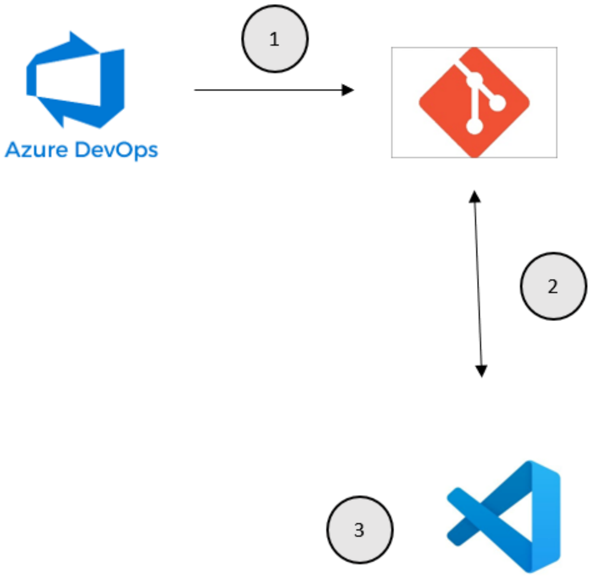
Getting a project started using Azure DevOps, Git and Visual Studio Code

This document will help guide you on how to get a project started using Azure DevOps to manage your project, a Git repository to manage your code and Visual Studio Code as your selected Integrated Deveopment Environment (IDE).

This guide is ideal if you want to get a small project started such as for a proof of concept or you would like to develop a better idea on how to use and integrate these tools. If you would like to use these tools for a substantial project within a larger team please refer to Roles and taks for an overiew of the different roles and repsonsibilities.

This page will give you further links for detailed instructions on your role and how to get started with your project. Details covered include how to get started with Azure DevOps, creating a project repository and setting up security control—adding project members and configuring permissions.



An overview of the tools used for this solution

1. Azure DevOps Use Azure DevOps to create and manage your project. You can use this to break down your project into smaller tasks for e.g. ingest data, preprocess data, build features, etc.

2. Git Repository Create a git repo for your project to manage your code. Git is a distributed version control system. Git repositories can live locally (such as on a developer's machine). Each developer has a copy of the source repository on their dev machine. Developers can commit each set of changes on their dev machine and perform version control operations such as history and compare without a network connection.

3. Visual Studio Code Launch an Azure Machine Learning (ML) workspace using Visual Studio Code, and clone the git repo onto your directory in the workspace's shared workspace file system. Once cloned you can develop your code in Visual Studio Code and use git to save a set o changes, perform version control operations such as history and share their code with colleagues. use Jupyter notebooks to write your code. With your workspace now connected to your git repo you can `commit` (save) changes and `push` them back to your git repo in Azure DevOps. If files are added to the git repo you and colleagues can `pull` these changes onto your workspace making collaborating with colleagues easy.

1. Create a project in Azure DevOps

Prerequisites

- You need an organization before you can create a project. If you haven't created an organization yet, create one by following the instructions in Sign up, sign in to Azure DevOps, which also creates a project. Or see Create an organization or project collection. If you have a Microsoft account follow the instructions fot the heading **Sign up with a personal Microsoft account** here and then refer back to this documentation. If you are going to sign up with a newly created Microsoft account (MWS), your project is automatically created and named based on your sign-in.

- To create a new project you must be a member of the Project Collection Administrators group or have the **Create new projects** permission set to **Allow**. If you're the Organization Owner, you're automatically added to the Project Collection Administrators group. If you aren't a member, get added before following this documentation. For more information, see Set permissions at the project- or collection-level.

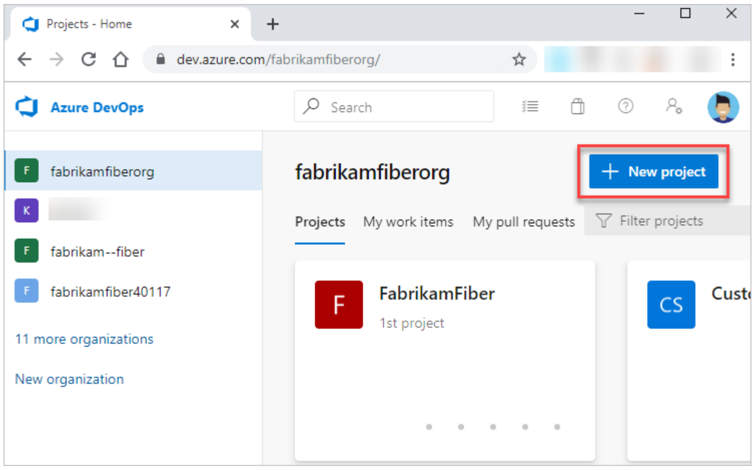
Important: To create a public project, or to make a private project public, see Create a public project in your organization or Change the project visibility, public or private. Additional policy settings must be enabled to work with public projects.

