Hybrid Devops Demo – Setup Guide

|  |  |
| --- | --- |
| Version | Author |
| 1.0 draft | Benjamin Kirnich |
| 1.1 draft ( Changed Github Repo) | Benjamin Kirnich |
|  |  |

Contents

[Preconditions 2](#_Toc515292422)

[Azurestack 2](#_Toc515292423)

[VSTS Account 3](#_Toc515292424)

[Build Agent 3](#_Toc515292425)

[Azure 3](#_Toc515292426)

[Buildfile Location 4](#_Toc515292427)

[Setup 5](#_Toc515292428)

[VSTS Project 5](#_Toc515292429)

[Import Code Repository 6](#_Toc515292430)

[Create Build Definition 7](#_Toc515292431)

[Create Release Definition 8](#_Toc515292432)

[Testing 13](#_Toc515292433)

[Preparing the demo 16](#_Toc515292434)

[Backlog and work items 16](#_Toc515292435)

[VSTS Dashboard 17](#_Toc515292436)

# Preconditions

## Azurestack

Have an installed Azure Stack instance either an One Not Devbox or your local MTC multimode installation available. The instance has to be deployed with SQL and App Service services since the demo is leveraging only Platform services to show the full potential.

For the full Demo potential and VSTS Dashboard functionality the App service later deployed on your local Azurestack instance has to be published to the internet and reachable via http:80 to make the Application Insights availability tests work.

The Demo is using the following Azurestack values. If you are using other naming in your installation please adjust the values either in the VSTS Environment variables for Azure Stack within the Hybrid Release pipeline or the Deployment files integrated in the code repo under src/cotosoweb/deployment. If you freshly install an azurestack environment for the demo, sticking to this values would save you the adjustments.

|  |  |  |
| --- | --- | --- |
| Variable | Value | Location |
| AppServicePlanSKUName | D1 | Pipeline Variable |
| Region | Local | Pipeline Variable |
| Sqlskutier | Sql | Deploy2.json |
| Sqlskuname | Standalone | Deploy2.json |
| Sqlskufamily | Sql2016 | Deploy2.json |
| Sqlserver in connectionstring | sqlworker1.local.cloudapp.azurestack.external | Deploy2.json |

Feel free to adjust all other variables and values to your needs if needed

## VSTS Account

We need an Visual Studio Team Services Account to host the project. You can use an already existing account if you already got one or you can create a new one under <https://www.visualstudio.com/de/team-services/> . A Free account is sufficient, the account should able to be shared between al persons using your demo installation, so you would maybe prefer to not use our personal one.

## Build Agent

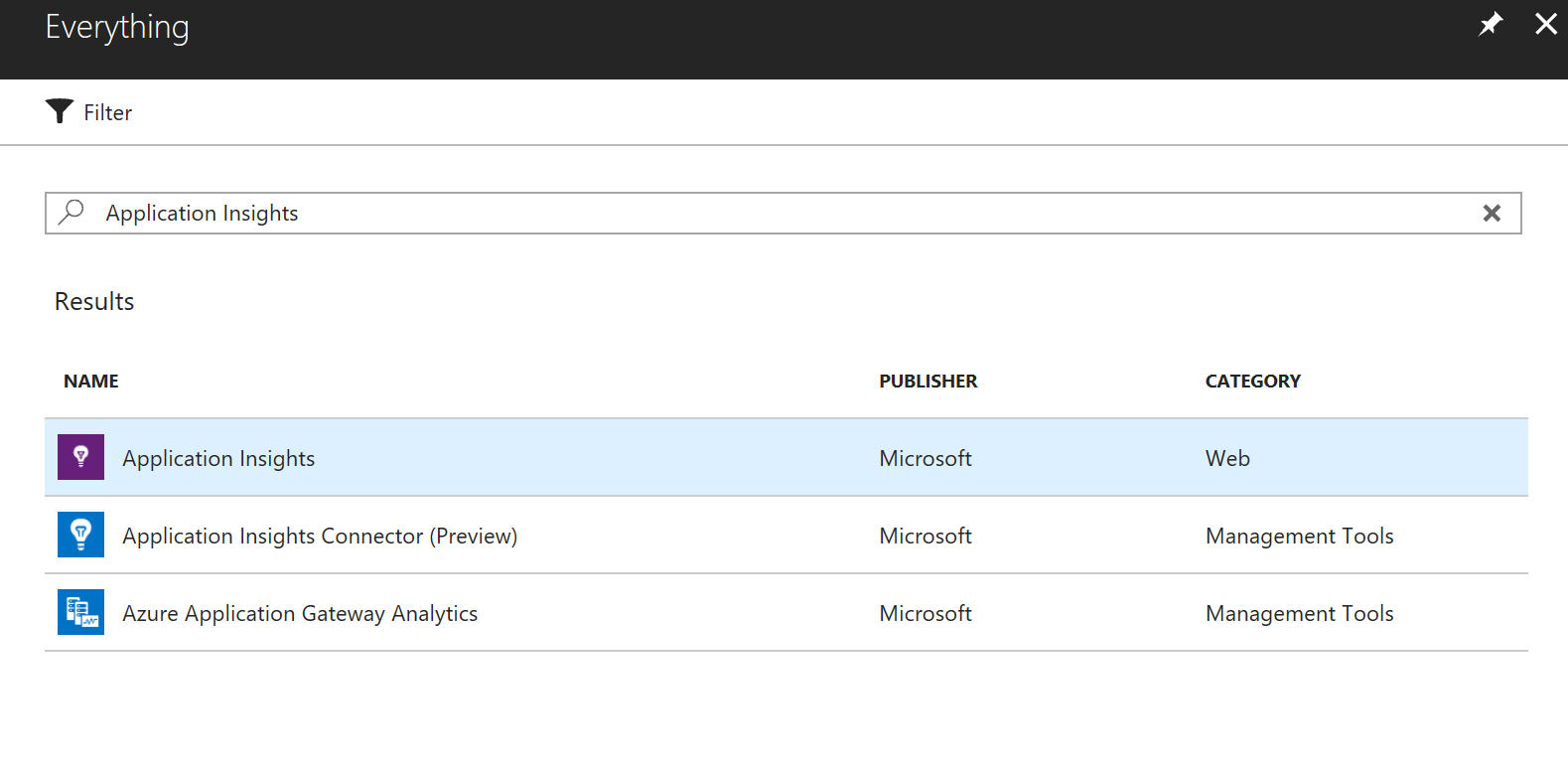
We must install our won build agent locally to ensure the agent can reach our local Azurestack. We use an W2k16 VM hosted on the Azurestack instance in a dedicated resource group. The Vm needs internet access. Install a Vm and then install Visualstudio Community 2017 via visualstudio.com and check all .net, web and cloud dependencies during the installation.

After visual studio has been installed follow this process to install the Build Agent and link it to your vsts account.

<https://docs.microsoft.com/de-de/vsts/build-release/actions/agents/v2-windows?view=vsts>

## Azure

Create a resource group in your Azure Subscription were we can host the Application Insights instances for the environments we will use in the Demo. Create a Application Insights Instance for each environment one for Dev, one for QA and one for PRD. Use ASP.Net Web application as Application Type.

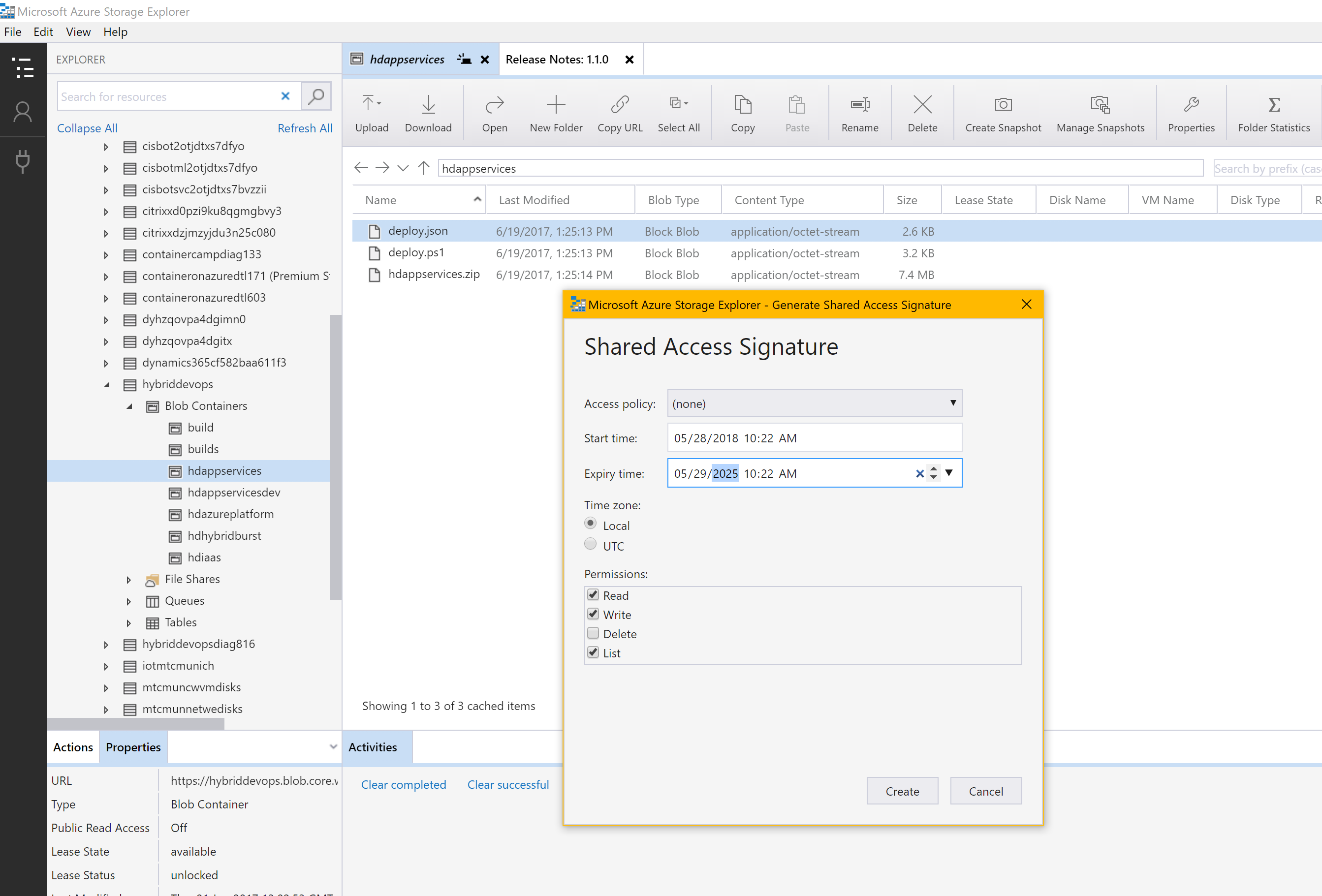


Also create an Azure Blob Storage Account we need in the next step-

## Buildfile Location

We need a location where we can place the Application Package after the build process where it is accessible from all locations we deploy to. In this Demo we are using an Azure Blob for this.

