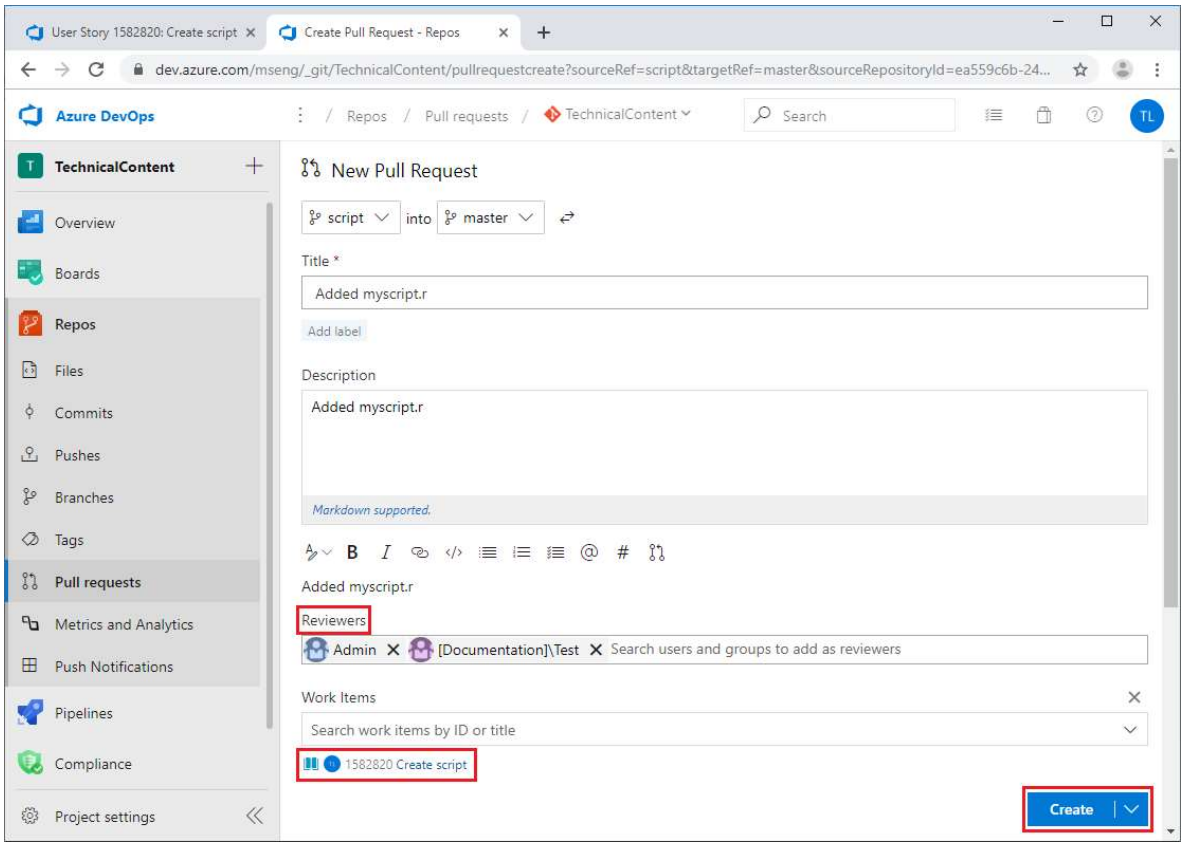
Create a pull request

After one or more commits and pushes, when you're ready to merge your current working branch into its base branch, you can create and submit a *pull request* in Azure Repos.

From the main page of your Azure DevOps project, point to **Repos** > **Pull requests** in the left navigation. Then select either of the **New pull request** buttons, or the **Create a pull request** link.

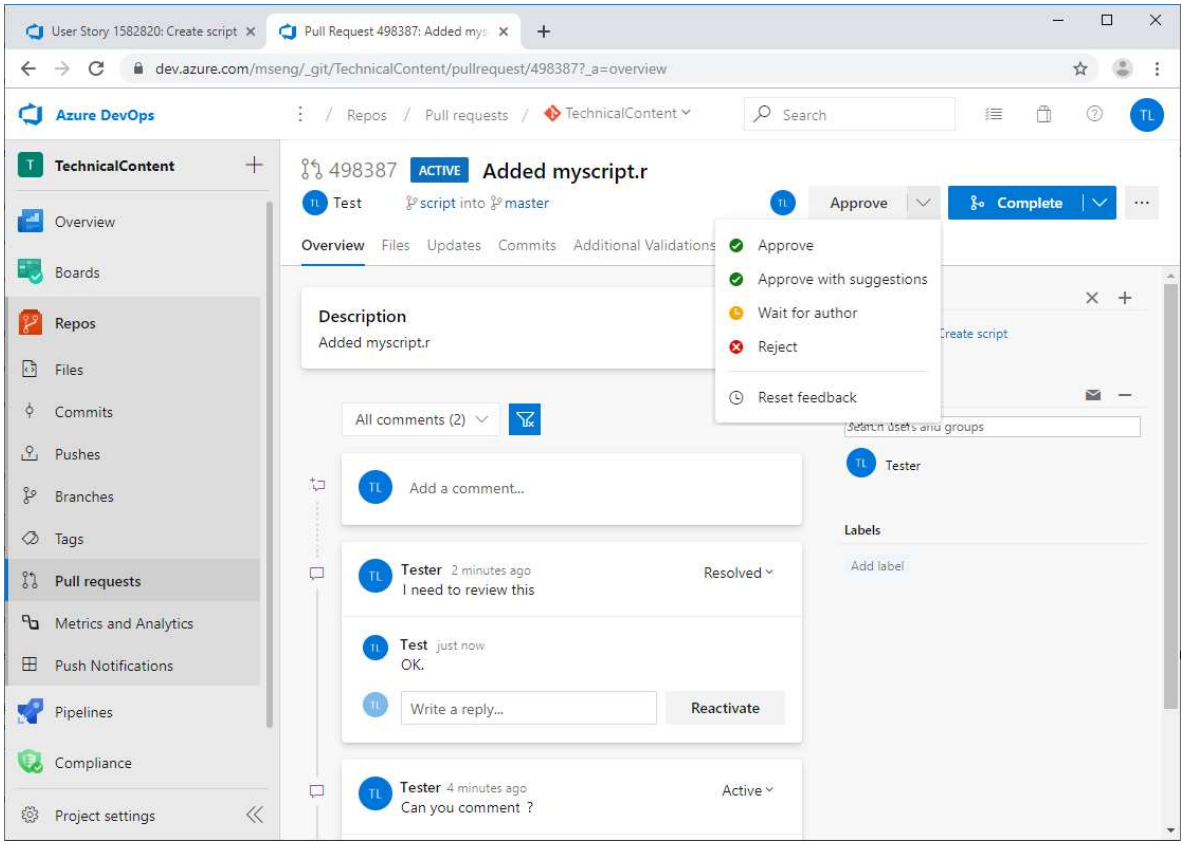


On the **New Pull Request** screen, if necessary, navigate to the Git repository and branch you want to merge your changes into. Add or change any other information you want. Under **Reviewers**, add the names of the reviewers, and then select **Create**.

