WORKBENCH DOT NET APPLICATION

Continuous Integration & Continuous Deployment User Manual

# **Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Serial No** | **Name** | **Page No.** |
| 1 | [Introduction](#_Introduction) | 2 |
| 2 | [CICD design for Dev (SIT) Environment](#_CICD_Design_for) | 4 |
| 3 | [CI pipeline Details for Dev (SIT) Environment](#_CI_pipeline_for) | 5 |
| 4 | [CD pipeline Details for Dev (SIT) Environment](#_CD_Pipeline_Details) | 6 |
| 5 | [CICD design for QA(UAT)/Training/Prod Environment](#_CICD_design_for_1) | 7 |
| 6 | [CI pipeline Details for QA(UAT)/Training/Prod Environment](#_CI_pipeline_for_1) | 8 |
| 7 | [CD pipeline Details for QA(UAT)/Training/Prod Environment](#_CD_pipeline_Details_1) | 9 |
| 8 | [CICD design for Demo Environment](#_CICD_for_Demo) | 10 |
| 9 | [CI pipeline Details for Demo Environment](#_CI_pipeline_for_2) | 11 |
| 10 | [CD pipeline Details for Demo Environment](#_CD_Pipeline_details_2) | 12 |
| 11 | [CI pipeline Details for Config](#_CI_pipeline_for_3) | 13 |
| 12 | [Procedure to trigger CI pipeline for Dev/QAT environments](#_Procedure_to_trigger) | 14 |
| 13 | Procedure to trigger CI pipeline for Demo environment | 15 |
| 14 | [CI pipeline logs](#_CI_pipeline_logs) | 17 |
| 15 | [Artifactory and Artifacts](#_Artifactory_and_Artifacts) | 18 |
| 16 | [Procedure to trigger CD pipeline for Dev/SIT environment](#_Procedure_to_trigger_2) | 21 |
| 17 | [Procedure to trigger CD pipeline for QA(UAT), Prod & Training environments](#_Procedure_to_trigger_3) | 23 |
| 18 | [Procedure to trigger CD pipeline for Demo environments](#_Procedure_to_trigger_4) | 25 |
| 19 | [Approvals](#_Approvals) | 27 |
| 20 | [Roll Back](#_Roll_back) | 29 |
| 21 | [CD pipeline logs](#_Deployment_Logs) | 30 |
| 22 | [Troubleshooting](#_Troubleshooting_1) | 33 |

# 

# **Introduction**

There are two types of pipelines created.

1. Build
2. Release

Build pipeline does below tasks:

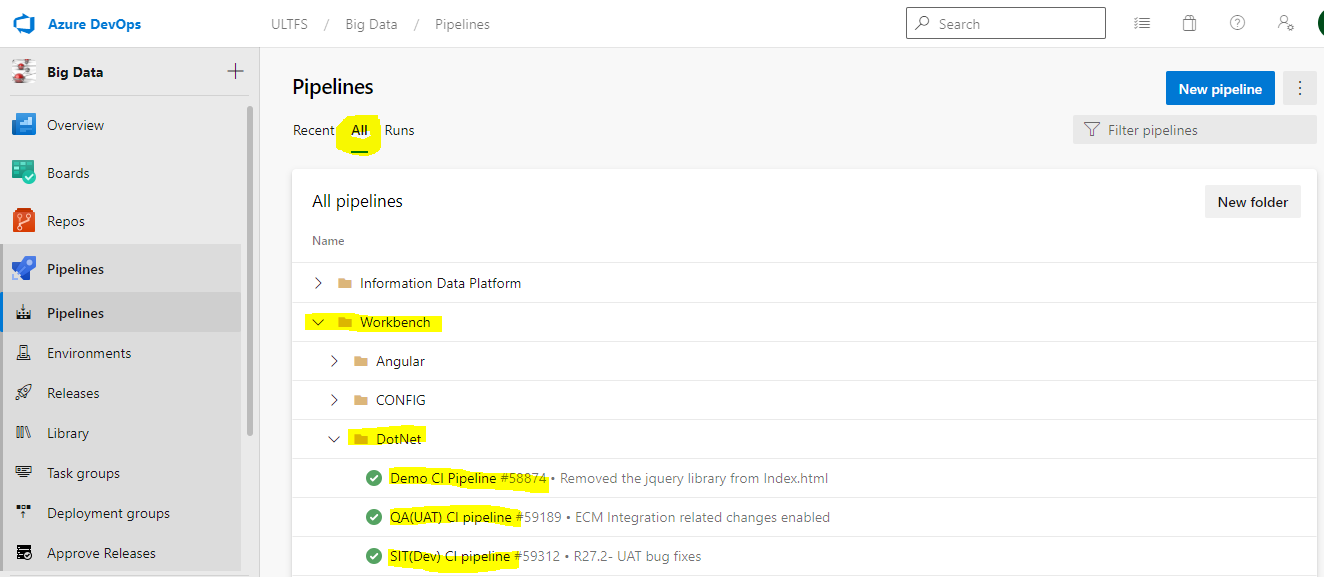
1. Fetch the latest code from source code repository
2. Build the code
3. Push the artifacts to Azure DevOps Artifactory

Release pipeline does the below tasks:

1. Connects to target server
2. Download package from Azure DevOps Artifactory
3. Deploy package to the target server

There are 3 build pipelines created for Dev, QAT and Demo environments and 3 release pipelines created for Dev, UAT, Prod and Demo environments.

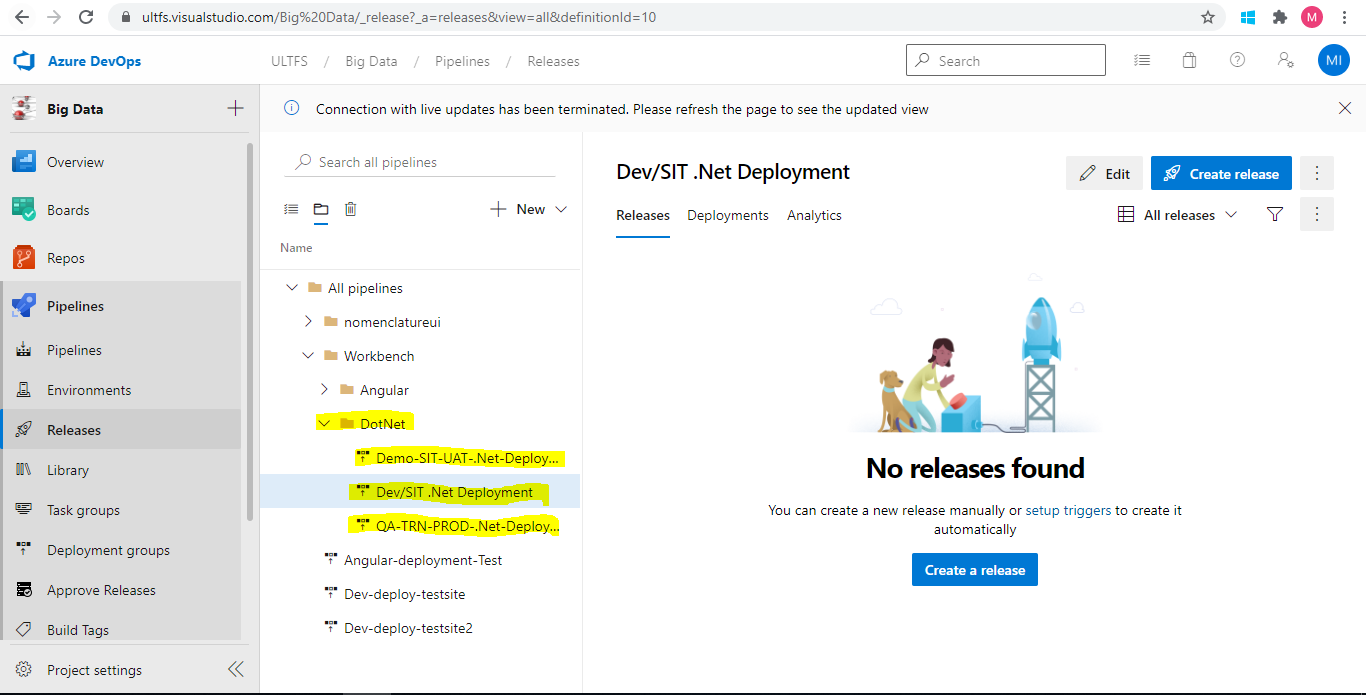
Here is the folder structure for Build pipelines:



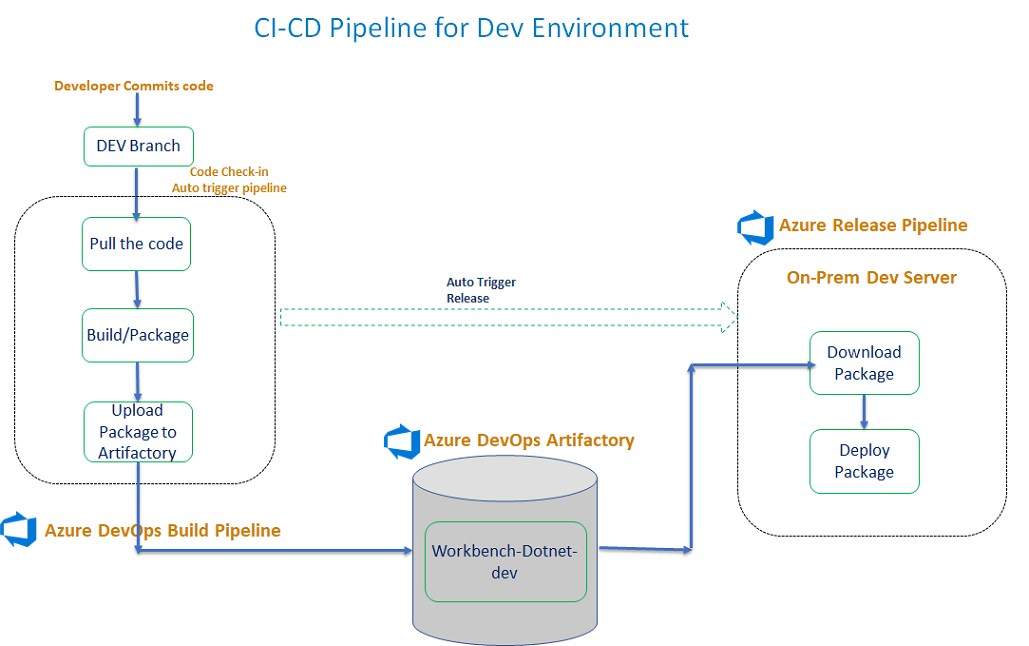
Below is list of Release Pipelines created and available in Azure DevOps for the deployment to various Environments.

1. **Dev/SIT .Net Deployment** – For Dev Env deployment
2. **Demo-SIT-UAT-.Net-Deployment** – For all Demo Env deployments
3. **QA-TRN-PROD-.Net-Deployment** – For QA, TRN and PROD deployments

Here is the folder structure for the release pipelines:



# **CICD Design for Dev(SIT) Environment**



# 

# **CI pipeline Details for Dev(SIT) Environment**

This pipeline builds the code from DEV branch and publishes the artifacts to Artifactory, and the details of the pipeline are listed in the below table:

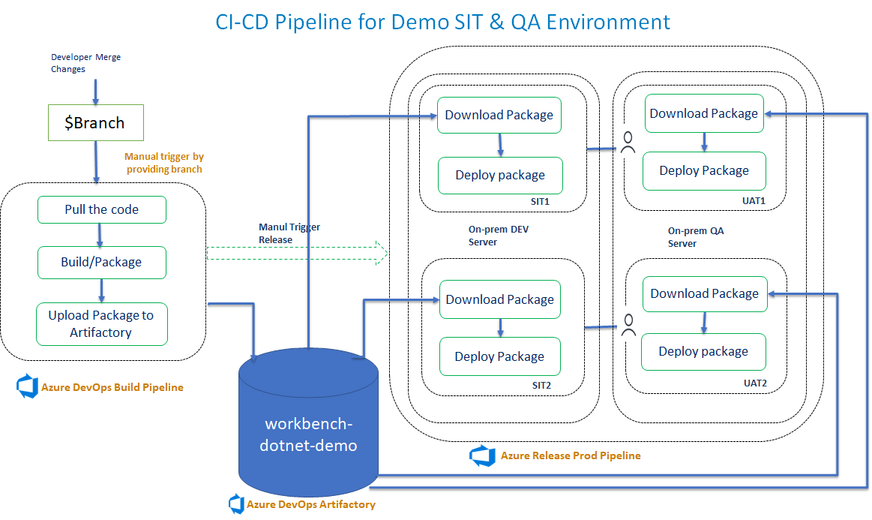
|  |  |  |
| --- | --- | --- |
| **Name** | **Value** | |
| **Pipeline Name** | SIT(Dev) CI pipeline | |
| **Pipeline URL** | <https://dev.azure.com/ULTFS/Big%20Data/_build?definitionId=701> | |
| **Source Code Repository URL** | <https://dev.azure.com/ULTFS/Big%20Data/_versionControl?path=%24%2FBig%20Data%2FUI%2FEngineering%20Assessment%20Process%202%2FDEV> | |
| **Artifactory URL** | <https://dev.azure.com/ULTFS/Big%20Data/_packaging?_a=feed&feed=WorkBench-DotNet> | |
| **Feed Name** | Workbench-dotnet | |
| **Artifact Name** | workbench-dotnet-dev | |
| **Trigger** | Automatic | |
| **Variables in pipeline** | dll\_folder\_Name | bin (Can be edited in pipeline) |
| Package\_Name | workbench-dotnet-dev (Can be edited in pipeline) |
| Solution\_Name | UL.DataCentricity.Eap.Service.Rest (Can be edited in pipeline) |

# **CD Pipeline Details for Dev/SIT environment**

This pipeline downloads the artifacts and config files from Azure DevOps Artifactory and deploys the artifacts on the **Dev** server.

|  |  |  |
| --- | --- | --- |
| **Name** | **Value** | |
| **Pipeline Name** | Dev/SIT .Net Deployment | |
| **Pipeline URL** | <https://ultfs.visualstudio.com/Big%20Data/_release?_a=releases&view=all&definitionId=10> | |
| **Artifactory URL** | <https://dev.azure.com/ULTFS/Big%20Data/_packaging?_a=feed&feed=WorkBench-DotNet> | |
| **Feed Name** | Workbench-dotnet | |
| **Artifact Name** | workbench-dotnet-dev | |
| **Trigger** | Automatic | |
| **Variables in pipeline** | Artifact\_Name | bin |
| Config\_Deploy\_path | D:\eap\api\ |
| Code\_Deploy\_path | D:\eap\api\bin\ |
| IIS-site | eap |
| Code\_Artifactory\_Package | workbench-dotnet-dev |
| Config\_Artifactory\_Package | dev |
| Staging-path | D:\devops-staging\dotnet |
| Code\_Staging\_folder | workbench-dotnet-dev |
| Config\_Staging\_folder | .Net Config |

# **CICD design for QA(UAT)/Training/Prod Environment**



# 

# **CI pipeline Details for QA(UAT)/Training/Prod Environment**

This pipeline builds the code from **QA** branch and publishes the artifacts to Artifactory, and the details of the pipeline are listed in the below table:

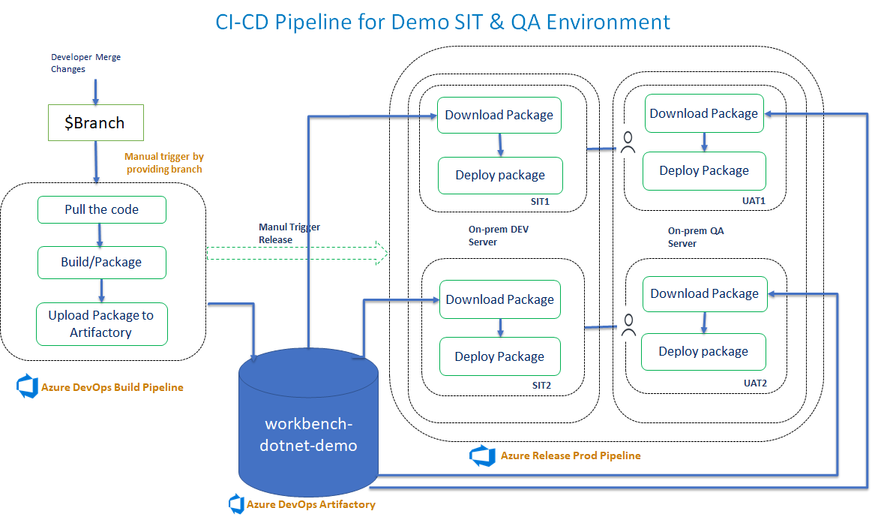
|  |  |  |
| --- | --- | --- |
| **Name** | **Value** | |
| **Pipeline Name** | QA(UAT) CI pipeline | |
| **Pipeline URL** | <https://dev.azure.com/ULTFS/Big%20Data/_build?definitionId=714> | |
| **Source Code Repository URL** | <https://dev.azure.com/ULTFS/Big%20Data/_versionControl?path=%24%2FBig%20Data%2FUI%2FEngineering%20Assessment%20Process%202%2FQA> | |
| **Artifactory URL** | <https://dev.azure.com/ULTFS/Big%20Data/_packaging?_a=feed&feed=WorkBench-DotNet> | |
| **Feed Name** | Workbench-dotnet | |
| **Artifact Name** | workbench-dotnet-release | |
| **Trigger** | Automatic | |
| **Variables in pipeline** | dll\_folder\_Name | bin (Can be edited in pipeline) |
| Package\_Name | workbench-dotnet-release (Can be edited in pipeline) |
| Solution\_Name | UL.DataCentricity.Eap.Service.Rest (Can be edited in pipeline) |