Open Azure Storage Explorer and navigate to your crated Storage Account. Create an Blob Container we will use to store the data. Right Click it and select get a shared access signature.



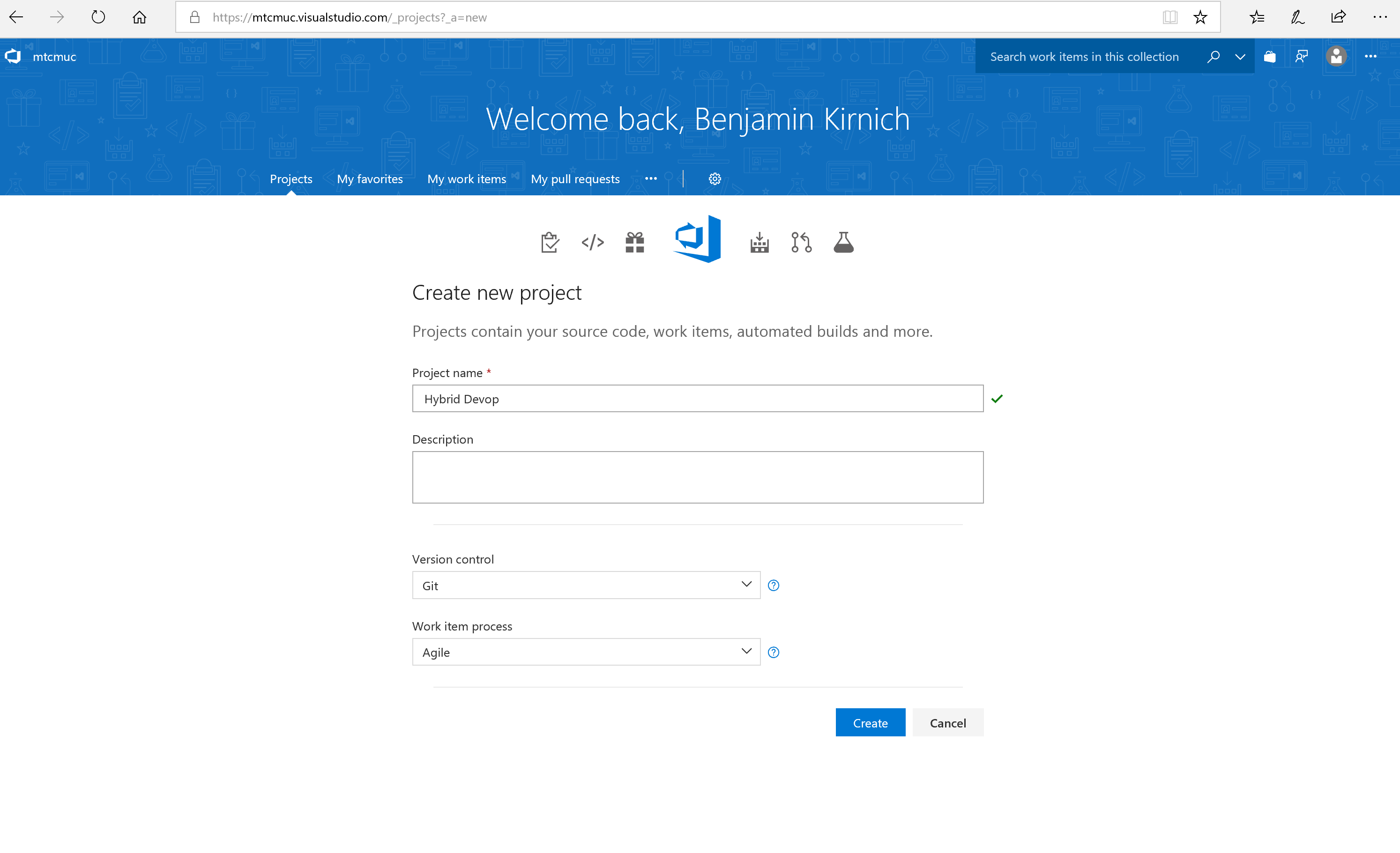
Choose an appropriate Expiry Date and select read, write and list and click create.

Copy the URL and save it for later.

# Setup

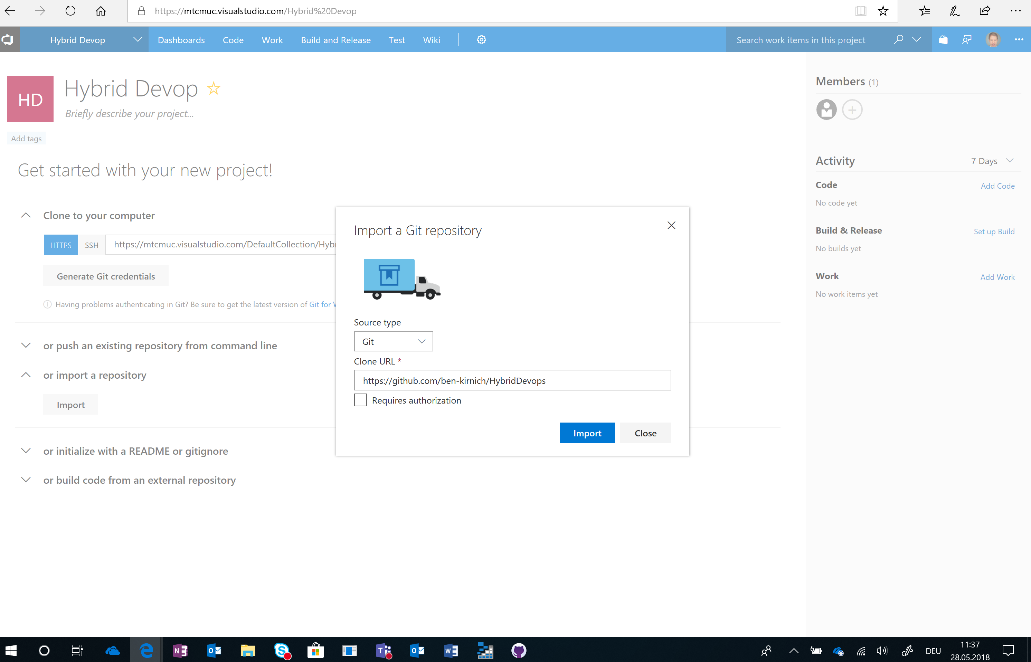
## VSTS Project

Login to your VSTS Account and crate a new Project with the following Settings. You can choose the name as you like:



## Import Code Repository

On the welcome Page of your new Project klick import a repository. Choose Source Type Git and the following repository link: <https://github.com/Microsoft/MTC_Hybrid_Devops_Demo> and klick import.