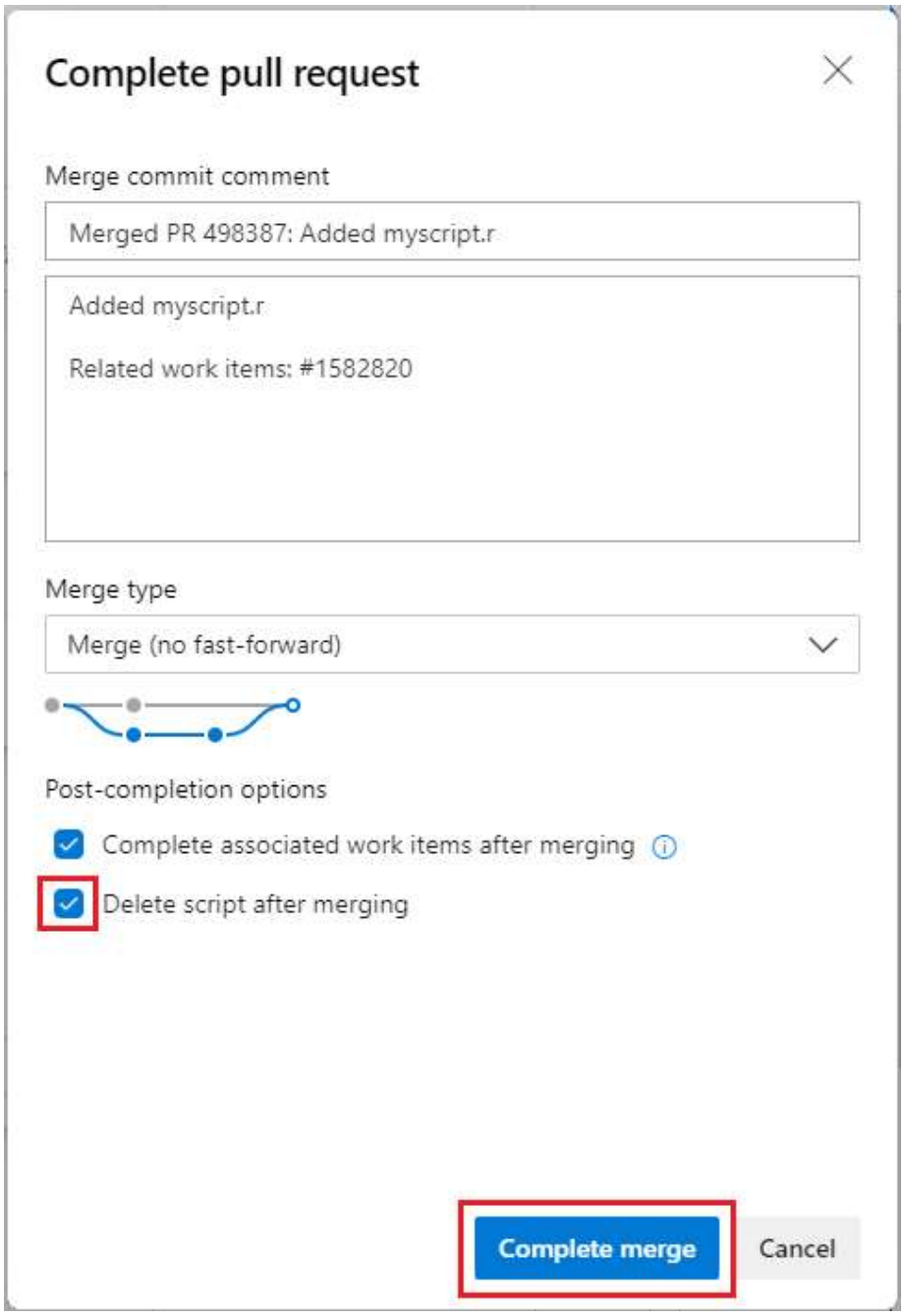


Review and merge

Once you create the pull request, your reviewers get an email notification to review the pull request. The reviewers test whether the changes work, and check the changes with the requester if possible. The reviewers can make comments, request changes, and approve or reject the pull request based on their assessment.

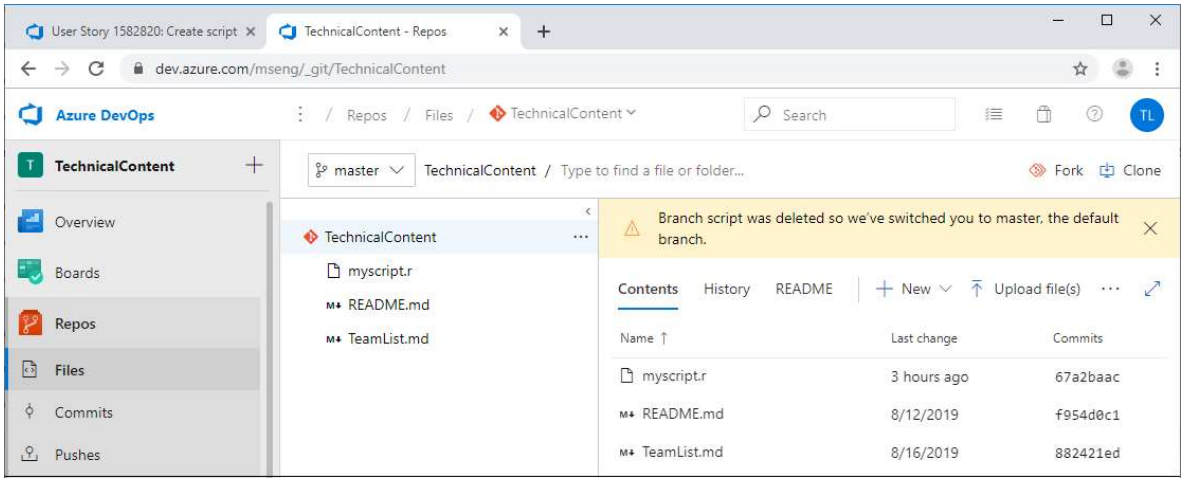


After the reviewers approve the changes, you or someone else with merge permissions can merge the working branch to its base branch. Select **Complete**, and then select **Complete merge** in the **Complete pull request** dialog. You can choose to delete the working branch after it has merged.



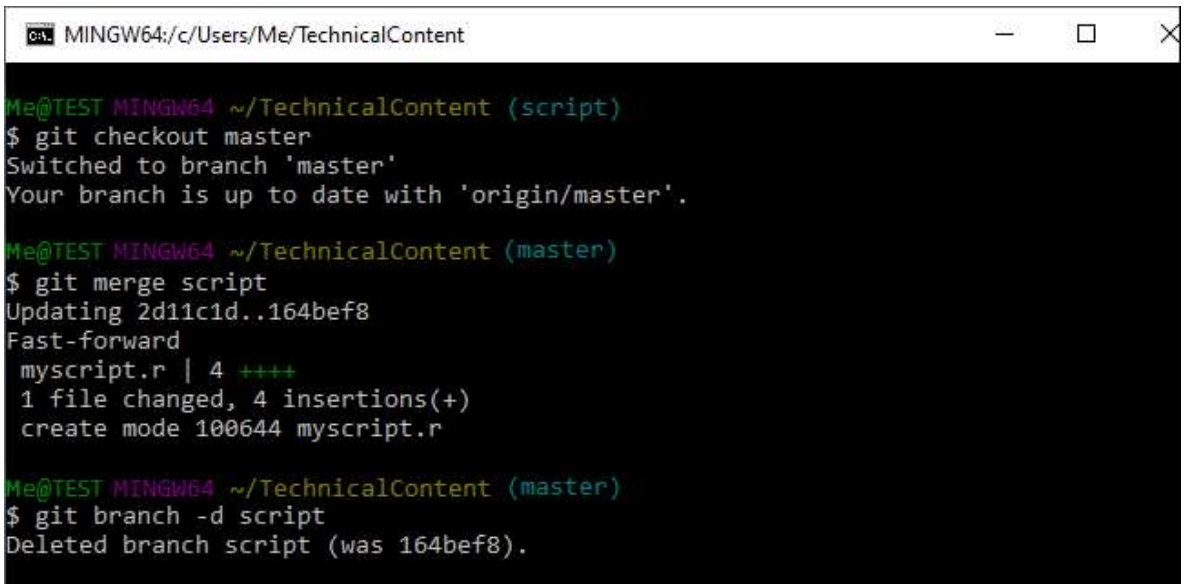
Confirm that the request is marked as **COMPLETED**.

When you go back to **Repos** in the left navigation, you can see that you've been switched to the main branch since the `script` branch was deleted.



You can also use the following Git bash commands to merge the `script` working branch to its base branch and delete the working branch after merging:

```bash
git checkout main
git merge script
git branch -d script
```



Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Execute data science tasks shows how to use utilities to complete several common data science tasks, such as interactive data exploration, data analysis, reporting, and model creation.

Related resources

- What is the Team Data Science Process?
- Compare the machine learning products and technologies from Microsoft
- Machine learning at scale

Execute data science tasks: exploration, modeling, and deployment

Article • 11/15/2022

Typical data science tasks include data exploration, modeling, and deployment. This article outlines the tasks to complete several common data science tasks such as interactive data exploration, data analysis, reporting, and model creation. Options for deploying a model into a production environment may include:

- Recommended: Azure Machine Learning
- Possible: SQL-Server with ML services

1. Exploration

A data scientist can perform exploration and reporting in a variety of ways: by using libraries and packages available for Python (matplotlib for example) or with R (ggplot or lattice for example). Data scientists can customize such code to fit the needs of data exploration for specific scenarios. The needs for dealing with structured data are different that for unstructured data such as text or images.