Create a project

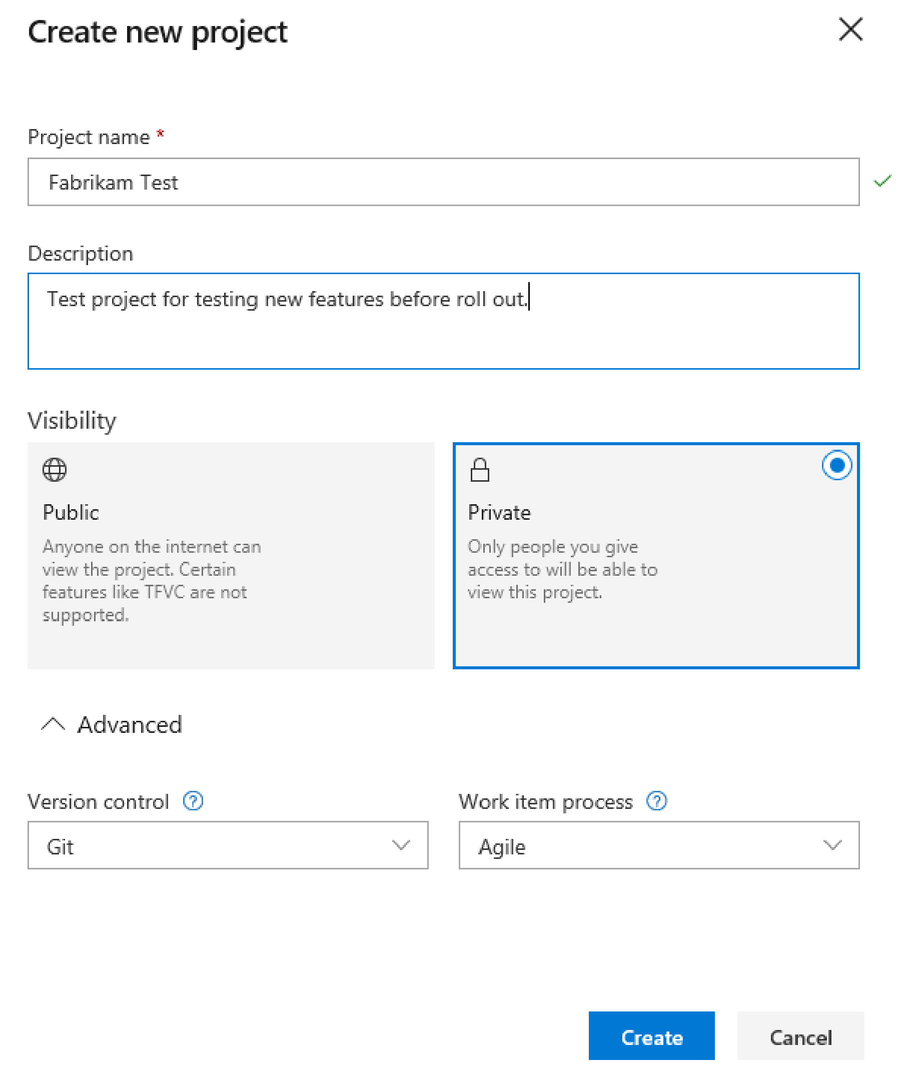
1. Navigate to your organization page `https://dev.azure.com/{yourorganization}`.

2. Select **Azure DevOps** to open the **Projects page**.

3. Choose the organization, and then select **New Project**.



4. Enter information into the form provided. Provide a name for your project. Your project name can't contain special characters, such as `/ : \ ~ & % ; @ ' " ? < > | # $ * } { , + = [ ]`, can't begin with an underscore, can't begin or end with a period, and must be 64 or fewer characters. Enter an optional description. Choose the visibility, **Git** as the source control type, and **Agile** as the work item process. For more information, see Choosing the right version control for your project and Choose a process.