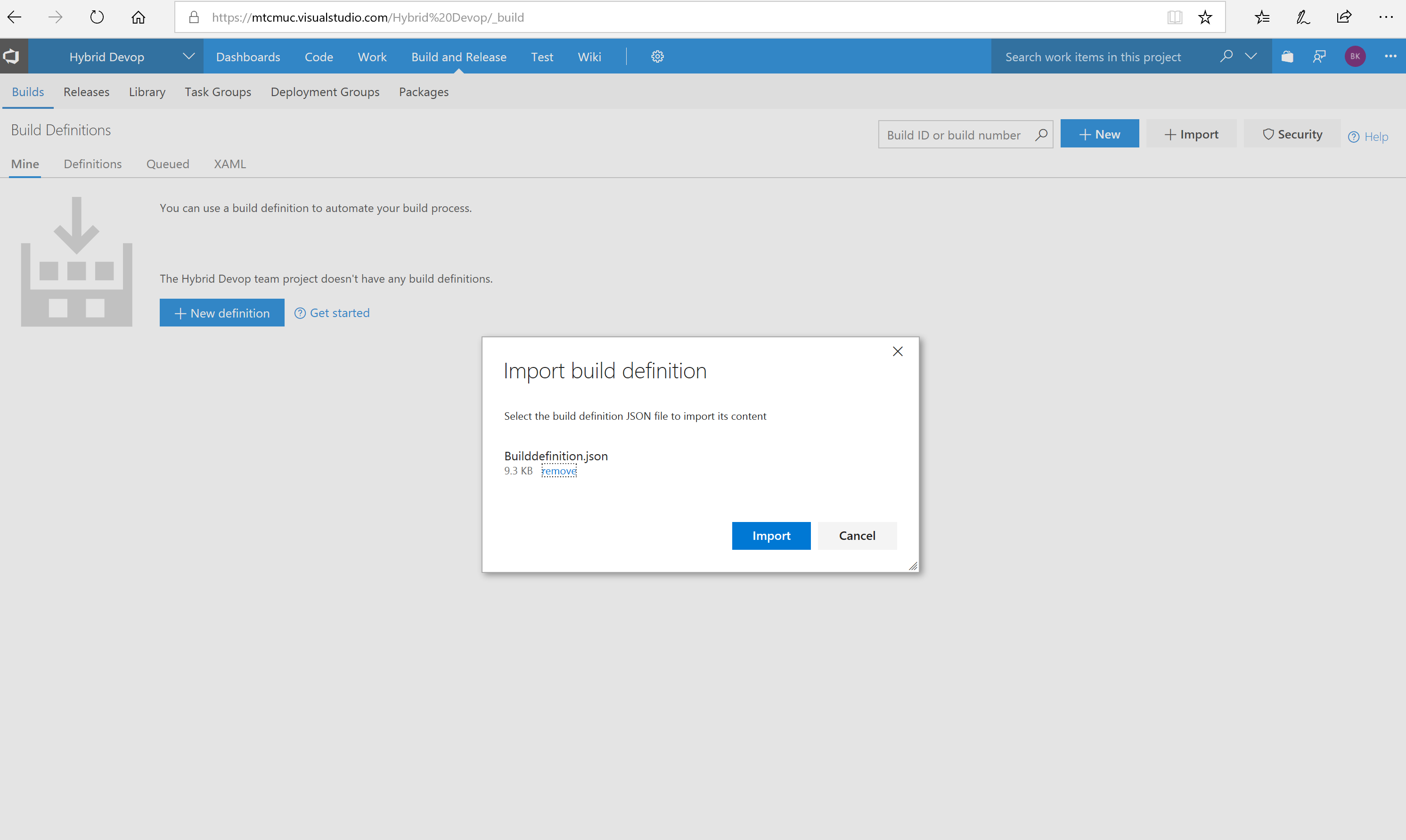


## Create Build Definition

Hoover over Build and Release and klick on Builds in the Top Menue.

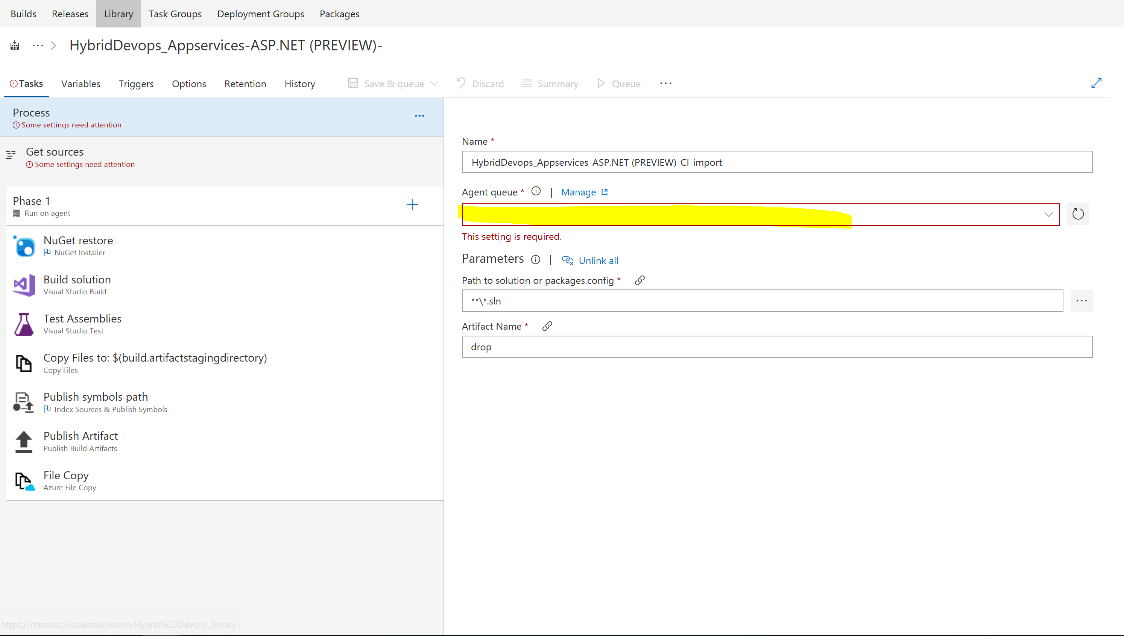
Click on Import

Choose the Builddefiniton.json file from the Template Folder



Click Import

Now Select your Build Agent you have created earlier

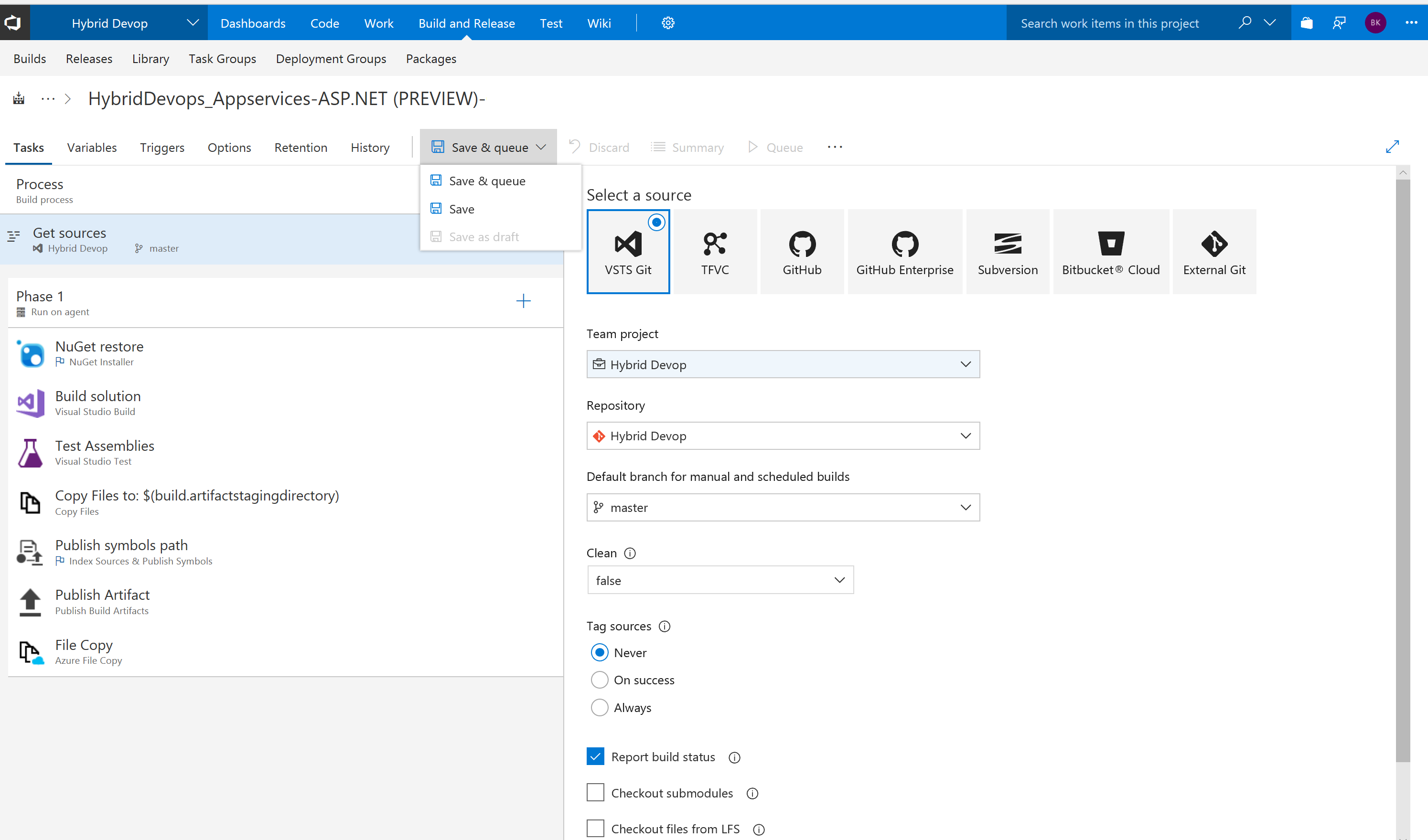


Click on the File Copy Task left and the End of the Task list.

Choose your Storage Account and Container you have created earlier.

Also change the Container Name if yours is named different

Klick on Get Sources and check if the appropriate repository is selected the repository is named like your project.