# **CD pipeline Details for QA/Training/Prod Environment**

This pipeline downloads the artifacts and config files from Azure DevOps Artifactory and deploys the artifacts on the **QA**, **Training and Prod** server.

|  |  |  |
| --- | --- | --- |
| **Name** | **Value** | |
| **Pipeline Name** | QA-TRN-PROD-.Net-Deployment | |
| **Pipeline URL** | <https://ultfs.visualstudio.com/Big%20Data/_release?view=all&_a=releases&definitionId=9> | |
| **Artifactory URL** | <https://dev.azure.com/ULTFS/Big%20Data/_packaging?_a=feed&feed=WorkBench-DotNet> | |
| **Feed Name** | Workbench-dotnet | |
| **Artifact Name** | workbench-dotnet-dev | |
| **Trigger** | Automatic | |
| **Variables in pipeline** | Artifact\_Name | bin |
| Config\_Deploy\_path | D:\eap\api\ |
| Code\_Deploy\_path | D:\eap\api\bin\ |
| IIS-site | eap, eap-trn.ul.com |
| Code\_Artifactory\_Package | workbench-dotnet-dev |
| Config\_Artifactory\_Package | qa, trn, prod |
| Staging-path | D:\devops-staging\dotnet |
| Code\_Staging\_folder | workbench-dotnet-dev |
| Config\_Staging\_folder | .Net Config |

# **CICD design for Demo Environment**



# 

# **CI pipeline Details for Demo Environment**

This pipeline builds the code from developer given **feature branch** and publishes the artifacts to Artifactory, and the details of the pipeline are listed in the below table:

|  |  |  |
| --- | --- | --- |
| **Name** | **Value** | |
| **Pipeline Name** | Demo CI pipeline | |
| **Pipeline URL** | <https://dev.azure.com/ULTFS/Big%20Data/_build?definitionId=722> | |
| **Source Code Repository URL** | <https://dev.azure.com/ULTFS/Big%20Data/_versionControl?path=%24%2FBig%20Data%2FUI%2FEngineering%20Assessment%20Process> | |
| **Branch Name** | Given during running the pipeline | |
| **Artifactory URL** | <https://dev.azure.com/ULTFS/Big%20Data/_packaging?_a=feed&feed=WorkBench-DotNet> | |
| **Feed Name** | Workbench-dotnet | |
| **Artifact Name** | workbench-dotnet-demo | |
| **Trigger** | Manual with branch name | |
| **Variables in pipeline** | dll\_folder\_Name | bin (Can be edited in pipeline) |
| Package\_Name | workbench-dotnet-release (Can be edited in pipeline) |
| Solution\_Name | UL.DataCentricity.Eap.Service.Rest(Can be edited in pipeline) |
| Branch\_Name | $(Branch\_Name) (Mandatory to give while executing pipeline) |

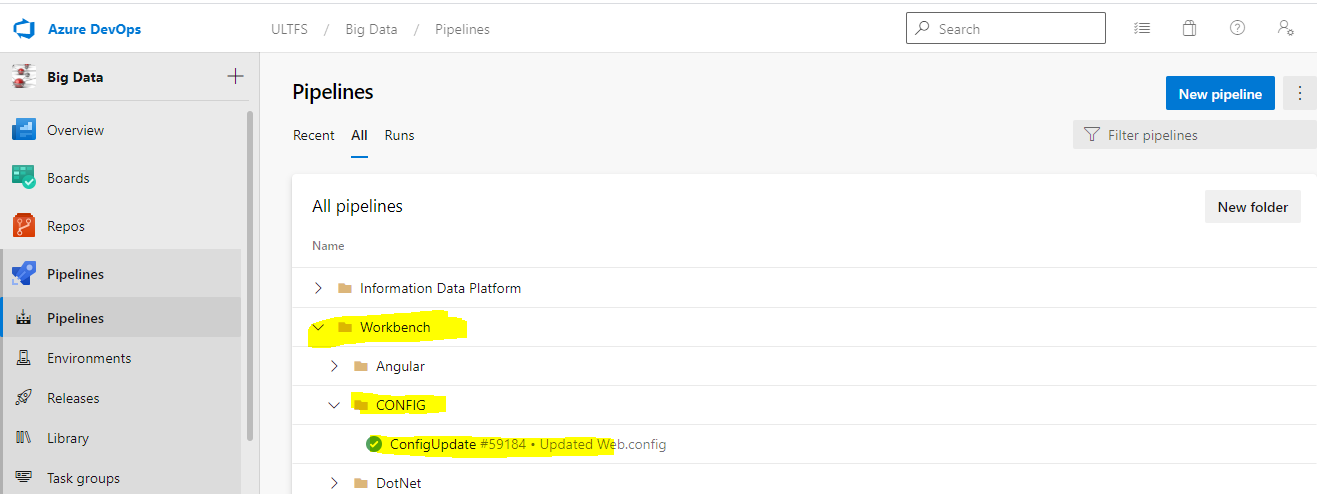
# **CD Pipeline details for Demo Environment**

This pipeline downloads the artifacts and config files from Azure DevOps Artifactory and deploys the artifacts on the **Demo SIT & UAT** server.

|  |  |  |
| --- | --- | --- |
| **Name** | **Value** | |
| **Pipeline Name** | Demo-SIT-UAT-.Net-Deployment | |
| **Pipeline URL** | <https://ultfs.visualstudio.com/Big%20Data/_release?view=all&_a=releases&definitionId=8> | |
| **Artifactory URL** | <https://dev.azure.com/ULTFS/Big%20Data/_packaging?_a=feed&feed=WorkBench-DotNet> | |
| **Feed Name** | Workbench-dotnet | |
| **Artifact Name** | workbench-dotnet-dev | |
| **Trigger** | Automatic | |
| **Variables in pipeline** | Artifact\_Name | bin |
| Config\_Deploy\_path | D:\eap\_7777\api\, D:\eap7777\api\, D:\eap\_8888\api\, D:\eap8888\api\ |
| Code\_Deploy\_path | D:\eap\_7777\api\bin\, D:\eap7777\api\bin\, D:\eap\_8888\api\bin\,  D:\eap8888\api\bin\ |
| IIS-site | eap\_7777, eap7777, eap\_8888, eap8888 |
| Code\_Artifactory\_Package | workbench-dotnet-demo |
| Config\_Artifactory\_Package | demo\_sit, demo\_qa |
| Staging-path | D:\devops-staging\dotnet |
| Code\_Staging\_folder | workbench-dotnet-demo |
| Config\_Staging\_folder | .Net Config |

# **CI pipeline folder structure & Details for Config files**

There is a build pipeline created for config files and the folder structure is as below:



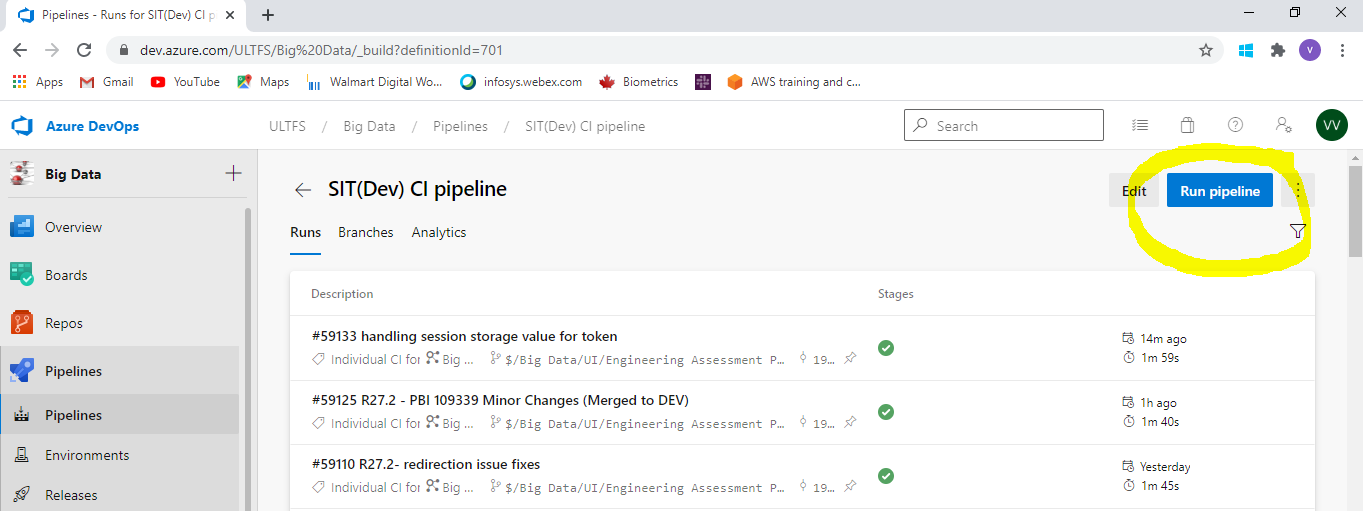
|  |  |
| --- | --- |
| **Name** | **Value** |
| **Pipeline Name** | ConfigUpdate |
| **Pipeline URL** | <https://dev.azure.com/ULTFS/Big%20Data/_build?definitionId=721> |
| **Source Code Repository URL** | <https://dev.azure.com/ULTFS/Big%20Data/_versionControl?path=%24%2FBig%20Data%2FUI%2FEngineering%20Assessment%20Process> |
| **Feed Name** | WorkBench-Config |
| **Artifact Name (Artifact created for each environment)** | Dev |
| Qa |
| Prod |
| Trn |
| demo\_sit |
| demo\_qa |
| **Trigger** | Automatic, will get triggered when config changes are done in Source Code Repository |

# **Procedure to trigger CI pipeline for DEV (SIT)/QA (UAT) environments**

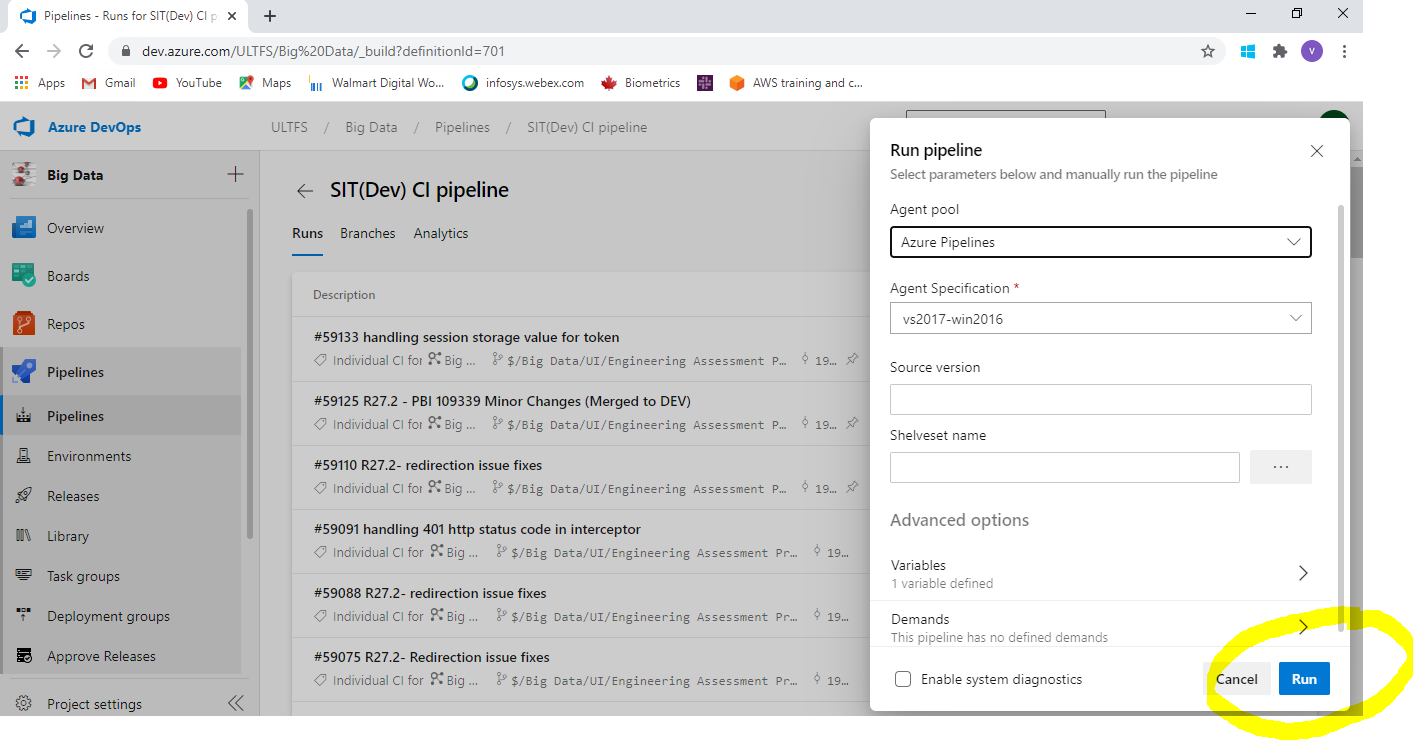
**Automatic** trigger is enabled for **DEV/SIT and QA/UAT** environments, i.e., whenever a developer commits code to corresponding branch (DEV in case of DEV/SIT environment and QA in case of QA/UAT environment), the CI pipeline gets triggered automatically.

