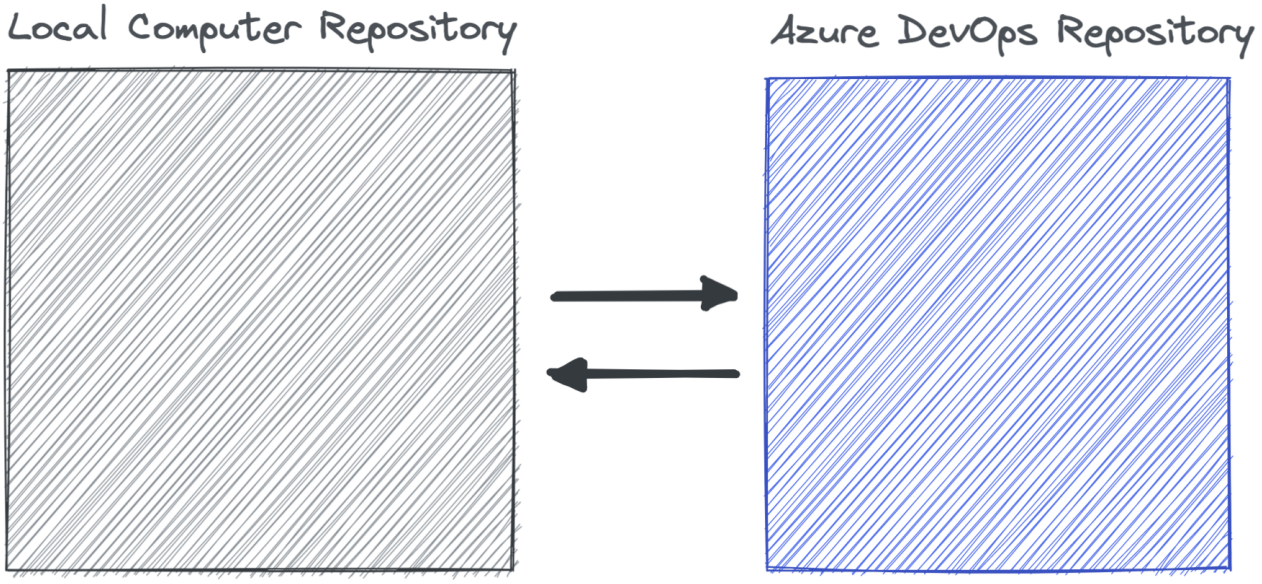
Working with Azure DevOps Repositories



December 15th, 2022

Introduction

This document describes how to set up your local computer and work with Azure DevOps. When working Azure DevOps project repositories, it can be confusing setting up SSH keys, adding existing local git repositories, and setting up the remote origin URLs to pull and push your commits.

This document covers the following:

- Install and use Azure CLI
- Create Azure DevOps Repositories using Azure DevOps CLI
- Create Local Repositories
- Create and assign SSH Keys
- Associate Remote Repositories
- Learn common Git Commands to manage code

Installing Azure CLI

The Azure Command-Line Interface (CLI) is a cross-platform command-line tool that can be installed locally on Windows computers. You can use the Azure CLI for Windows to connect to Azure and execute administrative commands on Azure resources.

To install the Azure CLI on your computer, visit the following site

https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli#install-or-update

Although the Azure CLI can be used in most aspects of Azure resources, we are primarily using it for Azure DevOps-related functions.

How to sign into the Azure CLI

Before using any Azure CLI commands with a local install, you need to sign in.

```
az login
(you will need to run this from Powershell)
```

1. If the CLI can open your default browser, it will initiate the authorization code flow and open the default browser to load an Azure sign-in page.

2. Sign in with your account credentials in the browser.

After logging in, you will see a list of subscriptions associated with your Azure account. The subscription information with `isDefault: true` is the currently activated subscription after logging in.

Azure DevOps CLI

There are many DevOps CLI commands. These are accessible using the az command.

