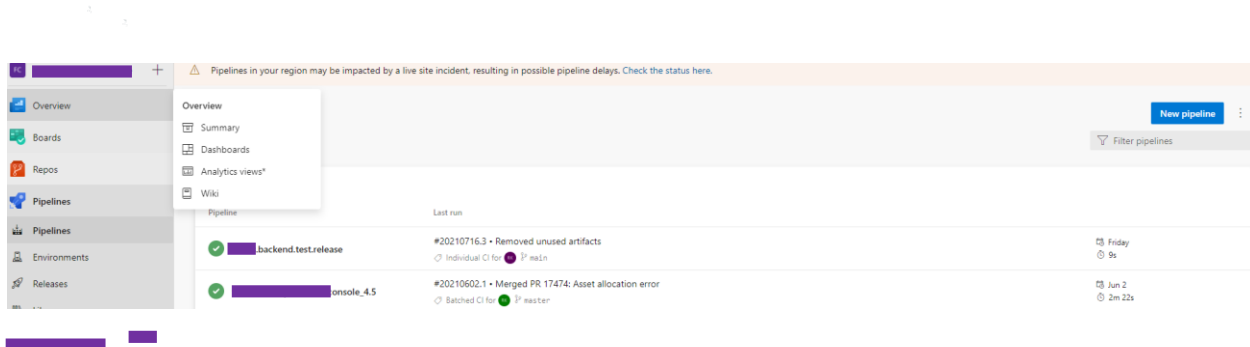


# Setup Azure Build Pipeline, Release Pipeline, and Stages

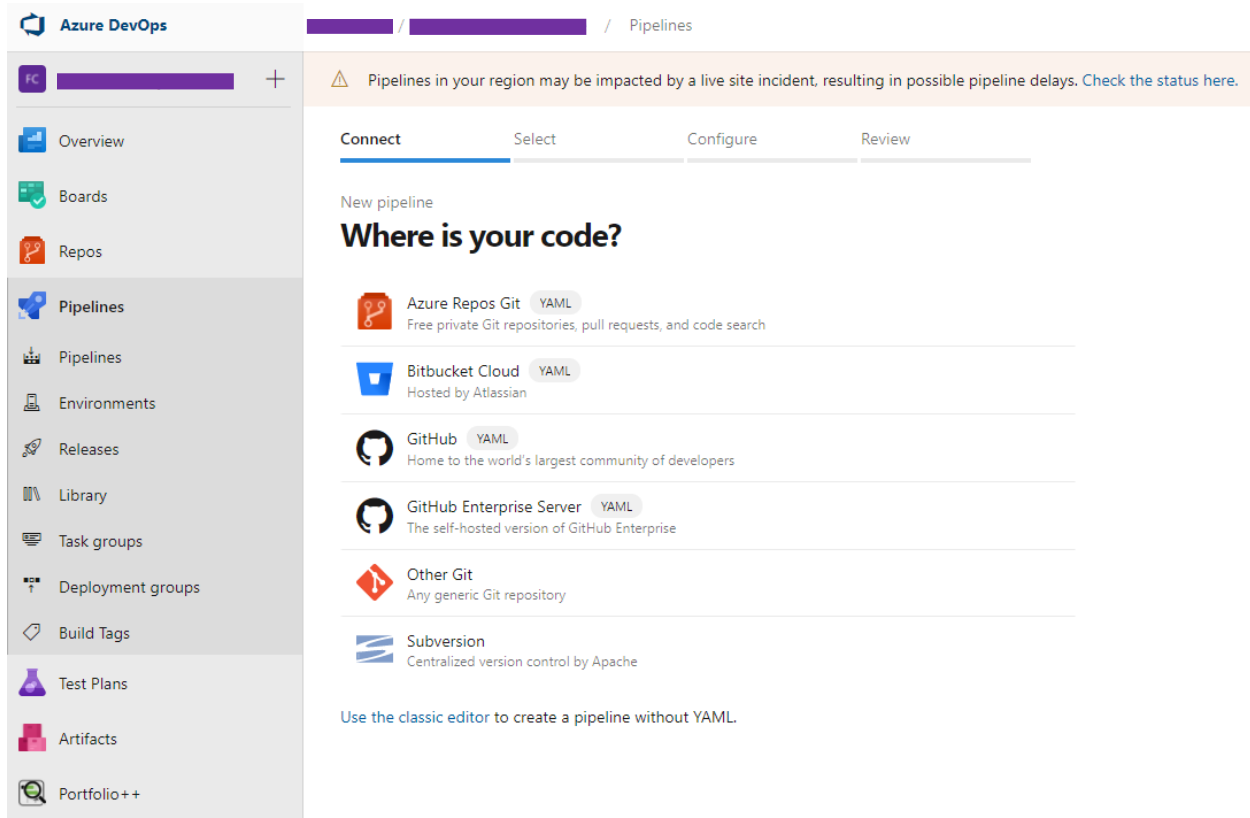
## Step 1: Create Build Pipeline.