If one wants to trigger CI pipeline manually, below procedure must be followed

1. Go to the SIT(DEV) or UAT(QA) CI pipeline and click on the run pipeline.

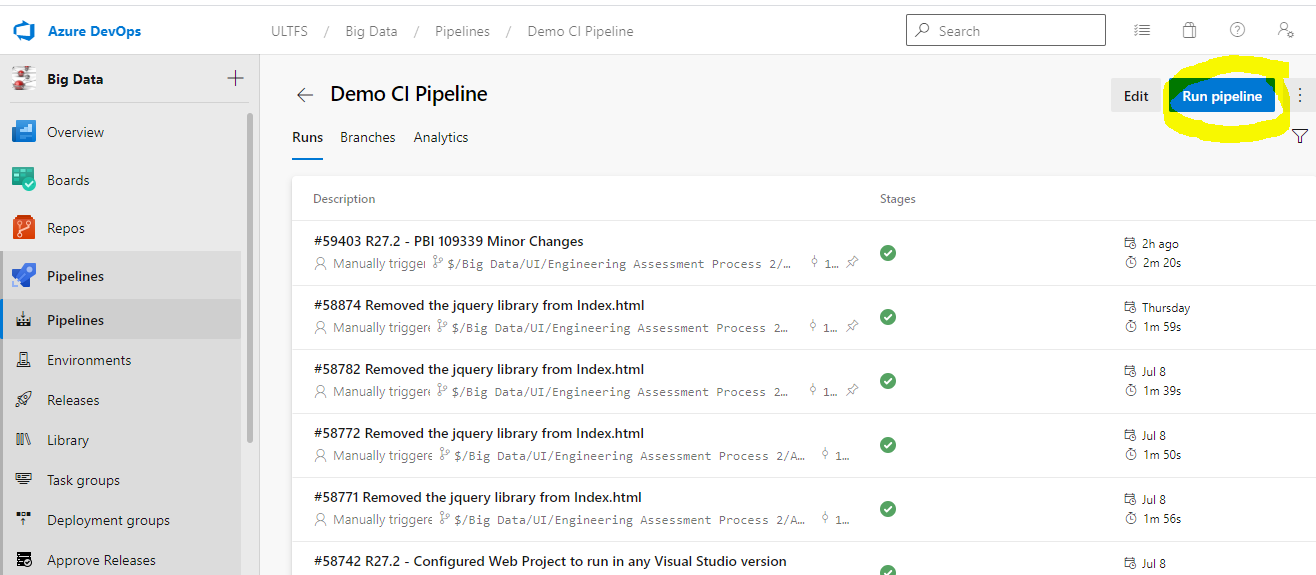


1. Click on run pipeline with the default values as shown in the below snapshot.

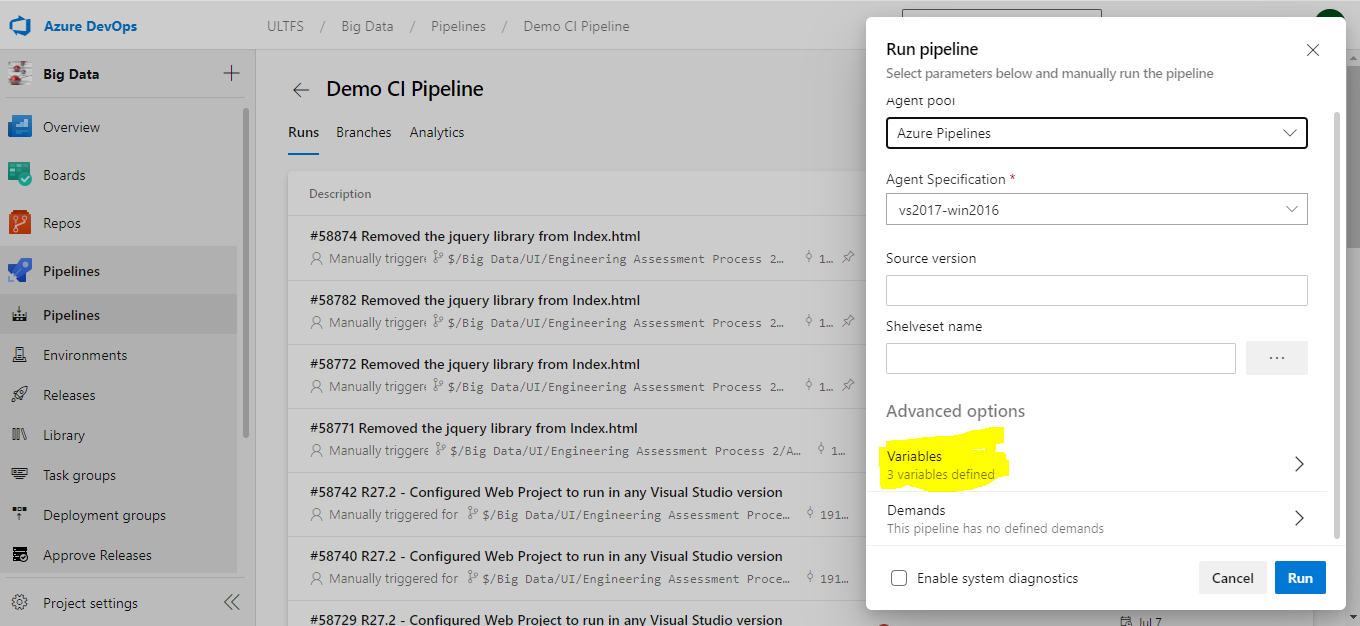


# **Procedure to trigger CI pipeline for Demo Environment**

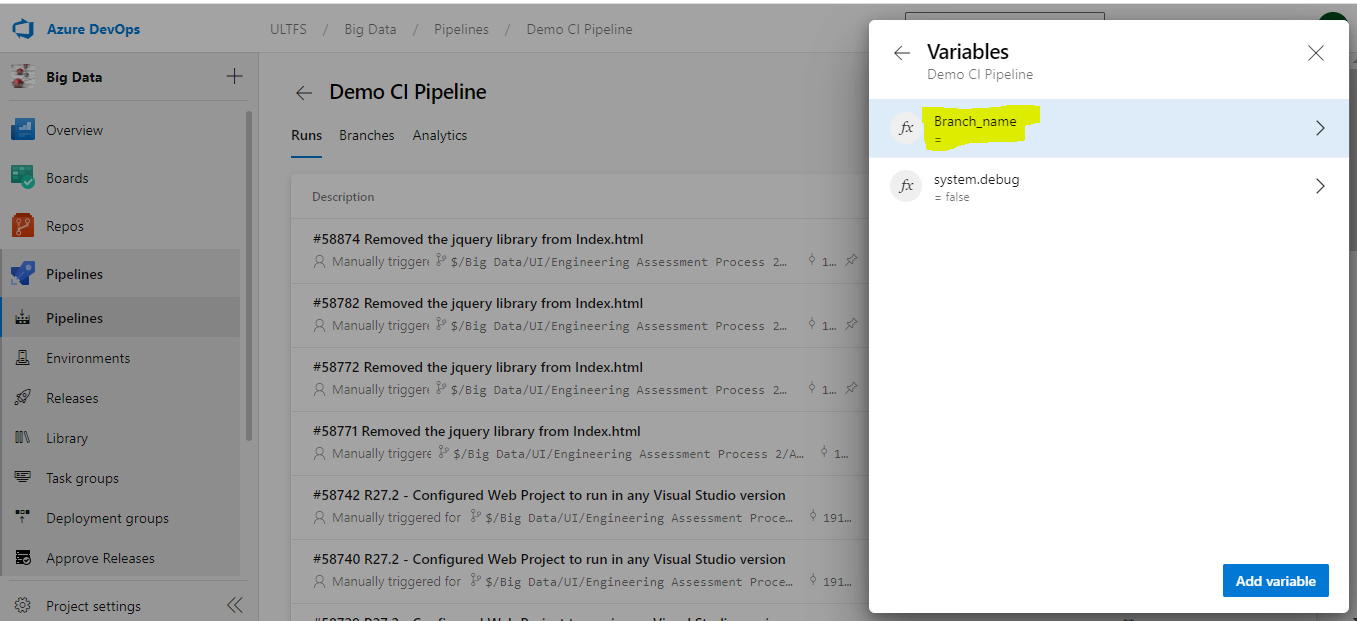
1. Go to the Demo CI pipeline and click on the run pipeline.



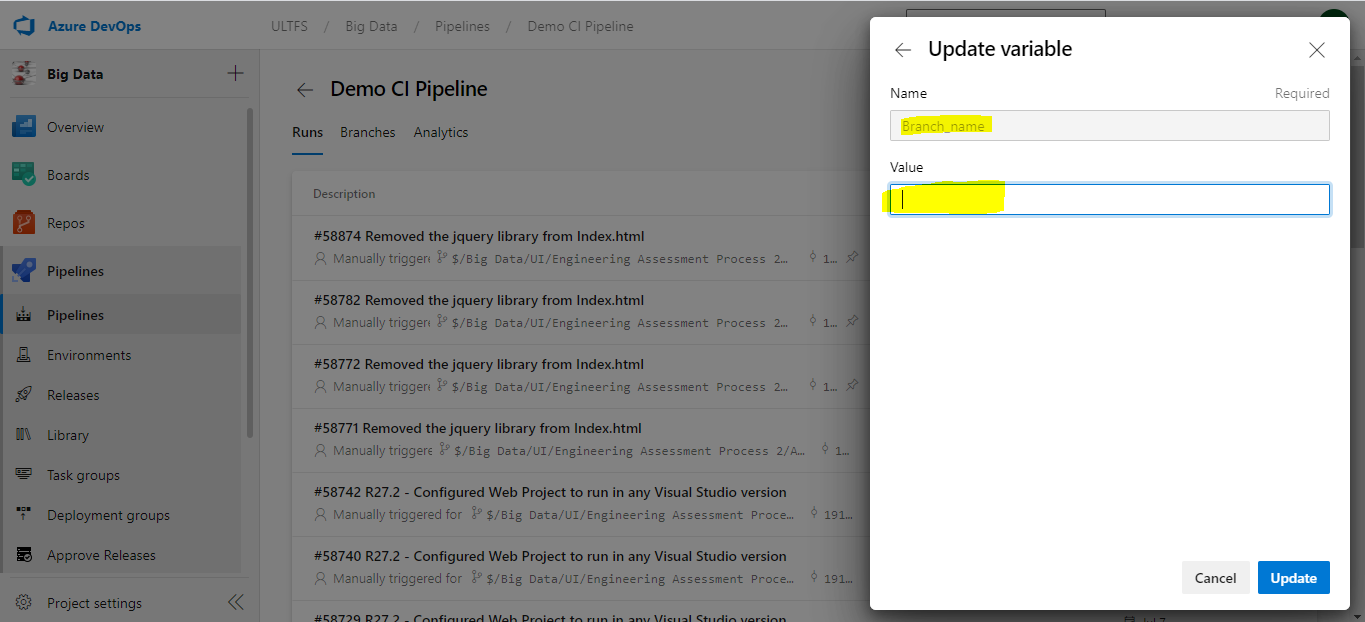
1. After clicking on Run pipeline, one can see variables as highlighted in the below snapshot.



1. Click on variables. A window opens as in snapshot. Click on the variable name as highlighted (Branch\_Name).

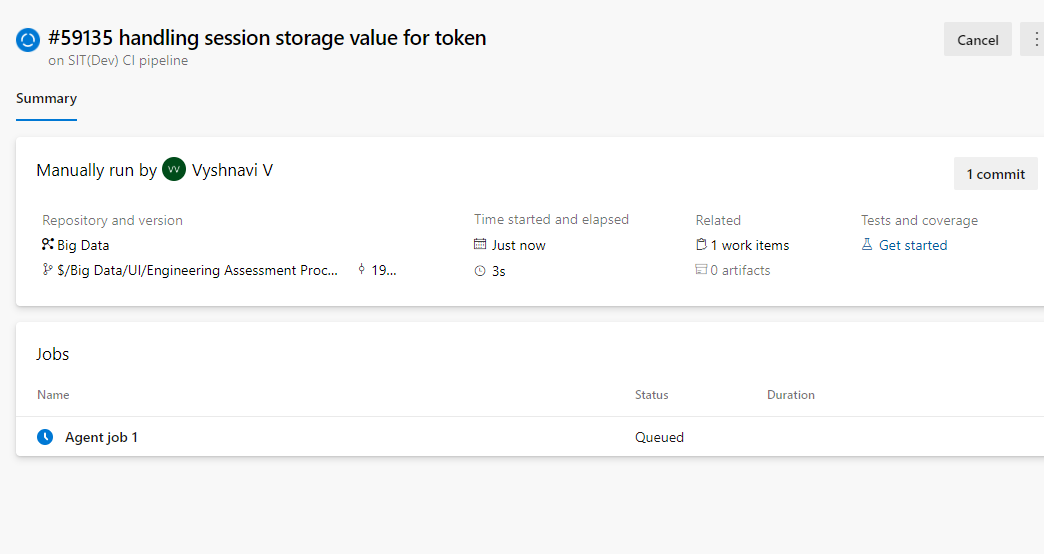


1. Provide the branch name or path to the source code from **Engineering Assessment Process 2** in the value section. See that the source code should be available in the given path/branch, click on update and run the pipeline. Ex: CERTIFICATE-FEATURES/FEATURE-85322

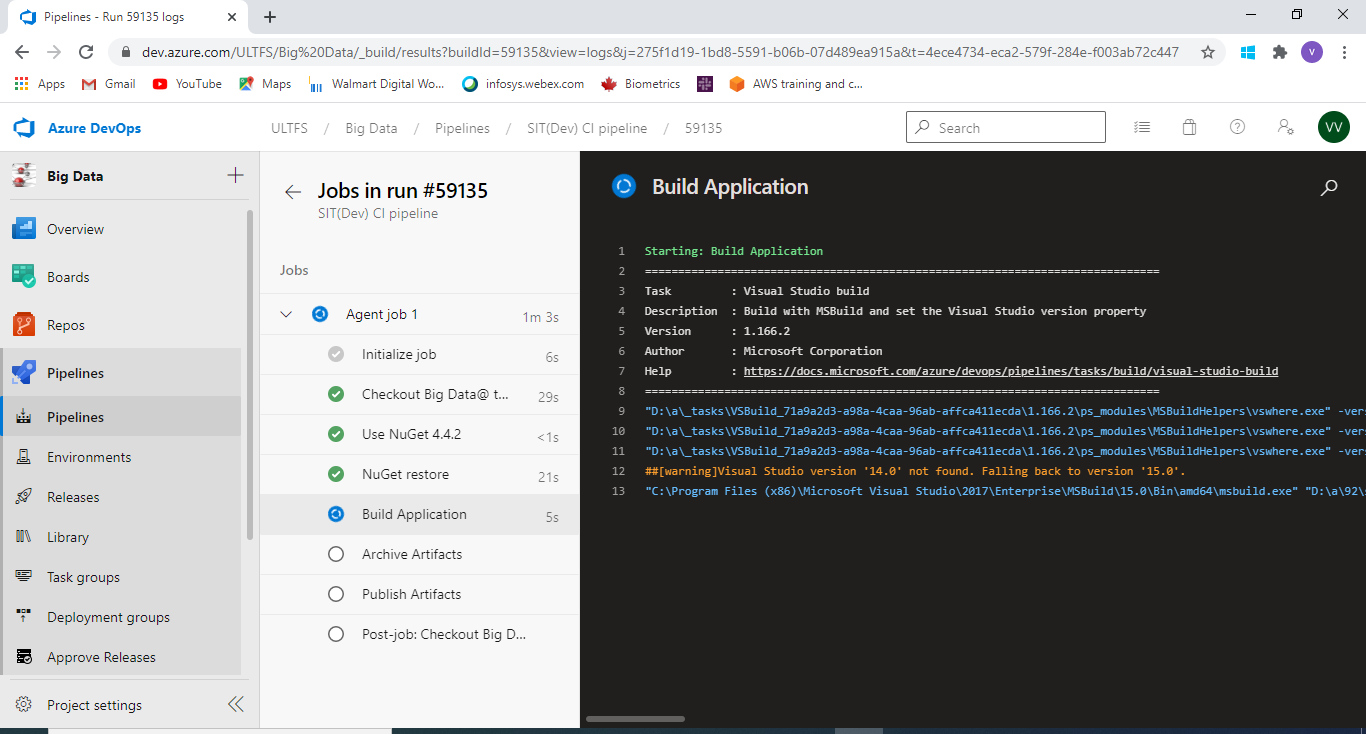


# **CI pipeline logs**

The pipeline overview can be found when one clicks on the build currently running or a previous build. A window with the job details opens as below



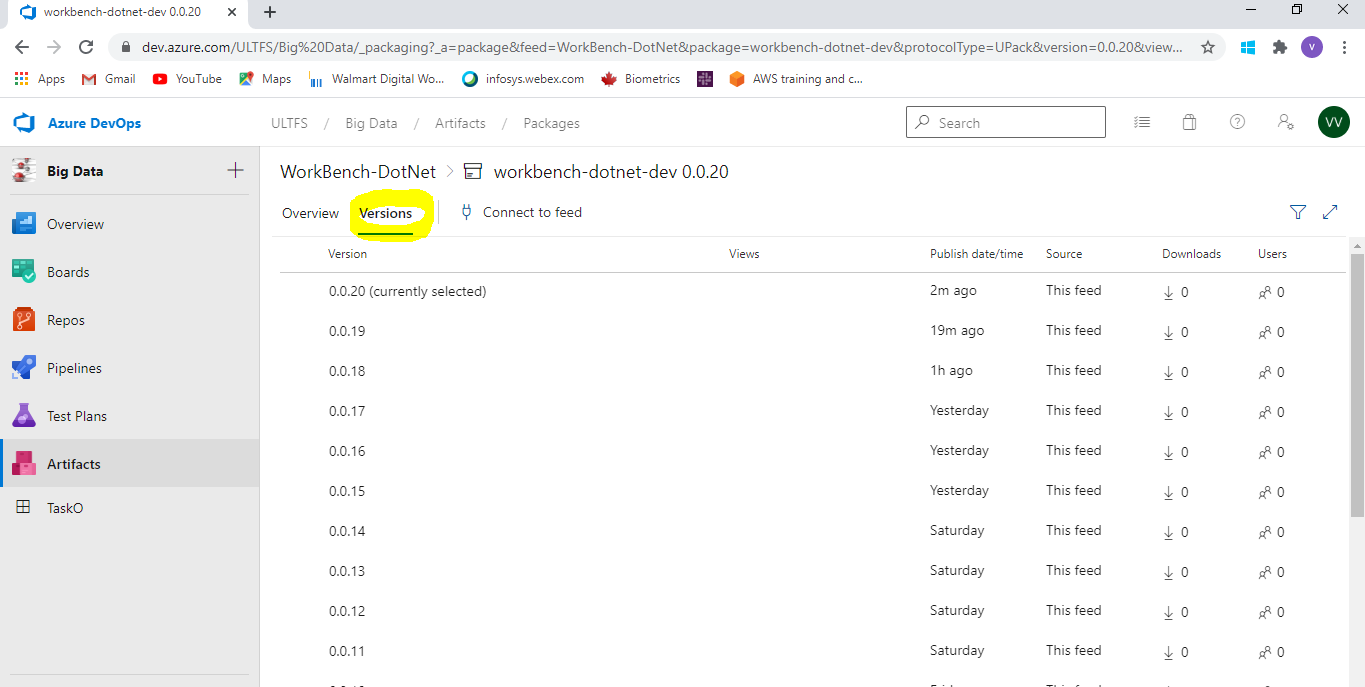
Click on Agent job 1 and the below window opens with the flow and logs