IMPORTANT NOTE: For seamless commanding, set the organization (and project) as defaults in configuration.

```
az devops configure --defaults
organization=https://dev.azure.com/levsena
```

az devops –help *(Returns a list of all the devops commands)*

```
levsen@Allans-MacBook-Pro devops1 % az devops --help

Group
    az devops : Manage Azure DevOps organization level operations.
        Related Groups
        az pipelines: Manage Azure Pipelines
        az boards: Manage Azure Boards
        az repos: Manage Azure Repos
        az artifacts: Manage Azure Artifacts.

Subgroups:
    admin            : Manage administration operations.
    extension        : Manage extensions.
    project          : Manage team projects.
    security         : Manage security related operations.
    service-endpoint : Manage service endpoints/connections.
    team             : Manage teams.
    user             : Manage users.
    wiki             : Manage wikis.

Commands:
    configure        : Configure the Azure DevOps CLI or view your configuration.
    invoke           : This command will invoke request for any DevOps area and resource. Please use
                       only json output as the response of this command is not fixed. Helpful docs -
                       https://docs.microsoft.com/en-us/rest/api/azure/devops/.
    login            : Set the credential (PAT) to use for a particular organization.
    logout           : Clear the credential for all or a particular organization.
```

az devops project –help *(Returns a list of all the devops project commands)*

```
levsen@Allans-MacBook-Pro devops1 % az devops project --help

Group
    az devops project : Manage team projects.

Commands:
    create : Create a team project.
    delete : Delete team project.
    list   : List team projects.
    show   : Show team project.
```

Creating an Azure DevOps Project

You can create a new Azure DevOps Project either from the Azure DevOps site or using the Azure DevOps CLI. This document only shows how to create a new project using the Azure DevOps CLI. To create a new project, use the following comment syntax:

```
az devops project create --name
                         [--description]
                         [--open]
                         [--org]
                         [--process]
                         [--source-control {git, tfvc}]
                         [--visibility {private, public}]
```

Parameter Explanation:

- **name**: Required. Name of the project to create.
- **description**: Optional. Short description of the project, enclosed in quotes.
- **open**: Optional. Once the command creates a project, it opens in the default web browser.
- **org**: Optional. Azure DevOps organization URL. Required if not configured as default or picked up by using `git config`. You can configure the default organization using `az devops configure -d organization=ORG_URL`. Example: `https://dev.azure.com/MyOrganizationName/`.
- **process**: Optional. The process model to use, such as *Agile*, *Basic*, *Scrum*, *CMMI*, or other custom process model. Agile is the default. To learn more, see About process customization and inherited processes.
- **source-control**: Optional. Type of source control repository to create for the project: *git* (default) or *tfvc*. If not, name or ID of the project. Example: `--project "Fabrikam Fiber"`.
- **visibility**: Optional. Project visibility. Accepted values: *private* (default), *public*.

Example

```
az devops project create --name HelloWorld --description "Hello World Project Description" --org https://dev.azure.com/levsena/ --process Agile --source-control  git
```

List Projects using Azure DevOps CLI

To list the projects within your Azure DevOps Organization, use the following command:

```
az devops project list
```

Below is a sample - a subset - of the output returned


```
{
  "abbreviation": null,
  "defaultTeamImageUrl": null,
  "description": null,
  "id": "b9f1095a-3886-4855-9c3d-313a6df387e3",
  "lastUpdateTime": "2022-12-16T16:12:34.800000+00:00",
  "name": "DevOpsOne",
  "revision": 275,
  "state": "wellFormed",
  "url": "https://dev.azure.com/levsena/_apis/projects/b9f1
  "visibility": "private"
},
{
  "abbreviation": null,
  "defaultTeamImageUrl": null,
  "description": null,
  "id": "5b949023-012f-40aa-8c59-994b3caf25db",
  "lastUpdateTime": "2022-12-16T18:00:35.510000+00:00",
  "name": "DevOpsTwo",
  "revision": 283,
  "state": "wellFormed",
  "url": "https://dev.azure.com/levsena/_apis/projects/5b94
  "visibility": "private"
}
```

