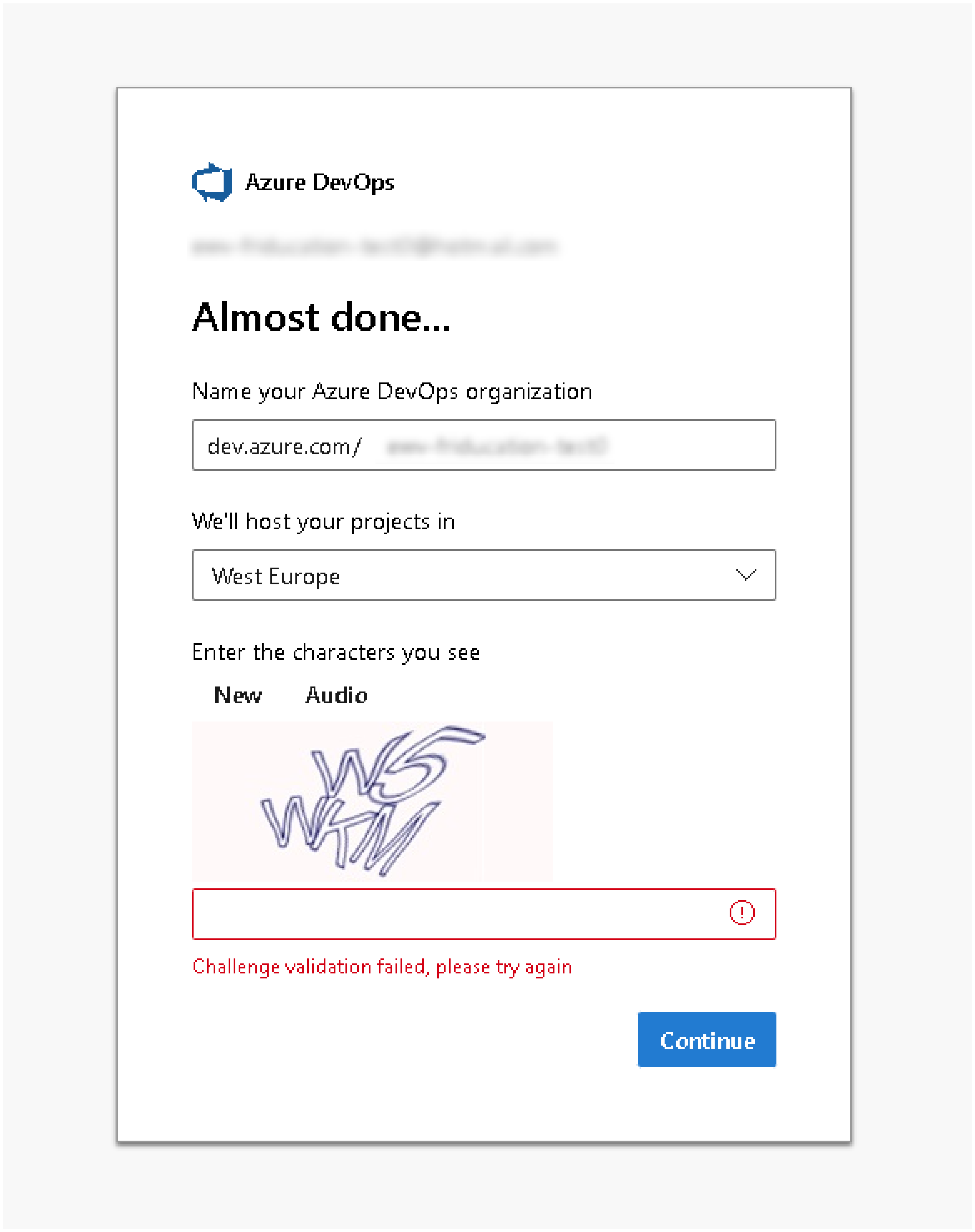
# Friducation DevOps Workshop

## Setting up the environment

1. Browse to www.hotmail.com and create a new hotmail account.
   Don't forget to store the username and password for later use.
2. Browse to dev.azure.com and click the button that says Start Free.
   *Note: you may need to log out of your existing Microsoft account first.*
3. Login with your new hotmail account.
4. Follow the prompt on the screen to create an organization.
   ⚠ The organization name must be unique in the world (so don't use the one in the screenshot)


Azure DevOps

[redacted]

Almost done...

Name your Azure DevOps organization

dev.azure.com/ [redacted]

We'll host your projects in

West Europe

Enter the characters you see

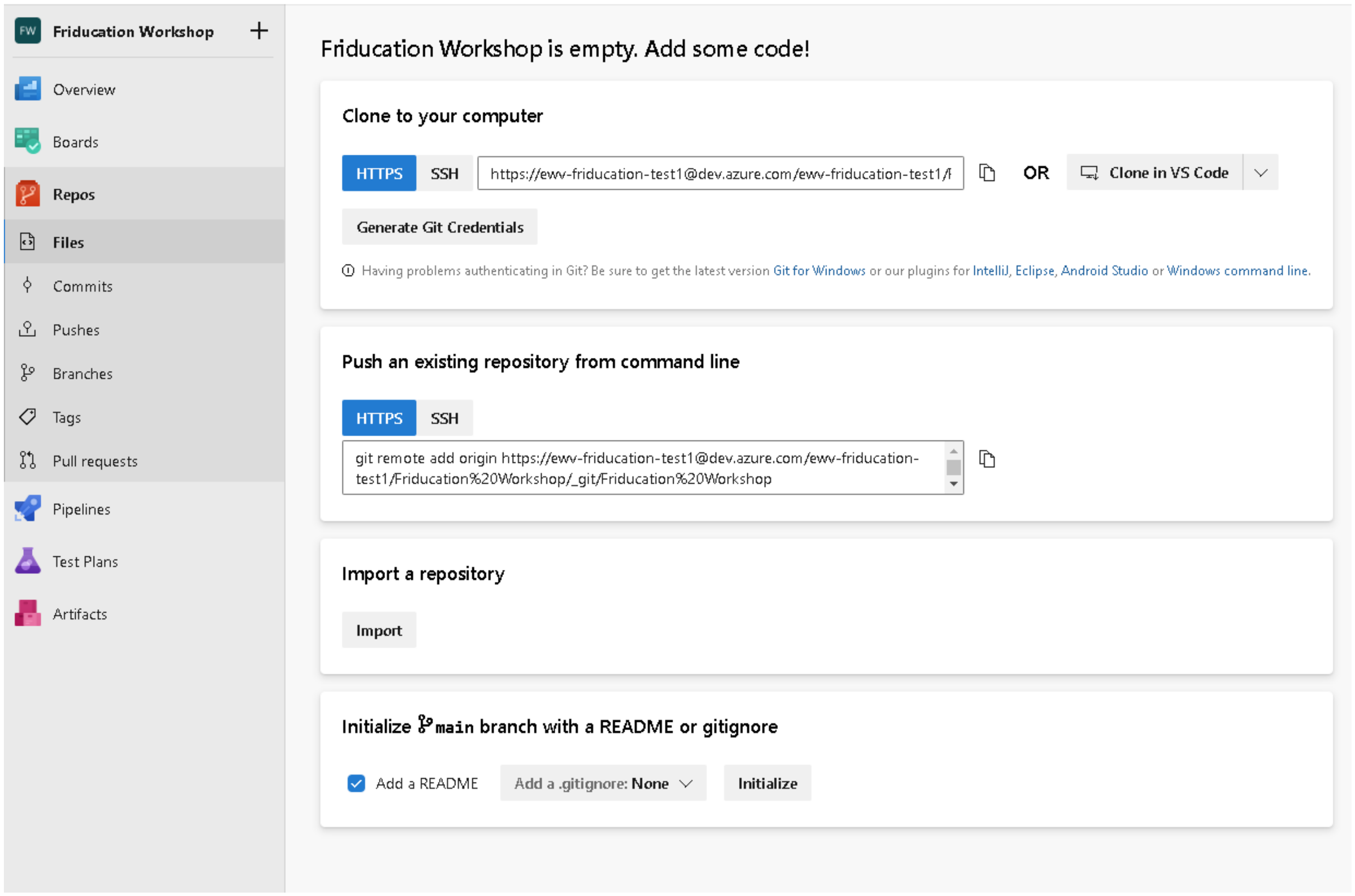
Challenge validation failed, please try again

5. Follow the instructions on the page to create a new Project. Make sure it is a Private project. Name it: `Friducation Workshop` .

## Import a code repository

Now that we have a project in Azure DevOps, we need to import the code for our website.

1. The website should automatically forward you to the Project Configuration page after creating it. In the menu on the left, click on 'Repos'.



Friducation Workshop is empty. Add some code!

Clone to your computer

HTTPS SSH https://ewv-friducation-test1@dev.azure.com/ewv-friducation-test1/f OR Clone in VS Code

Generate Git Credentials

Having problems authenticating in Git? Be sure to get the latest version Git for Windows or our plugins for IntelliJ, Eclipse, Android Studio or Windows command line.