Create new project

Project name * Fabrikam Test ✓

Description: Test project for testing new features before roll out.

Visibility

○ Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

● Private
Only people you give access to will be able to view this project.

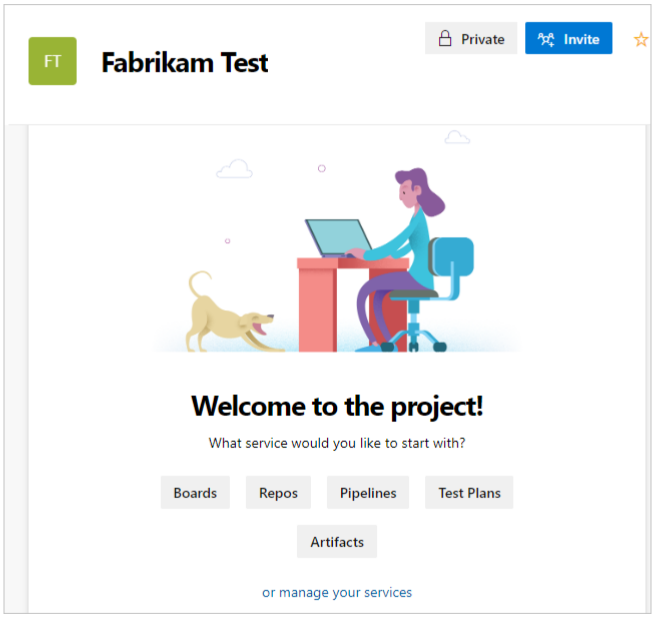
Advanced

Version control: Git

Work item process: Agile

Select visibility of either public or private. When you choose public visibility, anyone on the internet can view your project. With private visibility, only people who you give access to can view your project. For more information about public projects, see Create a public project in your organization. If the **Public** option isn't available, you need to change the policy

5. Select **Create**. The welcome page appears.



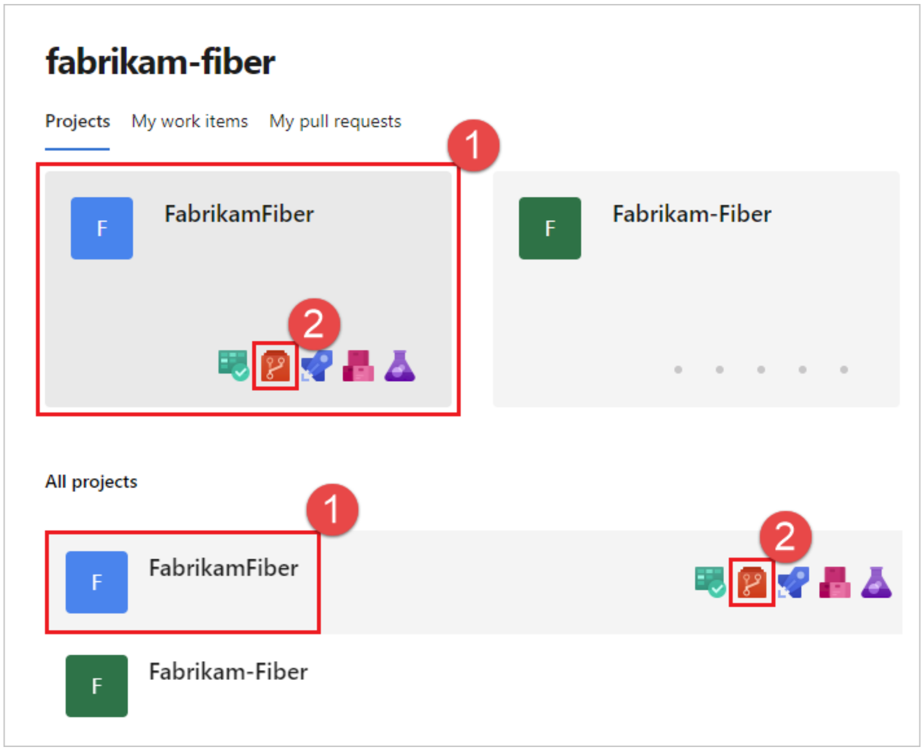
Welcome to the project!