Click on the Pipelines tab in order to create a new pipeline, Once, you invoke Pipelines, you can see a button **New pipeline** to create a new pipeline on Azure DevOps.




## Configure a newly created Pipeline.

- Select repository source e.g. Azure Repos Git.



- b. Select the repository name for the pipeline.

 Azure DevOps

FC [redacted] +

Overview

Boards

Repos

**Pipelines**

Pipelines

Environments

Releases

Library

Task groups

Deployment groups

Build Tags

Test Plans

Artifacts

Portfolio++

[redacted] / [redacted] / Pipelines

⚠ Pipelines in your region may be impacted by a live site incident, resulting in po:

✓ Connect

**Select**

Configure


Review


New pipeline


**Select a repository**


🔍 Filter by keywords


Funds Corres


 [redacted].backend.releases


 [redacted]


 [redacted]


 [redacted]

 [redacted]

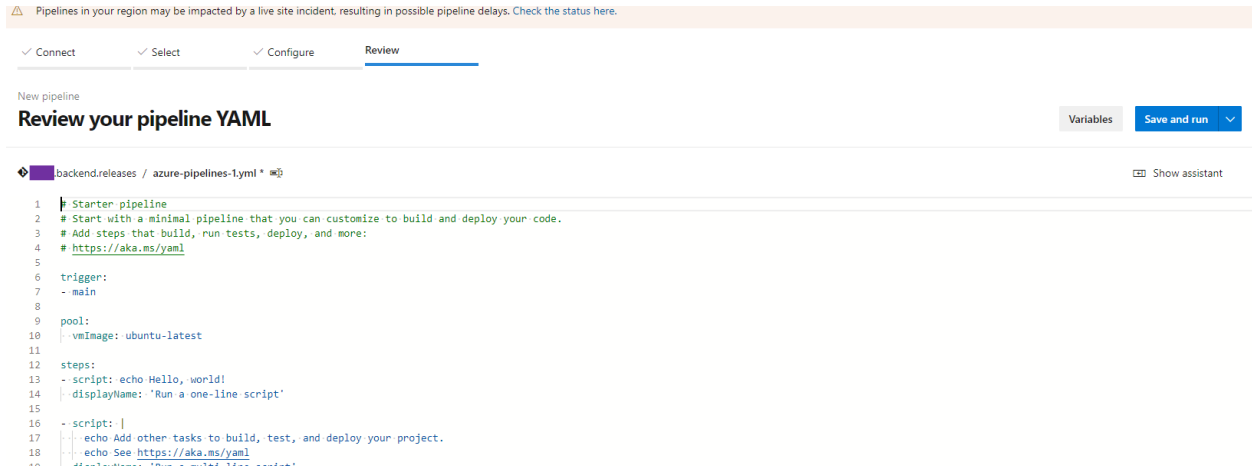
 [redacted]

 [redacted]