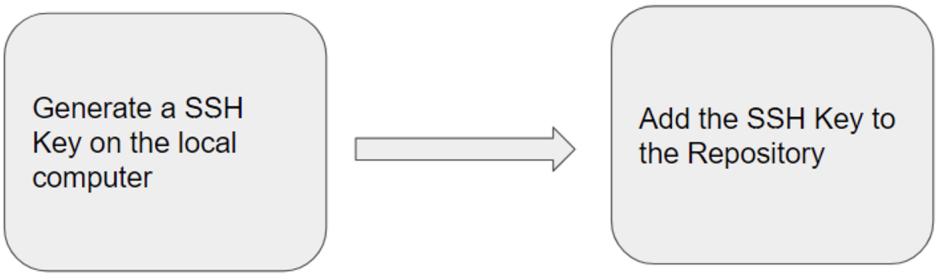

(Projects are listed as a JSON objects)

In-Class Exercise: From the command line, log in to your MacEwan Azure DevOps and list the projects.

Within Azure DevOps, Create two new projects 1) from within Azure DevOps named ‘DevOpsProject1’ and create another using the CLI named ‘DevOpsProject2’. Then, return to the command line and list the projects. Do you see your new projects?

Azure DevOps - Repository SSH Keys

Working with Azure Repositories from git Commands, we need to authenticate your computer to the remote repository - Azure DevOps Repository. One of the simplest ways of authenticating is using a SSH Key. This document describes generating an SSH Key and adding this key to Azure DevOps.



Generating SSH Keys (windows)

To generate an SSH Key on your local computer, you need to issue the following command using PowerShell

ssh-keygen -b 4096 -t rsa

```
PS C:\Users\vminstructor> ssh-keygen -b 4096 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\vminstructor/.ssh/id_rsa):
Created directory 'C:\Users\vminstructor/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\vminstructor/.ssh/id_rsa.
Your public key has been saved in C:\Users\vminstructor/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:p4fd+Ix5dJk/LlJqNQrthmowzpFUyWfNQtyG1vWrUNU vminstructor@ML-RefVm-326952
The key's randomart image is:
+---[RSA 4096]----+
|      . +.* .. ..|
|       + B *  o E|
|      . + o  . . |
|     .      .   .|
|    . . S o.   + |
|     =   * +o+=  |
|    o + o B.*o.. |
|     o . o %.. ..|
|      ... =.+ o..|
+----[SHA256]-----+
PS C:\Users\vminstructor>
```

Once you have successfully generated an SSH Key, we need to run the **cat** command within PowerShell to extra the key used to add to Azure DevOps. From the previous screenshot you can see that we used the default file name and options. Therefore, the following command assumes the default name.

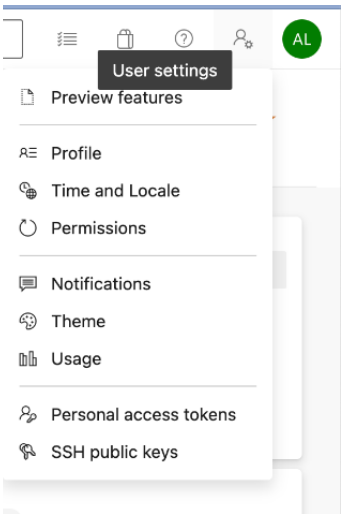
cat C:\Users\vminstructor\.ssh\id_rsa.pub