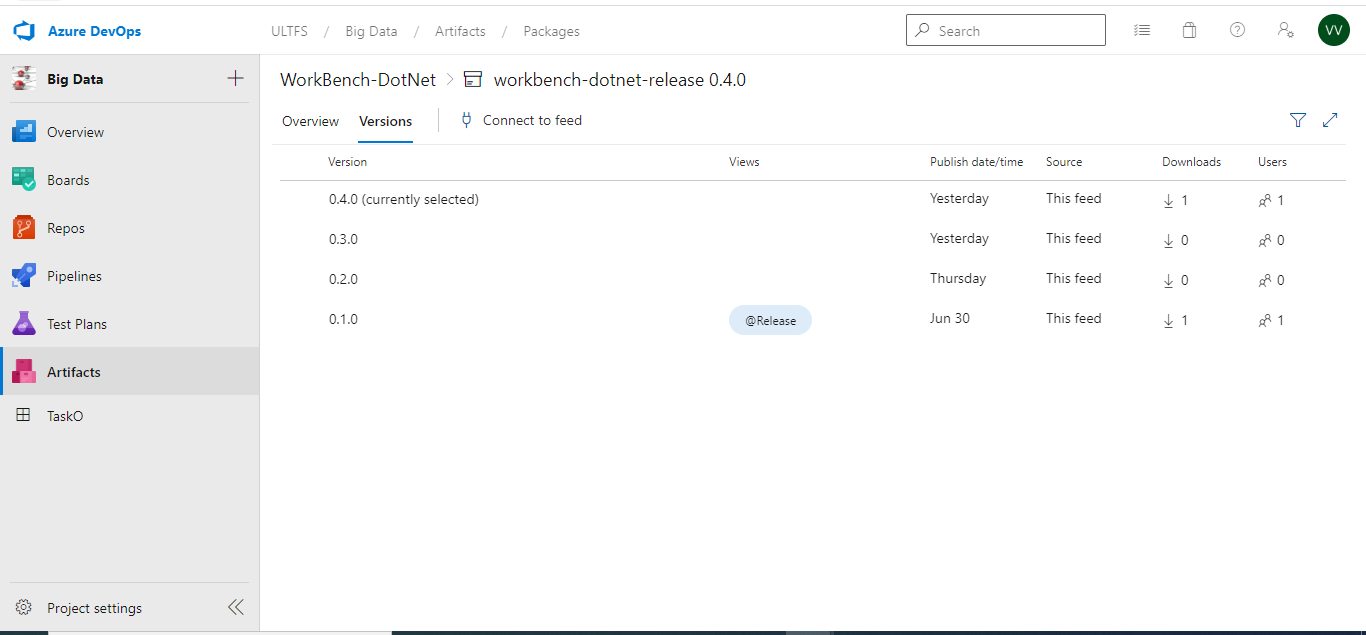
# **Artifactory and Artifacts**

Once CI pipeline build is completed, the artifacts created by CI pipeline are pushed to Artifactory.

The Artifactory snapshot for **Dev** pipeline is below:

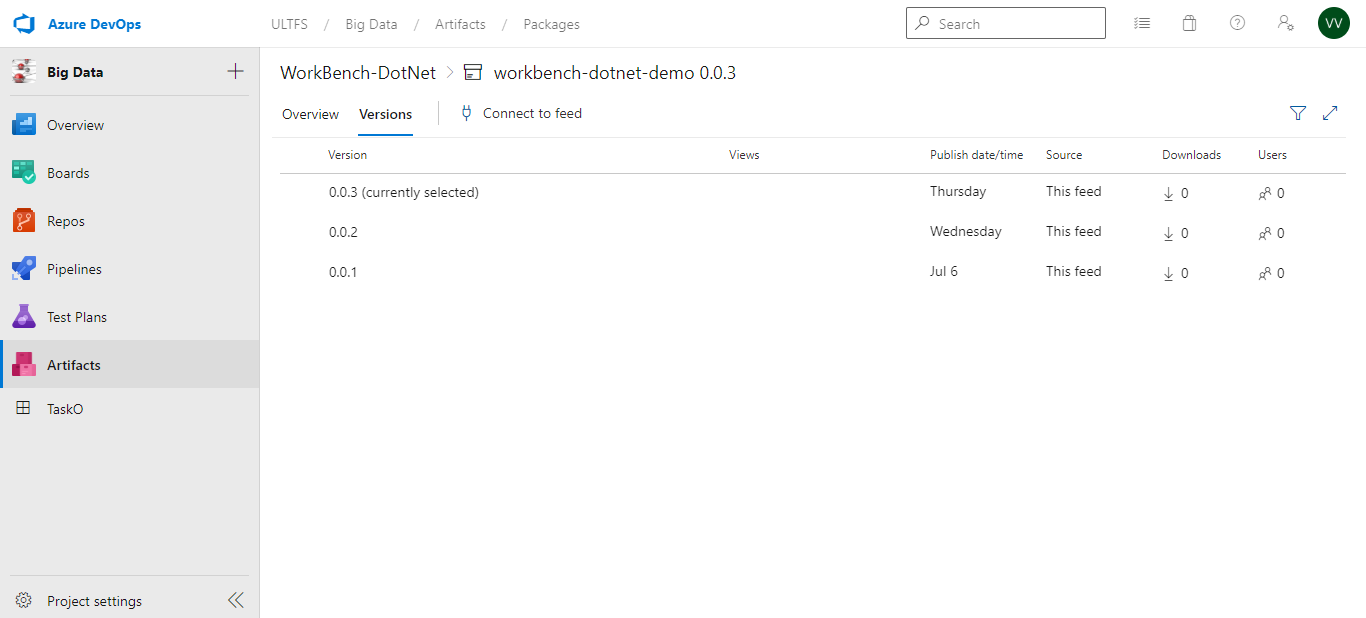


The Artifactory snapshot for **UAT** pipeline is below:

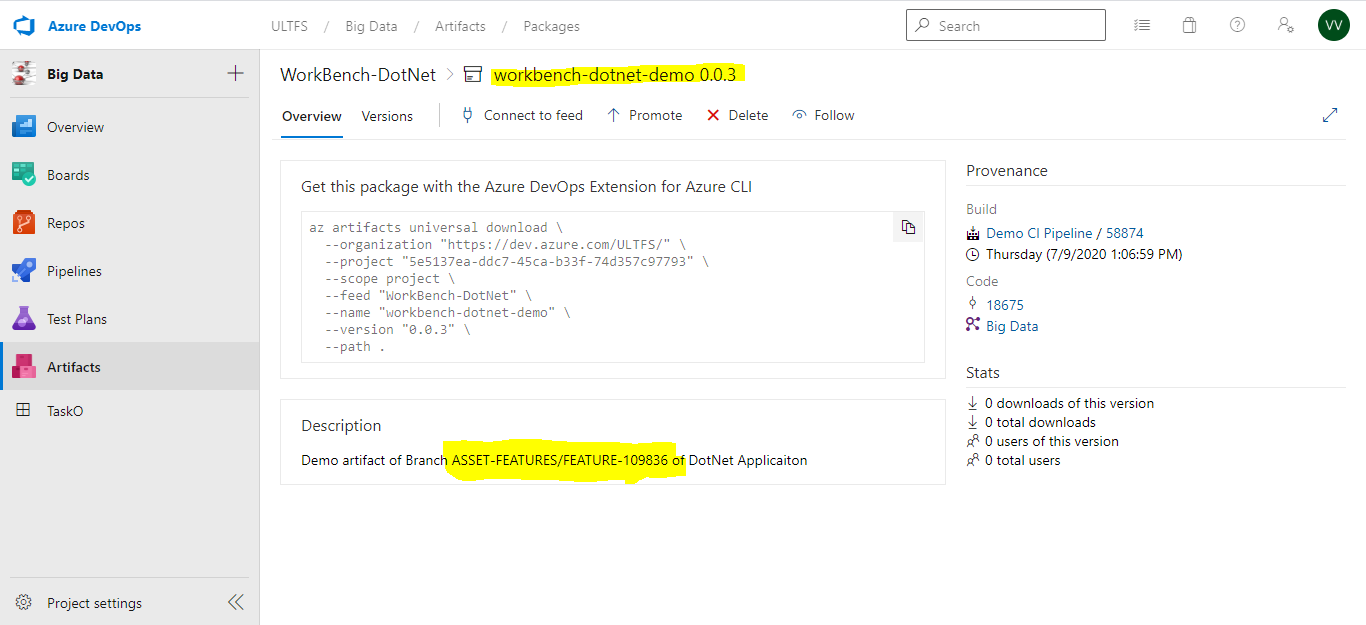


The Artifactory snapshot for **Demo** pipeline is below:

The below snapshot lists the versions available under the artifact for **Demo** workbench-dotnet-demo



The overview of Artifact shows the version of the artifact and the branch name for which the artifact is created as in below snapshot for **Demo** artifact

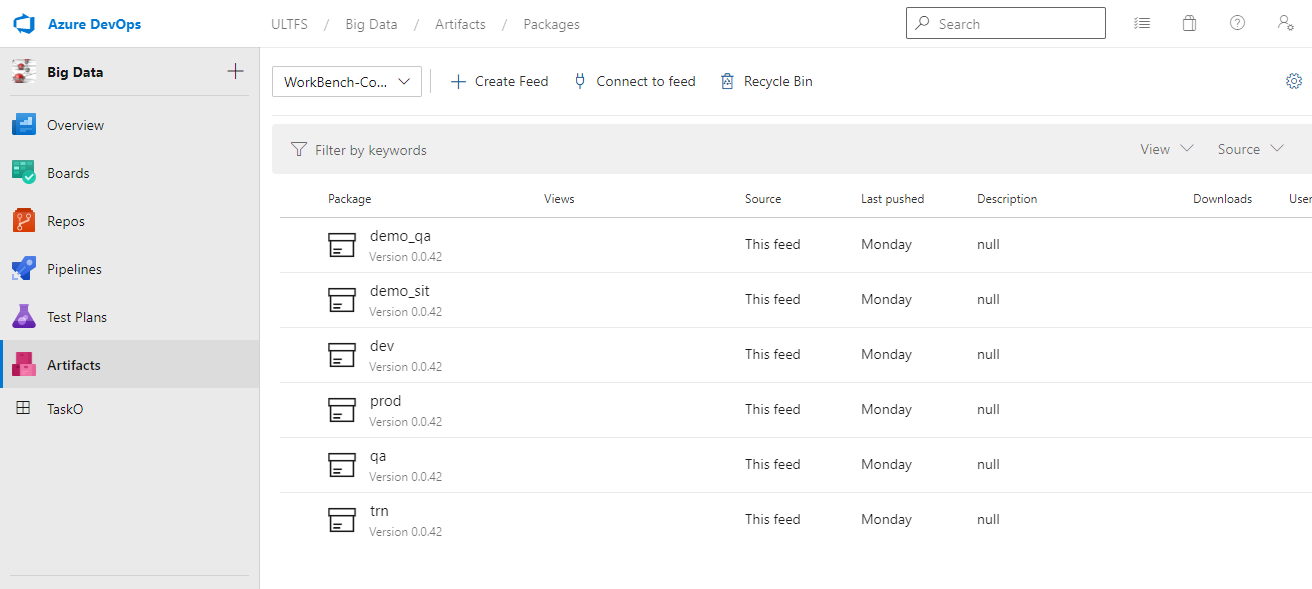


Artifact snapshot for **Config** files is below:

To publish artifacts there is a separate package created for each environment

For Dev environment the package named as dev

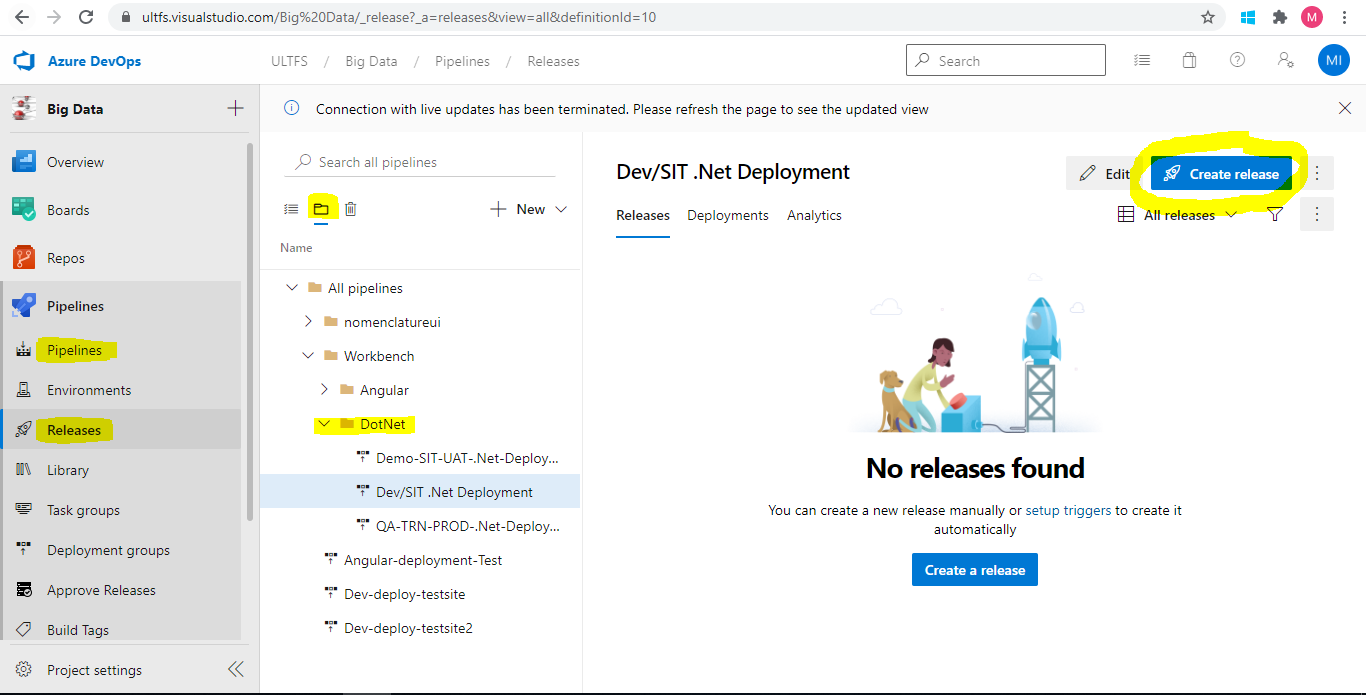
For Demo QA environment the package named as demo\_qa



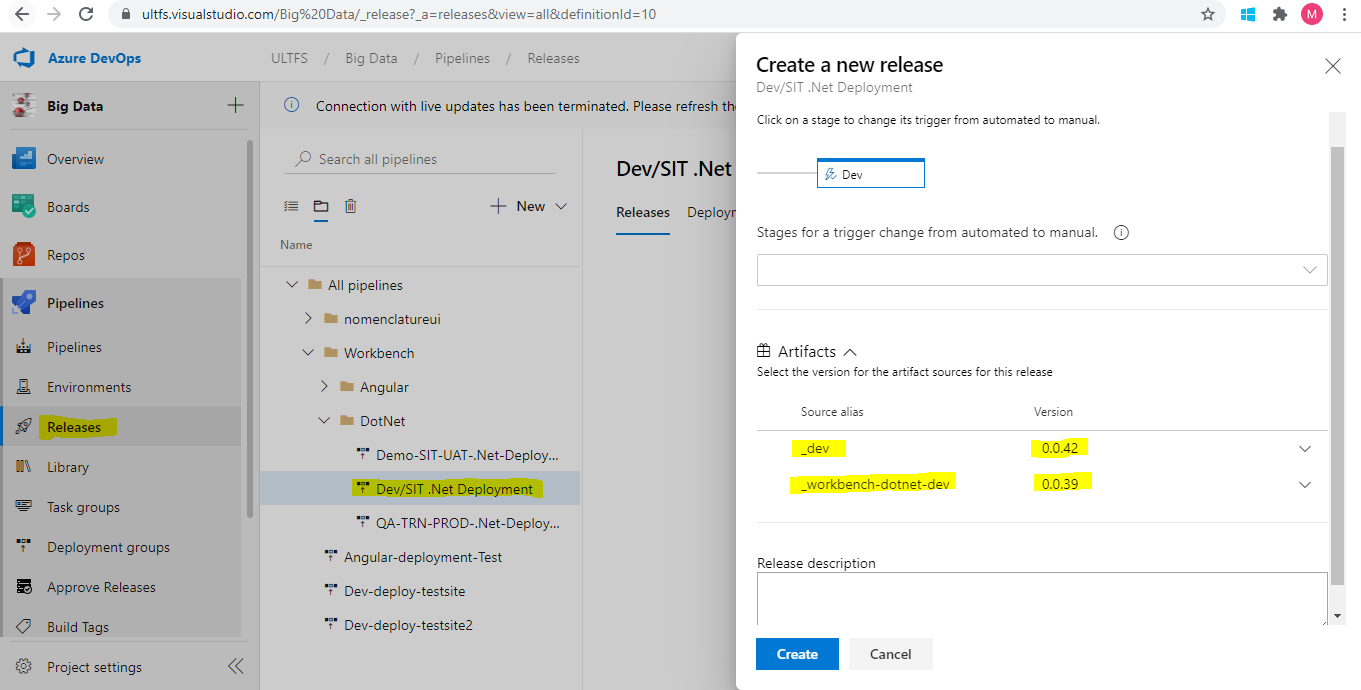
# 

# **Procedure to trigger CD pipeline for Dev/SIT environment**

1. Go to Dev/SIT .Net Deployment pipeline and click on **Create release** as shown in the below snapshot.



1. A window pops up as in the below snapshot with the artifact versions to be deployed.



1. By default, the latest version is selected, previous version can be changed from the drop down depending on the requirement & click on **Create**.
2. A release is created, and the deployment starts on the **Dev** server.