What service would you like to start with?

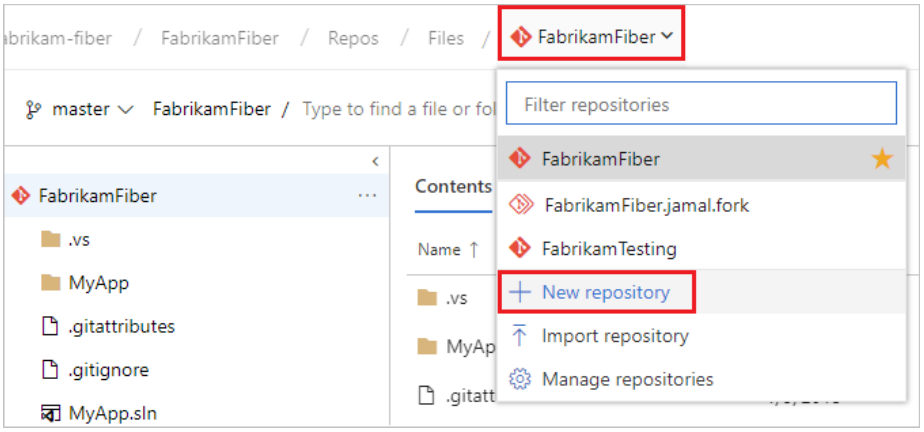
- **Invite**: on the top right if you want to add others to your project. See Add users to a project or team. You can only invite users who are already in your organization.

2. Add a Git repo to your project

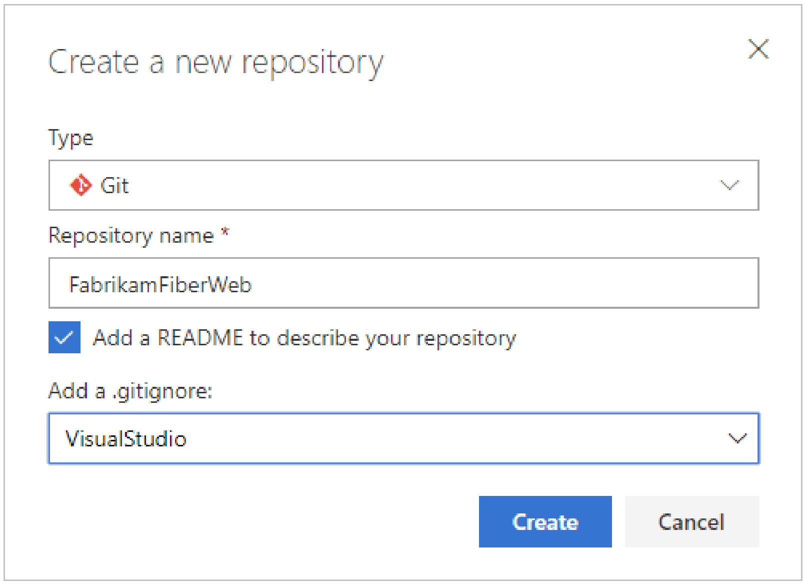
1. Navigate to the **Repos** page in your project, hover your mouse over the name of your project and select the **Repos** icon.



2. From the repo drow-down, select **New repository**.



3. In the **Create a new repository** dialog, verify that Git is the repo type and enter a name for your new repo. You can also choose to add a README and create a .gitignore for the type of code you plan to manage in the repo. A README contains information about the code in your repo, and a .gitignore file tells Git which types of files to ignore, such as temporary build files from your development environment.



Create a new repository

Type: Git

Repository name *: FabrikamFiberWeb

☑ Add a README to describe your repository

Add a .gitignore: VisualStudio

4. When you’re happy with the repo name and choices, select **Create**.

A new empty Git repo is now created in your project.