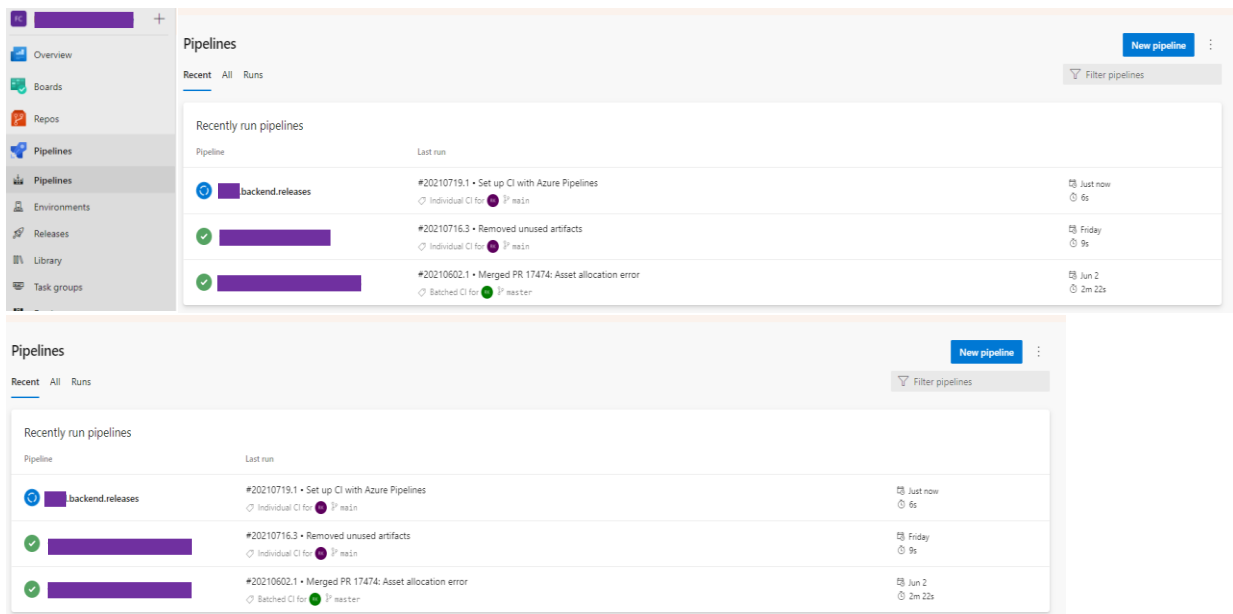
 [redacted]

 [redacted]

c. Run/Save and run the Pipeline in order to make it active.



Once the pipeline is created you can view it under the Pipelines tab. The below screen shows newly created pipeline is still executing.



## Step 2: Create a Release pipeline and Add an artefact.

Click on the + Add button under Artifacts. Select Azure Repos Git, the select project as repository **backend.releases**.

The screenshot shows the Azure DevOps Pipelines console. At the top, there's a warning banner about a live site incident. Below it, the breadcrumb is 'All pipelines > Backend Release QA and UAT'. The 'Pipeline' tab is selected. A message states: 'You cannot save a release pipeline that contains zero stages.' The main area has two sections: 'Artifacts | + Add' and 'Stages | + Add'. The 'Artifacts' section has a button 'Add an artifact' and a 'Schedule not set' icon. The 'Stages' section has a button '+ Add a stage'. On the right, the 'Add an artifact' dialog is open. It shows 'Source type' with options: Build, Azure Re... (selected), GitHub, TFVC, Azure Artifacts, GitHub Relea..., Azure Contai..., Docker Hub, and Jenkins. Below this, there are fields for 'Project \*' (with a dropdown), 'Source (repository) \*' (with a dropdown showing 'backend.releases'), and 'Default branch \*' (with a dropdown).

## Step 3: Add stage

Click on the + Add button under the Stages section. Once you click on Add button under the Stages section it asks you to select a template, and search Empty job, once you find it click on **Apply** button.

The screenshot shows the Azure DevOps Pipelines console. The breadcrumb is 'silica-net / Funds Correspondence / Pipelines / Releases'. A warning banner is at the top. The breadcrumb is 'All pipelines > New release pipeline (1)'. The 'Pipeline' tab is selected. The main area has two sections: 'Artifacts | + Add' and 'Stages | + Add'. The 'Artifacts' section has a button 'Add an artifact' and a 'Schedule not set' icon. The 'Stages' section has a button '+ Add a stage' and a 'Stage 1 Select a template' button. On the right, the 'Select a template' dialog is open. It shows a search bar with 'em' and a dropdown. Below the search bar, there's a section 'Others' with two options: 'Empty job' (with a button 'Apply') and 'Azure Policy Deployment' (with a description: 'Create and assign Azure Policy to your management group or subscription.').

### Step 3: Add a task for a stage.

Once you added a Stage, you can see a task link on the Stage box. Click on the task button. Once, you click on the task button, you can see the Agent job.

#### a. Configure Agent Job.

Click on + to add an agent, you can see the below screen to configure a specific agent pool for the Agent job.