**Artifact details in the release:**

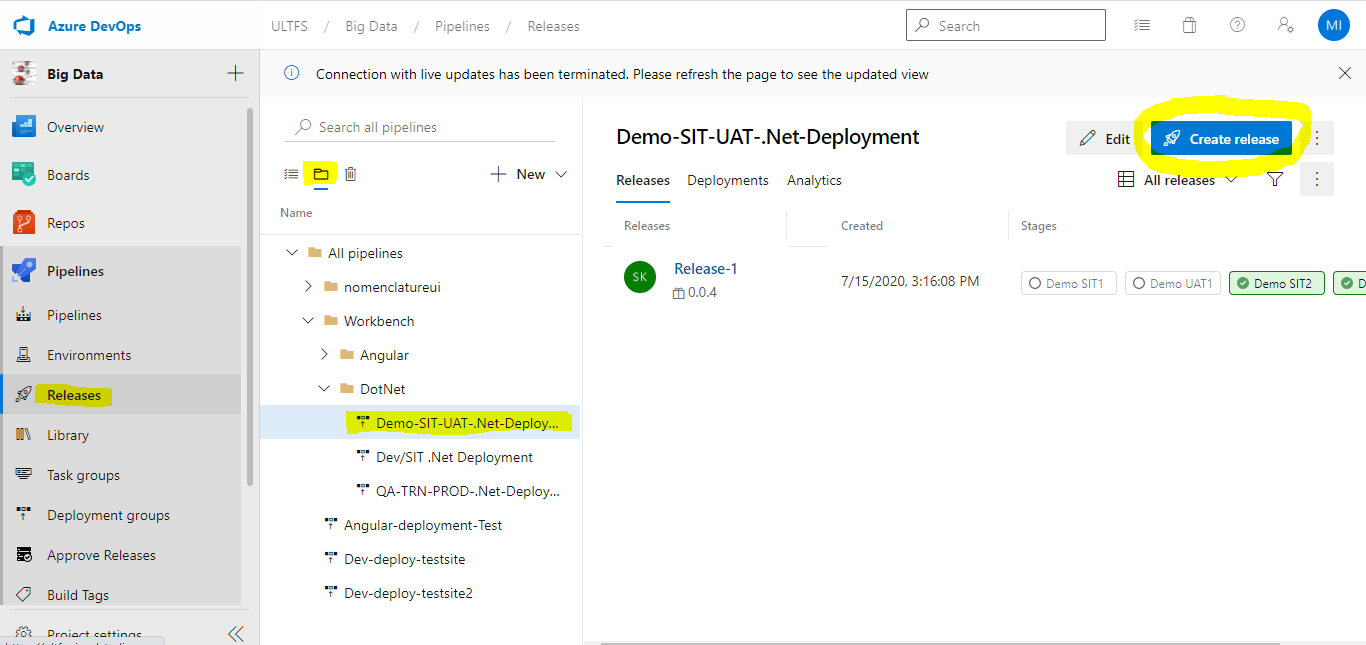
Under Artifacts section, there are 2 packages for Dev environment

1. **\_dev** – This package contains conf related file like web.config
2. **\_workbench-dotnet-dev** – This package contains dll files from dev branch

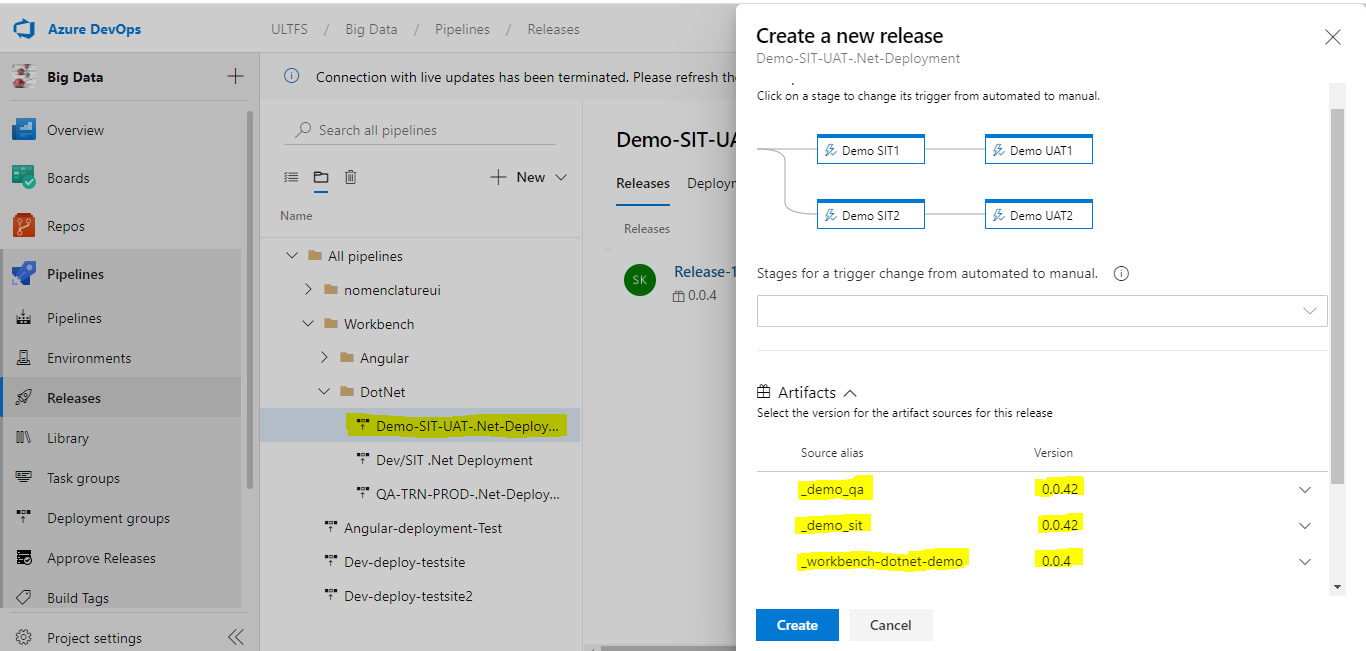
# **Procedure to trigger CD pipeline for QA, Prod & Training Environments**

A single Release pipeline is created for deployment on QA, Prod and TRN environments in different stages with approvals at each stage. Follow the below procedure to deploy:

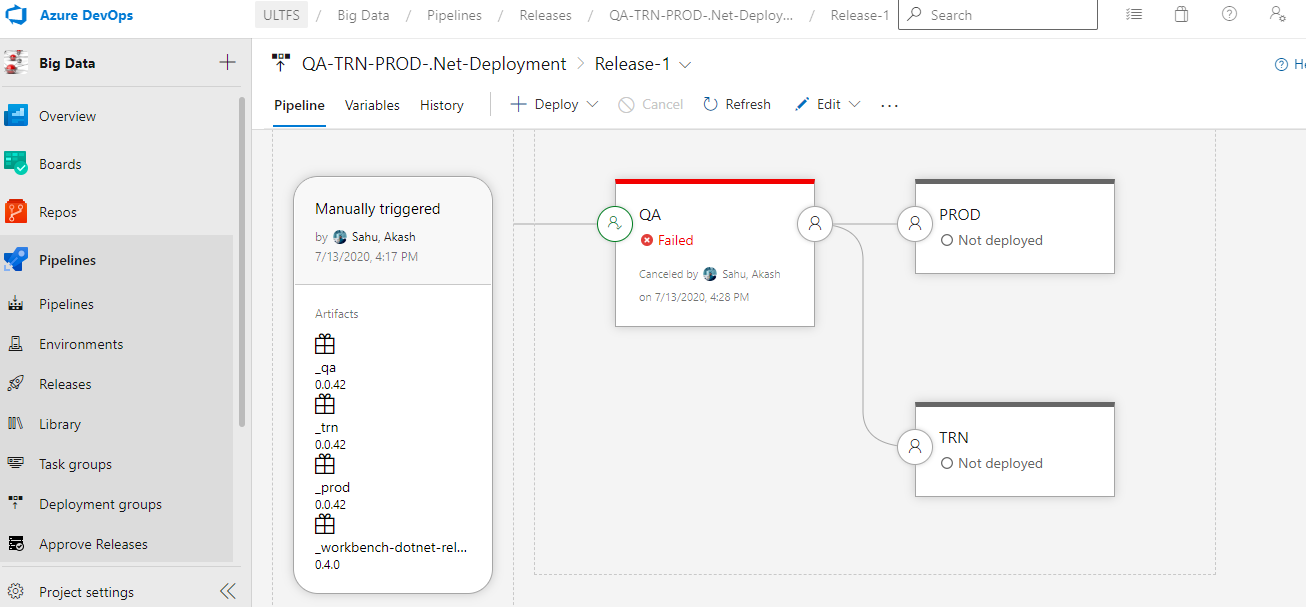
Go to QA-TRN-PROD-.Net-Deployment release pipeline and click on **Create release** as shown in the snapshot.



1. A window pops-up as in the below snapshot with the artifacts and their versions.



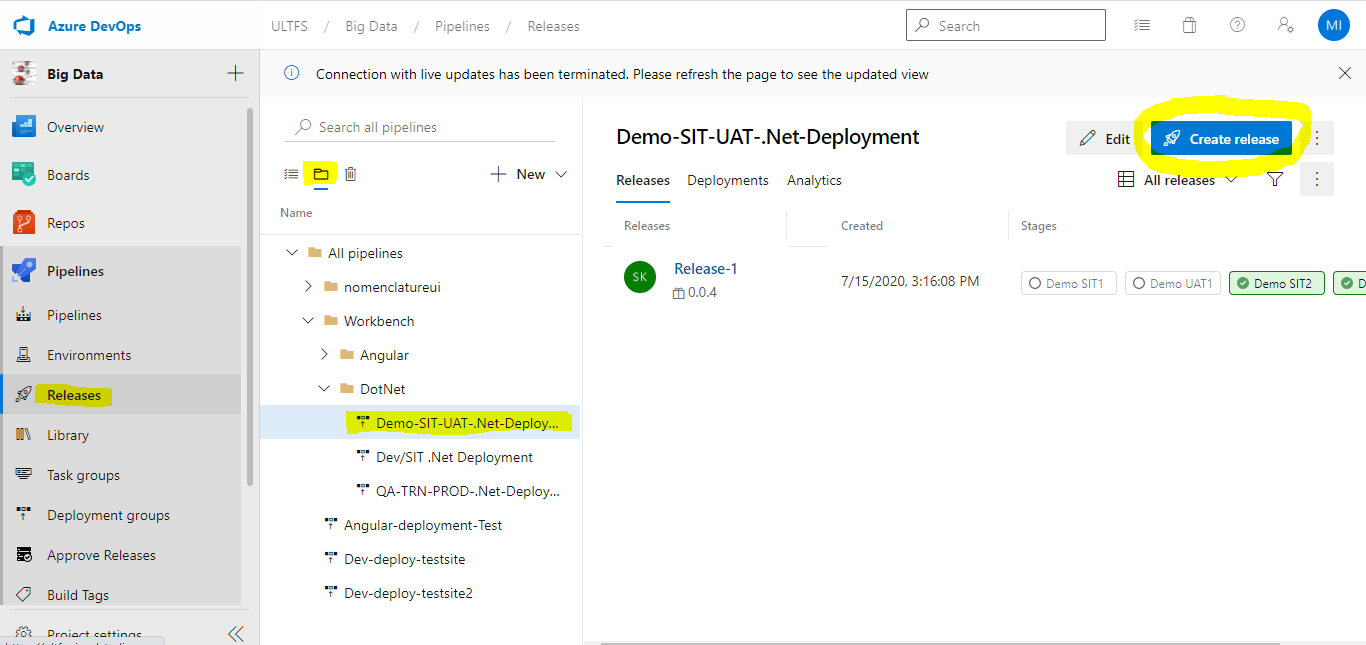
1. By default, Artifacts with the latest versions are selected. Version of the artifact can be changed accordingly depending on the requirement. Below are the artifact packages that will be deployed.
2. **\_prod, \_qa and \_trn** – This package contains conf related files like web.config. \_prod is for production, \_qa is for QA and \_trn is for Training environment.
3. **\_workbench-dotnet-release** – This package contains dll files from demo branch
4. After selecting the version, click on **Create**. A new release is created, and deployment starts
5. As the approval is required for all the stages, every stage waits until approved.
6. On Create release, emails are triggered for approvals to deploy on QA environment.
7. Once deployment is completed on **QA** environment, emails are triggered for **post deployment approval**.
8. Once post deployment is approved for QA, then emails triggered for **PROD and TRN deployments** for approvals as in the below snapshot.



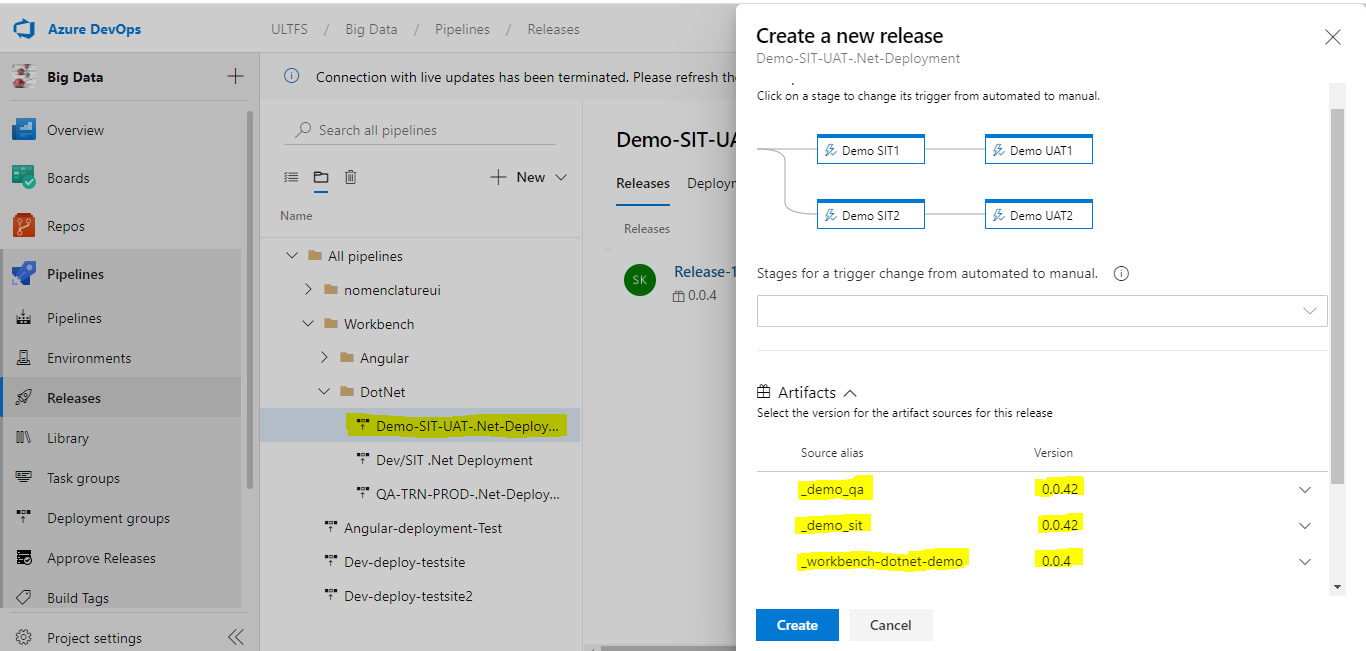
# **Procedure to trigger CD pipeline for Demo Environments**

A single pipeline is created to deploy for all the Demo environments.