Products such as Azure Machine Learning also provide advanced data preparation for data wrangling and exploration, including feature creation. The user should decide on the tools, libraries, and packages that best suite their needs.

The deliverable at the end of this phase is a data exploration report. The report should provide a fairly comprehensive view of the data to be used for modeling and an assessment of whether the data is suitable to proceed to the modeling step.

2. Modeling

There are numerous toolkits and packages for training models in a variety of languages. Data scientists should feel free to use which ever ones they are comfortable with, as long as performance considerations regarding accuracy and latency are satisfied for the relevant business use cases and production scenarios.

Model management

After multiple models have been built, you usually need to have a system for registering and managing the models. Typically you need a combination of scripts or APIs and a

backend database or versioning system. Azure Machine Learning provides deployment of ONNX models or deployment of ML Flow models.

3. Deployment

Production deployment enables a model to play an active role in a business. Predictions from a deployed model can be used for business decisions.

Production platforms

There are various approaches and platforms to put models into production. We recommend deployment to Azure Machine Learning.

> **Note**
>
> Prior to deployment, one has to ensure the latency of model scoring is low enough to use in production.

A/B testing

When multiple models are in production, it can be useful to perform A/B testing to compare performance of the models.

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Track progress of data science projects shows how a data scientist can track the progress of a data science project.

Model operation and CI/CD shows how CI/CD can be performed with developed models.

Test data science code with Azure DevOps

Article • 11/15/2022

This article gives preliminary guidelines for testing code in a data science workflow, using Azure DevOps. Such testing gives data scientists a systematic and efficient way to check the quality and expected outcome of their code. We use a Team Data Science Process (TDSP) project that uses the UCI Adult Income dataset that we published earlier to show how code testing can be done.

Introduction on code testing

"Unit testing" is a longstanding practice for software development. But for data science, it's often not clear what "unit testing" means and how you should test code for different stages of a data science lifecycle, such as:

- Data preparation
- Data quality examination
- Modeling
- Model deployment

This article replaces the term "unit testing" with "code testing." It refers to testing as the functions that help to assess if code for a certain step of a data science lifecycle is producing results "as expected." The person who's writing the test defines what's "as expected," depending on the outcome of the function--for example, data quality check or modeling.

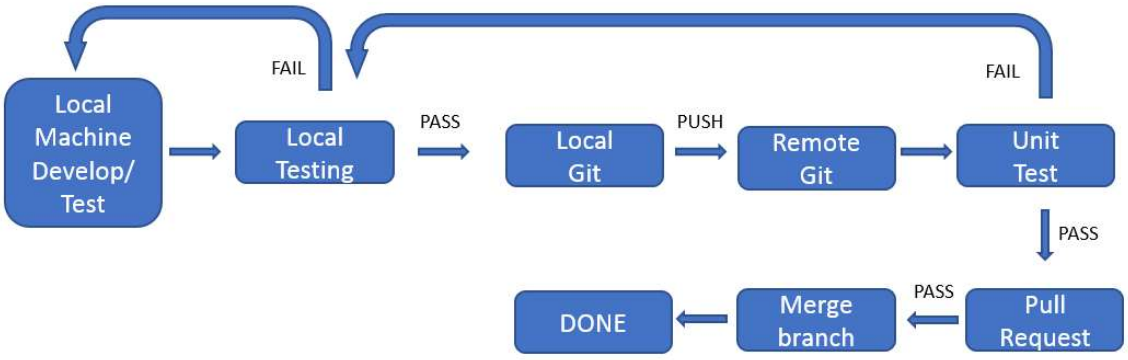
This article provides references as useful resources.

Azure DevOps for the testing framework

This article describes how to perform and automate testing by using Azure DevOps. You might decide to use alternative tools. We also show how to set up an automatic build by using Azure DevOps and build agents. For build agents, we use Azure Data Science Virtual Machines (DSVMs).

Flow of code testing

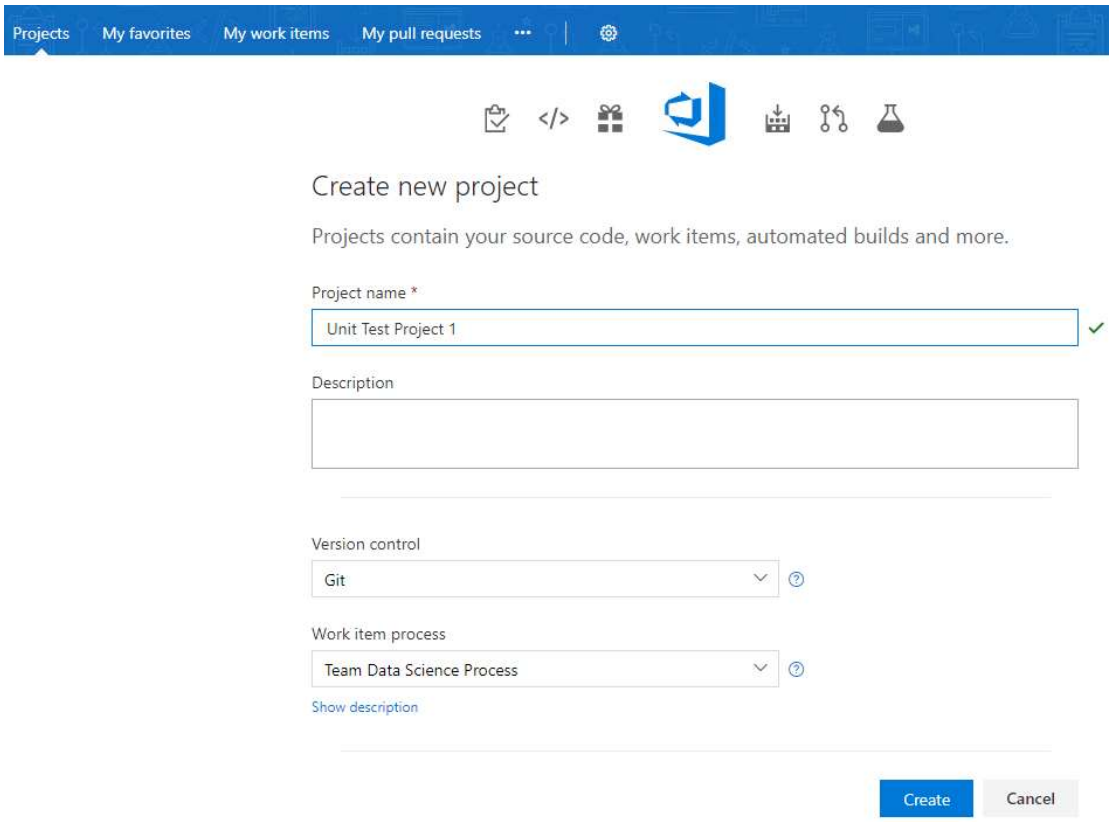
The overall workflow of testing code in a data science project looks like this:



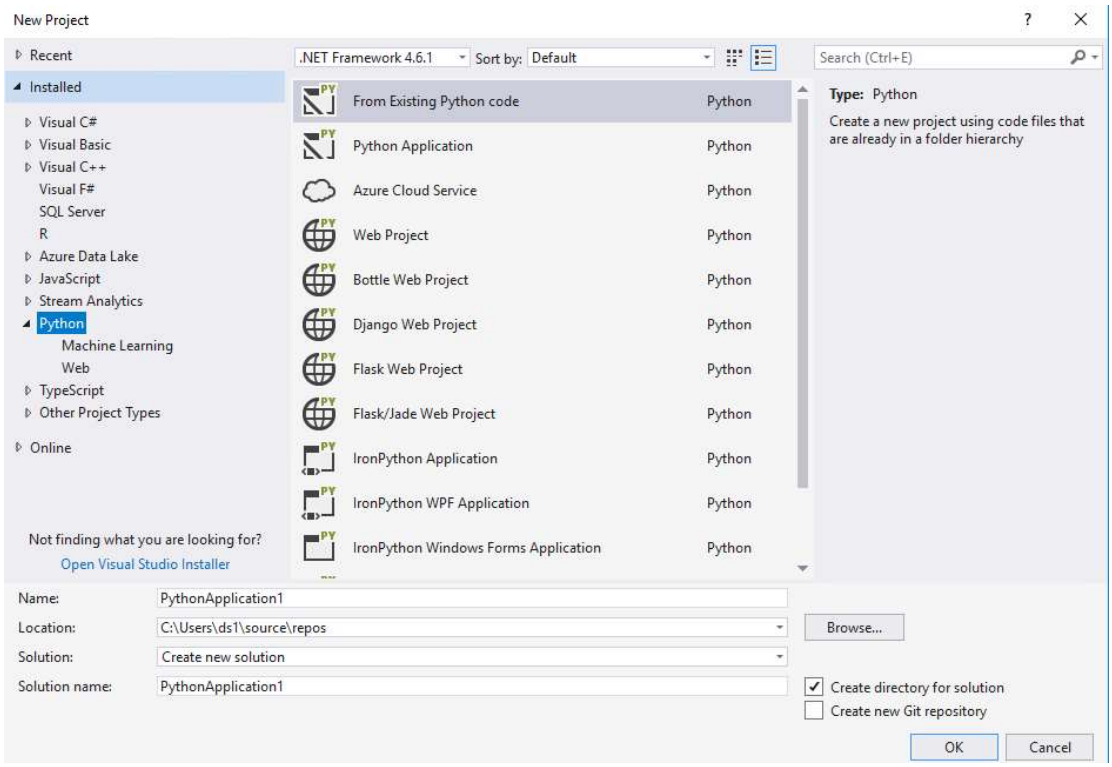
Detailed steps

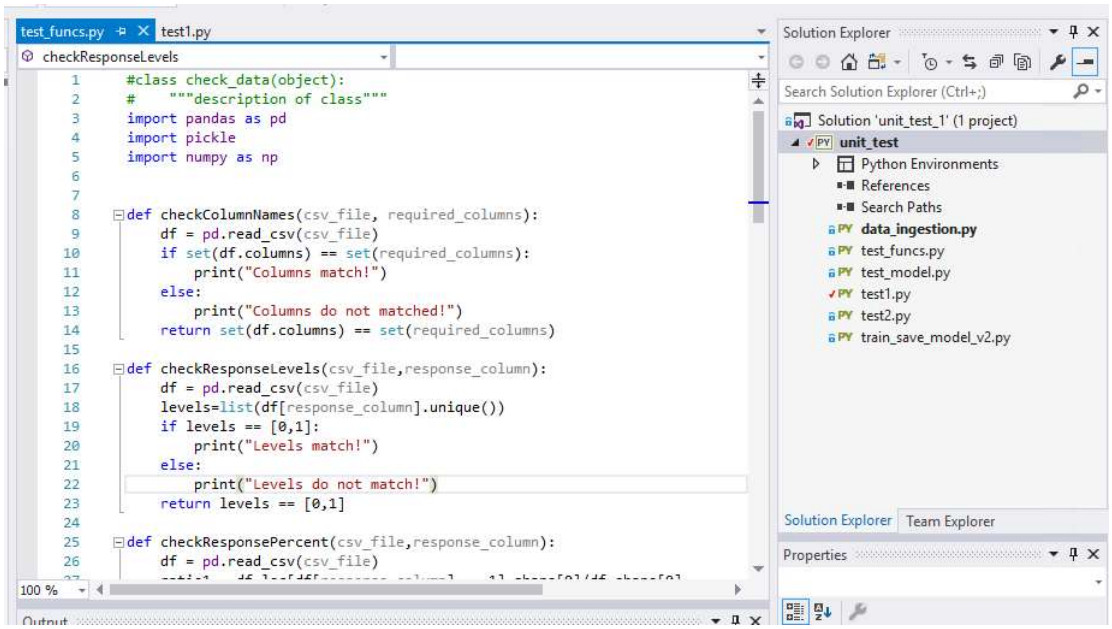
Use the following steps to set up and run code testing and an automated build by using a build agent and Azure DevOps:

1. Create a project in the Visual Studio desktop application:

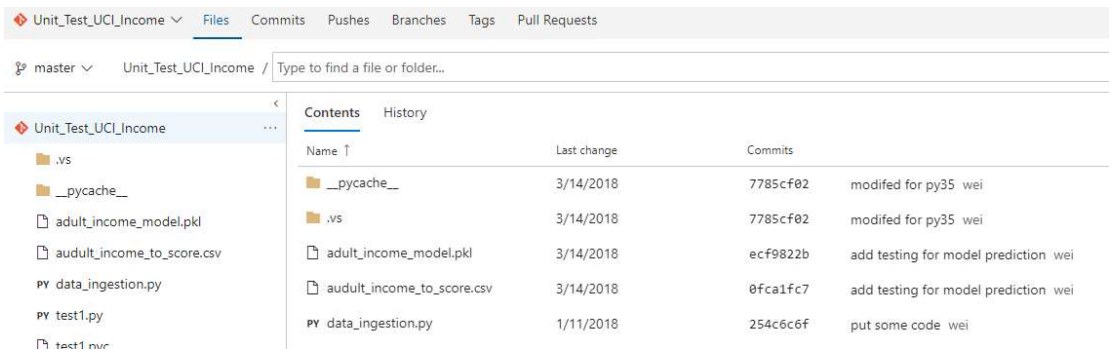


After you create your project, you'll find it in Solution Explorer in the right pane:





2. Feed your project code into the Azure DevOps project code repository:



3. Suppose you've done some data preparation work, such as data ingestion, feature engineering, and creating label columns. You want to make sure your code is generating the results that you expect. Here's some code that you can use to test whether the data-processing code is working properly:

   - Check that column names are right:

```
def checkColumnNames(csv_file, required_columns):
    df = pd.read_csv(csv_file)
    if set(df.columns) == set(required_columns):
        print("Columns match!")
    else:
        print("Columns do not matched!")
    return set(df.columns) == set(required_columns)
```

   Check that response levels are right:

```
def checkResponseLevels(csv_file,response_column):
    df = pd.read_csv(csv_file)
    levels=list(df[response_column].unique())
    if levels == [0,1]:
        print("Levels match!")
    else:
        print("Levels do not match!")
    return levels == [0,1]
```

   - Check that response percentage is reasonable:

```
def checkResponsePercent(csv_file,response_column):
    df = pd.read_csv(csv_file)
    ratio1 = df.loc[df[response_column] == 1].shape[0]/df.shape[0]
    if ratio1 > 0.5:
        print("Response levels(0/1) are messed up!")
    else:
        print("Response 0/1 are OK, and OK, Happy Friday!")
    return ratio1 < 0.5
```

   - Check the missing rate of each column in the data:

```
def checkMissingRate2(csv_file, threshold):
    df = pd.read_csv(csv_file)
    for x in df.columns:
        miss_rate = df[x].isnull().sum()/df.shape[0]
        if miss_rate > threshold:
            print("{} has more than {} missing values!".format(x,threshold))
            return False
    print("No column has missing values more than {}!".format(threshold))
    return True
```

4. After you've done the data processing and feature engineering work, and you've trained a good model, make sure that the model you trained can score new datasets correctly. You can use the following two tests to check the prediction levels and distribution of label values:

- Check prediction levels:

```
def checkPredictionLevels(csv_file, model_file):
    df = pd.read_csv(csv_file)
    X_to_score = df[['education_num','age','hours_per_week']].values
    loaded_model = pickle.load(open(model_file, 'rb'))
    y_hat = loaded_model.predict(X_to_score)
    if list(set(y_hat)) == [0,1]:
        print("Prediction Levels match!")
    else:
        print("Prediction Levels do not match!")
    return list(set(y_hat)) == [0,1]
```

- Check the distribution of prediction values:

```
def checkPredictionPercent(csv_file,model_file):
    df = pd.read_csv(csv_file)
    X_to_score = df[['education_num','age','hours_per_week']].values
    loaded_model = pickle.load(open(model_file, 'rb'))
    y_hat = loaded_model.predict(X_to_score)
    ratio1 = sum(y_hat == 1)/len(y_hat)
    if ratio1 > 0.5:
        print("Response levels(0/1) are messed up!")
    else:
        print("Response 0/1 percent is OK, Happy prediction!")
    return ratio1 < 0.5
```