Push an existing repository from command line

HTTPS SSH

```
git remote add origin https://ewv-friducation-test1@dev.azure.com/ewv-friducation-
test1/Friducation%20Workshop/_git/Friducation%20Workshop
```

Import a repository

Import

Initialize `main` branch with a README or gitignore

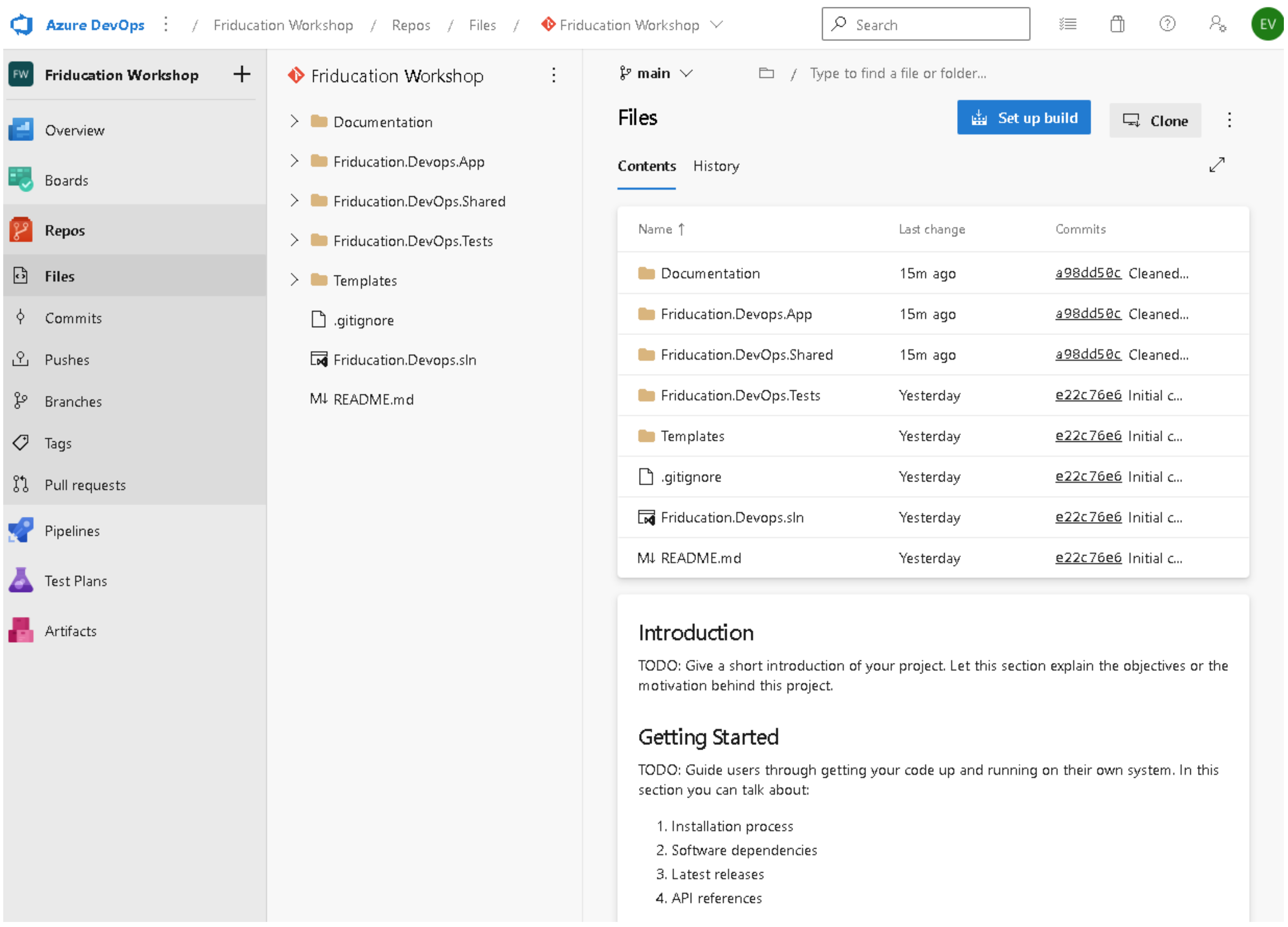
✓ Add a README   Add a .gitignore: None   Initialize

2. Click on the button that says 'Import' under 'Import Repository'.
3. In the 'Clone Url' field, copy paste the following url:

   https://github.com/everboom/FriducationWorkshop.git

   This will copy the code from the demo website into our project.

   *Note: You can leave the option "Requires Authentication" unchecked.*

4. 

| Name ↑ | Last change | Commits |
|---|---|---|
| Documentation | 15m ago | a98dd50c Cleaned... |
| Fridrucation.Devops.App | 15m ago | a98dd50c Cleaned... |
| Fridrucation.DevOps.Shared | 15m ago | a98dd50c Cleaned... |
| Fridrucation.DevOps.Tests | Yesterday | e22c76e6 Initial c... |
| Templates | Yesterday | e22c76e6 Initial c... |
| .gitignore | Yesterday | e22c76e6 Initial c... |
| Fridrucation.Devops.sln | Yesterday | e22c76e6 Initial c... |
| README.md | Yesterday | e22c76e6 Initial c... |

Introduction

TODO: Give a short introduction of your project. Let this section explain the objectives or the motivation behind this project.

Getting Started

TODO: Guide users through getting your code up and running on their own system. In this section you can talk about:

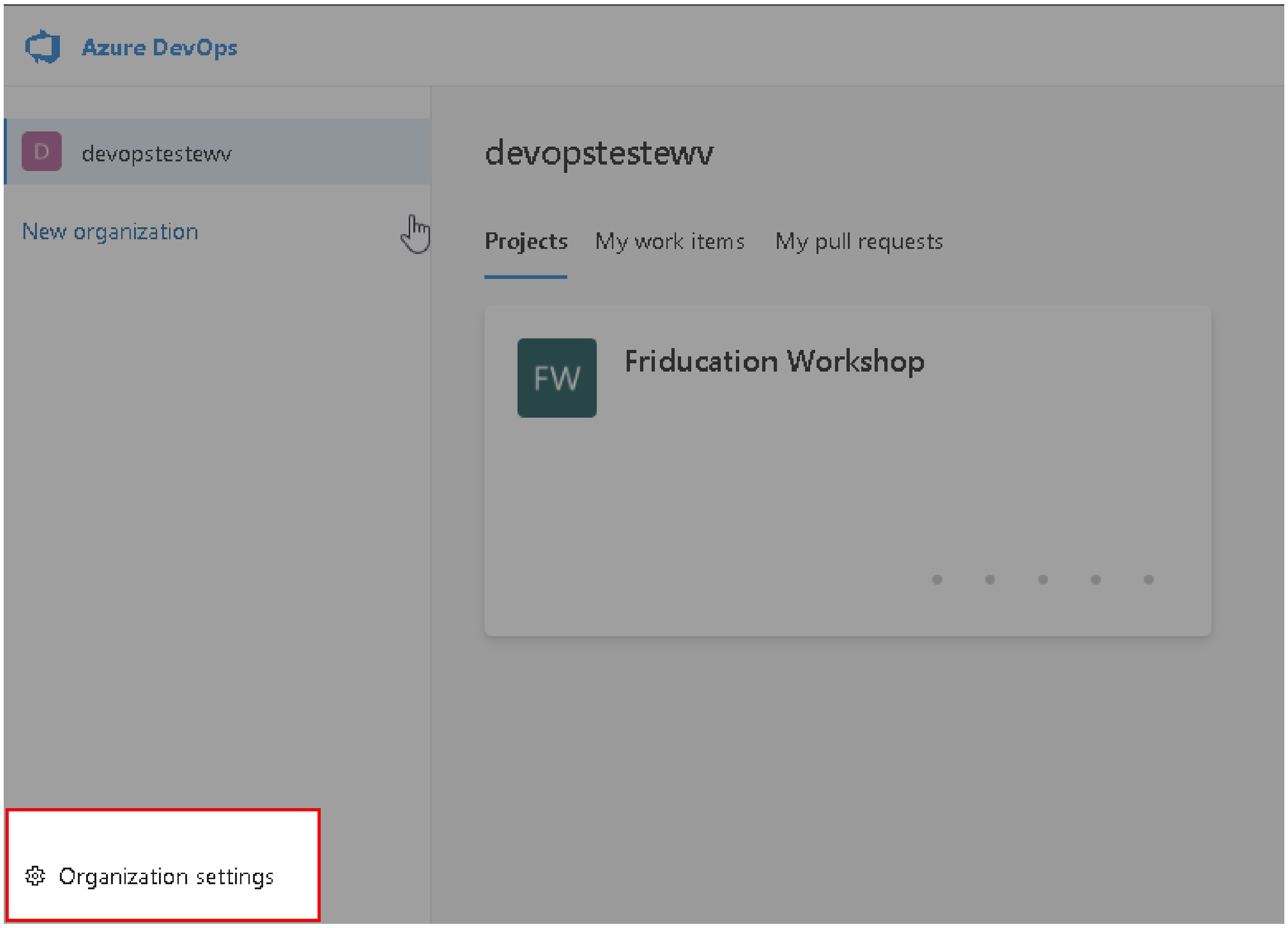
1. Installation process
2. Software dependencies
3. Latest releases
4. API references

5. Feel free to take a look around the code in the page that opens. You can also find the page through the Project Configuration menu under Repos -> Files.

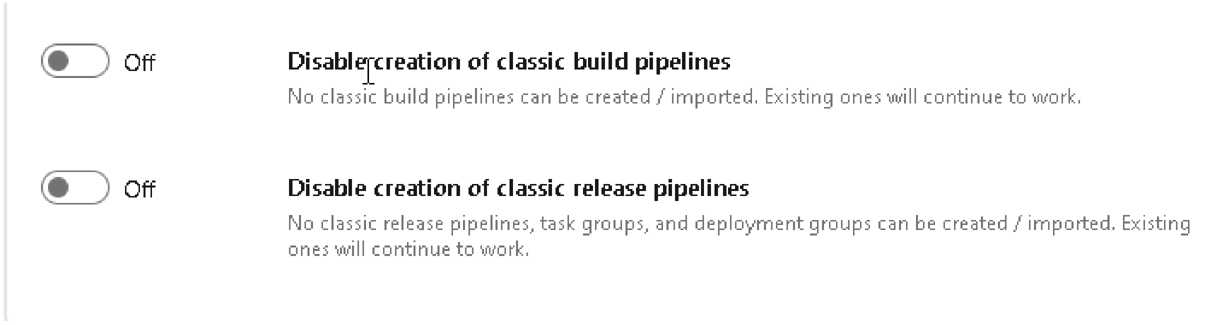
## Enable Classic Pipelines

In the next step we will create our own build pipeline. Microsoft is in the process of deprecating classic pipelines, so we need to turn this feature on manually from the organization settings.

1. Click on the **Organization Settings** button in the bottom left.



2. In the menu on the left, under the header **Pipelines** find the **Settings** page and navigate to it.
3. Then under **General** find the following to options and uncheck them, so they look like this:



Off — **Disable creation of classic build pipelines**
No classic build pipelines can be created / imported. Existing ones will continue to work.