NOTE: CAT is available in only available in a PowerShell window

```
PS C:\Users\vminstructor> cat C:\Users\vminstructor\.ssh\id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC/JpA/phJJqGpkLN/pGFC3WR8qewmkgzzmWY5evj1S8VTYWI3JIuD1xJMy0XhpyGNwBgzg
N4xaE1V2xUNdKTwPQS1nQ4VMP6jZ7vSQ9HvTh1OFVfamhArrfiW4NZ43NTh7PIpOWfLUWZEqUZjT7MOv15jnIUD9h11m+EYwL8/Se6bzfkvN
usyLgAUgB/5fgj4acrctBgObBMtDUzuc7dZNq73VYjs73ZHRb21BiBtGLuG3z6x/hngMZWnMsGmQtLj/ajAATJdbPV0EXFuQJBnIL0yN7G6Q
jdtDd6W3OZdPh1PSCPxH9sGGUSUFHN4W766g5uw77ua5Kk85bGaDR1YDnwRniDbd1W6oP6IRbz4UgyWT1heDgYi7JqXODHiv4VfWHtwmcw2D
J3mmQ8yGUH1S9aWUK4YfgJBSzwfnSkTtjONU6bacC8LgmLmxLY5L/0Axapho23EZ5X2DWNtKXbUSgljpK8S4Bt6YbWmWk6VPVUqra56Udj3F
Jii/woWKwWeMNkT9AwnNgpkG71fuTSMvE1qpuDDNw83afOC76EbUJ8pZXSMcDNrluEvSVLsrqt97082CFgODm2awraJr8yPpKhcJd53yIOXZ
NGqqLGaQMMaNaV9/OY0nt8ZaPRx9+Urs9u4wgsN49edlDsaSQiB0FCuZ1nzVIHH/PnMvaZutXw== vminstructor@ML-RefVm-326952
PS C:\Users\vminstructor>
```

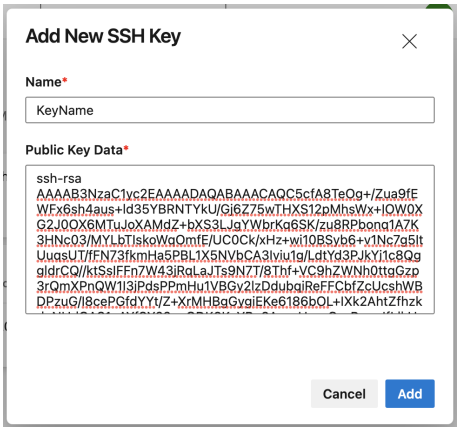
Highlight and copy the output from this command starting from the ssh-rsa text. We will need to paste this into the add key form.

Adding the SSH Key into Azure DevOps



Step 1: From the User Settings, select the SSH Public Keys menu option

Step 2: Click on the + New Key button… you should see the following popup



From the cat command's output, we need to copy and paste it into the Public Key Data field shown on the left.

The Name isn't important - although it should describe the computer this key is associated with.

Working with git and Azure DevOps Repo

You typically obtain a Git repository in one of two ways:

1. You can take a local directory that is currently not under version control and turn it into a Git repository, or
2. You can clone an existing Git repository from elsewhere.

Either way, you end up with a Git repository on your local machine, ready for work.

Cloning an Existing Repository

If you want to get a copy of an existing Git repository — for example, a project on your Azure DevOps Organization — the command you need is **git clone**.

Git receives a full copy of nearly all data that the server has. Every version of every file for the history of the project is pulled down by default when you run git clone.

You clone a repository with git clone <url>. For example, if you want to clone the Git linkable library called libgit2, you can do so like this:

git clone https://github.com/libgit2/libgit2

That creates a directory named **libgit2**, initializes a .git directory inside it, pulls down all the data for that repository, and checks out a working copy of the latest version. If you go into the new **libgit2** directory that was just created, you'll see the project files ready to be worked on or used.

If you want to clone the repository into a directory named something other than libgit2, you can specify the new directory name as an additional argument:

git clone https://github.com/libgit2/libgit2 mylibgit

That command does the same thing as the previous one, but the target directory is called **mylibgit**

Initializing our new Remote Repo

Suppose the project (and, therefore, the repository) is a newly created one. In that case, it's a good idea to **initialize** the repository from within Azure DevOps before cloning the project - this will set up the branch main and place a readme.md file (a Markdown) file.