5. Put all test functions together into a Python script called **test_funcs.py**:

test_funcs.py

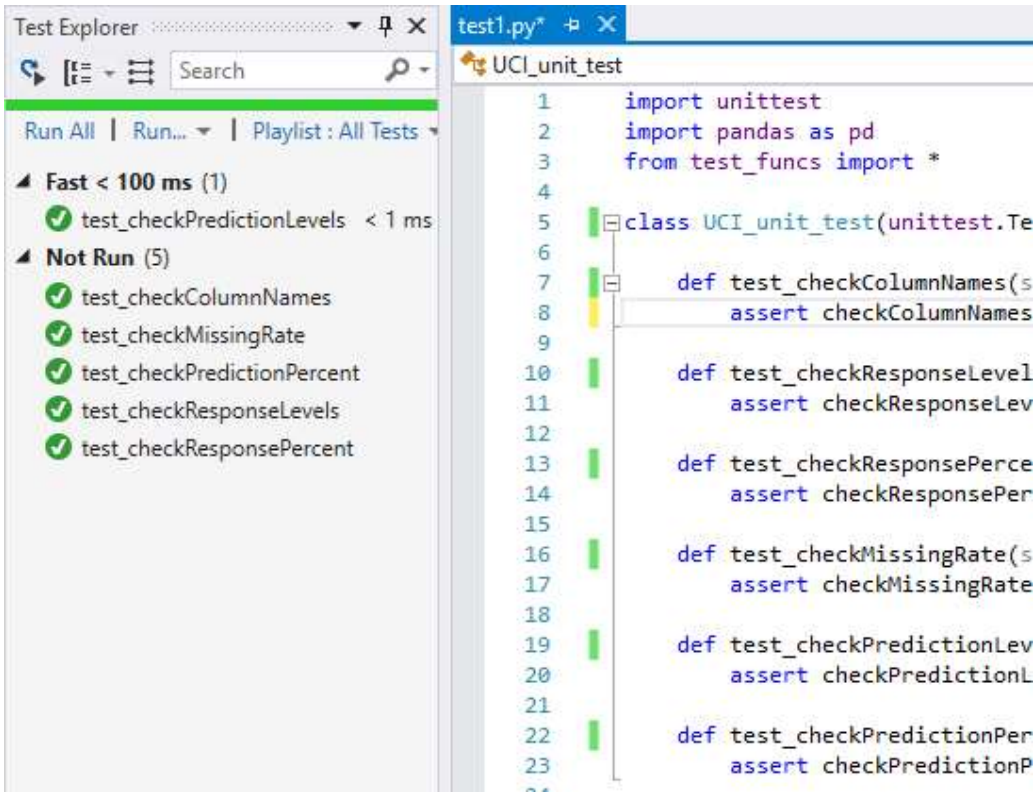
```python
#class check_data(object):
#    """description of class"""
import pandas as pd
import pickle
import numpy as np


def checkColumnNames(csv_file, required_columns):
    df = pd.read_csv(csv_file)
    if set(df.columns) == set(required_columns):
        print("Columns match!")
    else:
        print("Columns do not matched!")
    return set(df.columns) == set(required_columns)

def checkResponseLevels(csv_file,response_column):
    df = pd.read_csv(csv_file)
    levels=list(df[response_column].unique())
    if levels == [0,1]:
        print("Levels match!")
    else:
        print("Levels do not match!")
    return levels == [0,1]

def checkResponsePercent(csv_file,response_column):
    df = pd.read_csv(csv_file)
    ratio1 = df.loc[df[response_column] == 1].shape[0]/df.shape[0]
    if ratio1 > 0.5:
        print("Response levels(0/1) are messed up!")
    else:
        print("Response 0/1 are OK, and OK, Happy Friday!")
    return ratio1 < 0.5
```

6. After the test codes are prepared, you can set up the testing environment in Visual Studio.

   Create a Python file called **test1.py**. In this file, create a class that includes all the tests you want to do. The following example shows six tests prepared:

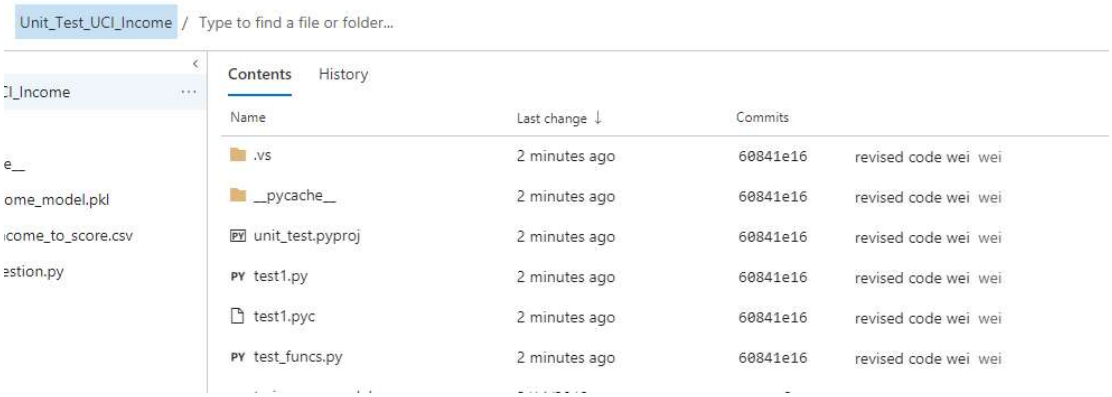
```python
class Test_A(unittest.TestCase):

    def checkColumnNames(self):
        assert checkColumnNames(file_name, required_columns) == True

    def checkResponseLevels(self):
        assert checkResponseLevels(file_name, response_column) == True

    def checkResponsePercent(self):
        assert checkResponsePercent(file_name, response_column) == True

    def checkMissingRate(self):
        assert checkMissingRate2(file_name, threshold) == True

    def checkPredictionLevels(self):
        assert checkPredictionLevels(score_file_name, model_file_name) == True

    def checkPredictionPercent(self):
        assert checkPredictionPercent(score_file_name, model_file_name) == True
```

1. Those tests can be automatically discovered if you put **codetest.testCase** after your class name. Open Test Explorer in the right pane, and select **Run All**. All the tests will run sequentially and will tell you if the test is successful or not.



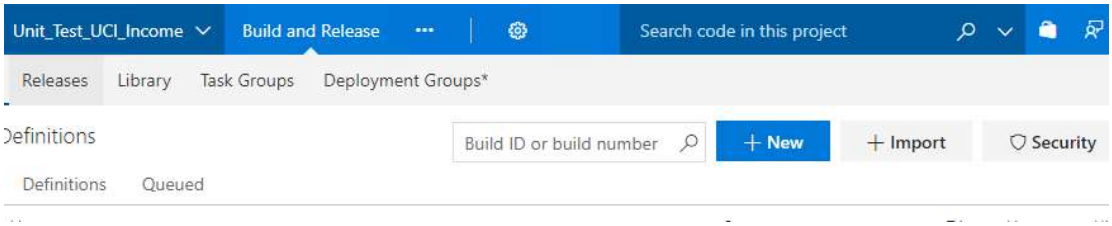
2. Check in your code to the project repository by using Git commands. Your most recent work will be reflected shortly in Azure DevOps.

```
PS C:\Unit_Test_UCI_Income> git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .vs/unit_test_1/v15/.suo
        modified:   __pycache__/test_funcs.cpython-35.pyc
        modified:   test1.py
        modified:   test1.pyc
        modified:   test_funcs.py
        modified:   unit_test.pyproj

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        __pycache__/test1.cpython-35.pyc
        __pycache__/test2.cpython-35.pyc
        __pycache__/test_model.cpython-35.pyc

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Unit_Test_UCI_Income> git add .
PS C:\Unit_Test_UCI_Income> git commit -m"revised code wei"
[master 60841e1] revised code wei
 9 files changed, 7 insertions(+), 19 deletions(-)
 rewrite .vs/unit_test_1/v15/.suo (63%)
 create mode 100644 __pycache__/test1.cpython-35.pyc
 create mode 100644 __pycache__/test2.cpython-35.pyc
 create mode 100644 __pycache__/test_model.cpython-35.pyc
 rewrite test1.pyc (64%)
PS C:\Unit_Test_UCI_Income> git push
Counting objects: 15, done.
Delta compression using up to 24 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (15/15), 6.16 KiB | 2.05 MiB/s, done.
Total 15 (delta 6), reused 0 (delta 0)
remote: Analyzing objects... (15/15) (26 ms)
remote: Storing packfile... done (213 ms)
remote: Storing index... done (32 ms)
To https://dg-ads.visualstudio.com/_git/Unit_Test_UCI_Income
   7785cf0..60841e1  master -> master
PS C:\Unit_Test_UCI_Income>
```