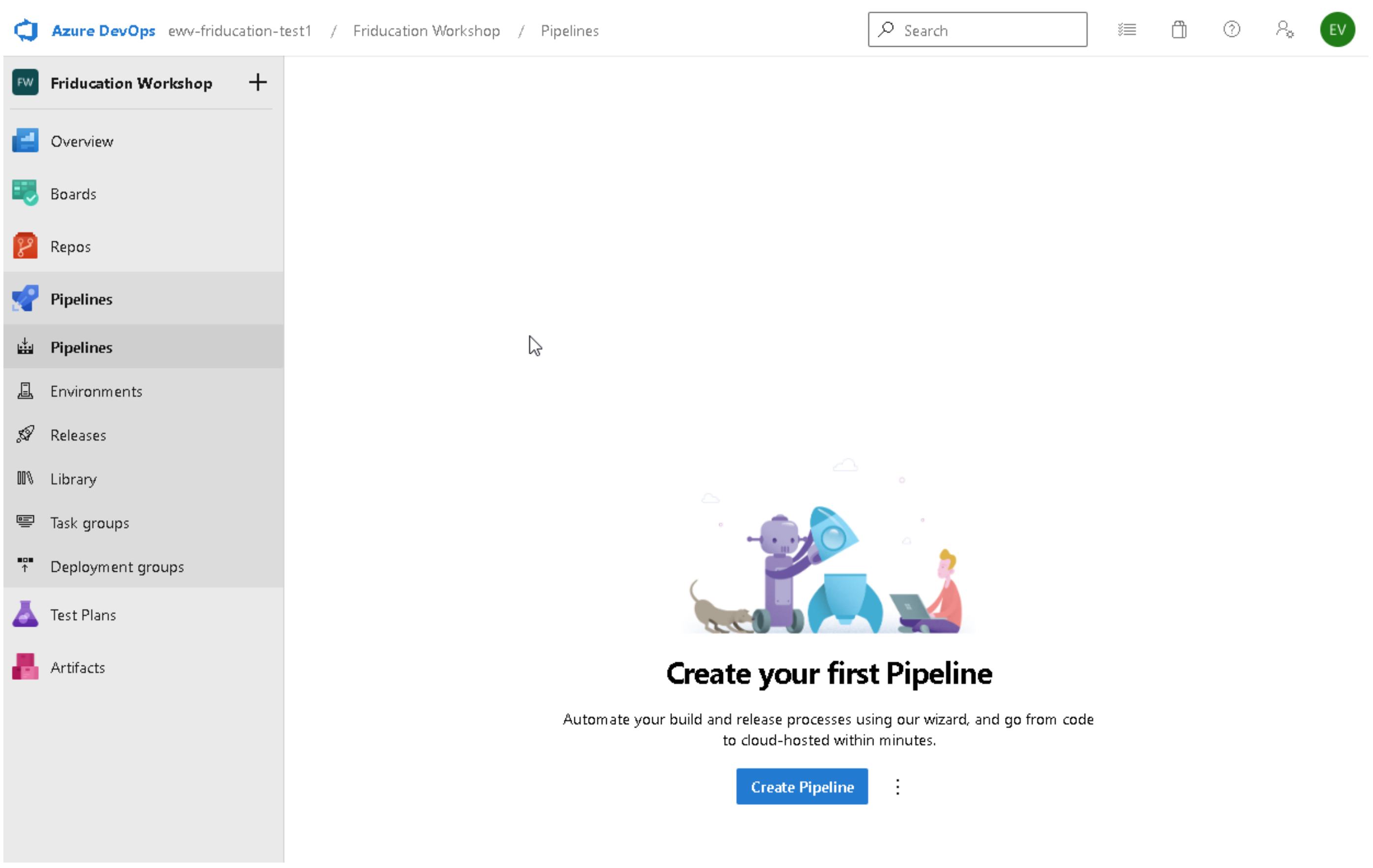
Off — **Disable creation of classic release pipelines**
No classic release pipelines, task groups, and deployment groups can be created / imported. Existing ones will continue to work.

You will now be able to create classic pipelines.

## Create a build pipeline

In the a previous step we imported the code for a simple website into our project.
Now we need to automate the build process of this website.

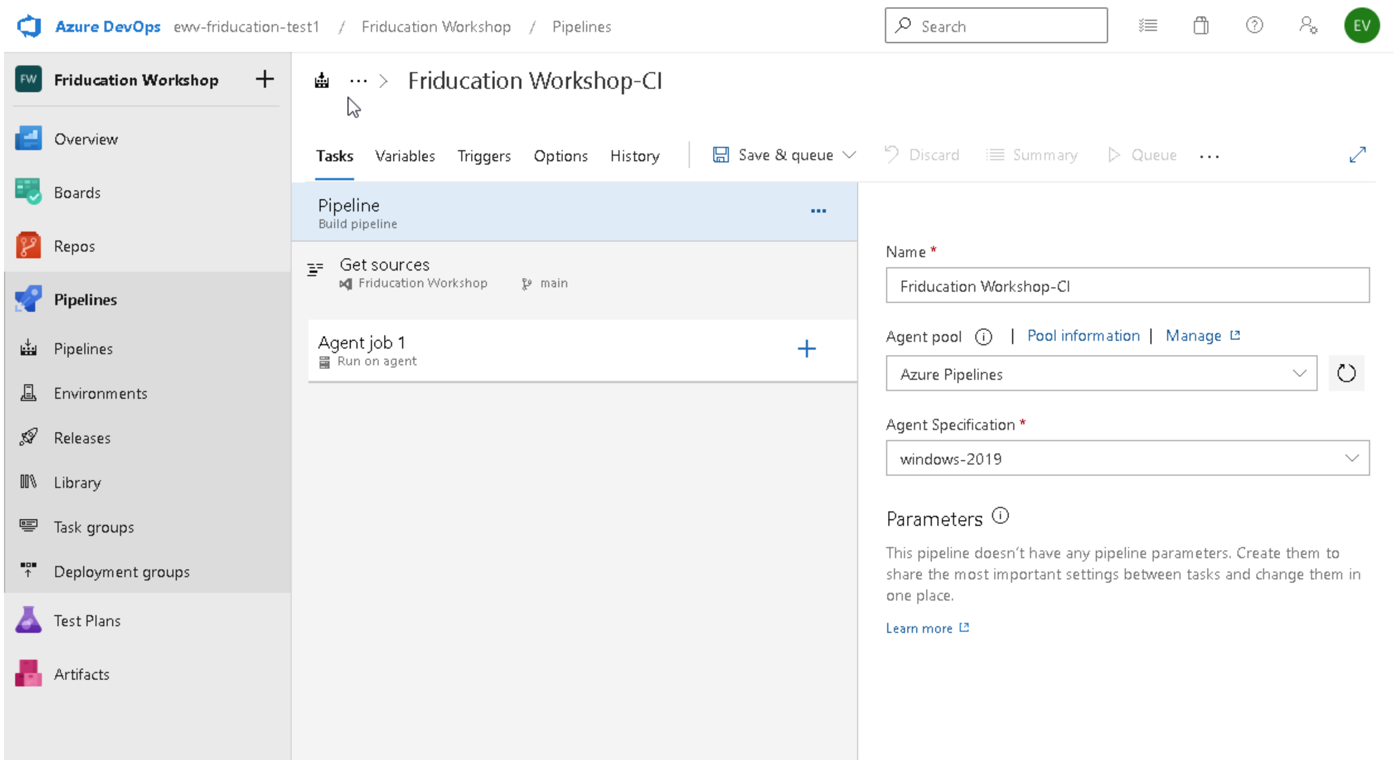
1. Browse to the Project Configuration page. On the menu in the left, go to Azure Pipelines -> Pipelines. In other words, in the menu you can see the Azure Pipelines section consists of a number of sub-pages, including Pipelines, Releases, Library, etc. Navigate to the top one, called Pipelines.



Create your first Pipeline

Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.

2. Press the button 'Create Pipeline' to start creating a new pipeline.
3. At the bottom of the page, click the "Use the classic editor" link.
4. Make sure 'Azure Repos Git' is selected as your top option. It should also automatically select the project and repository we've created during these exercises. If you followed the earlier suggestion, these would both be called 'Friducation Workshop'.
5. Press 'Continue'.
6. At the top of the page, click 'Start with an empty job'. Your page should now look like this:



Name *
Friducation Workshop-CI

Agent pool | Pool information | Manage
Azure Pipelines

Agent Specification *
windows-2019

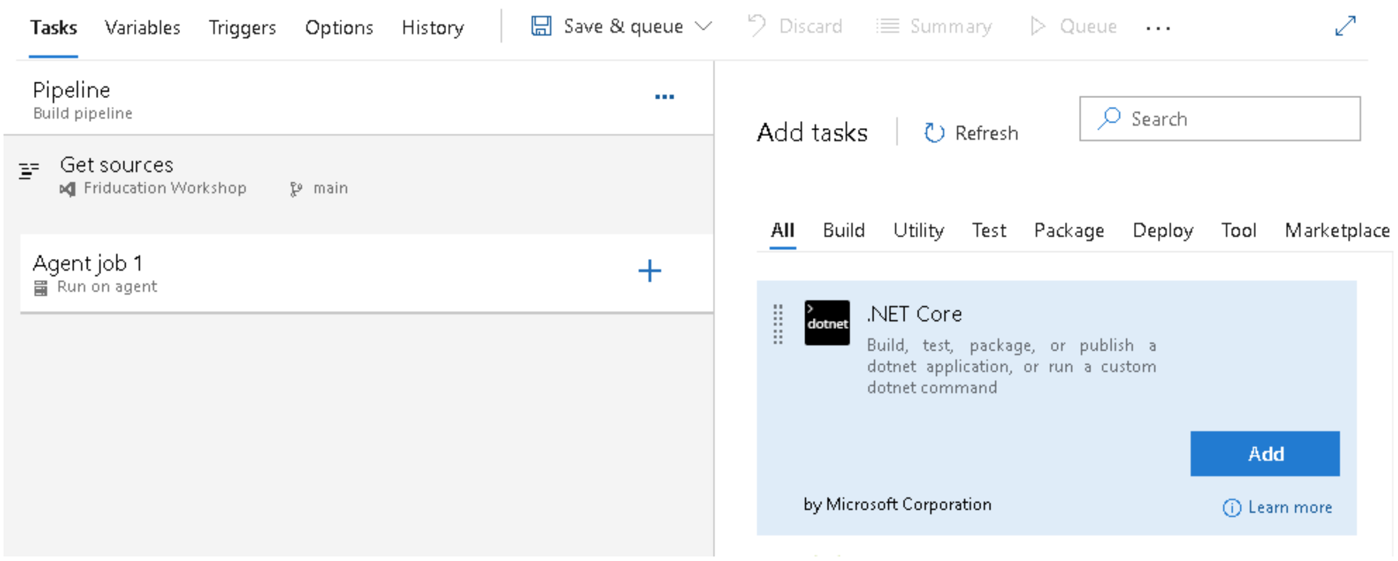
Parameters

This pipeline doesn't have any pipeline parameters. Create them to share the most important settings between tasks and change them in one place.