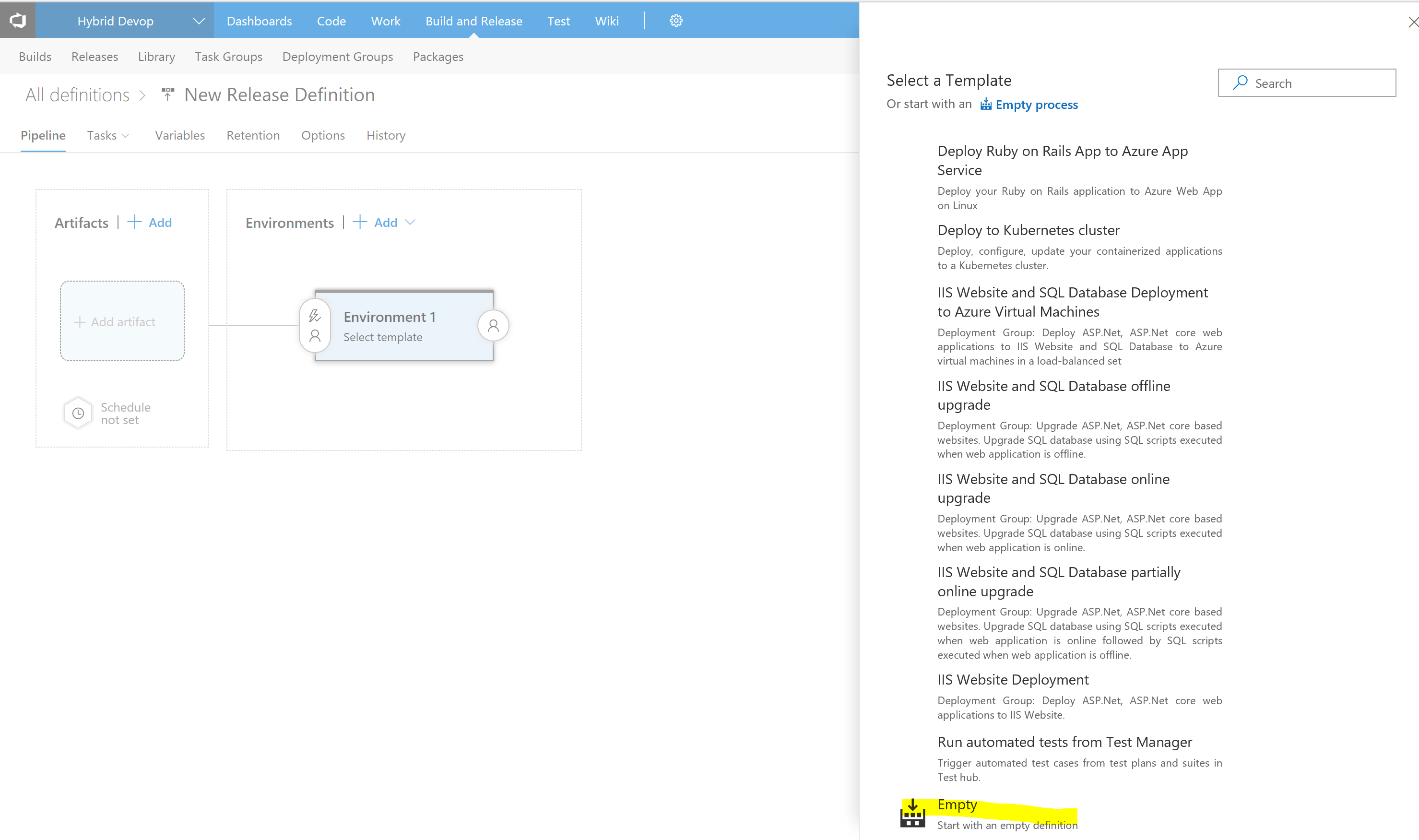


Hoover over Save & queue and just klick save

## Create Release Definition

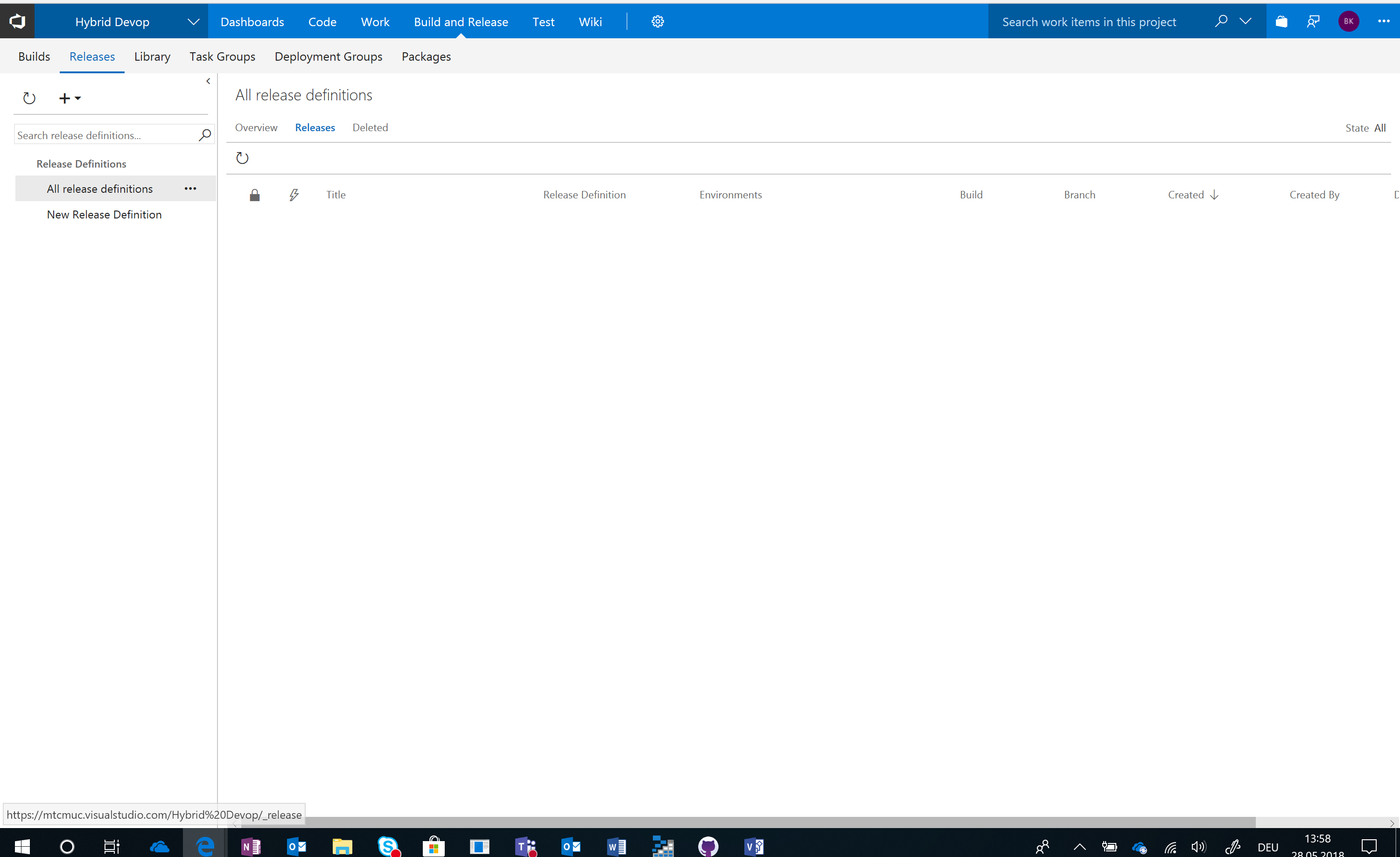
Hover over Buid and Releases in the top menu and click releases

We now have to work around a little VSTS bug because we cant import a definition as long as our Definition List is Empty. Therefore, click on + New Definition



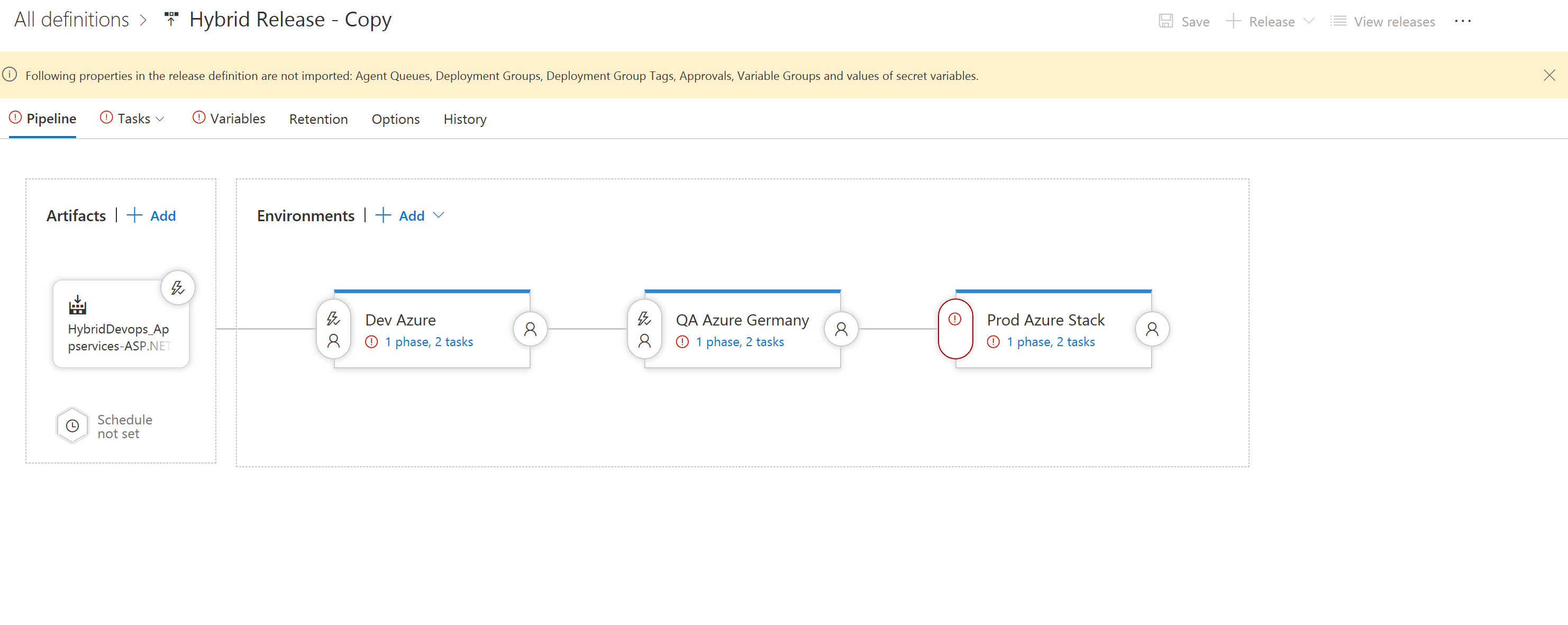
And select Empty as Template and create it.