3. Visual Studio Code

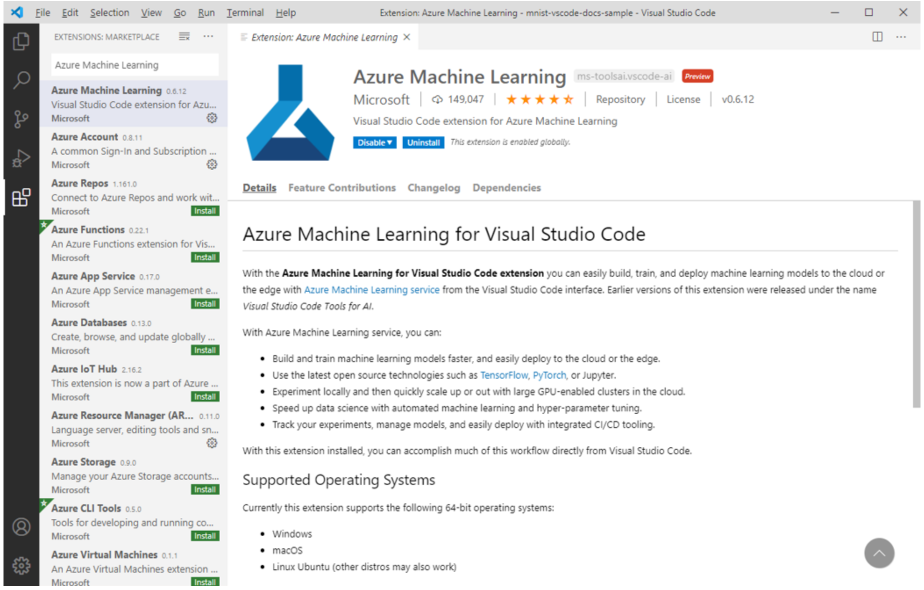
Set up Azure Machine Learning Visual Studio Code extension

Prerequisites

- Visual Studio Code. If you don’t have it, install it.
- Python 3

Install the extension

1. Open Visual Studio Code
2. Select **Extensions** icon from the **Activity Bar** to open the Extensions View.
3. In the Extensions view, search for “Azure Machine Learning”.
4. Select **Install**.


File Edit Selection View Go Run Terminal Help
Extension: Azure Machine Learning - mnist-vscode-docs-sample - Visual Studio Code
EXTENSIONS: MARKETPLACE
Azure Machine Learning
Azure Machine Learning 0.6.12
Visual Studio Code extension for Azu...
Microsoft
Azure Account 0.8.11
A common Sign-In and Subscription ...
Microsoft
Azure Repos 1.161.0
Connect to Azure Repos and work wit...
Microsoft
Install
Azure Functions 0.22.1
An Azure Functions extension for Vis...
Microsoft
Install
Azure App Service 0.17.0
An Azure App Service management e...
Microsoft
Install
Azure Databases 0.13.0
Create, browse, and update globally ...
Microsoft
Install
Azure IoT Hub 2.16.2
This extension is now a part of Azure ...
Microsoft
Install
Azure Resource Manager (AR... 0.11.0
Language server, editing tools and sn...
Microsoft
Azure Storage 0.9.0
Manage your Azure Storage accounts...
Microsoft
Install
Azure CLI Tools 0.5.0
Tools for developing and running co...
Microsoft
Install
Azure Virtual Machines 0.1.1
An Azure Virtual Machines extension ...
Microsoft
Install
Extension: Azure Machine Learning
Azure Machine Learning
ms-toolsai.vscode-ai
Preview
Microsoft | 149,047 | ★★★★☆ | Repository | License | v0.6.12
Visual Studio Code extension for Azure Machine Learning
Disable ▾
Uninstall
This extension is enabled globally.
Details
Feature Contributions
Changelog
Dependencies

Azure Machine Learning for Visual Studio Code

With the **Azure Machine Learning for Visual Studio Code extension** you can easily build, train, and deploy machine learning models to the cloud or the edge with Azure Machine Learning service from the Visual Studio Code interface. Earlier versions of this extension were released under the name *Visual Studio Code Tools for AI.*

With Azure Machine Learning service, you can:

- Build and train machine learning models faster, and easily deploy to the cloud or the edge.
- Use the latest open source technologies such as TensorFlow, PyTorch, or Jupyter.
- Experiment locally and then quickly scale up or out with large GPU-enabled clusters in the cloud.
- Speed up data science with automated machine learning and hyper-parameter tuning.
- Track your experiments, manage models, and easily deploy with integrated CI/CD tooling.