Learn more

Now that we've created a pipeline, we should make it do something! Let's add some tasks to build and test our website code.

7. On the bar that says "Agent Job 1", Press the + button to start adding a new Task.
8. Select the .NET Core task in the Marketplace.

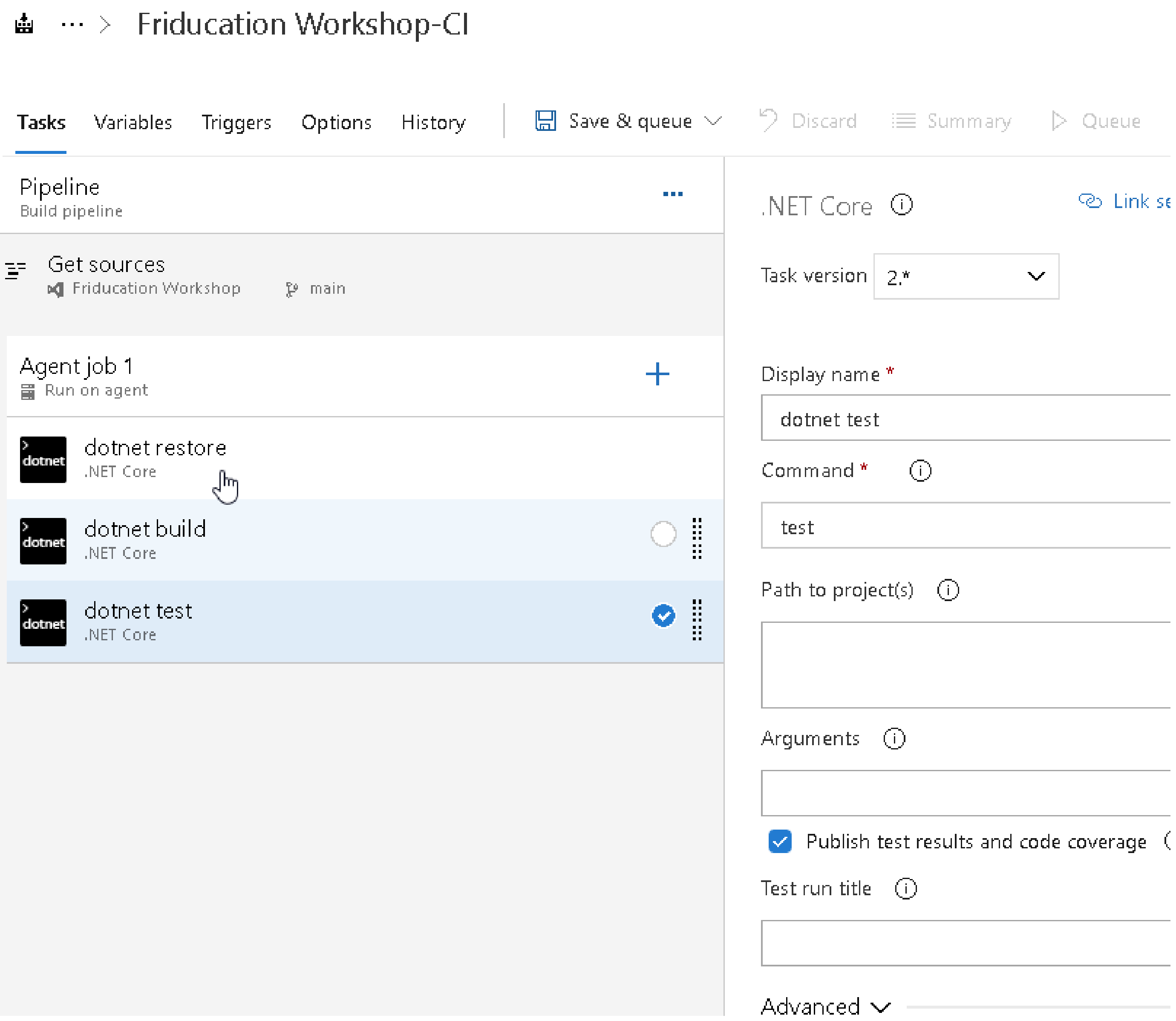


.NET Core
Build, test, package, or publish a dotnet application, or run a custom dotnet command
by Microsoft Corporation

9. Add the same task 2 more times.
10. Select the first task you added and set the 'Command' option to 'restore'. This will instruct DevOps to download the dependencies of the project.
11. Select the second task you added and set the 'Command' option to 'build'.
    This will turn the C# code into compiled code, including .DLL and .EXE files.
12. Select the third task you added and set the 'Command' option to 'test'.
    This will tell DevOps to run the project's unit tests and publish their results where you can easily

inspect them in the DevOps portal.

13. At the top of the page, Click 'Save and Queue', then 'Save'.
14. Your pipeline should look like this:



Pipeline
Build pipeline

Get sources
Friducation Workshop
main

Agent job 1
Run on agent

dotnet restore
.NET Core

dotnet build
.NET Core

dotnet test
.NET Core

.NET Core

Task version 2.*

Display name * dotnet test

Command * test

Path to project(s)

Arguments

☑ Publish test results and code coverage

Test run title

Advanced

We've constructed a pipeline to automatically download the dependencies for the project, then build the code and turn it into an application and finally, to run automated tests on them.

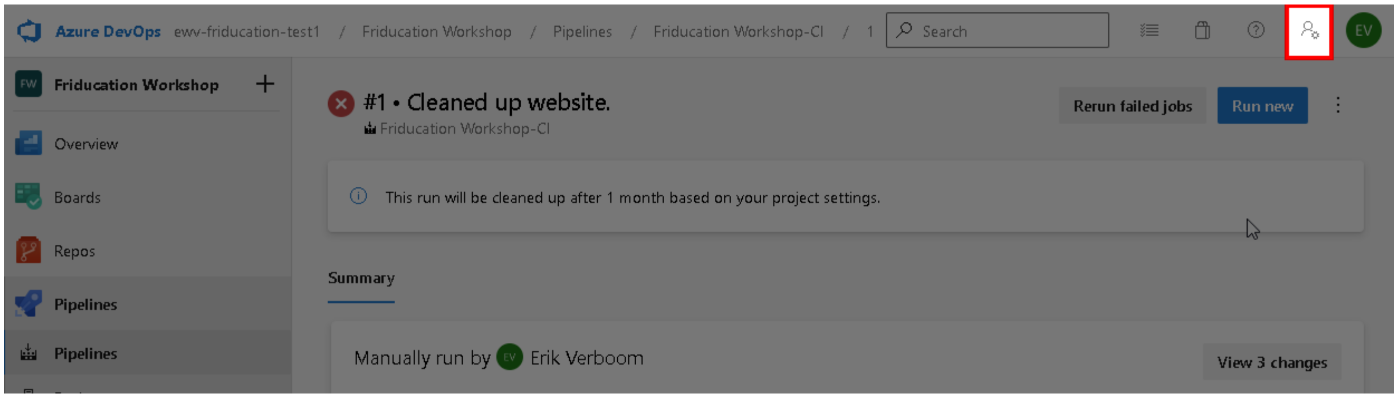
Let's see if the pipeline succeeds when we run it!

1. Navigate to the pipeline editing page if you're not there already. Press the Queue button, then in the new page click the 'Run' button to start the pipeline.
2. ⚠ The build will fail with the following error:
   No hosted parallelism has been purchased or granted.