Then click save, hoover over Build and Release and click Release to go back to the Release Overview

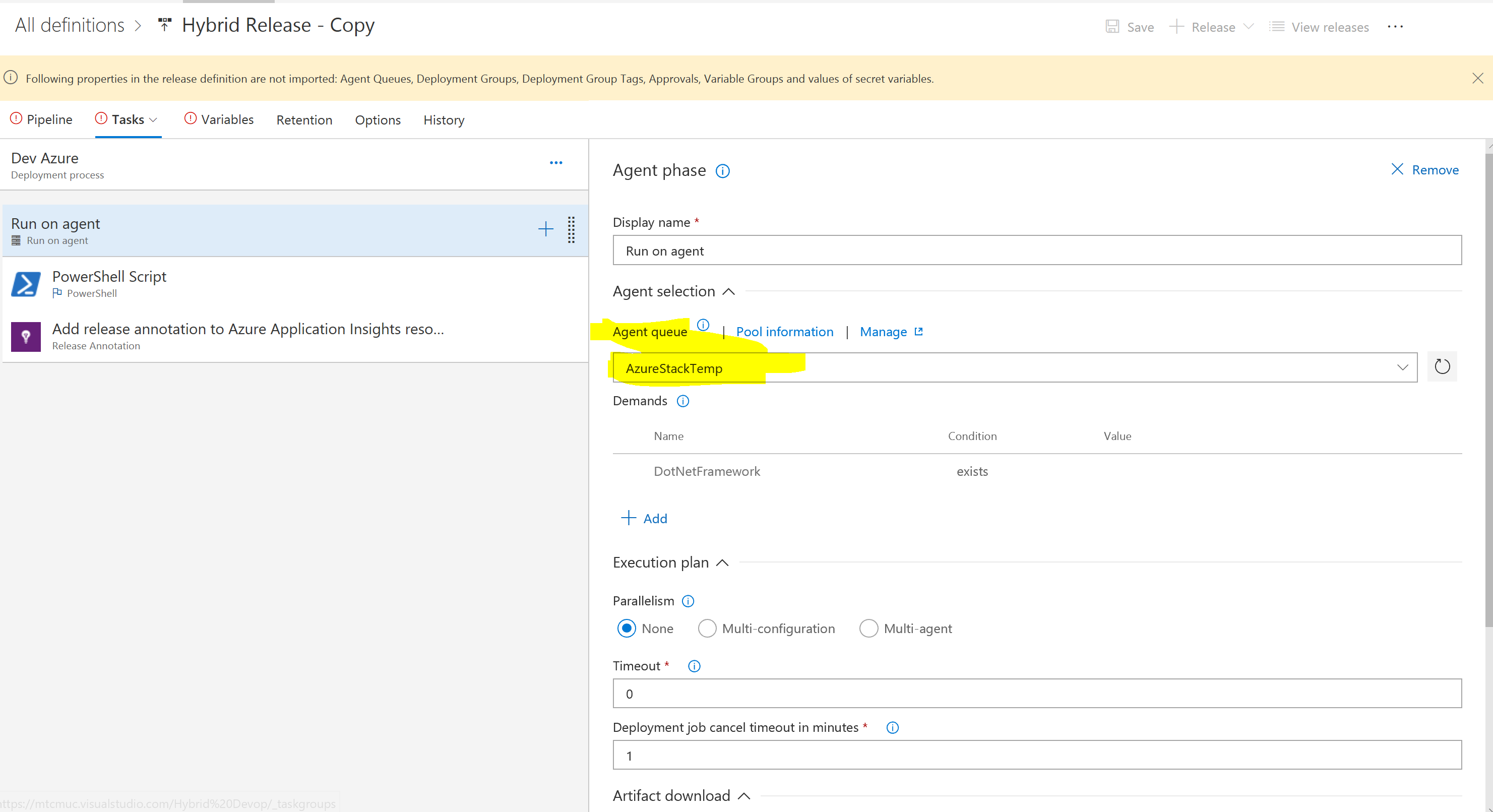


Then Click the + sign and select Import Release Definition. Select the Releasedefinition.json from the template folder and import it. You can later delete the “New Relese Definition” Dummy we have created as workaround to clan up.

Your Release Pipeline should now look like this:



Now Click on the first Environment and choose your earlier created Build Agent as Agent Queue



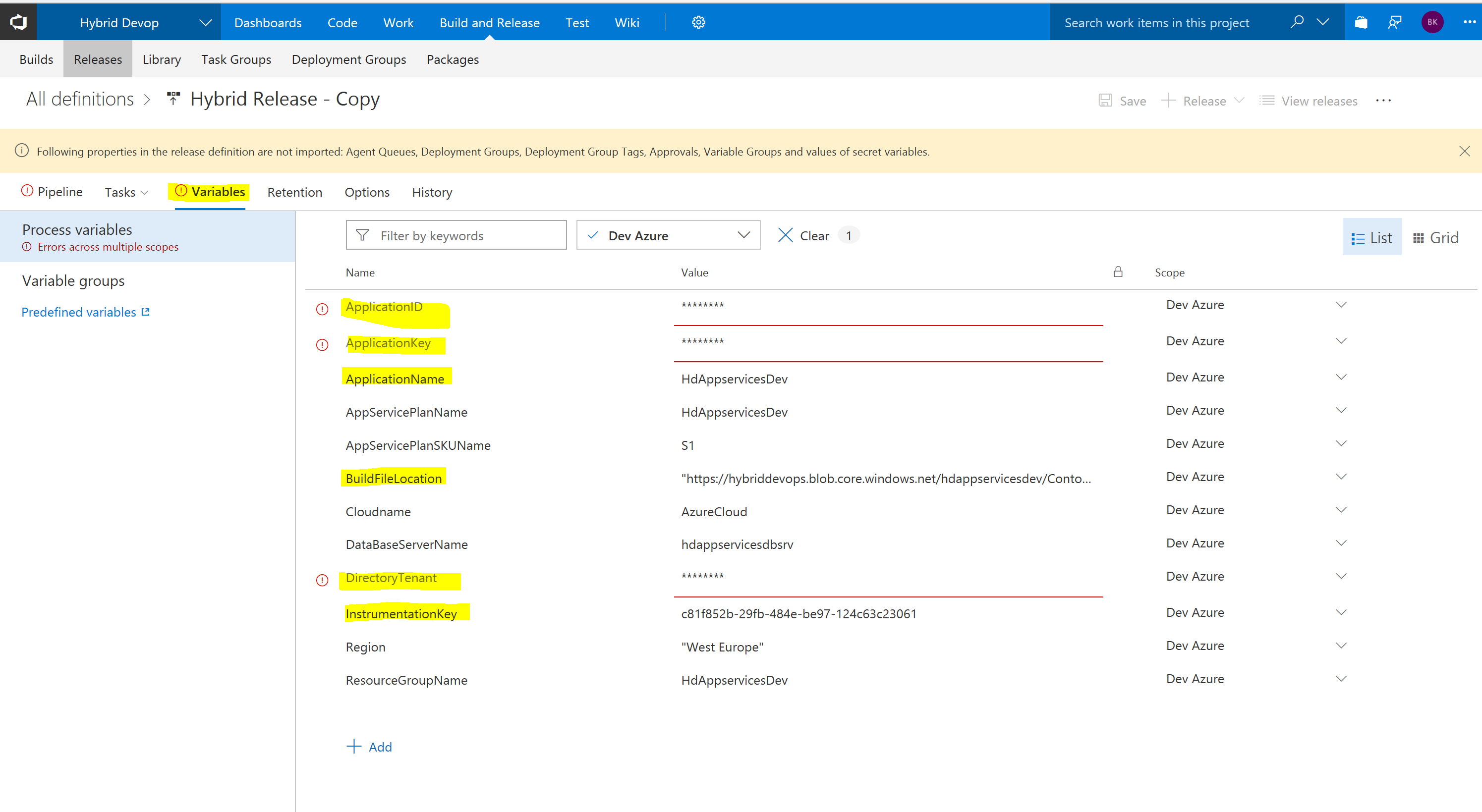
After that we have to adjust the Application Insights Instance we are writing Release Annotations for this Environment Click on the Task Add release Annotation on the left



And set the Application ID and API Key of the Application Insights Instance you have created for this Environment at the beginnen.

Repeat the adjustment of the Agent Queue and the Applications Insights Settings for the remaining two Environments. You can choose the Environments vie Tasks in the Menue.

After you have adjusted all Environments klick on Variables



In this Dialog you must adjust the Variables for all Environments, you can select Every Environment vie the Scope drop down Filter. I assume you don’t want to use AzurGermany as one Environment. If you would like to change that to public Azure or any other Azure Cloud you have to change the Cloudname Variable to Azure Cloud for public Azure for instance or any other Azure Instance you have access to.

To deploy to any Azure Environment we need access to talk to azure, and we use an Service Principle for that. Here is how you do that: <https://docs.microsoft.com/de-de/azure/azure-resource-manager/resource-group-create-service-principal-portal>

You have to create an Service Principle in every AAD used to manage a subscription you want to deploy to at least in Azure and Azure Stack and maybe in a third environment if you use a special cloud like AzureGermany.