With this extension installed, you can accomplish much of this workflow directly from Visual Studio Code.

Supported Operating Systems

Currently this extension supports the following 64-bit operating systems:

- Windows
- macOS
- Linux Ubuntu (other distros may also work)

Note—Alternatively, you can install the Azure Machine Learning extension via the Visual Studio Marketplace by downloading the installer directly.

Sign into your Azure Account In order to provision resources and run workloads on Azure, you have to sign in with your Azure account credentials. To assist with account management, Azure Machine Learning automatically installs the Azure Account extension. Visit the following site to learn more about the Azure Account extension.

1. Open the command palette by selecting **View > Command Palette** from the menu bar or pressing **Ctrl+Shift+P**. The Command Palette provides an easy and convenient way to access a wide variety of tasks, including those provided by 3rd party extensions.

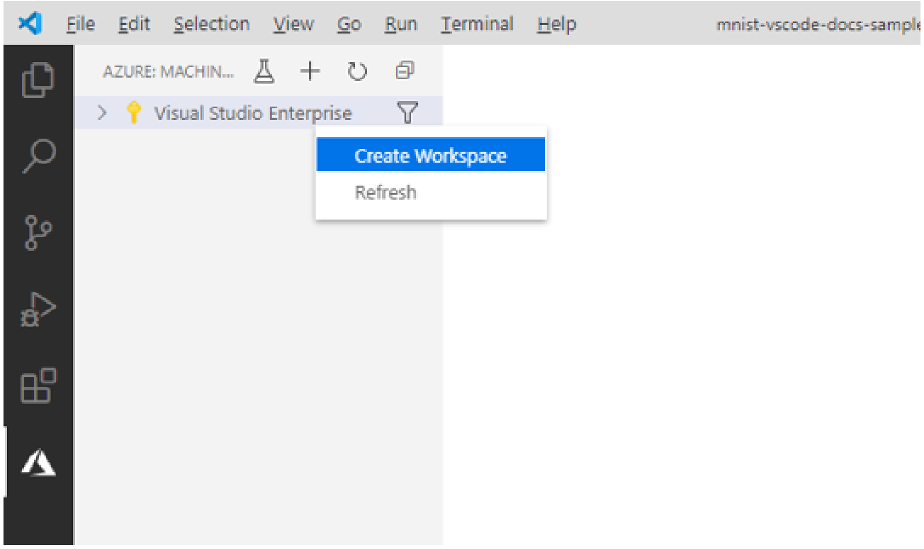
2. Enter the command "Azure: Sign In" into the commad palette to start the sign in process.

Create a workspace

The first thing you have to do to build an application in Azure Machine Learning is to create a workspace. A workspace contains the resources to train models as well as the trained models themselves. For more information, see what is a workspace.

1. On the Visual Studio Code activity bar, select the **Azure** icon to open the Azure Machine Learning view.

2. Right-click your Azure subscription and select **Create Workspace**.



3. By default a name is generated containing the date and time of creation. In the text input box, change the name to “TeamWorkspace” or something more sutitable and press **Enter**.
4. Select **Create a new resource group**.
5. Name your resource group for e.g. “TeamWokspace-rg” and press **Enter**.
6. Choose a location for your workspace. It’s recommended to choose a location that is closest to the location you plan to deploy you model. For example, “West US 2”.
7. When prompted to select the type of workspace, choose **basic**.

At this point, a request to Azure is made to create a new workspace in your account. After a few minutes, the new workspace appears in your subscription node.