The screenshot shows the configuration page for an Agent job. The left sidebar displays the project structure: 'All pipelines > Backend Release QA and UAT'. The main content area is titled 'Agent job' and includes a 'Remove' button. The configuration is organized into several sections:

- Display name \***: A text input field containing 'Agent job'.
- Agent selection ^**: A section with a link 'Agent pool | Pool information | Manage 12'. Below it is a dropdown menu showing 'QA\_RELEASE'.
- Demands ^**: A table with columns 'Name', 'Condition', and 'Value'. Below the table is a '+ Add' button.
- Execution plan ^**: A section with a 'Parallelism' dropdown set to 'None' (other options: 'Multi-configuration', 'Multi-agent').
- Timeout \***: A text input field with the value '0'.
- Job cancel timeout \***: A text input field with the value '1'.
- Artifact download ^**: A section with a dropdown menu showing 'backend.test.release' and 'Latest', and a 'Selected all artifacts' button.
- Additional options ^**: A section with a checkbox 'Allow scripts to access the OAuth token' (unchecked) and a 'Run this job' dropdown menu set to 'Only when a previous job has failed'.

b. Add first tasks to a Stage.

Once you added the agent job, you can create a + button to add a task, and search the command line, once you find the command line, hover the mouse on it, then get add button, and click on add button.

Add the following command to prepare the deployment script.

The screenshot shows the Jenkins pipeline configuration for a stage named 'I05D'. The 'Agent job' is set to 'Run on agent'. The 'Command Line Script' task is selected. The 'Script' field contains the following commands:

```
echo Write your commands here
echo "Extracting Release and Preparing for Deployment"
SET CUR_PATH=%CD%
echo CUR_PATH
set REL_PATH=%CUR_PATH%\backend.releases Pending_Releases_Deployment
echo %REL_PATH%
%REL_PATH%Common_Release_Extractor.bat $(username) $(password) $(DB_Name)
```

c. Add the second task, like the first task, click on the + button to add the second task, and search PowerShell, once you find it add it.

Add the below command in a script to execute the shell command.

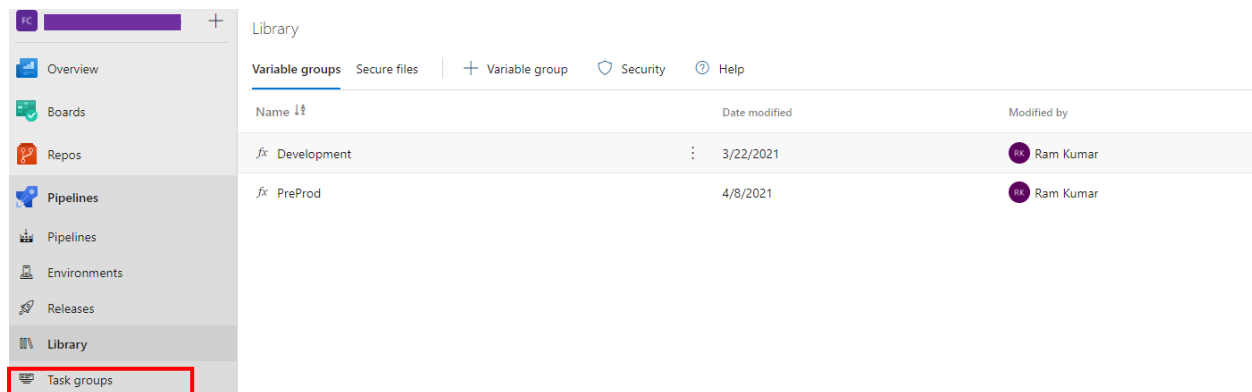
The screenshot shows the Jenkins pipeline configuration for the same stage 'I05D'. The 'PowerShell Script' task is selected. The 'Script' field contains the following commands:

```
Write-Host "***** Release deployment Started on $(DB_Name) *****"
Write-Host "*****"
cd .
cd $(DB_Name)
sh runall.sh $(username)/$(password)@$(DB_Name)
Write-Host "***** Release deployment Ended on $(DB_Name) *****"
Write-Host "*****"
```

Step 4: - Add variable group.

- a. Create a Variable group.

Click on Library (Shown in the red box in the below image) under Azure DevOps. Once you click on the Library tab you can see + Variable group, click on it in order to create a new group. The below screen shows that two variable groups **Development** and **Pre prod** are created.