From within Azure DevOps, you can also add a .gitignore file

IMPORTANT: we are using SSH to authenticate between our local computer and Azure DevOps. Therefore, when getting the remote repository URL, we need to use the SSH URL when cloning a repository to our local computer.


```
git clone
git@ssh.dev.azure.com:v3/levsena/HelloWorld/HelloWorld
```

REMEMBER: When you use git clone, it creates a new folder in the current folder using the repo name - in our case, 'HelloWorld'.

Pushing our existing Repo to Azure

In this scenario, a project is being developed on a computer - this can include many commits to the local repository. Then, we want to connect to a remote repository - such as Azure DevOps - and maintain a remote repository. Or, you could start a new project, and initialize and configure your local computer first before pushing it to a remote repository.

To accomplish this, in general, these are the steps required:

- Create a new folder where the local repository will live
- Initialize git
- Add some files and commit the files
- Push to the Azure DevOps Repo

The following steps need to be followed to push a local repository to Azure DevOps:

Step 1: Create a folder in source/repos/ called HelloWorldTwo. Navigate to the new folder

Step 2: Using Azure CLI (or from the DevOps site), create a HelloWorldTwo project

```
az devops project create --name HelloWorldTwo
--description "Hello World Two Project Description" --org
https://dev.azure.com/levsena/ --process Agile
--source-control  git
```

Step 3: Initialize the folder within Git

```
git init -b main
```

```
PS C:\Users\vminstructor\source\repos\Hello\helloworld2> git init -b main
Initialized empty Git repository in C:/Users/vminstructor/source/repos/Hello/HelloWorld2/.git/
PS C:\Users\vminstructor\source\repos\Hello\helloworld2> git status
On branch main

No commits yet
```

Step 4: Assign Name and Email to the global git settings (if this has not been done previously)

```
git config --global user.email "levsena@macewan.ca"
git config --global user.name "Allan Levsen"
```

You can verify this - as follows

```
PS C:\Users\vminstructor\source\repos\Hello\helloworld2> git config --global user.name
Allan Levsen
PS C:\Users\vminstructor\source\repos\Hello\helloworld2> git config --global user.email
levsena@macewan.ca
PS C:\Users\vminstructor\source\repos\Hello\helloworld2>
```

Step 5: Assign the remote url

git remote add origin
git@ssh.dev.azure.com:v3/levsena/HelloWorldTwo/HelloWorldTwo

We can verify that this worked using the **git remote -v** command, as follows

```
PS C:\Users\vminstructor\source\repos\helloworld> git remote -v
origin  git@ssh.dev.azure.com:v3/levsena/HelloWorld/HelloWorld (fetch)
origin  git@ssh.dev.azure.com:v3/levsena/HelloWorld/HelloWorld (push)
PS C:\Users\vminstructor\source\repos\helloworld>
```

Step 6: Using Visual Studio Code, add a .gitignore and index.html file
The following URL has a common Angular version of gitignore
https://github.com/angular/angular-cli/blob/main/.gitignore

Then, create an HTML file and add the following text within it.

Step 7: Save, Add, and commit these files

```
PS C:\Users\vminstructor\source\repos\Hello\helloworld2> git add .
PS C:\Users\vminstructor\source\repos\Hello\helloworld2> git commit -m "Initial Commit"
[main (root-commit) dc9fb24] Initial Commit
 2 files changed, 53 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 index.html
PS C:\Users\vminstructor\source\repos\Hello\helloworld2>
```

Step 8: Push the commit to the Remote - Azure DevOps Repo

```
PS C:\Users\vminstructor\source\repos\Hello\helloworld2> git push -u origin --all
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 763 bytes | 127.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Analyzing objects... (4/4) (53 ms)
remote: Storing packfile... done (104 ms)
remote: Storing index... done (87 ms)
To ssh.dev.azure.com:v3/levsena/HelloWorldTwo/HelloWorldTwo
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```