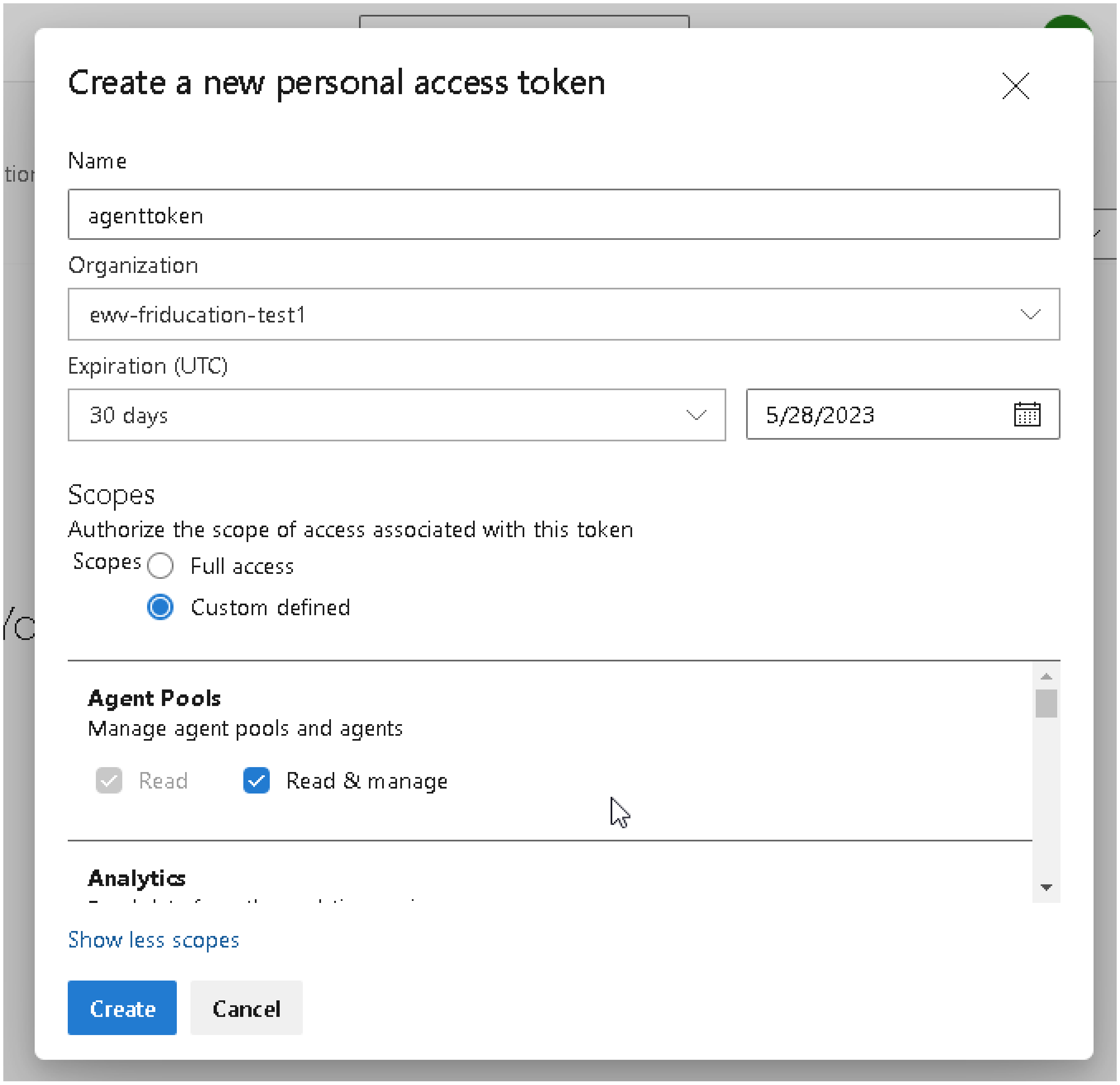
What does this mean? It means we do not currently have an agent available to run the pipeline. We could request free resources from Microsoft to do this, but this will take a day or two. We could also pay for the agents, but we probably want to just play around for now. The easiest and cheapest option then is to set up our own pipeline agent, so let's do that now (with the help of the teacher).

## Setting up a pipeline agent

To register an agent in our DevOps organization, we first need to select a computer or docker container that can do the work. Then we need to connect that machine to DevOps. In order to do that, we need a 'Personal Access Token'. This is a way for the agent application to show to Azure DevOps that it is authorized to connect to our DevOps organization.

1. Find the User Settings button in the top right of the DevOps screen, next to your user information. Near the bottom of the menu, find the option 'personal access tokens' and click it.


#1 • Cleaned up website.
Friducation Workshop-CI
This run will be cleaned up after 1 month based on your project settings.
Manually run by Erik Verboom
2. On the next page, press the button that says '+ New Token'.
3. Under Name, choose a name for your token, like 'agenttoken'.
4. At the bottom of the new panel, find the link that says 'Show all scopes' and click it.
5. Now at the top of the list, find the item 'Agent Pools'. Check the checkbox that says 'Read & Manage'. Your page should look like this:


Create a new personal access token

Name: agenttoken

Organization: eww-friducation-test1

Expiration (UTC): 30 days — 5/28/2023

Scopes

Authorize the scope of access associated with this token

Scopes: Full access [✓ Custom defined]

**Agent Pools**

Manage agent pools and agents

✓ Read ✓ Read & manage

**Analytics**

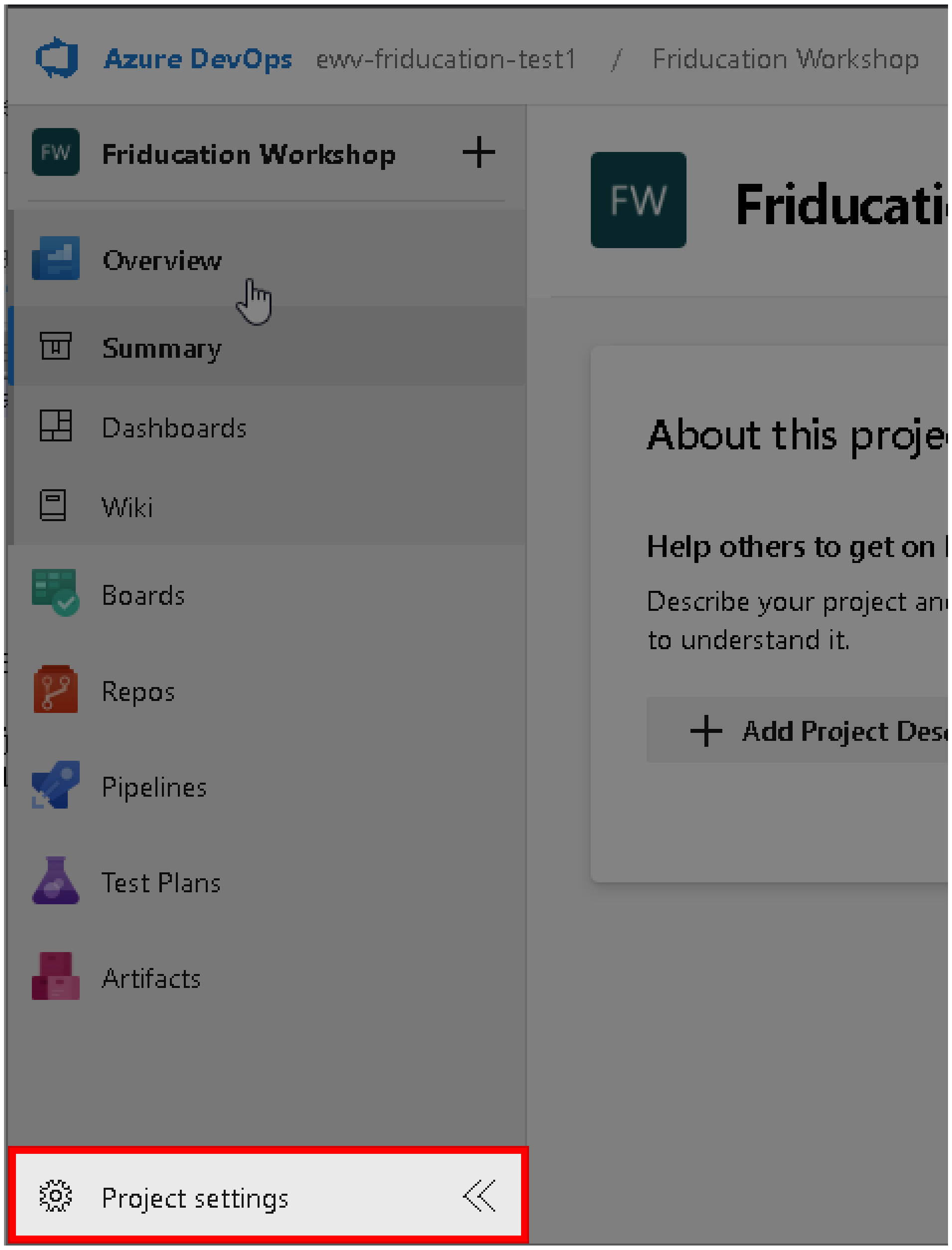
6. At the bottom of the panel, Press the 'Create' button.
7. On the next page, copy paste the code and store it in a text file. This is the Personal Access Token we'll need to create an agent. After this, press the 'Close' button to go back to the main Azure DevOps window.
8. Now navigate back to your DevOps project and copy paste the URL from your browser into the same text file. It should look similar to this:

```
https://dev.azure.com/[OrganizationName]/[ProjectName]
```

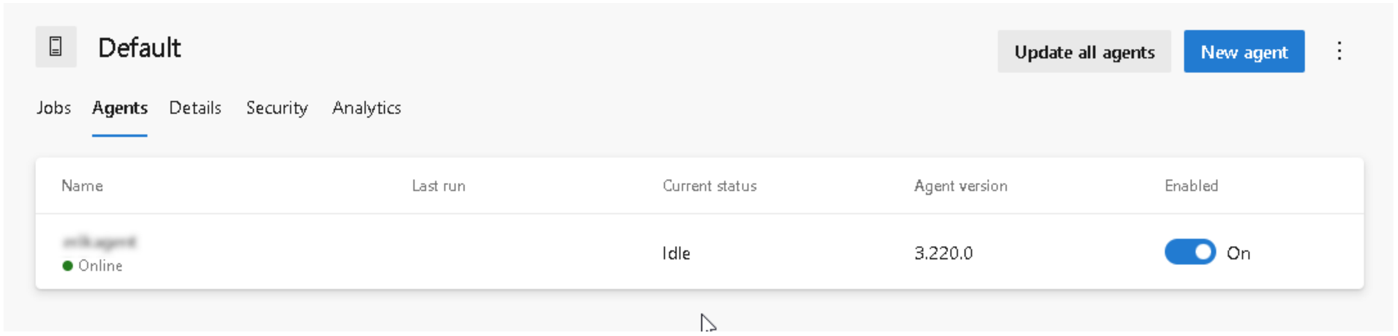
9. ⚠ Now copy paste both of the values to your teacher in Microsoft Teams, and he will create a new Self-Hosted Agent for you to use.