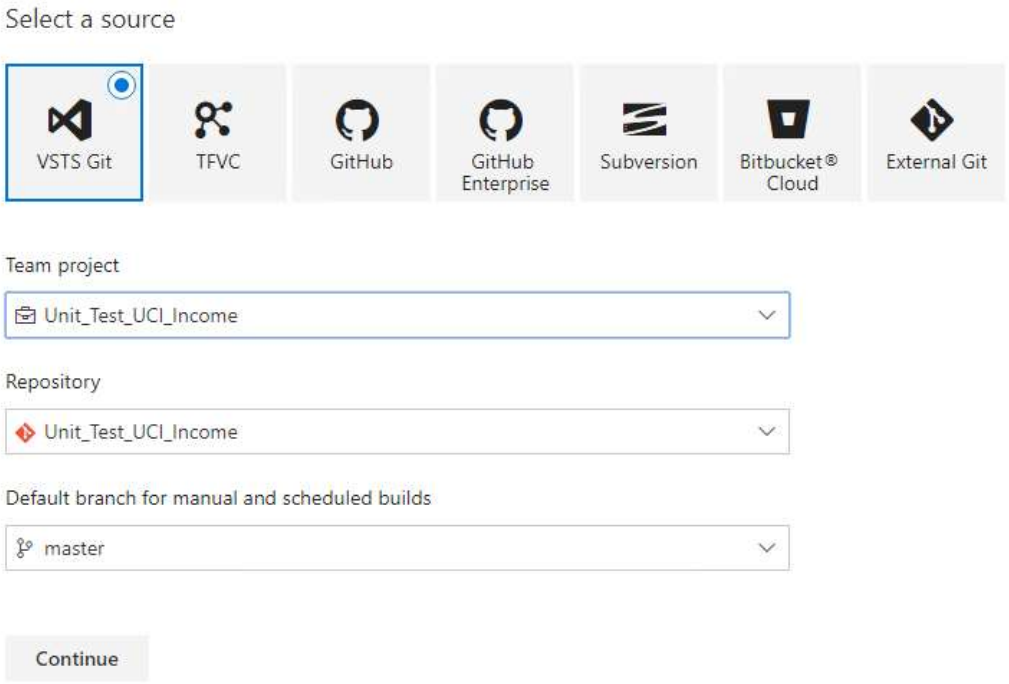


3. Set up automatic build and test in Azure DevOps:

   a. In the project repository, select **Build and Release**, and then select **+New** to create a new build process.

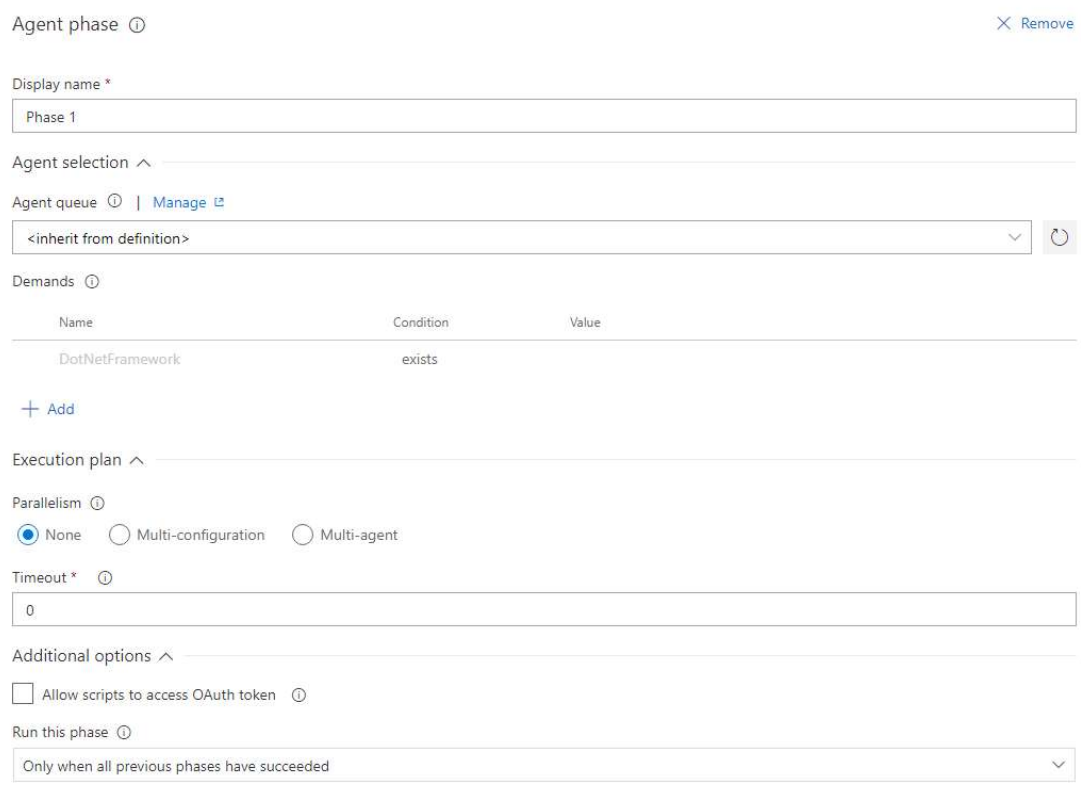


b. Follow the prompts to select your source code location, project name, repository, and branch information.

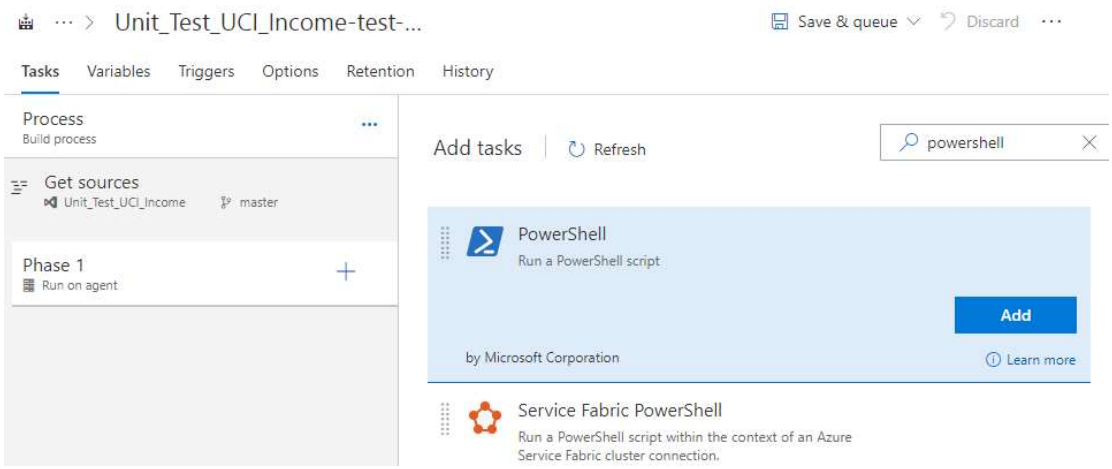


c. Select a template. Because there's no Python project template, start by selecting **Empty process**.

d. Name the build and select the agent. You can choose the default here if you want to use a DSVM to complete the build process. For more information about setting agents, see Build and release agents.