1. Click on the pipeline URL and click on **Create release**



1. Select the stages(environments) to deploy as shown in the below snapshot.



Under Artifacts section, there are 3 packages for Demo environment

1. **\_demo\_qa, \_demo\_sit** – This package contains conf related file like web.config and environment.json files.
2. **\_workbench-dotnet-demo** – This package contains dll files from demo branch

# **Approvals**

Below are the approval details added as per environments

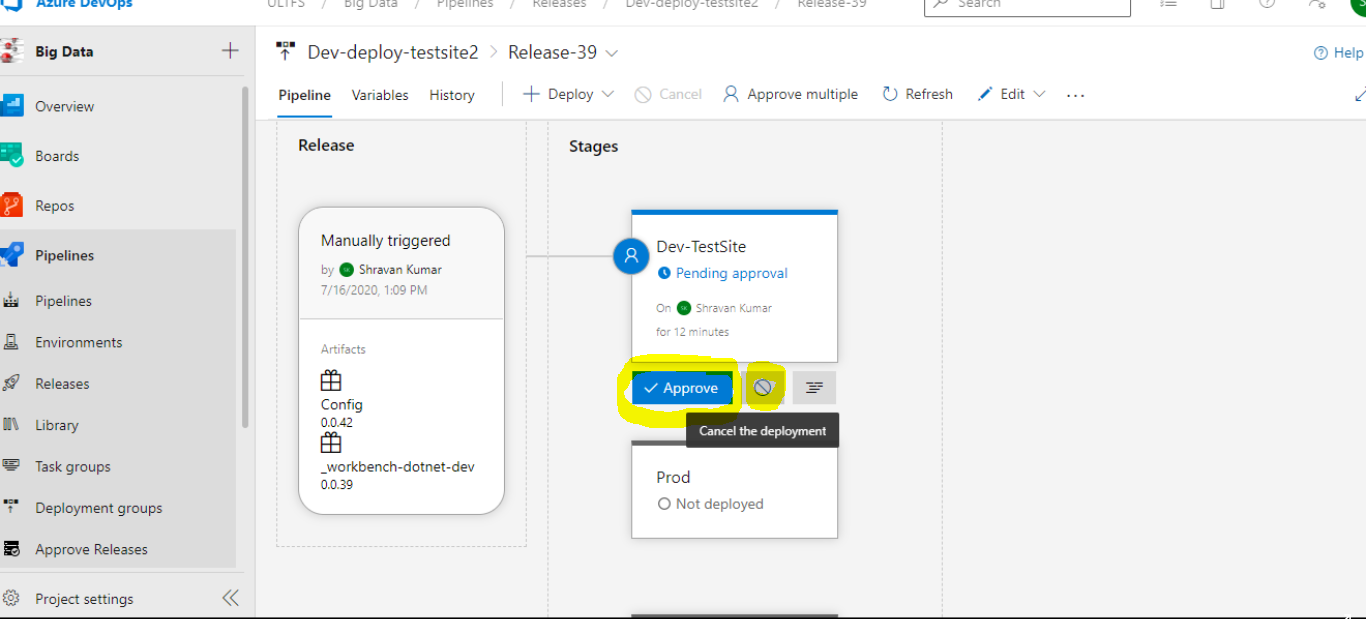
|  |  |  |
| --- | --- | --- |
| **Environment** | **Pre-Deployment** | **Post-Deployment** |
| Dev | No | No |
| QA | Yes | Yes |
| Prod | Yes | No |
| Training | Yes | No |
| Demo SIT | No | Yes |
| Demo QA | Yes | No |

## **Procedure to Approve Deployment**

An email is sent to the approvers for pre or post deployment and below is the sample of the same. Email has all the details like Pipeline name, Release number, artifacts and their versions.



To approve the deployment, click on **View approval**, then it will take you to the Release pipeline which needs to be approved in Azure DevOps portal as shown in below screenshot.



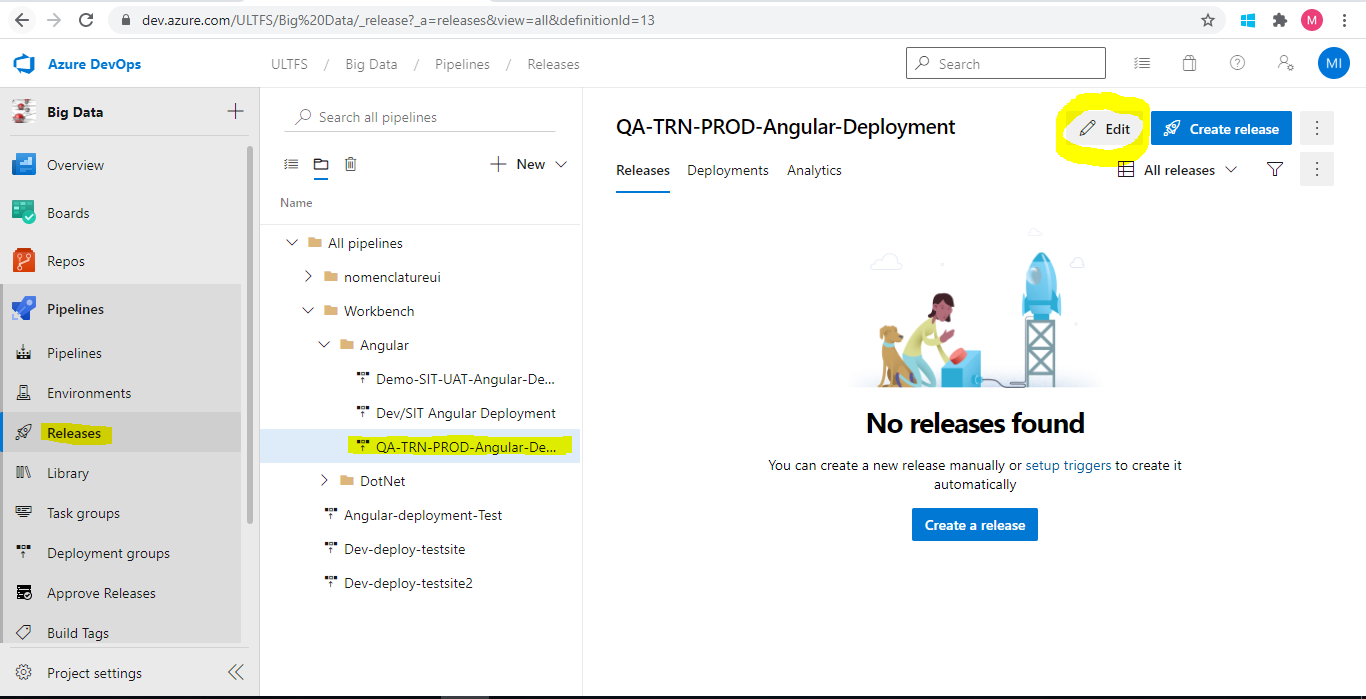
In this page approver has 2 options, one is **Approve** and second one is **Cancel the Deployment**. To approve the deployment, click on **Approve**, to cancel click on **Cancel the deployment**

If approver don’t take any action, it will wait till the time out and deployment get cancelled.

## **Adding or Removing an Approver:**

1. To add or remove an approver, follow the below steps.

* Go to Releases
* Select any deployment pipeline and click on “Edit”



Click on “Pre-Deployment Condition” on any stage, then it will open pop up window where we can search to add or remove an approver.

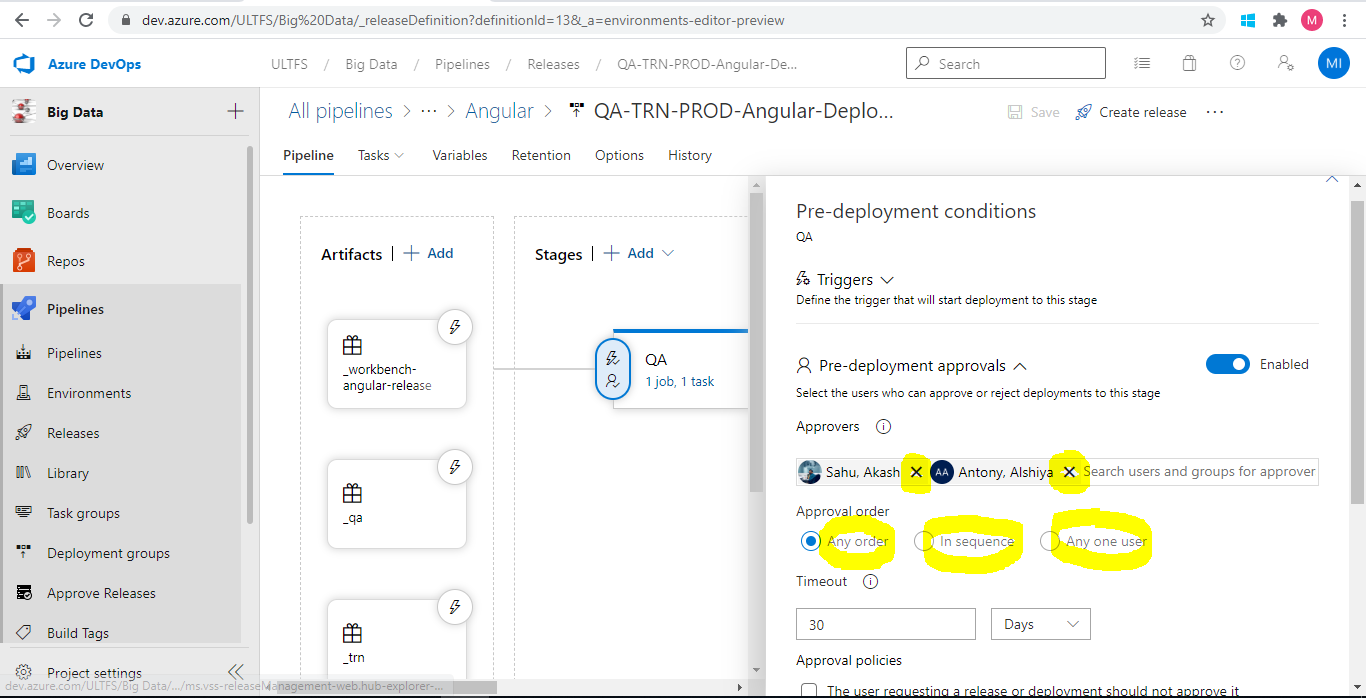


Under Pre-deployment approvals

* To add approver search and select the user
* To remove just click on “X” to remove the approver

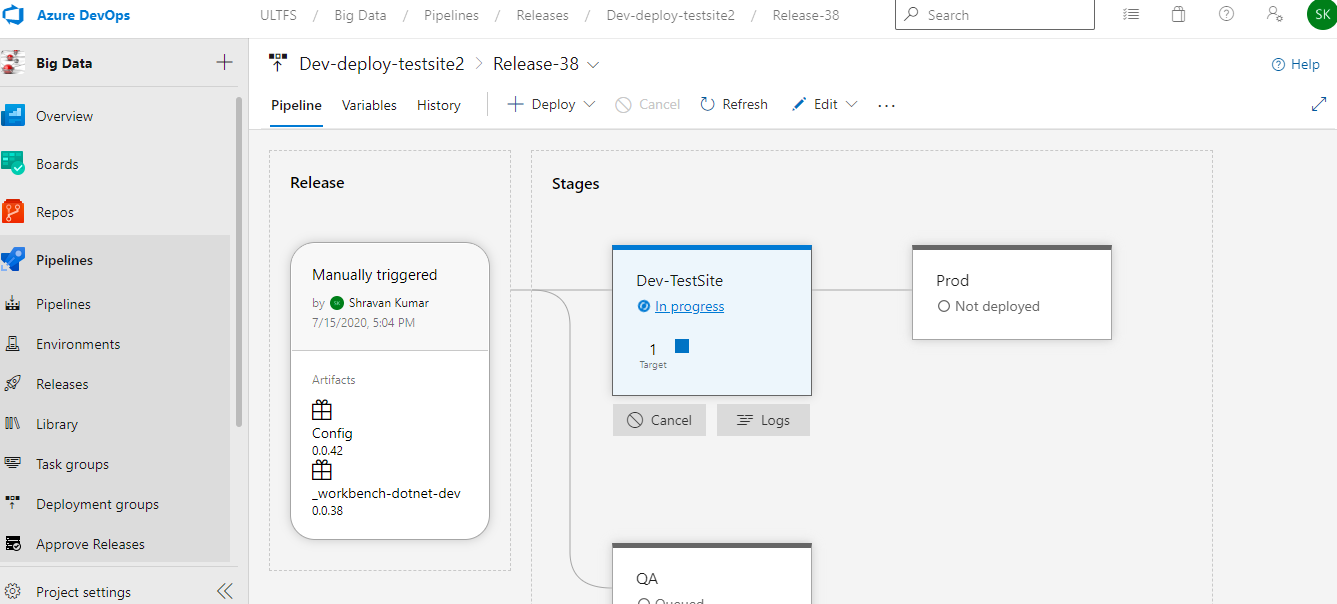
Approval order

* Any order -> Approvers can approver in any order, without waiting for other approver.
* In Sequence -> If this option selected, the second approver can approve only when first approver approves it.
* Any one user -> If this option selected, any one of approval is enough out of all the approvers.



## **Cancel Deployment**

During Deployment any time we can cancel the deployment. As shown in below mouse over to the Deployment stage, then we can find the option of Cancel.

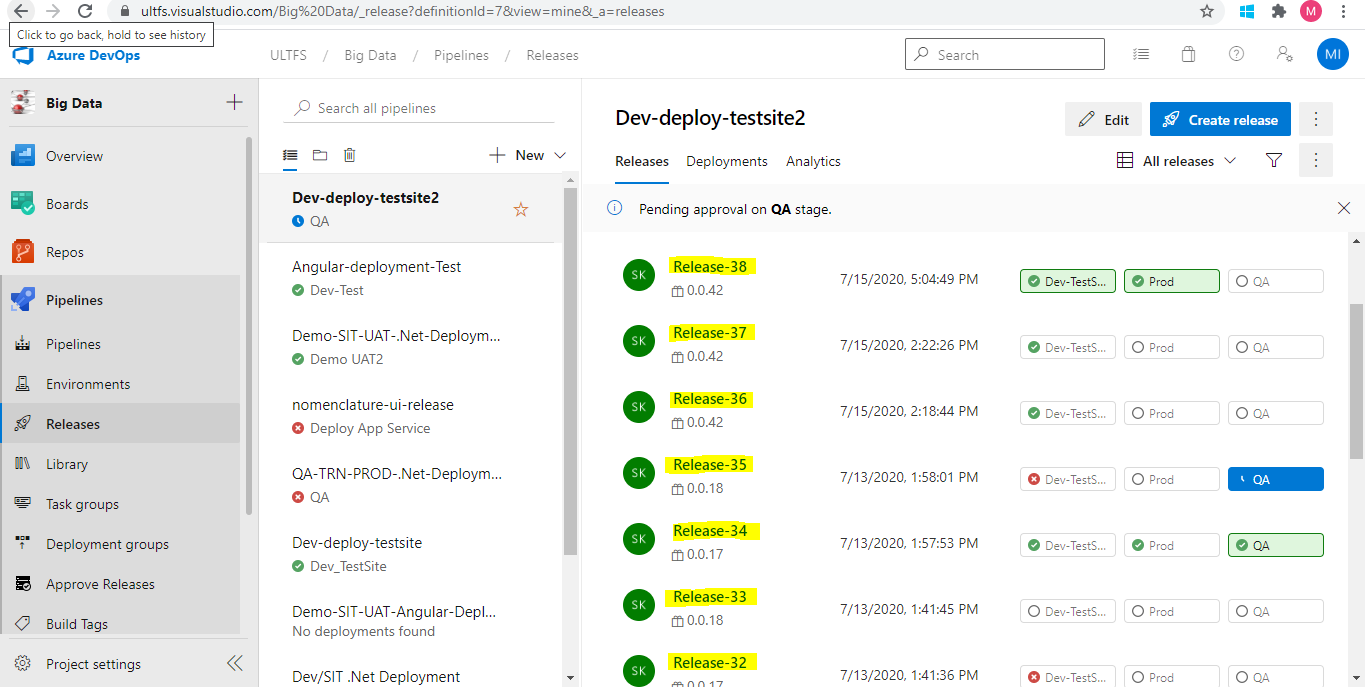


# 

# **Roll back**

Rollback can be done in 2 ways.