**Let's see if the new agent was added correctly!**

1. Navigate to your project configuration page and find the 'Project Settings' button in the bottom left, then click it.



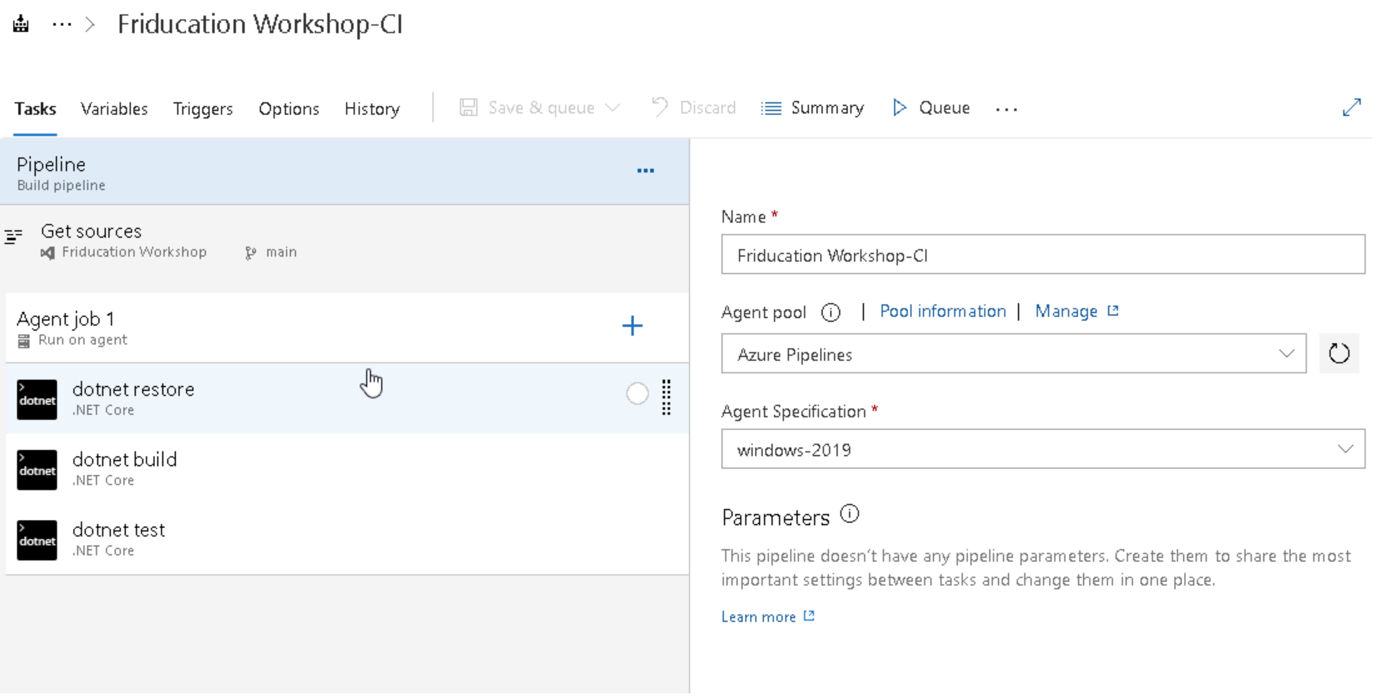
2. Under the Pipelines section, find the 'Agent Pools' page and navigate there.
3. Navigate to the 'Default' Agent pool.
4. Navigate to the 'Agents' tab.
5. The teacher will have started the setup of the build agent in the meantime. When this process is complete and the agent is up and running, the page will look like this:



| Name | Last run | Current status | Agent version | Enabled |
|---|---|---|---|---|
| [redacted] ● Online | | Idle | 3.220.0 | On |

We must now configure our pipeline to make use of the new agent.

1. Navigate back to the project configuration page, then to Pipelines -> Pipelines. Then click the pipeline we created earlier, called 'Friducation Workshop-CI'. Then in the top-right, click 'Edit'. You should see the following:



Name *
Friducation Workshop-CI

Agent pool
Azure Pipelines

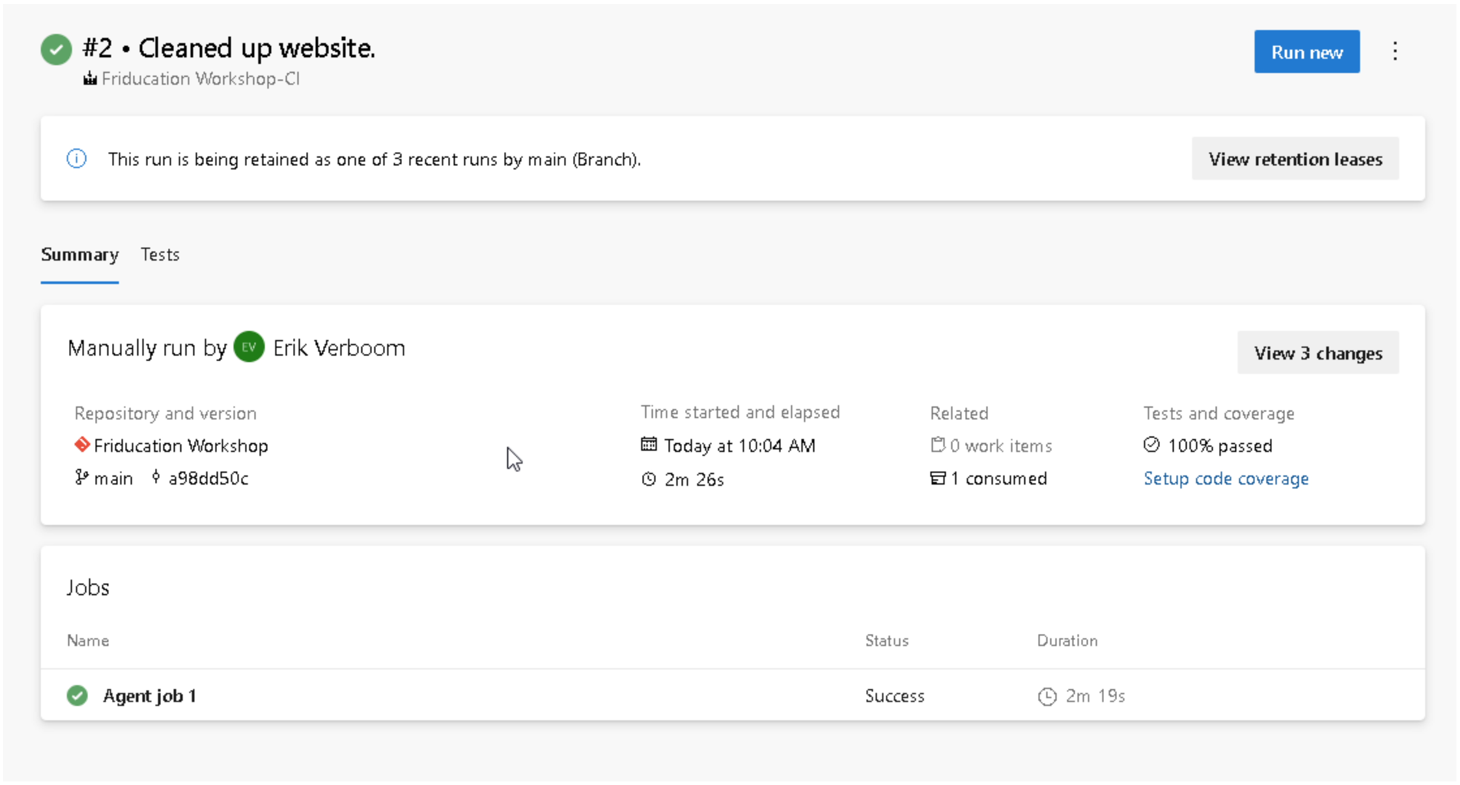
Agent Specification *
windows-2019

Parameters

This pipeline doesn't have any pipeline parameters. Create them to share the most important settings between tasks and change them in one place.

2. Click on the top bar, that says 'Pipeline'. Then change the setting for Agent pool from 'Azure Pipelines' to 'Default'.
3. At the top of the page, click on 'Save and Queue', then on the 'Save and run' button. You will be redirected to a page where you can see if the pipeline runs successfully.

The pipeline should run correctly this time. It may take a few minutes for it to complete. You can see the results in the pipeline run page.


#2 • Cleaned up website.
F培ducation Workshop-CI

This run is being retained as one of 3 recent runs by main (Branch).

Manually run by Erik Verboom

| Repository and version | Time started and elapsed | Related | Tests and coverage |
|---|---|---|---|
| Friducation Workshop | Today at 10:04 AM | 0 work items | 100% passed |
| main a98dd50c | 2m 26s | 1 consumed | Setup code coverage |

Jobs

| Name | Status | Duration |
|---|---|---|
| Agent job 1 | Success | 2m 19s |

- You can see the log messages of the restore, build and test phases by clicking on "Agent Job 1", next to the green checkmark, at the bottom.
- You can also find a detailed report for the Unit tests under 'Tests and coverage'.

One thing that is missing is that we're not storing the build artifact yet. We need to change our pipeline to publish it, so we can use it later on in our CD pipeline.

1. Navigate back to the pipeline editing page and add another .NET Core task. This one needs 'Publish' for its 'command' setting.
2. In the new task, copy paste the following text into the 'Arguments' field.
   ```
   --configuration Release --output $(Build.ArtifactStagingDirectory)
   ```
3. Finally, add a new task from the marketplace called 'Publish build artifacts'. You can find this task quickly by typing its name into the search bar.
4. Press the 'Save and Queue' button, then the 'Save and run' button.