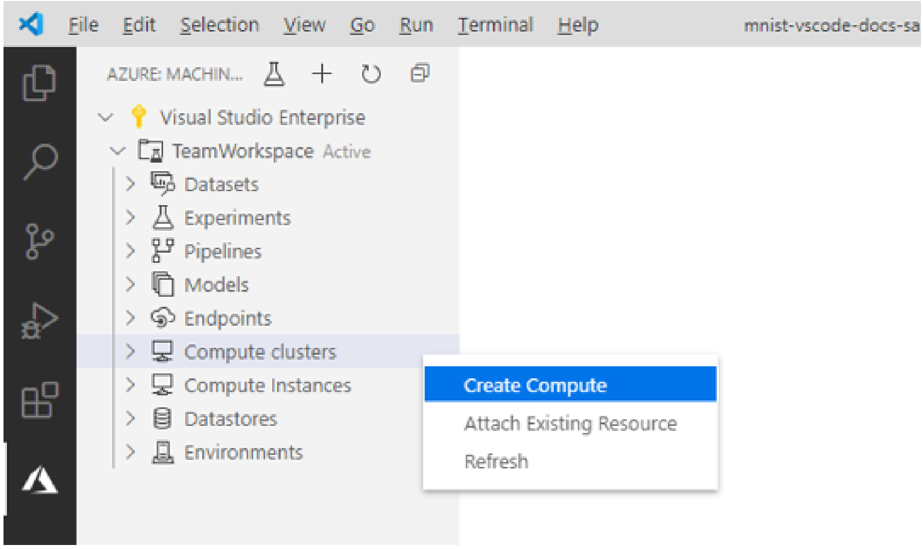
Configure Compute Targets

A compute target is the computing resource or environment where you run scripts and deploy trained models. For more information, see the Azure Machine Learning compute targets documentation.

To create a compute target:

1. On the Visual Studio Code activity bar, select the **Azure** icon. The Azure Machine Learning view appears.
2. Expand your subscription node.
3. Expand the **Workspace name** e.g., “TeamWorkspace” node.

4. Under the workspace node, right-click the **Compute clusters** node and choose **Create Compute**.

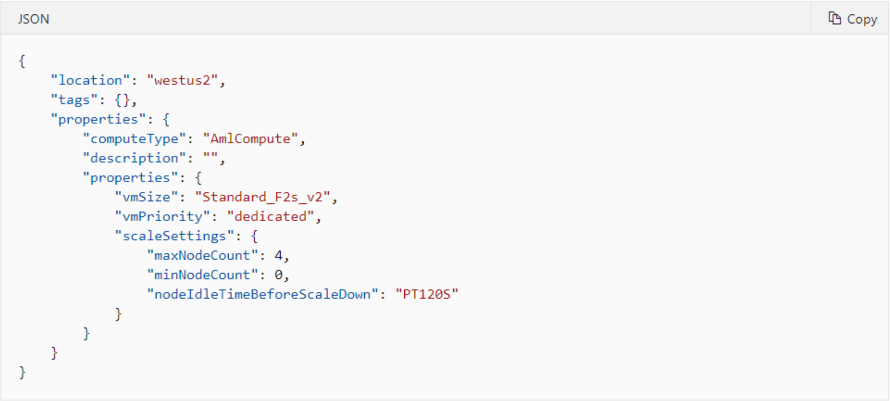


5. Select **Azure Machine Learning Compute (AmlCompute)**. Azure Machine Learning Compute is a managed-compute infrastructure that allows the user to easily create a single or multi-node compute that can be used with other users in your workspace.

6. Choose a VM size. You can select **Standard_F2s_v2** from the list of options. The size of your VM has an impact on the amount of time it takes to train your models. For more information on VM sizes, see sizes for Linux virtual machines in Azure.

7. Name your compute for e.g., “TeamWkspc-com” and press **Enter** to create your compute.

A file appears in VS Code with content similar to the one below:



```json
{
    "location": "westus2",
    "tags": {},
    "properties": {
        "computeType": "AmlCompute",
        "description": "",
        "properties": {
            "vmSize": "Standard_F2s_v2",
            "vmPriority": "dedicated",
            "scaleSettings": {
                "maxNodeCount": 4,
                "minNodeCount": 0,
                "nodeIdleTimeBeforeScaleDown": "PT120S"
            }
        }
    }
}
```

8. When satisfied with the configuration, preess **Ctrl+Shift+P** to show the **Command Palette**.
9. Enter the following command into the command palette to save your run configuration file.

text

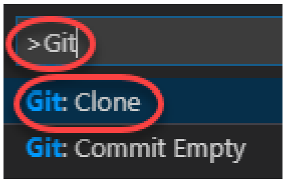
```
Azure ML: Save and Continue
```

After a few minutes, the new compute target appears in the Compute clusters node of your workspace.