1. Redeploy the previous release. Select required previous release and click on **Redeploy**

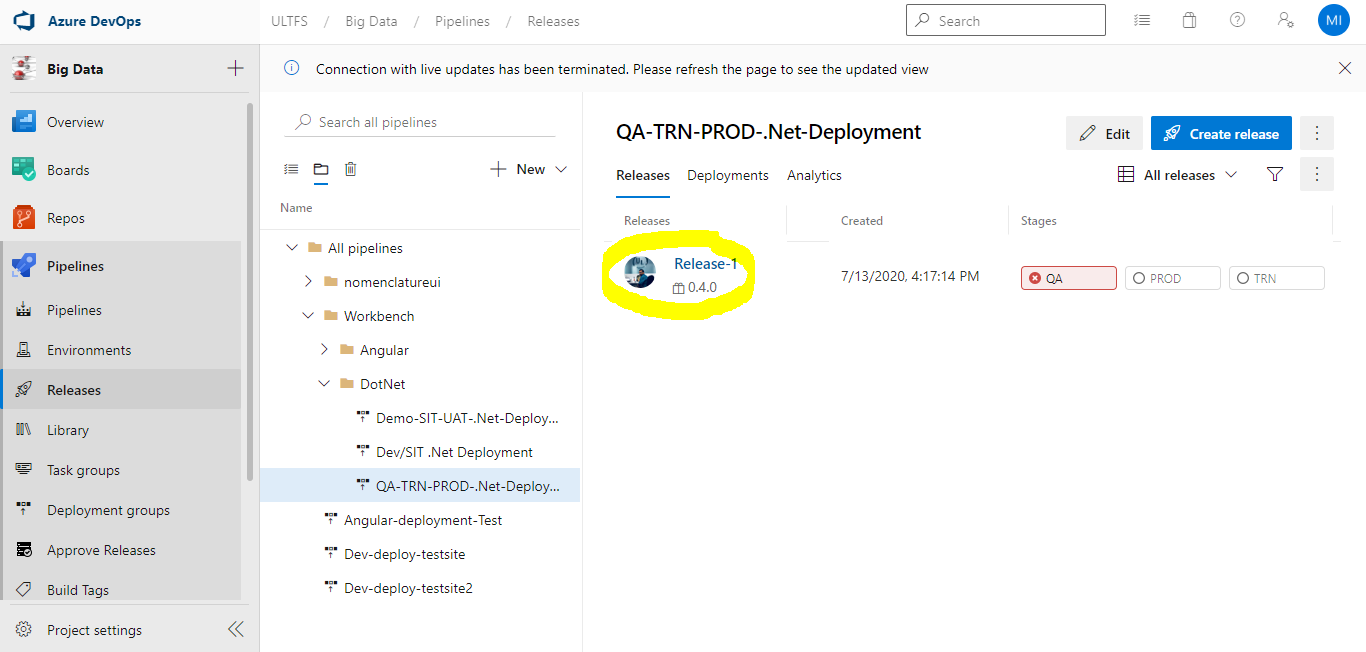


1. Create a new release and select the previous versions from drop down list.

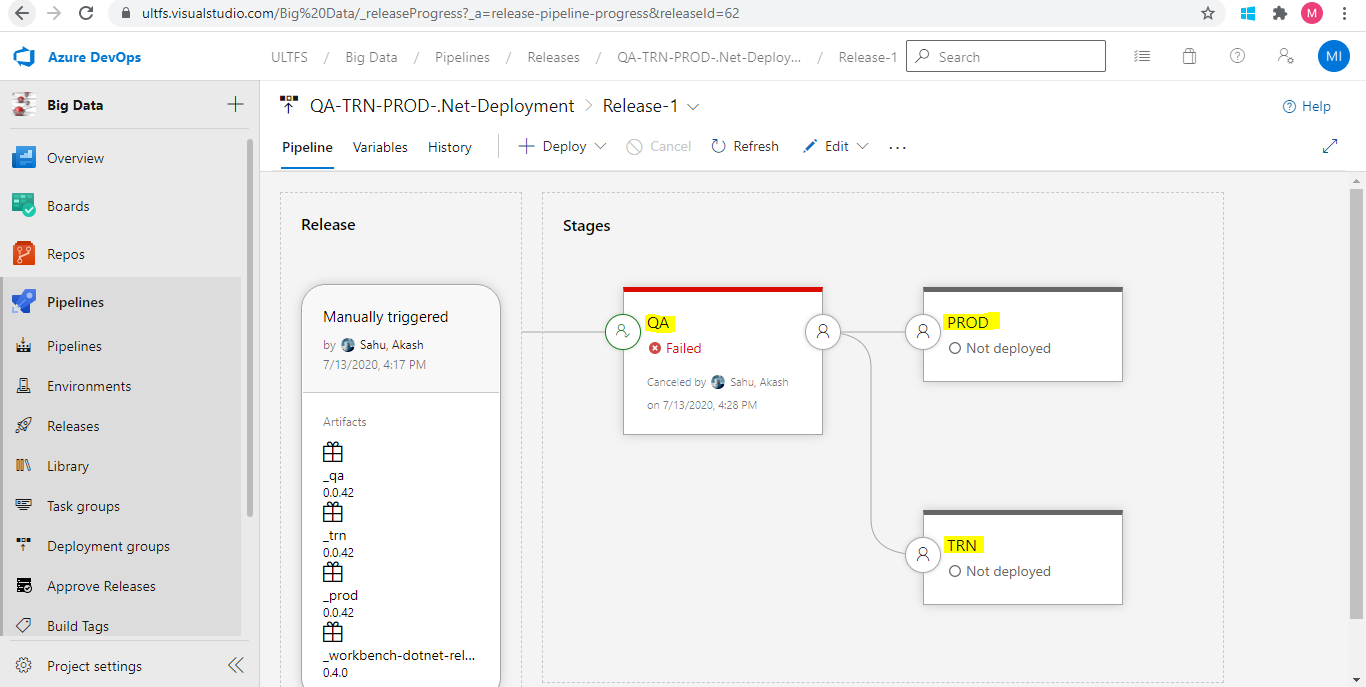
# 

# 

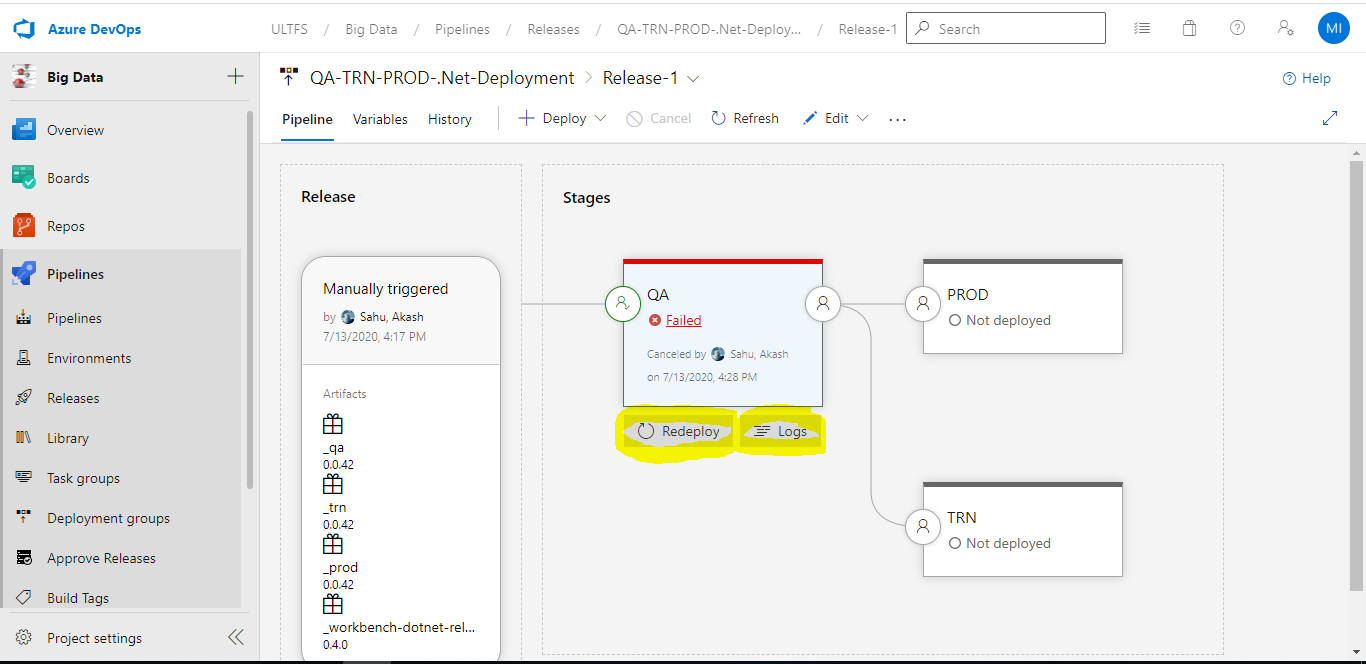
# **Deployment Logs**

Once deployment completed, select the recent release, in below screen shot the recent release is **Release-1**. 

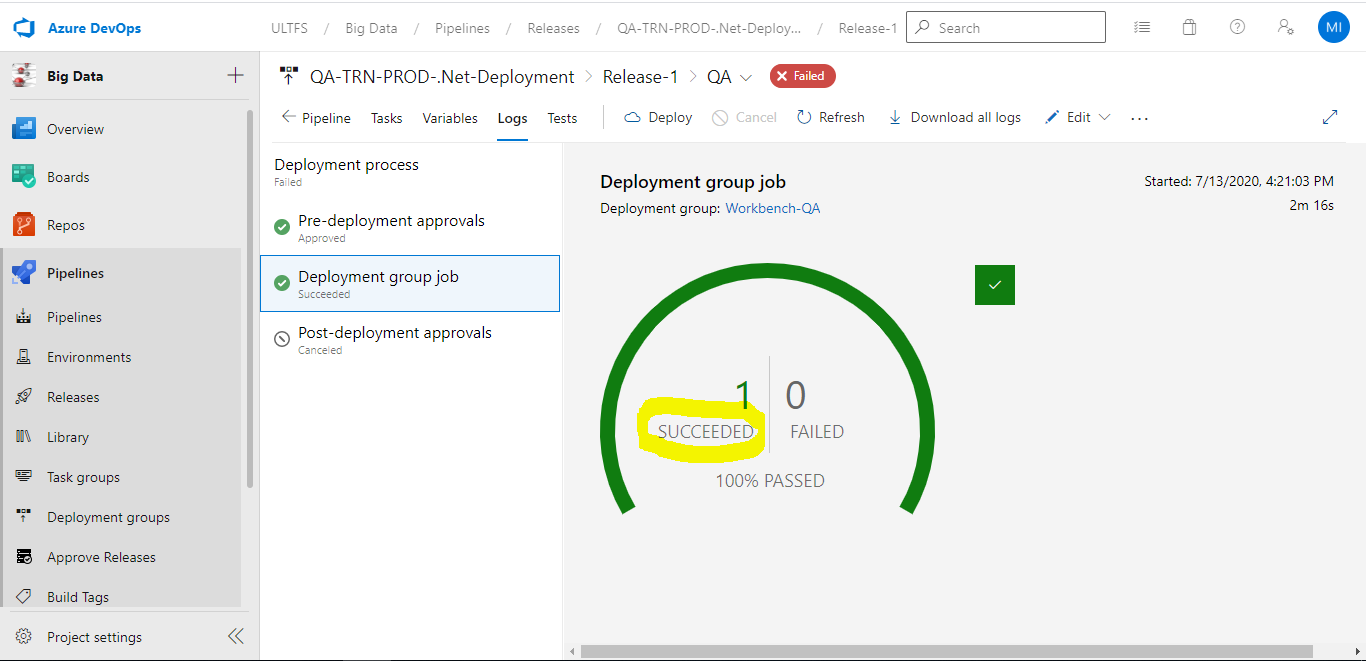
One clicking **Release-1**, the below window opens with all the stages.



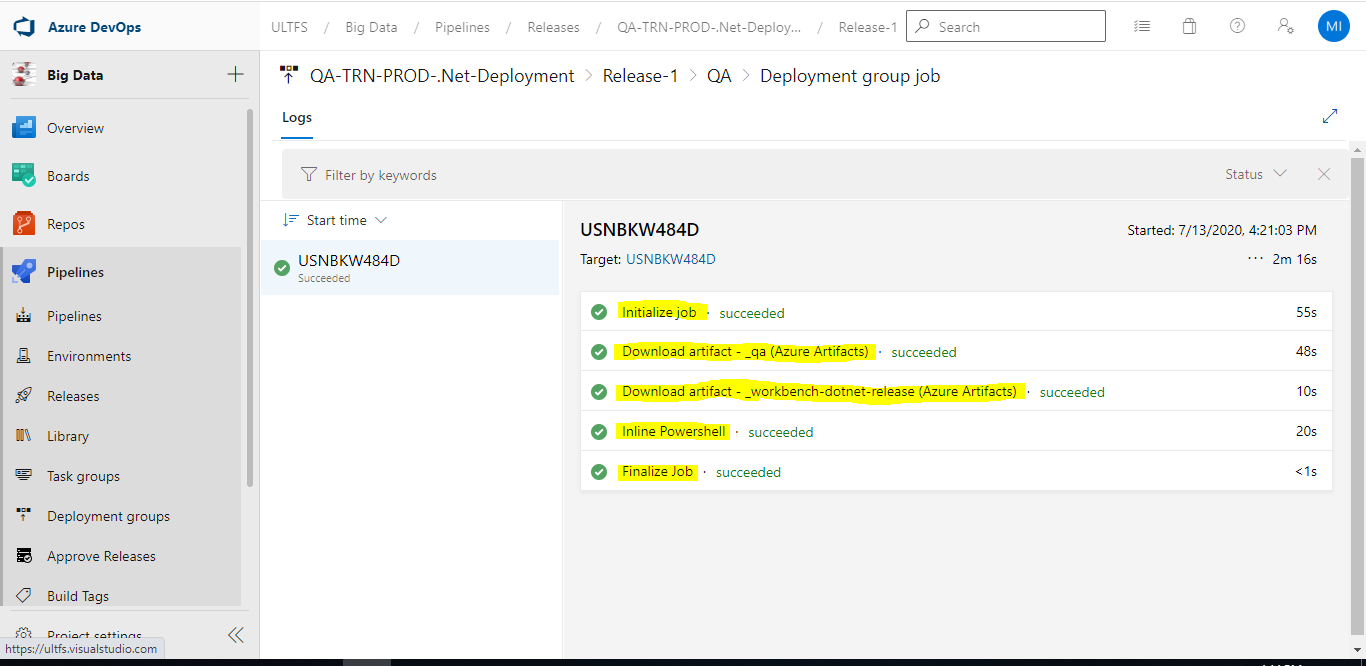
On hovering over any stage, one can see 2 options -**Redeploy** and **Logs** as in the below snapshot.