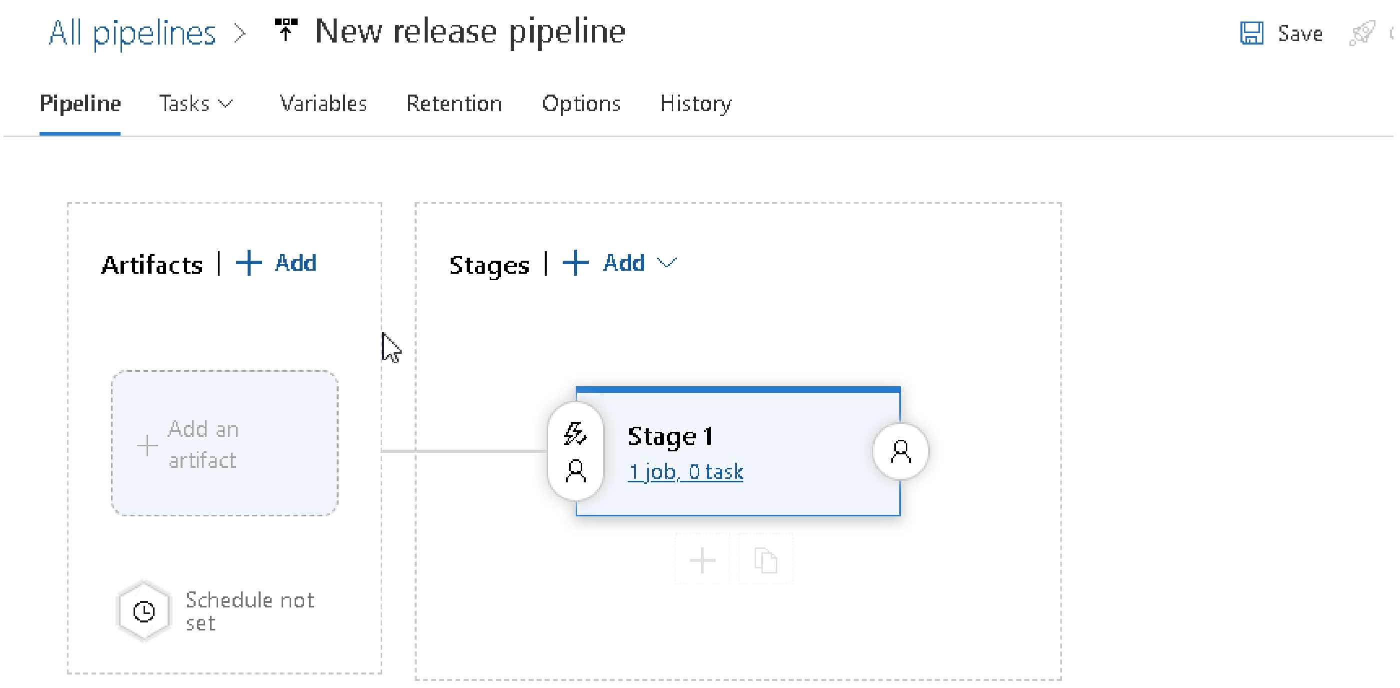
This will ensure that the pipeline produces a build artifact which we can publish to Azure and put our website online.

## Create a release pipeline

Let's now create a release pipeline which will publish the website to Azure.

1. In the project menu, click on Pipelines, then Releases.

2. Click the 'New Pipeline' button.
3. Under 'Select a template', click 'Empty job'.
4. Your page should now look like this:

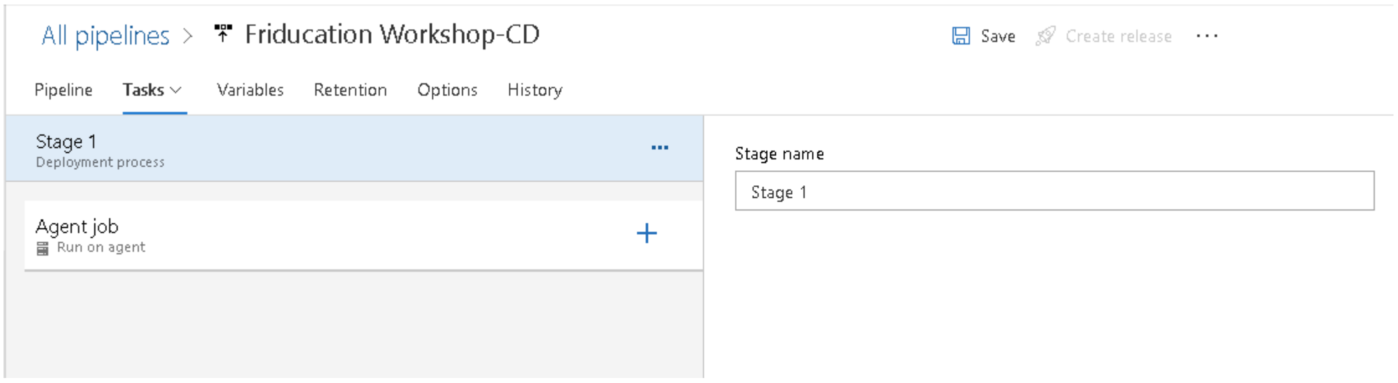


First, we must ensure the release pipeline has access to the original repository, because it contains a template that we will use to create our web application resource in Azure. This essentially allows us to host our website in the cloud.

1. Click on 'Add an Artifact'.
2. In the 'Source type' field, select 'Azure Repos Git' (it should be the second option).
3. For 'Project', select our project.
4. For 'Source (repository)', select its Repository.
5. For 'Default branch', select 'main'.
6. ⚠ Under the 'Source alias' field, remove the space so that the value becomes: `_FriducationWorkshop` .
7. Click 'Add' to add the artifact.

Let's continue configuring the pipeline.

1. At the top of the page, replace 'New Release Pipeline' with `Friducation Workshop-CD` to rename our pipeline to something more readable.
2. Next, click the Variables tab to set up a new variable.
3. Press '+ Add' to start adding a new variable. Under name, type `WebAppName` .

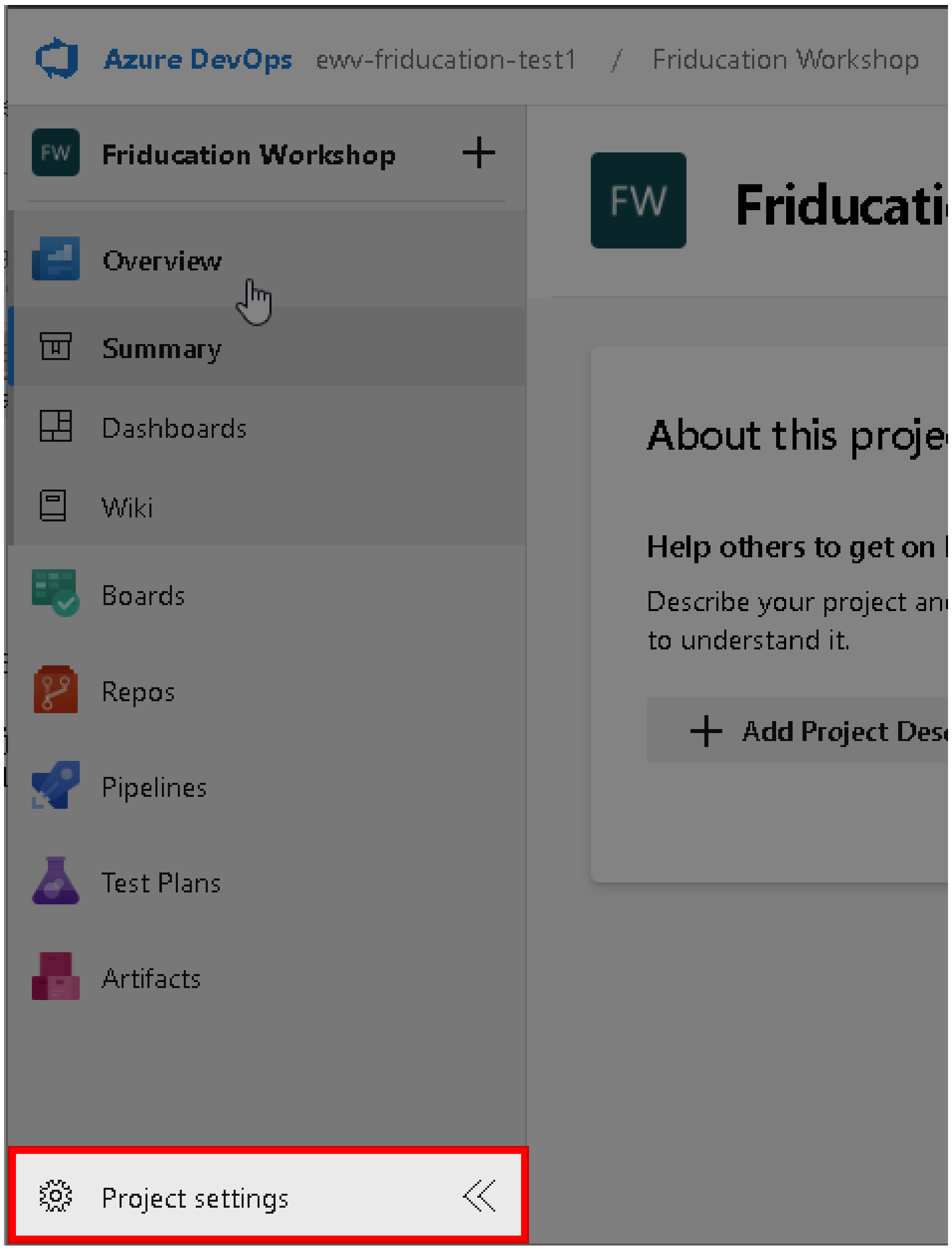
4. Under value, type a name with the format: `friducation-[yourname]-[randomnumber]` .
   We will use this variable in various places in our pipeline.
   ⚠ Only use small letters, dashes `-` and numbers in the name.
5. Go back to the Tasks tab in the Release Pipeline Editing page. Your page should look like this:
   

6. Rename the stage from 'Stage 1' to 'Deploy and Publish'.
7. Then click on the Agent Job and change the value under Agent Pool to 'Default', so that this pipeline can also use our custom pipeline agent.
8. Click the 'Save' button to save the pipeline.

## Create Azure Service Connection

We must now add a task that will create a new web application in Azure, but to do this we must authorize Azure DevOps to do some work in Azure. Normally you would link Azure DevOps to your own Azure environment during this step, but for this exercise this is simplified by using an existing Azure Subscription managed by the teacher.