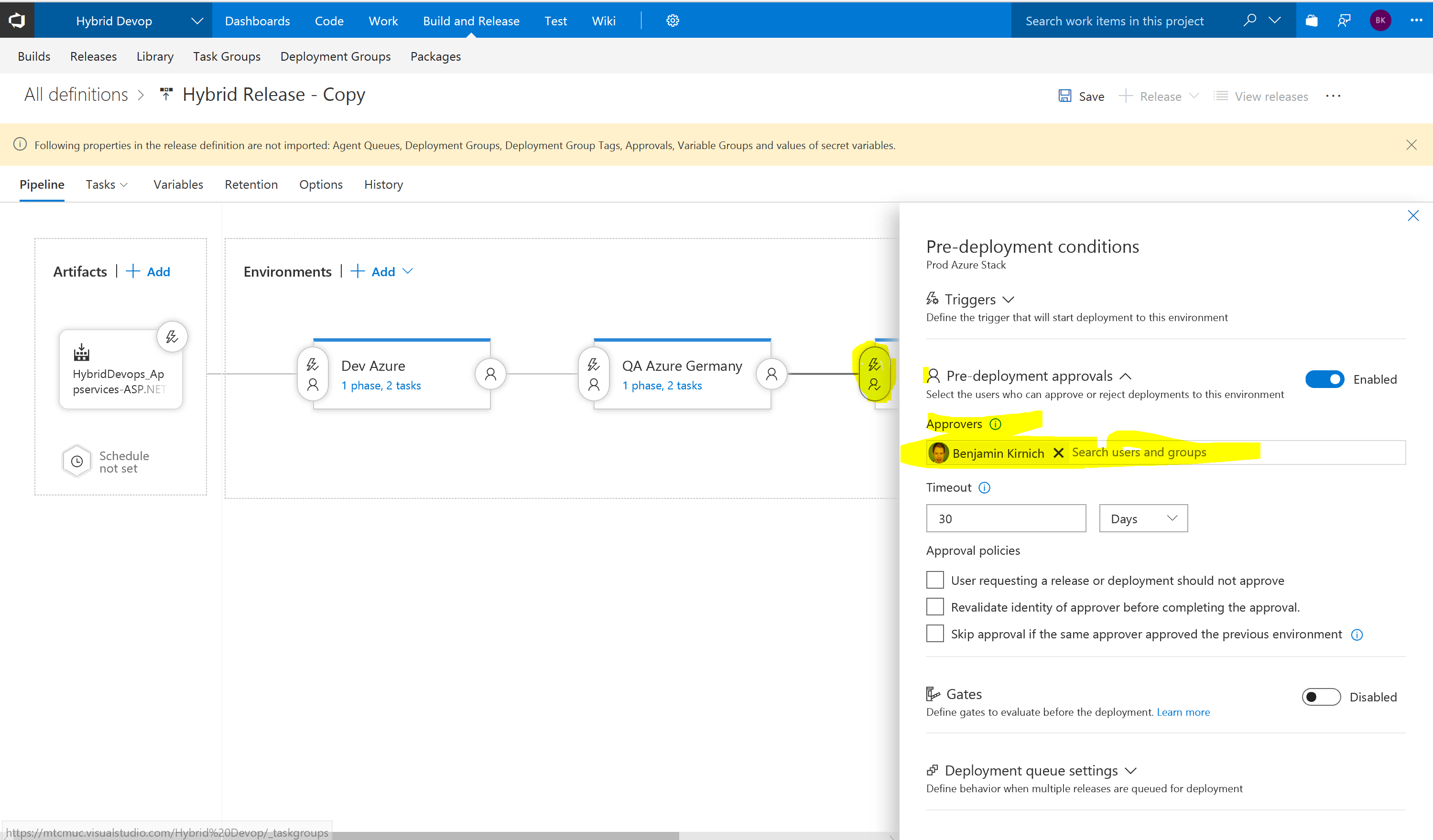
**If you have created the Service Principle don’t forget to assign the SP at least contributor permission within the subscription you want to deploy to. Otherwise the Execution will fail because no active subscription will be found.**

You have to Adjust the following Variables for each Environment:

|  |  |
| --- | --- |
| **Variable** | **Value** |
| ApplicationID | Service Principle ID |
| ApplicationKey | Service Principle Key |
| Directory Tenant | Azure AD ID the Service Principle belongs to |
| Buildfile Location | URL of the Shared Acces Secret you saved earlier |
| Instrumentation Key | Instrumentation Key of the Application Insights Instance the Environment should log to |

After you have adjusted all Environments click on Pipeline in the Menu.

Than Click on the “Predeployment Condition Button” of the last Environment to reenable the approval Step



Assign an approver out of your VSTS Team to the Environment. All Users doing the demo should be able to approve the Deployment.

After this step you should be able to click Save in the top Menue to save the release definition. Please do so

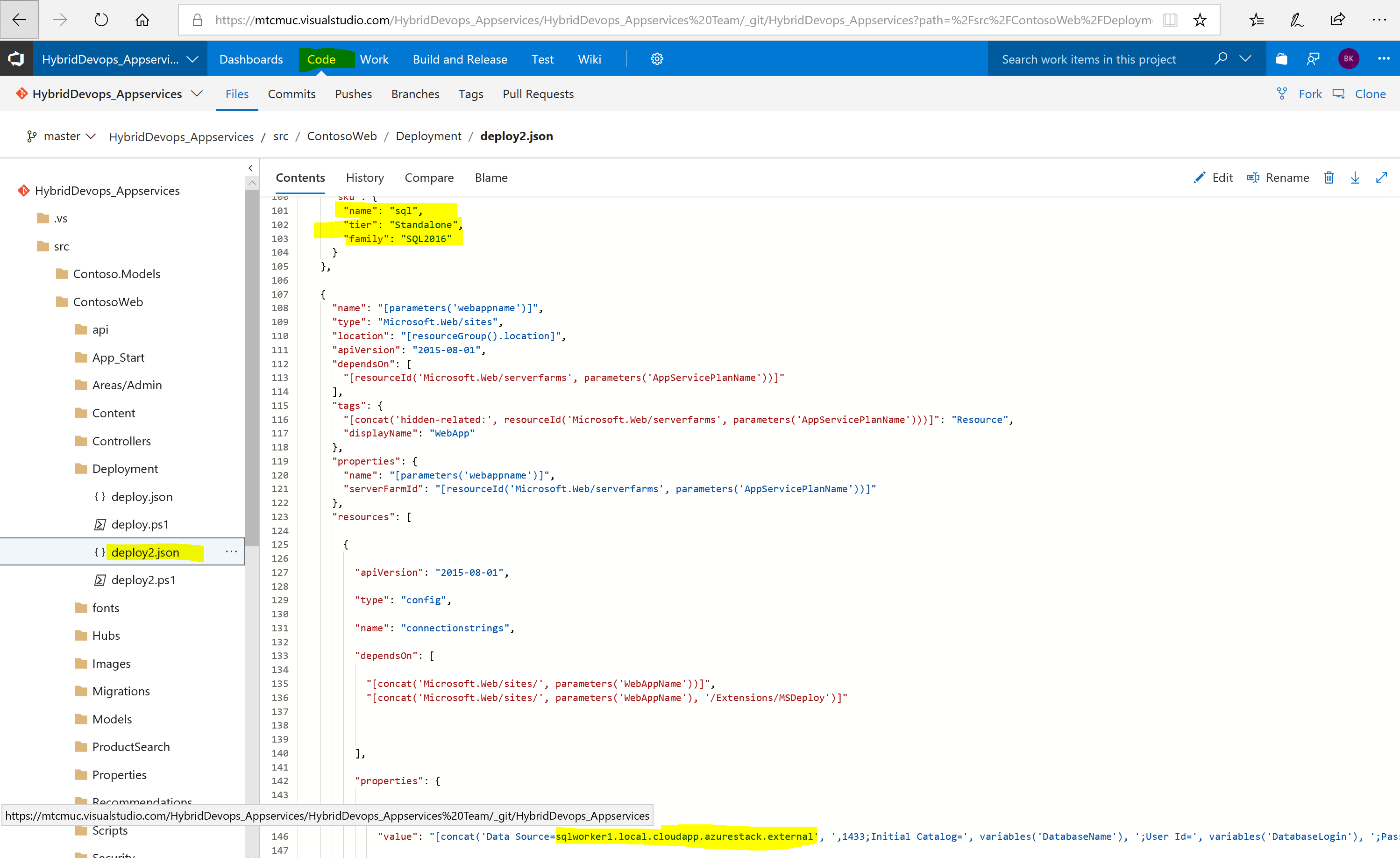
# Testing

The Azure Dev environment should now already work, and if you have adjusted the QA Environment to Azure that as well. The Azurestak Environment could still have issues if your environment uses different naming as described under the Azurestack prerequisites.

|  |  |  |
| --- | --- | --- |
| Variable | Value | Location |
| AppServicePlanSKUName | D1 | Pipeline Variable |
| Region | Local | Pipeline Variable |
| Sqlskutier | Sql | Deploy2.json |
| Sqlskuname | Standalone | Deploy2.json |
| Sqlskufamily | Sql2016 | Deploy2.json |
| Sqlserver in connectionstring | sqlworker1.local.cloudapp.azurestack.external | Deploy2.json |

Adjust the Variables like the ones before for the Azure Stack Environment. To Adjust the SQL Variables if needed which are special to Azurestack and therefore in the specific Deploy2.json Template for Azure Stack go this way:

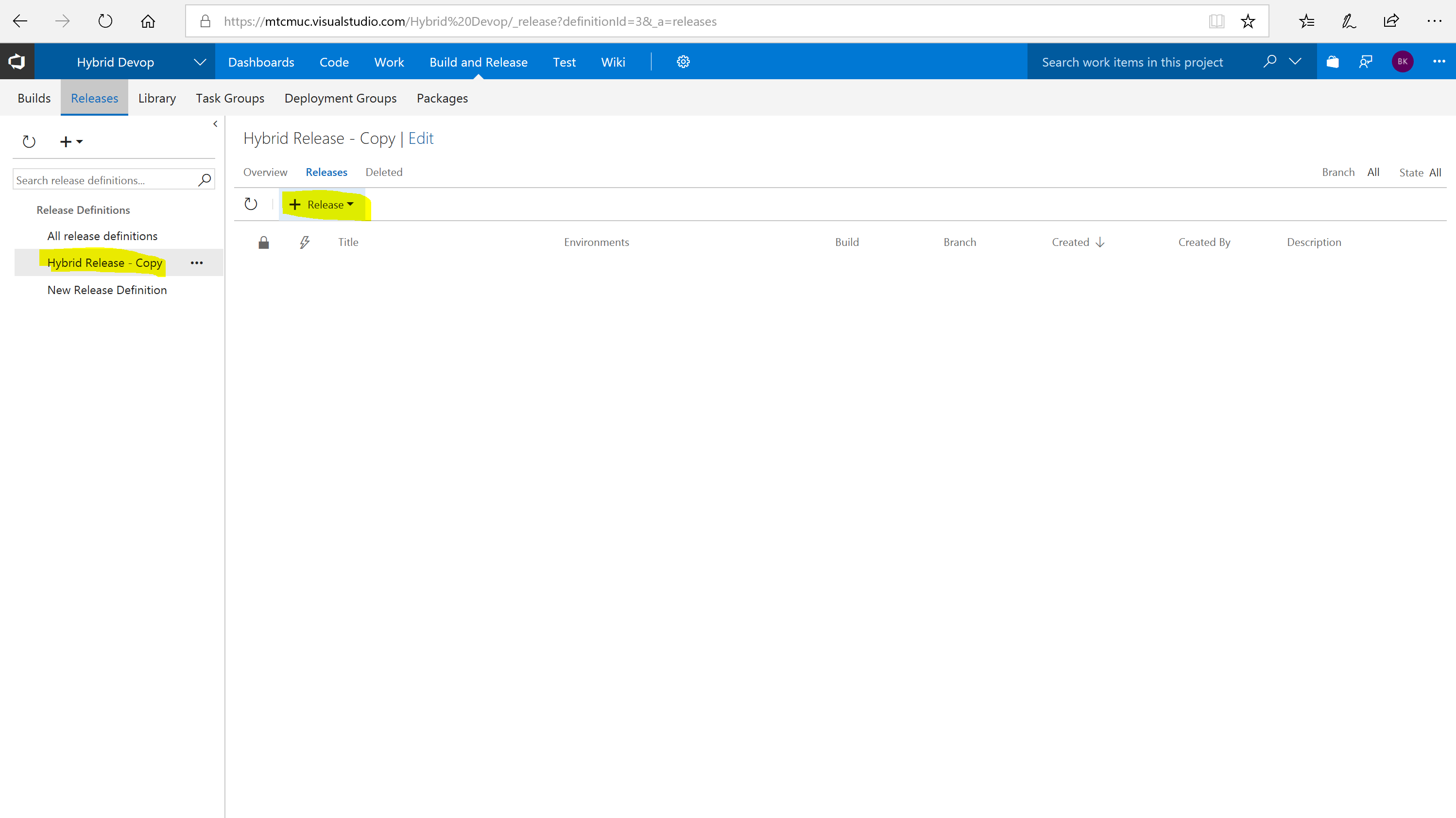
Hover over Code in the topo menue and click on files:



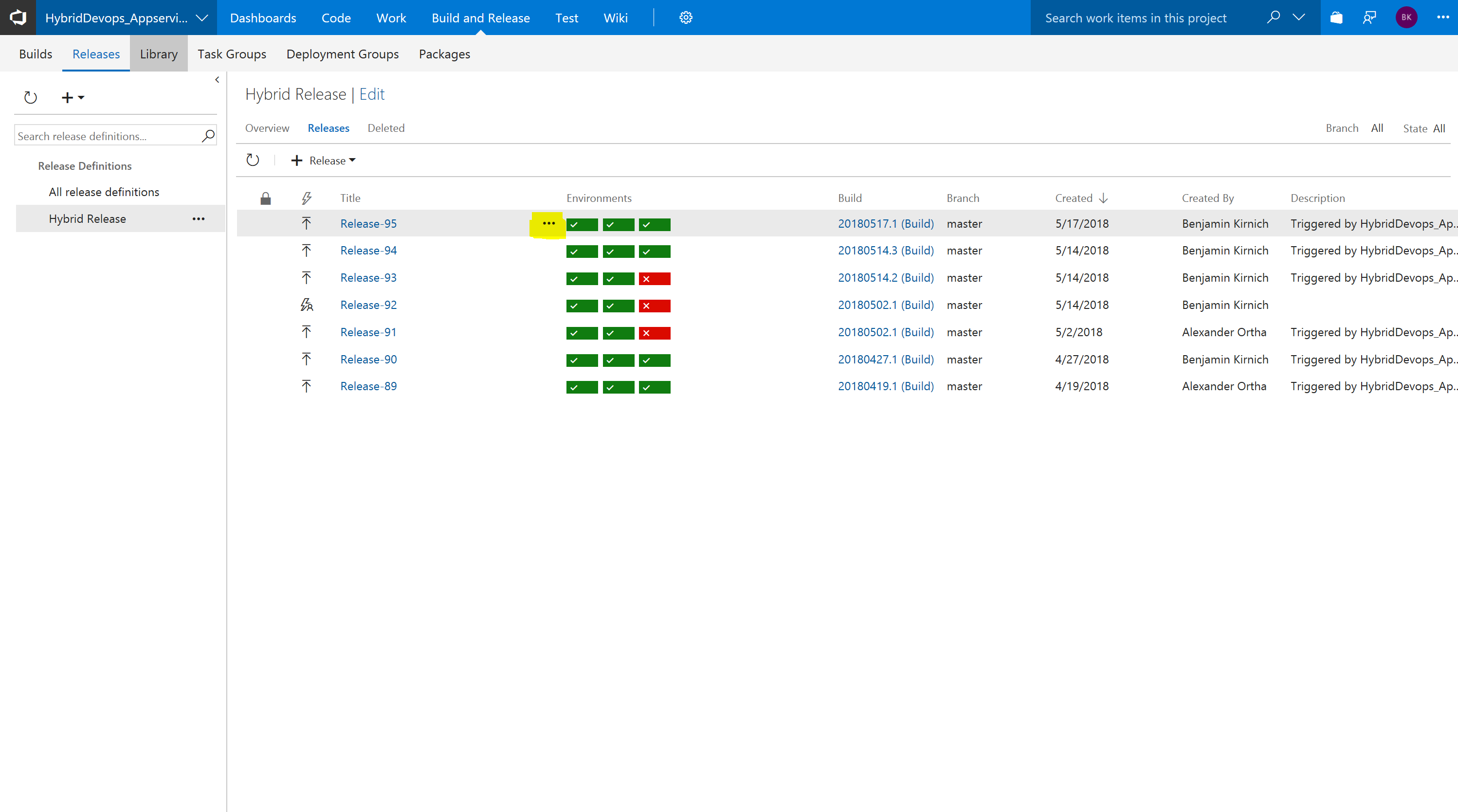
Then browse to the deploy2.json on the left, click edit top right and adjust the template and then save it.

Now you should be ready to test the Deployment. You can do that by manually creating a release from the release definition overview:

Select the release definition and click + Release and New Release.



Then you can open The Release after it was created and is now running, click on the … and then Open



Here you can click on Logs to be able to do some troubleshooting if something goes wrong. Typically, something with the variables, access rights or some settings within the arm templates are wrong in case of errors.

# Preparing the demo

The CICD Pipeline should be up and running and now we have to prepare some additional Stuff for the Demo.

## Web availability tests

Open your Azure portal and the Application Insights instance for your dev environment. Add an availability test to your Instance to monitor the environment.

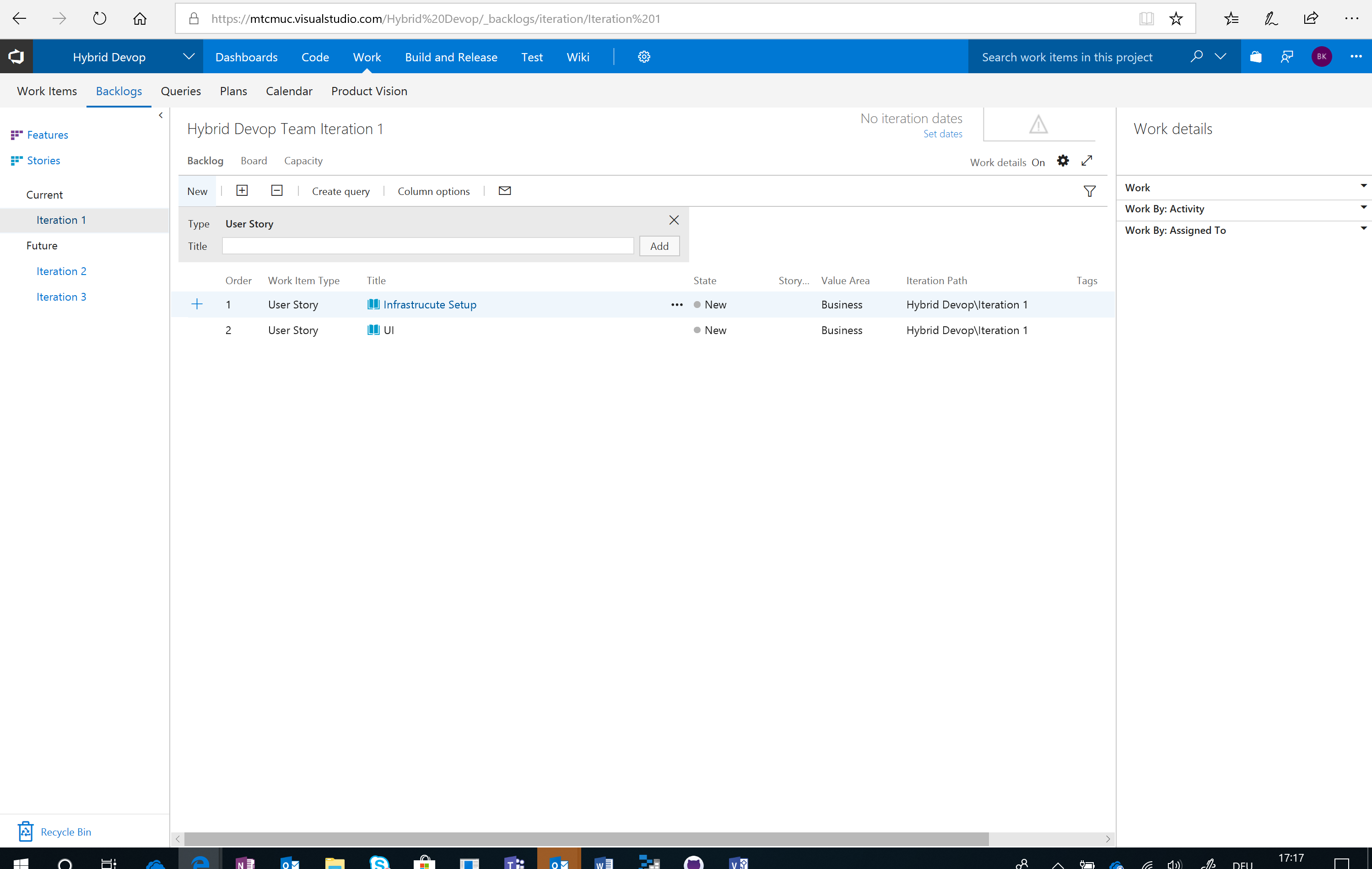


Add a test t09 each of your application insights instance for each environment. Remember your production environment on Azure Stack has to be available from azure for successful testing.