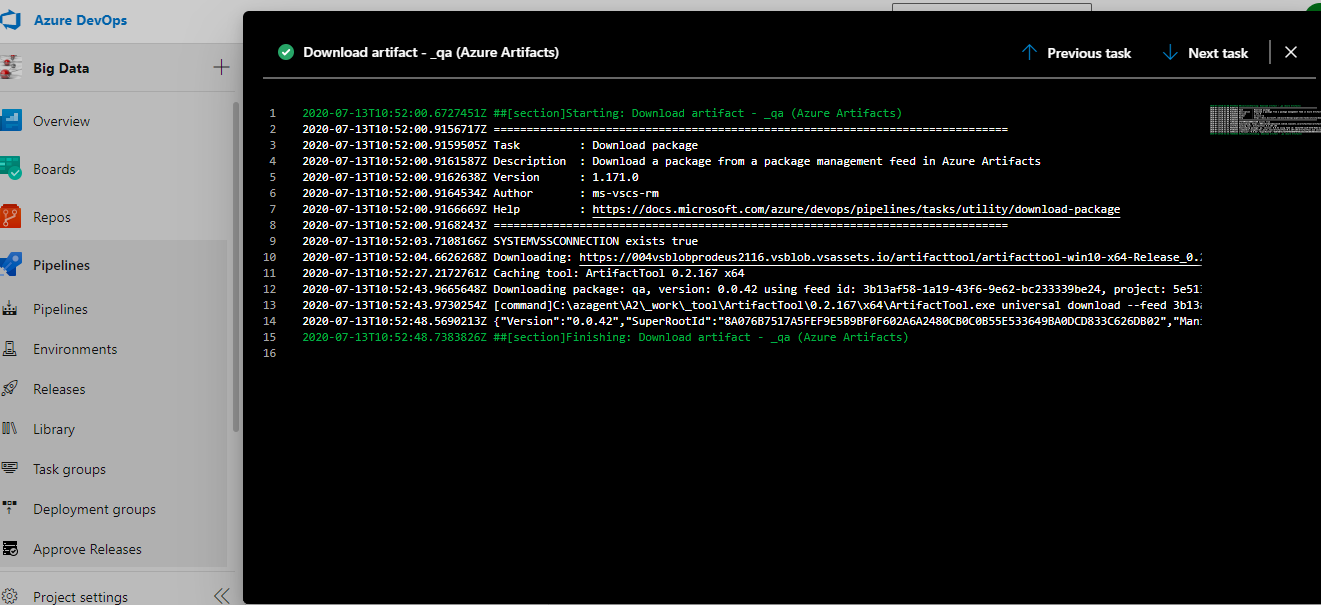
Click on **Logs**. The below window opens with the log details.



Click on **Succeeded** or **Failed** or **In-progress**, then one can see log flow as in the snapshot below

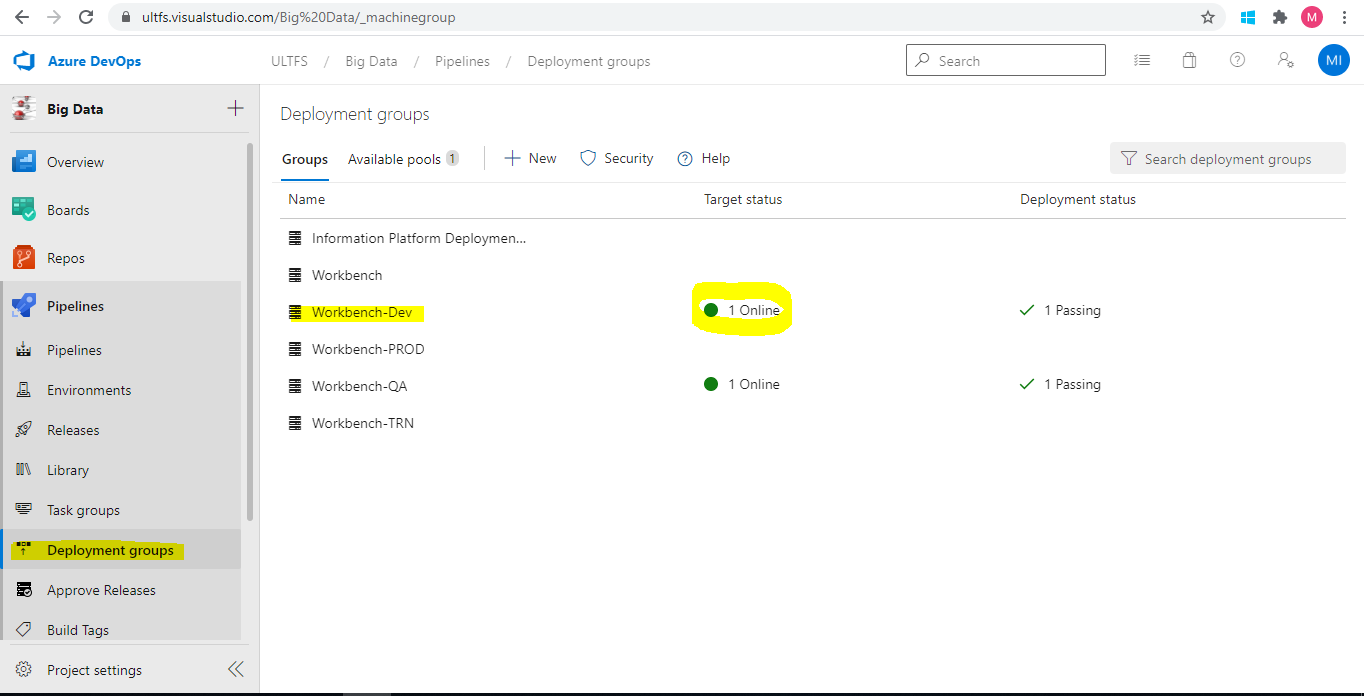


Click on any Step, then we can see the log related to step

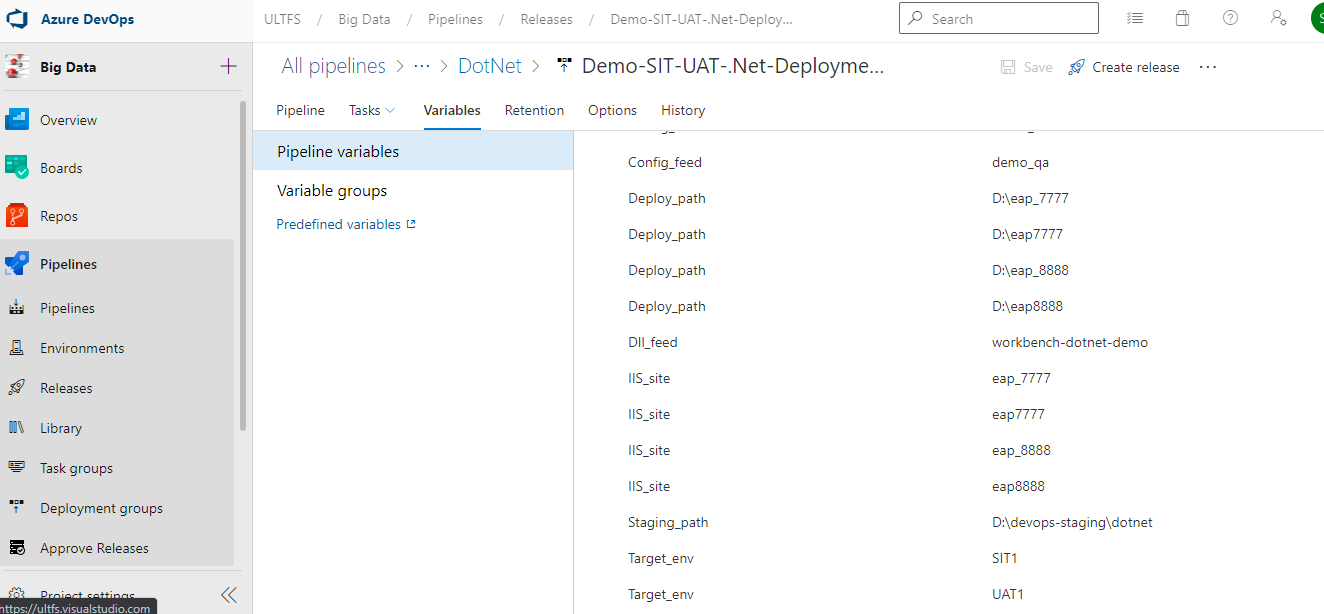


# **Troubleshooting**

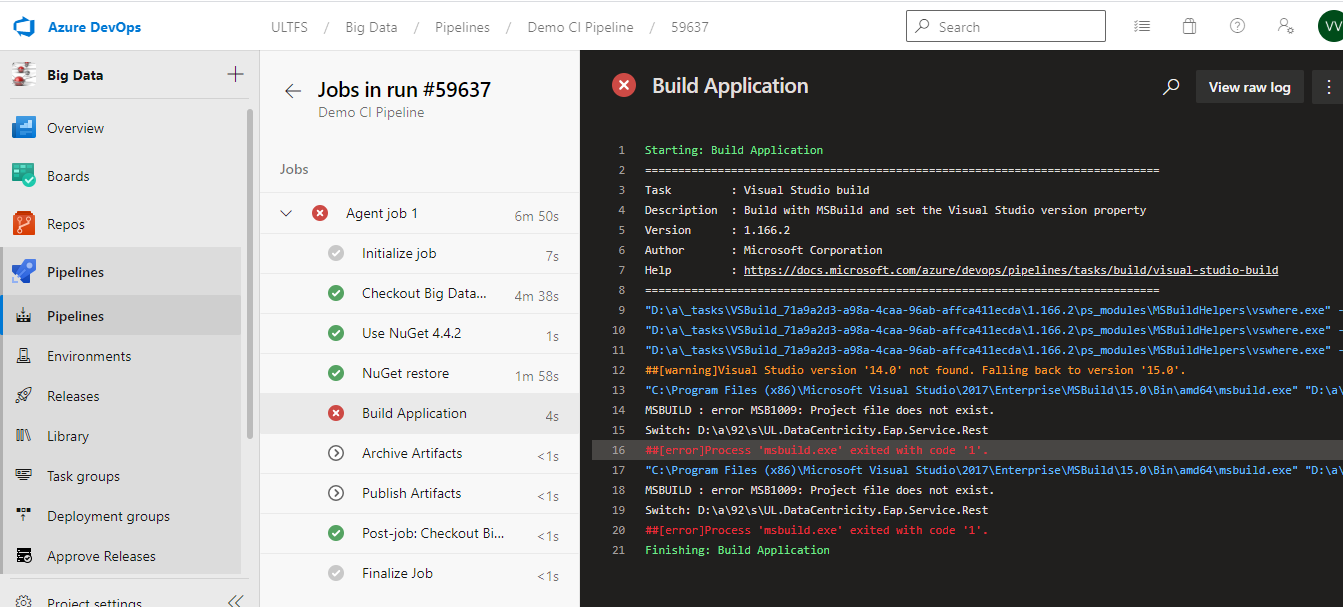
1. If the deployment is failed or isn’t starting, then check the targets online under Deployment groups.



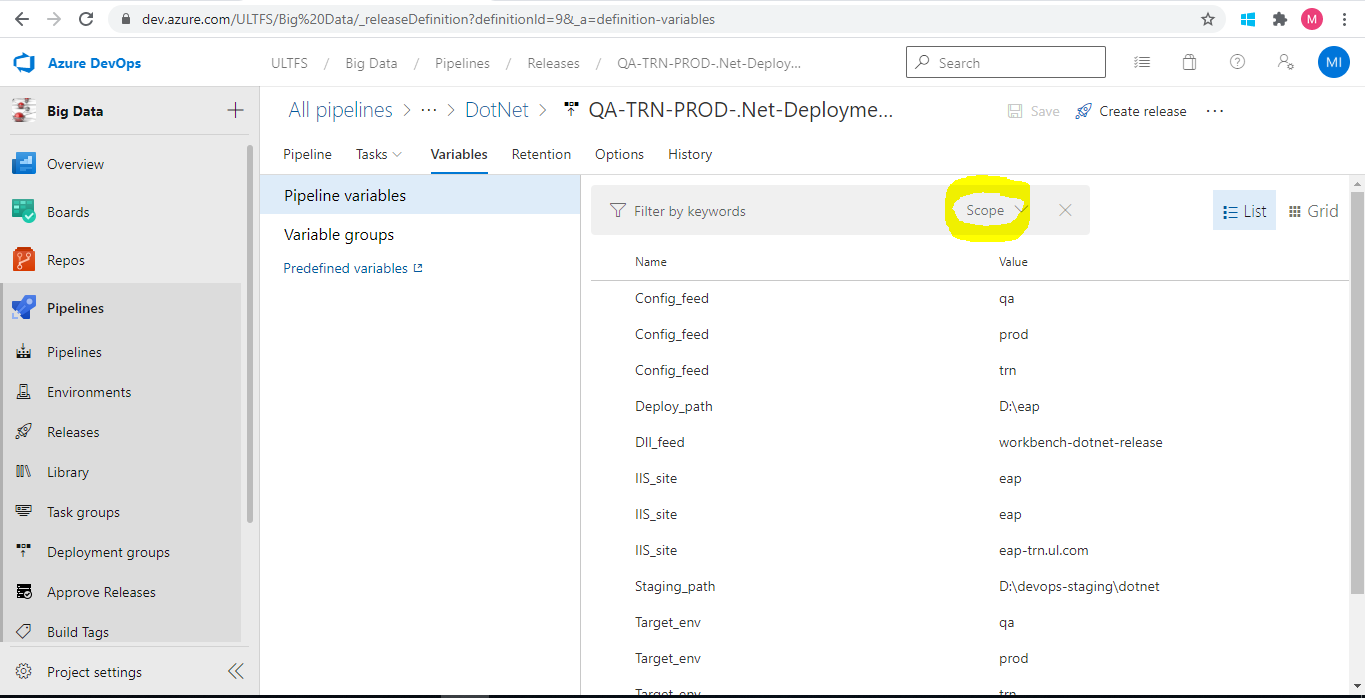
1. Make sure the variables are in sync with the Deployment environment. In below screen shot we can see variables configured for Demo environment. In case if there is any change in Deploy path or IIS site name, make sure to update the variables in Pipelines also.



1. In the Demo CI pipeline, make sure to provide the correct Branch Name. Otherwise the build fails as in below snapshot



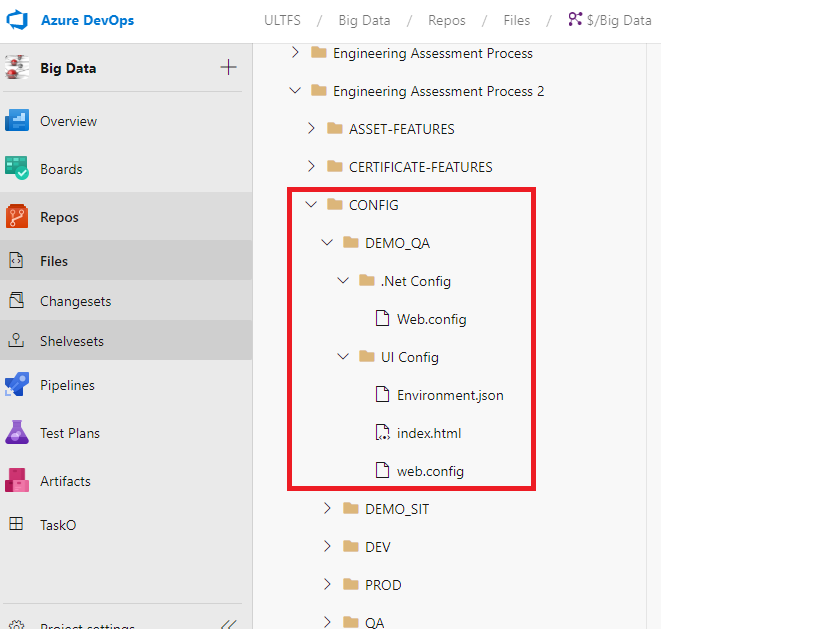
1. Make sure the variables are in sync with Scope as per the environment as shown below.



1. If a build/deployment fails, try to rerun the pipeline again. There can be connection issues. If it fails, go through the logs and check the error. Correct the values given accordingly.

For example:

1. If agent can’t be assigned/started, change the agent to Windows in the run pipeline configuration
2. If variable values are wrong, change the variable values accordingly and re-run the pipeline.
3. For deployment on server for Dev, QA, TRN or PROD environment (cannot add screenshot of script here due to security reasons, covered the same in demo session):
4. Go to corresponding Deployment group
5. Open the details
6. Copy the script
7. Execute on the target server by opening PowerShell as admin
8. Below is the folder structure for Config files in Source Code repository:



Under config folder, we are maintaining separate folder for each environment. Folder named with environment consists of folder for .Net and UI/Angular config files.

For addition of new or removal of existing config file, add or remove the same from Source Code repo (from corresponding environment and application folder). For removal of config file, remove the same manually from server also.