1. Click Project Settings in the bottom left.

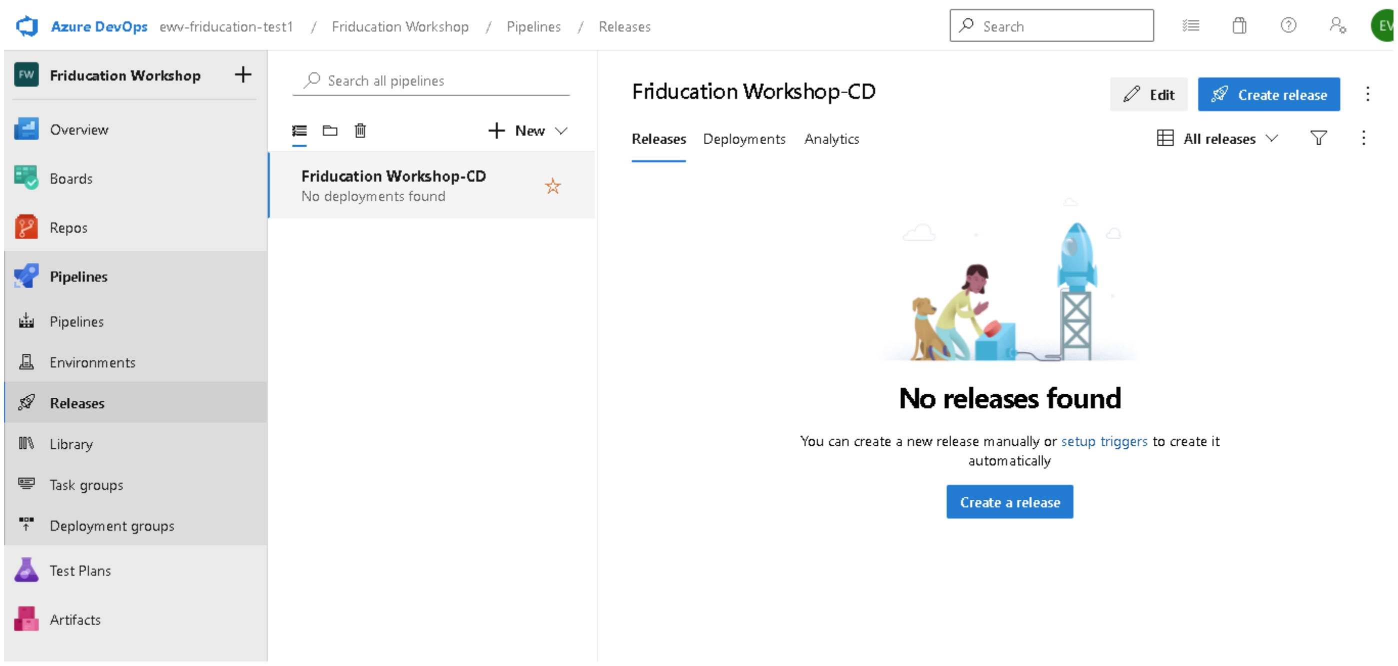


2. Then, navigate to 'Service Connections' under the 'Pipelines' section.
3. Click 'Create Service Connection'
4. Select 'Azure Resource Manager' and click on 'Next'.
5. Then select 'Service Principal (manual)' and click 'Next'.
6. For the 'Subscription Id' field, set value to: `f05495e0-9123-402d-89b6-0c8f63339b73`
7. For the 'Subscription Name' field, set value to: `DevOpsWorkshops`
8. For the 'Service Principal Id', set value to `3e45a629-8654-49f4-87a2-482f529c1996`
9. For the 'Service Principal Key', set value to: `9SU8Q~9MJaCJxZbLLInaZrtfAnG-_7f8vfC1KaXF`
10. For the 'Tenant Id', set value to: `d9e835f3-6333-4986-ba37-74778153ebc5`
11. Click 'Verify' to ensure you've set the values correctly.
12. For 'Service Connection name', set value to: `Azure Connection`
13. Check the box that says 'Grant access permission to all pipelines'.
14. Save the Service Connection.

We can now finish setting up the release pipeline. It needs two tasks: one to create a web application in Azure, and one to publish our build artifact to it.

## Deploy a web application to Azure

To create a web application service in Azure which can host our website, we are going to use a task that deploys it by using a template.

1. Navigate back to the Release Pipeline we created in an earlier step.
   You should see the following:
   

   No releases found

   You can create a new release manually or setup triggers to create it automatically
2. Click the 'Edit' button in the topright.
3. Navigate to the tasks header.
4. Use the '+' button on the 'Agent Job' bar to add an ARM Template Deployment task.
5. Click on the new task to edit it.
6. In the dropdown menu for 'Azure Resource Manager' select our new Service Connection, called `Azure Connection` .
7. Under the Subscription field, select `DevOpsWorkshops` .
8. In the Resource Group field, type: `$(WebAppName)`
9. In the Location field, select `West Europe`
10. Under the Template field, select the Ellipses button (three dots), to find the template we need to deploy.
11. In the new window, navigate to the item that ends in (Azure Repos Git), then to the folder Templates, then select webapp.bicep. Then press 'Ok'.
12. The value under Template should look like this:
    `$(System.DefaultWorkingDirectory)/[Project Name]/Templates/webapp.bicep`

13. Under **'Override template parameters'**, copy paste: `-webAppName $(WebAppName)`
    ⚠ Be sure to place value in **'Override template parameters'**, not 'Template parameters'.
14. Save the Release Pipeline.

If you run the release pipeline now, it should create a new web app in Azure that we can then publish to.

1. You can go ahead and run the release pipeline now if you wish, by clicking the 'Create Release' button, then 'Create'. While it's running, we can go back and add the final step to the pipeline.

## Publish our code

We are now going to publish our website to Azure. First, we must ensure that the release pipeline knows how to find the build artifact from the previous pipeline.

1. Navigate to the Pipeline tab on Release Pipeline editing page. On the left under Artifacts, click `+ Add` to add a source artifact to the Release Pipeline.
2. In the 'Source type' field, make sure `Build` is selected.
3. In the source field, select `Friducation Workshop-CI` . This is the build pipeline we created earlier.
4. At the bottom of the panel, click 'Add'.

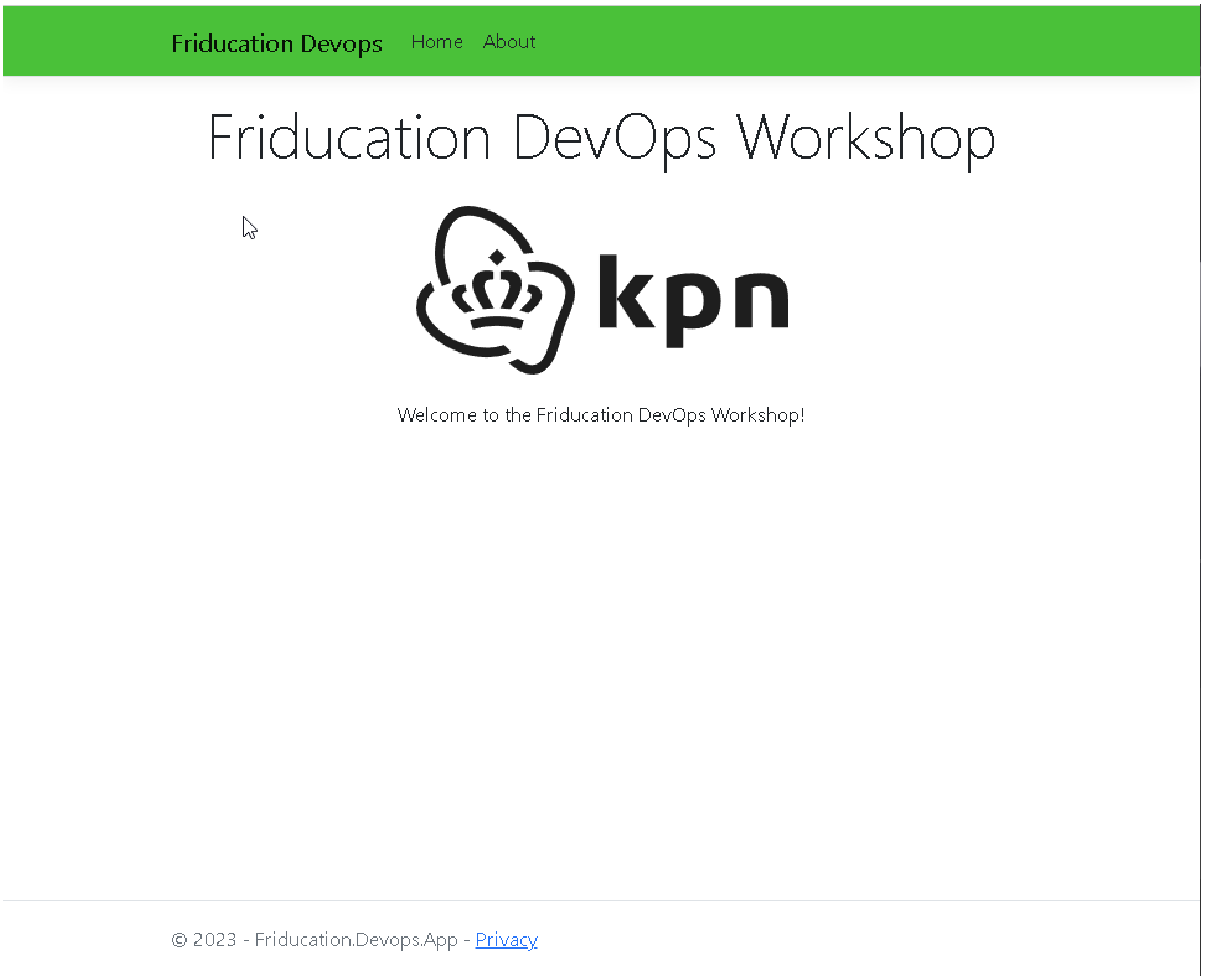
Now we can add a step to our release pipeline to publish our website to Azure.

1. Navigate to the Tasks editing page of your release pipeline and add a task to the Agent Job. Use the marketplace to find the task with the name: 'Azure App Service Deploy'.
2. Click on the newly added task to start editing its settings.
3. Under 'Azure Subscription', select the Service Connection we created earlier, which should be called `Azure Connection` .
4. Under App Service Name, type `$(WebAppName)`
5. Save the pipeline and press the button 'Create Release'.

## End result

Once the release finishes running, you should be able to visit your website at:
*https://[yourname].azurewebsites.net*
Where [yourname] is the value you chose for the variable `$WebAppName` .

If all went well, you will see this page:



Friducation DevOps Workshop

Welcome to the Friducation DevOps Workshop!

© 2023 - Friducation.Devops.App - Privacy

## Next steps

So what's next? Well, you can play around with the project you've created. Try creating some work items to plan work, then change the code in the repository to make changes to the website. Run your CI/CD pipelines and track the result!

If you want to learn more, look into the official Azure DevOps course, called AZ-400. There, you'll learn all sorts of advanced techniques about how to set up your pipelines and use them to maximum effect, and all sorts of techniques to make you more productive and allow you to spend time on creating cool new features for your application!