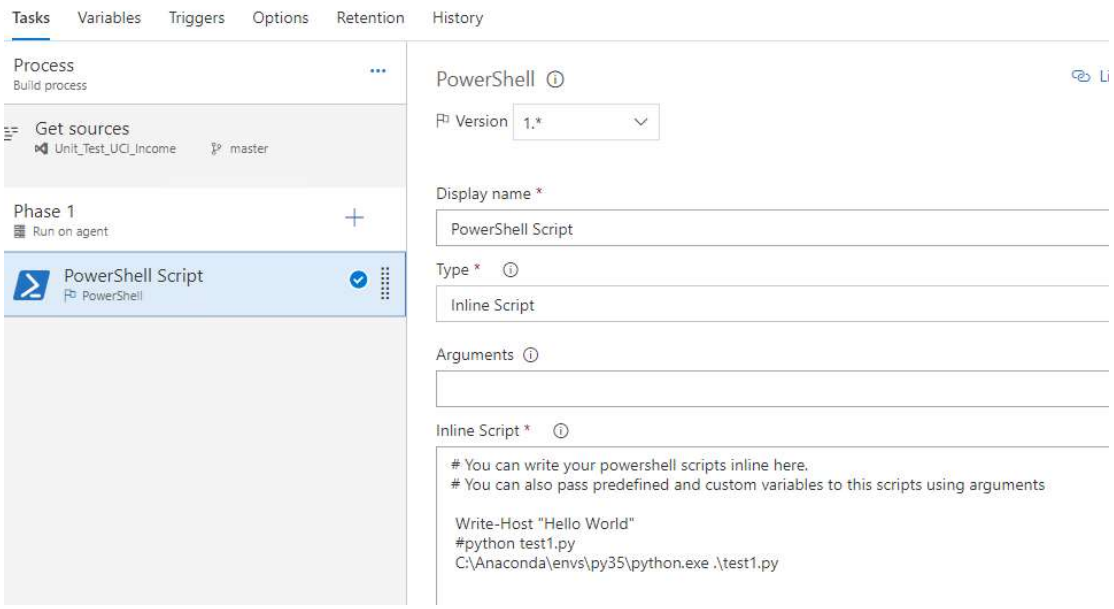


e. Select + in the left pane, to add a task for this build phase. Because we're going to run the Python script **test1.py** to complete all the checks, this task is using a PowerShell command to run Python code.

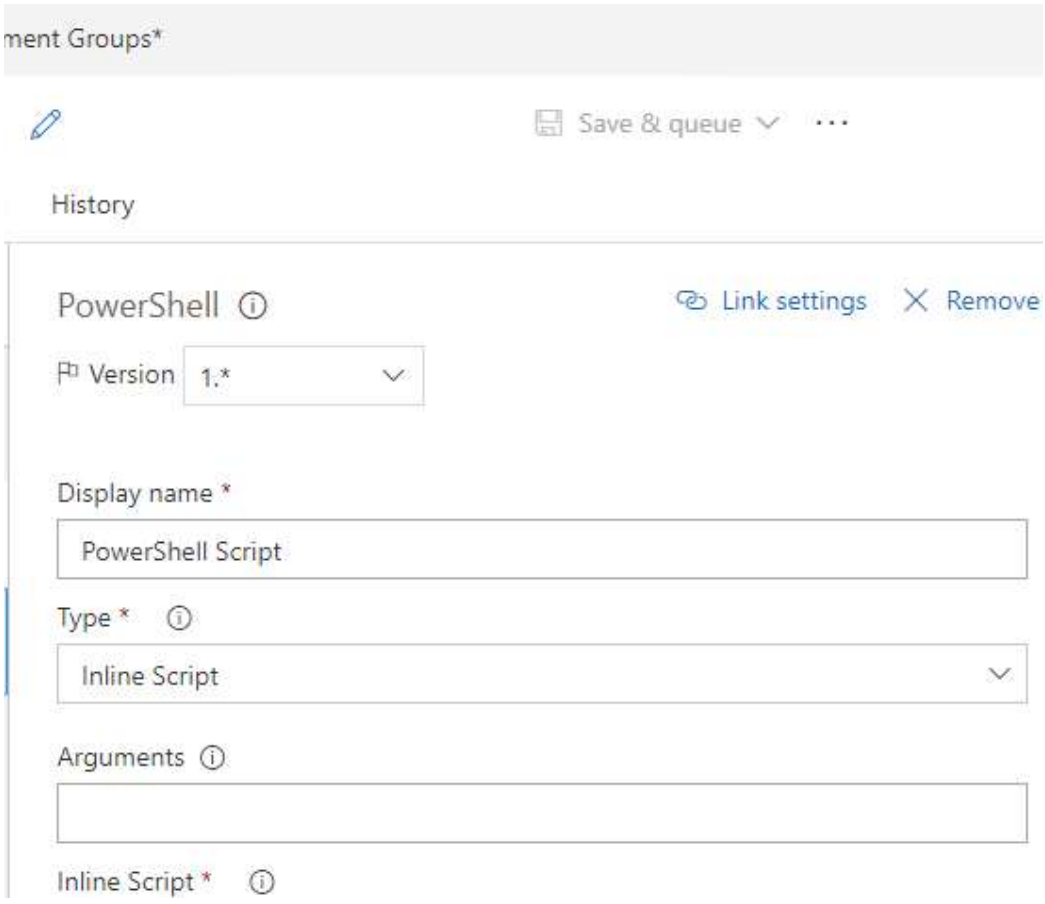


f. In the PowerShell details, fill in the required information, such as the name and version of PowerShell. Choose **Inline Script** as the type.

In the box under **Inline Script**, you can type **python test1.py**. Make sure the environment variable is set up correctly for Python. If you need a different version or kernel of Python, you can explicitly specify the path as shown in the figure:



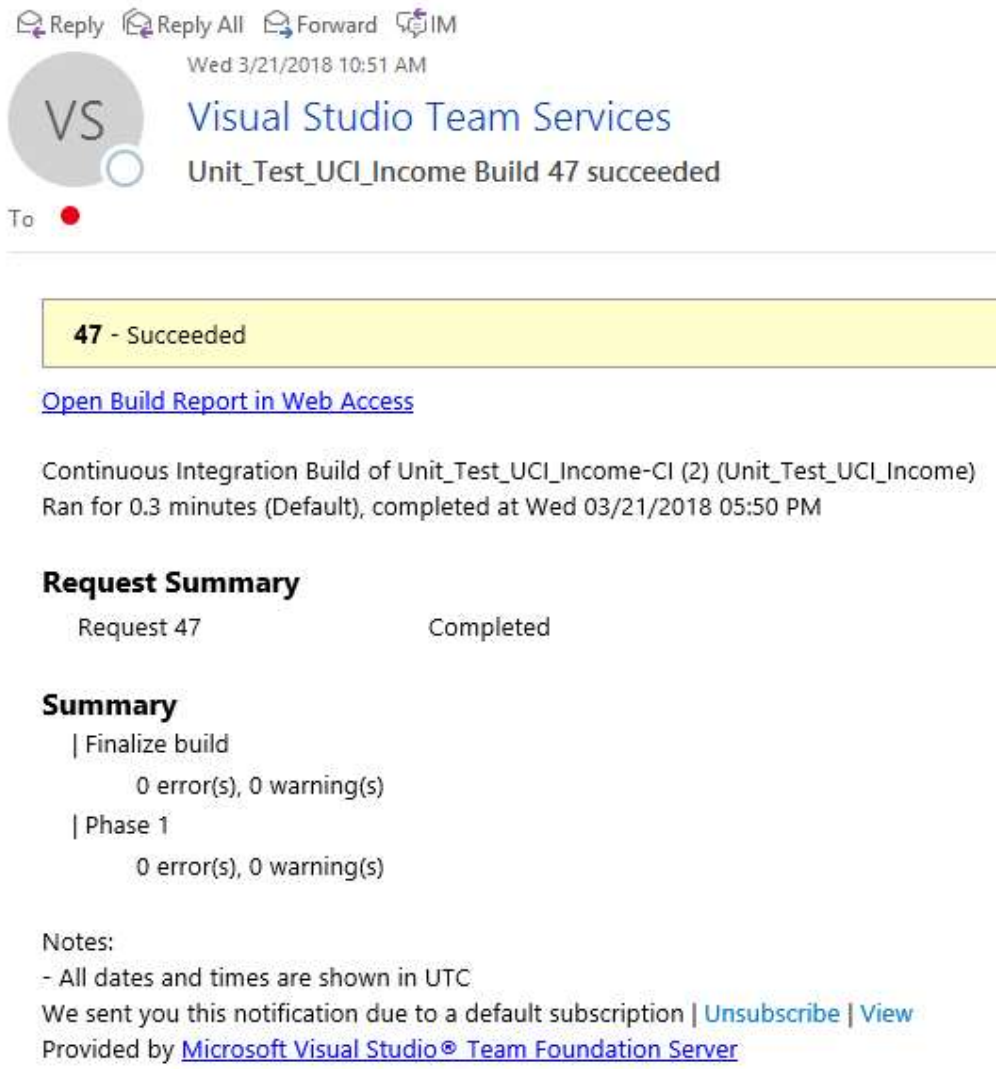
g. Select **Save & queue** to complete the build pipeline process.