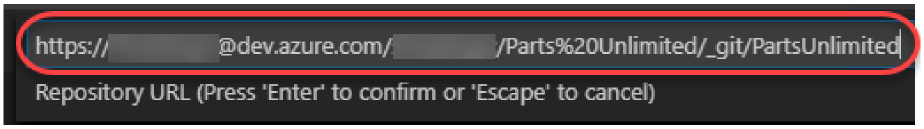
Clone Git repositories into your workspace file system

Azure Machine Learning provides a shared file system for all users in the workspace. To clone a Git repository into this file share:

1. Follow the instructions here to copy the details needed to clone the Git repo.
2. Go back to your **Viusa Studio Code** instance.
3. Press **Ctrl+Shift+P** to show the **Command Palette**.
4. Execute the **Git: Clone command**. It may help to type **"Git"** to bring it to the shortlist.



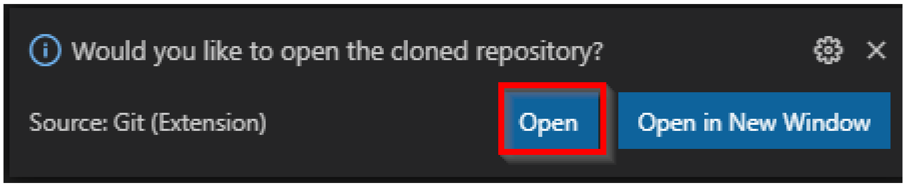
5. Paste in the URL to your repo and press **Enter**.



https://[redacted]@dev.azure.com/[redacted]/Parts%20Unlimited/_git/PartsUnlimited

Repository URL (Press 'Enter' to confirm or 'Escape' to cancel)

6. Select a local path to clone the repo to—it is advised that you clone the repository into your users directory so that others will not make collisions directly on your working branch. An option is to clone it in your own ‘users’ directory.
7. When prompted, log in to your Azure DevOps account.
8. Once the cloning has completed, click **Open** to open the cloned repository.



Would you like to open the cloned repository?

Source: Git (Extension)

Notes on working with Git commands on Visual Code

Saving work with commits When you make changes to your files, Git will record the changes in the local repository. You can select the changes that you want to commit by staging the changes. Commits are always made against your local Git repository, so you don’t have to worry about the commit being perfect or ready to share with others. You can make more commits as you continue to work and push the changes to others when they are ready to be shared.

What’s in a commit?

Git commits consists of the following:

- The file(s) changed in the commit. Git keeps the contents of all file changes in your repo in the commits. This keeps it fast and allows intelligent merging.
- A reference to the parent commit(s). Git manages your code history using these references.
- A message describing a commit. You give this message to Git when you create the commit. It’s a good idea to keep this message descriptive, but to the point.

For more detailed examples see Committing changes, Reviewing commits and Staging commits.

Reviewing history Git uses the parent reference information stored in each commit to manage a full history of your development. You can easily review this commit history to find out when file changes were made and determine differences between versions of your code using the terminal or from one of the many Visual Studio Code extensions available. You can also review changes using the Azure DevOps portal.

Git's use of the **Branches and Merges** feature works through pull requests, so the commit history of your development doesn't necessarily form a straight, chronological line. When you use history to compare versions, think in terms of file changes between two commits instead of file changes between two points in time. A recent change to a file in the master branch may have come from a commit created two weeks ago in a feature branch but was only merged yesterday.

For more detailed examples on this see Comparing files.

Working with branches You can manage the work in your Azure DevOps Git repo from the **Branches** view on the web. You can also customize the view to track the branches you care most about so you can stay on top of changes made by your team.

Committing changes to a branch will not affect other branches and you can share branches with others without having to merge the changes into the main project. You can also create new branches to isolate changes for a feature or a bug fix from your master branch and other work. Since the branches are lightweight, switching between branches is quick and easy. Git does not create multiple copies of your source when working with branches, but rather uses the history information stored in commits to recreate the files on a branch when you start working on it. Your Git workflow should create and use branches for managing features and bugfixes. The rest of the Git workflow, such as sharing code and reviewing code with pull requests, all work through branches. Isolating work in branches makes it very simple to change what you are working on by simply changing your current branch.

For more detailed examples on this see Creating a new branch in your local repository and Working with branches.

Managing branches from Azure DevOps In addition to all the functionality available in Visual Studio Code, you can also manage your repo branches from the Azure DevOps portal. See Creating a new branch, Deleting a branch, Locking a branch, and Tagging a release.