## Backlog and work items

First, we have to populate some tasks to the work management to have a real looking project.

Hoover over work in the top menu and klick on backlogs

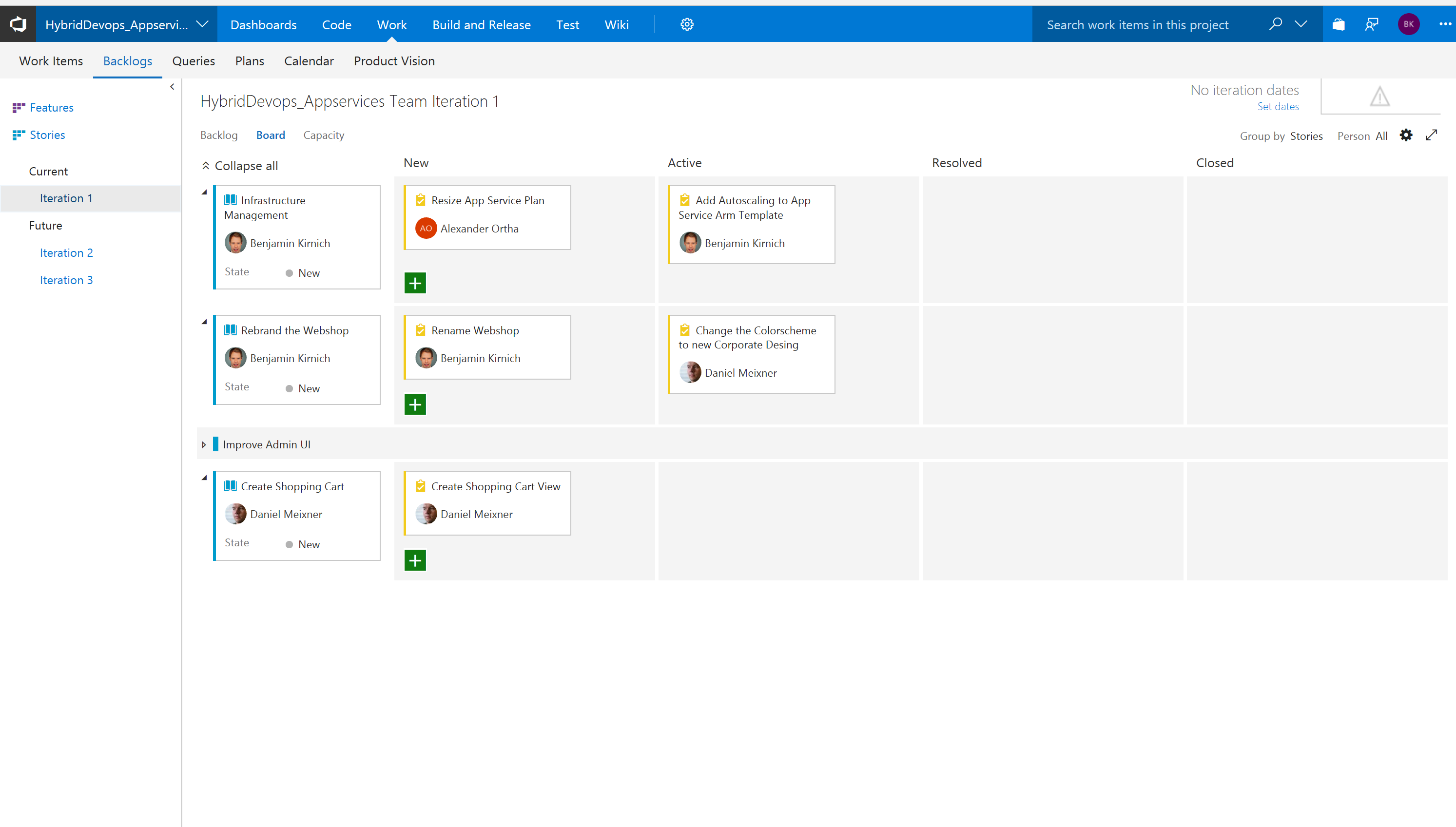


Select Iteration on the left and create some User Stories (Workstreams)

Then klick on Board and populate some tasks to the Kanban board to have a good example project.

You should have on task to Rename the Website which is the task we use in the demo.

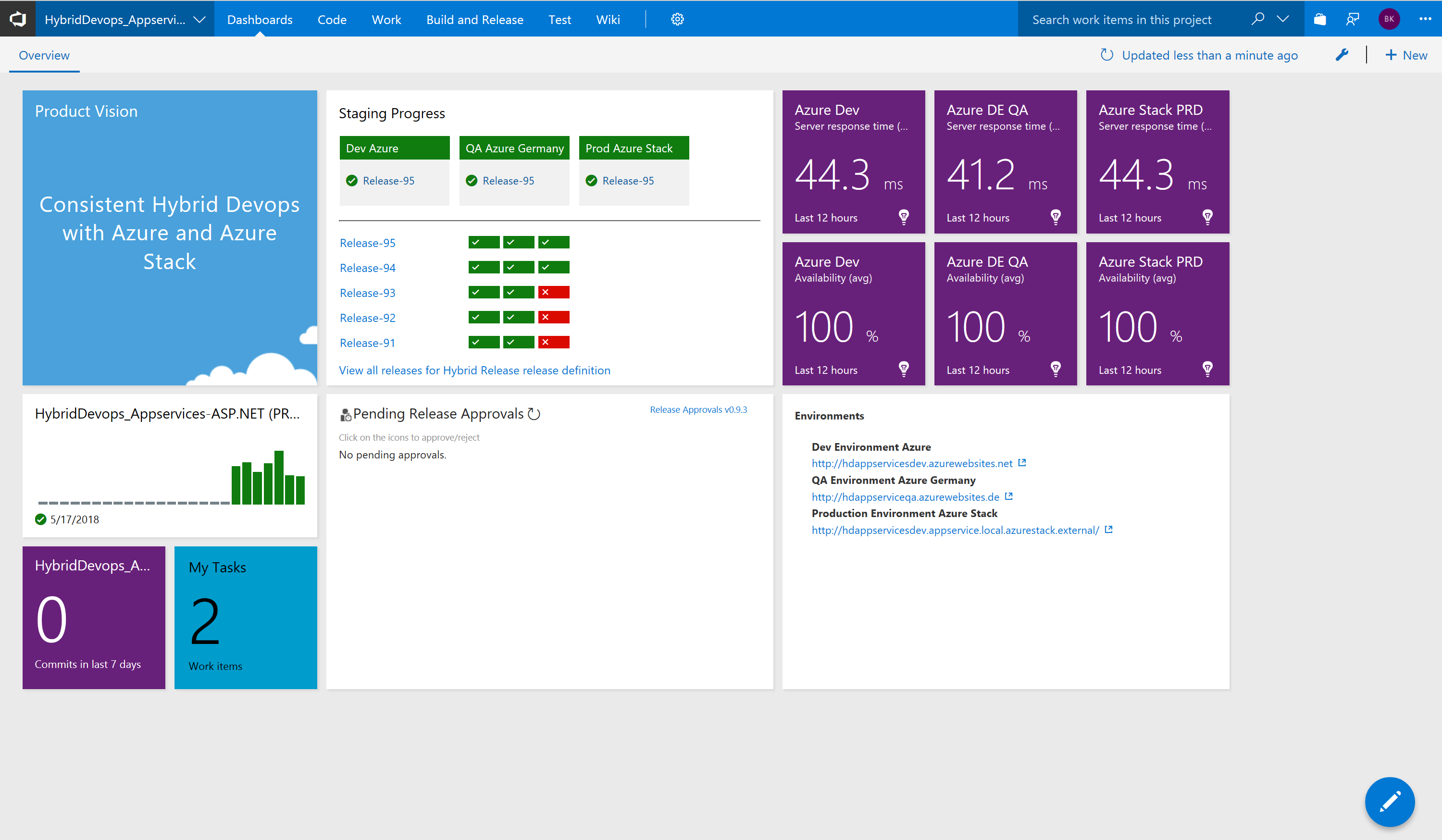
Finally, your board should look somehow like this:



## VSTS Dashboard

The last thing we need is a nice looking Dashboard to present the demo unfortunately there is no way to export a dashboard so you have to create your own one.

Click on dashboards in the top menue. You can customize the dashboard how ever you like it to be four your demo. However, the dashboard should include some things like the staging overview widget, a pending release approval widget and application insights metrics. A markup widget with the links to all environments makes the Demo easier as well. Here is how ours look like:



Just klick on the blue pen in the bottom right corner and adjust your empty dashboard. Some widgets we used are not build in and have first to be installed via the extension gallery.