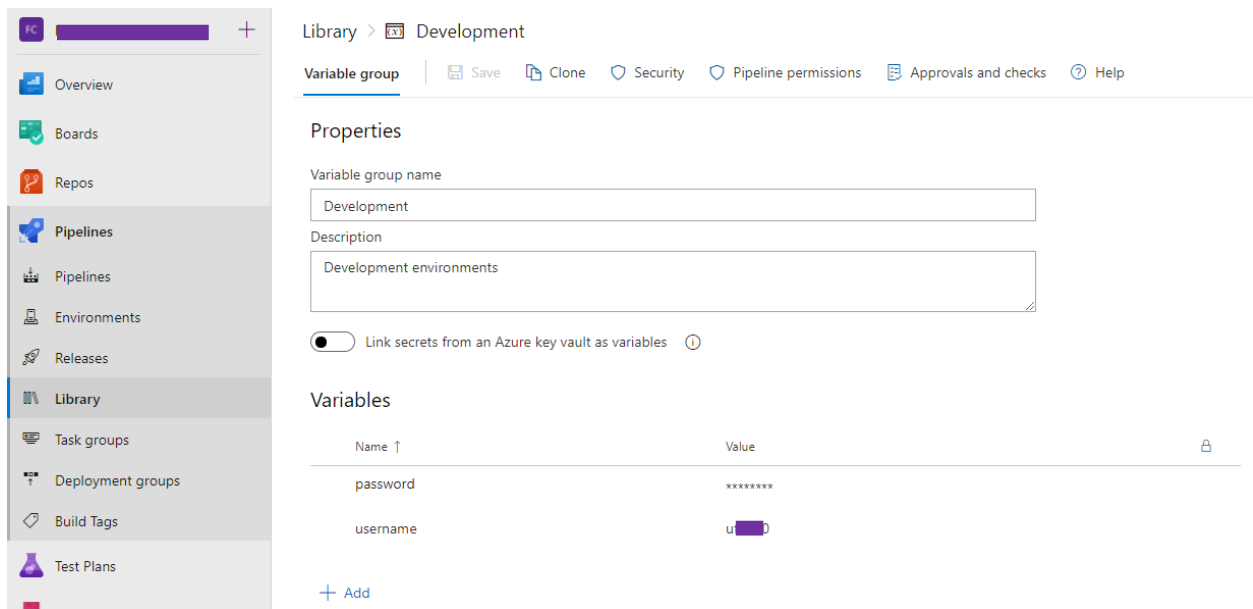


- b. Add variables in a variable group.

Once you added a variable group, click on the variable group name. Once you click on the variable group name you can see another tab to add a variable. To add a variable click on the + button under the Variable tab.

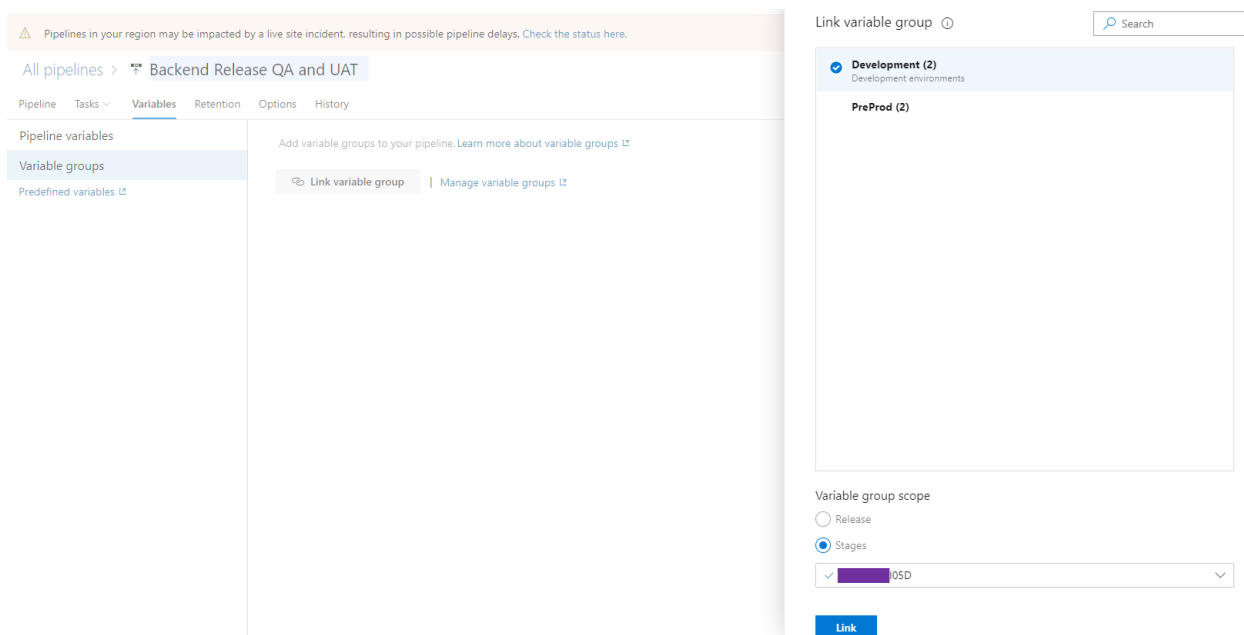
The below screenshot shows that username and password variables are added to Development variable group.