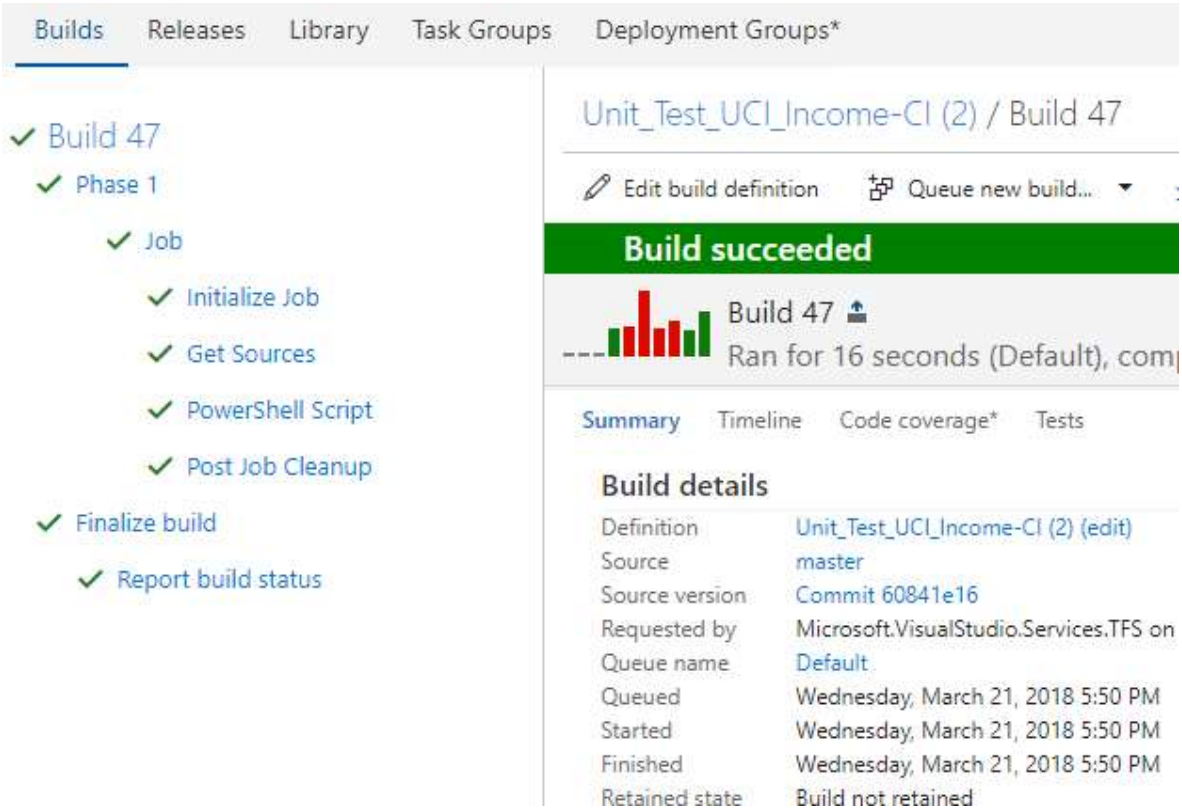


Now every time a new commit is pushed to the code repository, the build process will start automatically. You can define any branch. The process runs the **test1.py** file in the agent machine to make sure that everything defined in the code runs correctly.

If alerts are set up correctly, you'll be notified in email when the build is finished. You can also check the build status in Azure DevOps. If it fails, you can check the details of

the build and find out which piece is broken.





Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- See the UCI income prediction repository for concrete examples of unit tests for data science scenarios.
- Follow the preceding outline and examples from the UCI income prediction scenario in your own data science projects.

References

- Team Data Science Process
- Visual Studio Testing Tools
- Azure DevOps Testing Resources

- Data Science Virtual Machines

Track the progress of data science projects

Article • 08/31/2023

Data science group managers, team leads, and project leads can track the progress of their projects. Managers want to know what work has been done, who did the work, and what work remains. Managing expectations is an important element of success.

Azure DevOps dashboards

If you're using Azure DevOps, you can build dashboards to track the activities and work items associated with a given Agile project. For more information about dashboards, see Dashboards, reports, and widgets.

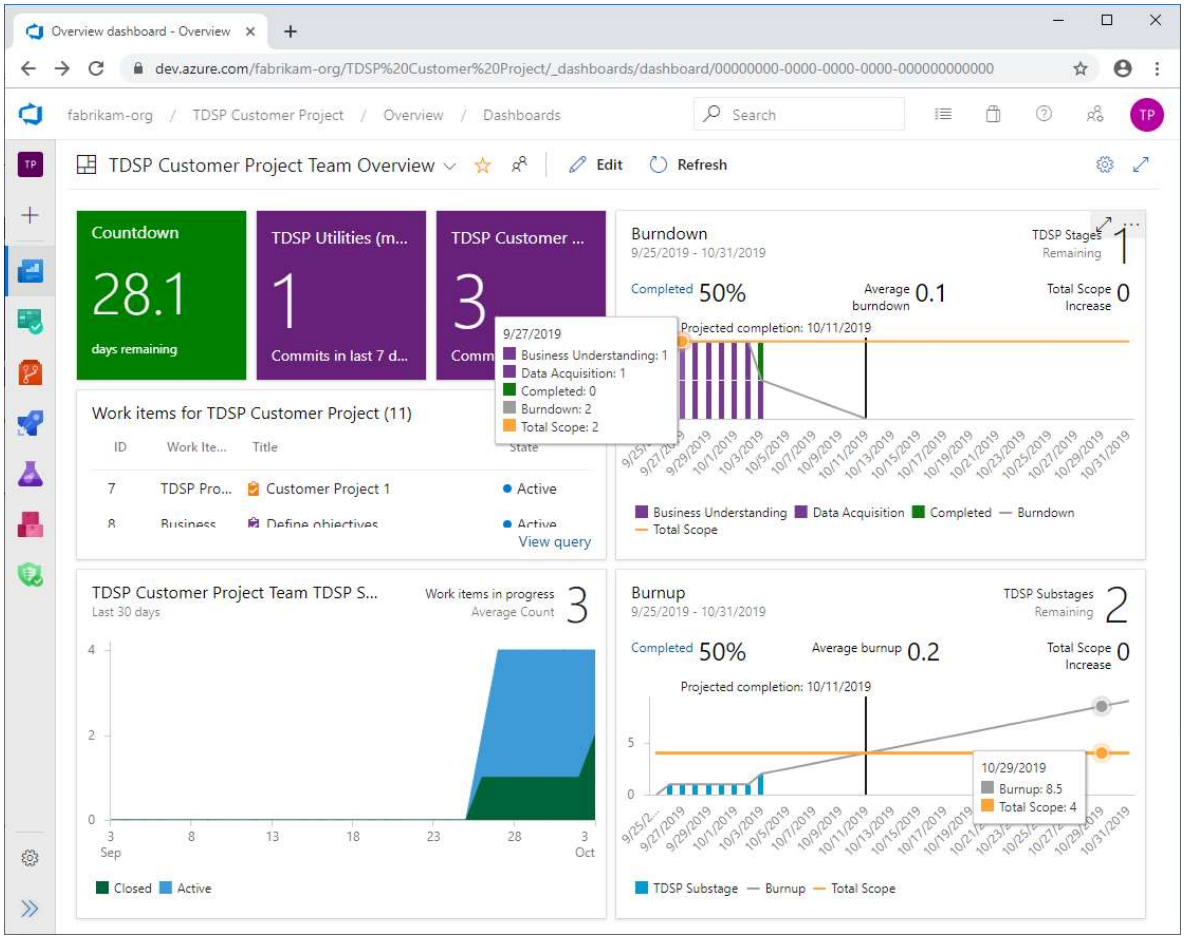
For instructions on how to create and customize dashboards and widgets in Azure DevOps, see the following quickstarts:

- Add and manage dashboards
- Add widgets to a dashboard

Example dashboard

Here is a simple example dashboard that tracks the sprint activities of an Agile data science project, including the number of commits to associated repositories.

- The **countdown** tile shows the number of days that remain in the current sprint.
- The two **code tiles** show the number of commits in the two project repositories for the past seven days.
- **Work items for TDSP Customer Project** shows the results of a query for all work items and their status.
- A **cumulative flow diagram** (CFD) shows the number of Closed and Active work items.
- The **burndown chart** shows work still to complete against remaining time in the sprint.
- The **burnup chart** shows completed work compared to total amount of work in the sprint.



Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal author:

- Mark Tabladillo | Senior Cloud Solution Architect

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

- What is the Team Data Science Process?
- Compare the machine learning products and technologies from Microsoft