Note: for the password when you hover on value, you can get the lock option, if you click on the lock option password will be not visible. Once you have locked it you can't open it.



c. Link a variable group to a Stage.

Click on Variable tab of Pipeline, you can find an option to link a variable group to the Stage.



d. Pipeline Variable.

Pipeline variable can be created under Variables of the specific pipeline. Below screen shows that two pipeline variables created with same for with different scope. Scope represents



different Stages.

Backend Release QA and UAT > Release-11

Pipeline Variables History Refresh Edit ...

Pipeline variables

Variable groups

Predefined variables

Filter by keywords Scope

Name	Value	Scope
DB_Name	005D	005D
DB_Name	002Q	UAT 002Q

e. Once the Variable group is created, it can be seen under Pipeline's variables section.

All pipelines > Backend Release QA and UAT

Pipeline Tasks Variables Retention Options History

Pipeline variables

Variable groups

Predefined variables

Name	Value
<b>Development (2)</b> Development environments	Scopes: 005D
password	*****
username	ui 0

Link variable group | Manage variable groups

Step 5: - Auto **Azure DevOps release (Note. it is not backend release)** creation and manual deployment (It can include backend release) setting.

- a. Below setting can be done by clicking on the Continuous deployment trigger button (shown in the red box). **Enable** the option for Continuous deployment trigger. **Enable** the option for the Pull request trigger.

The screenshot displays the Azure DevOps pipeline configuration for 'Backend Release QA and UAT'. The pipeline is structured with two stages: 'Backend Release' and 'UAT'. The 'Backend Release' stage contains a task named 'backend.release', which has a red box highlighting the 'Continuous deployment trigger' button. The 'UAT' stage contains a task named 'UAT'. On the right side of the interface, a panel titled 'Continuous deployment trigger' and 'Pull request trigger' is visible. Both triggers are set to 'Enabled'. The 'Continuous deployment trigger' panel shows 'Type' as 'Branch' and 'Branch filters' as 'main'. The 'Pull request trigger' panel shows 'Target Branch Filters' as 'main'.

- b. Below setting can be done by clicking on Pre-deployment conditions (shown in red box).

Select the **Manual only** option under the Triggers section.

The image displays two screenshots of the Argo CD pipeline configuration interface for a pipeline named "Backend Release QA and UAT".

**Top Screenshot:** The "Stages" section shows two stages: "IO5D" (1 job, 1 task) and "UAT, IO5D" (1 job, 1 task). The "IO5D" stage is highlighted with a red box. The "Pre-deployment conditions" panel on the right shows the "Triggers" section with "Manual only" selected.

**Bottom Screenshot:** The "Stages" section shows the same two stages. The "Pre-deployment conditions" panel on the right shows the "Triggers" section with "Manual only" selected. The "Pre-deployment approvals" and "Gates" sections are both disabled.

- c. Below setting can be done by clicking on Pre-deployment conditions (shown in red box). Select the **Manual only** option under the Triggers section. For UAT and PROD environments select Pre-deployment approvals as **Enabled**.

⚠ Pipelines in your region may be impacted by a live site incident, resulting in possible pipeline delays. [Check the status here.](#)

All pipelines > Backend Release QA and UAT

Save Create release View releases ...

Pipeline Tasks Variables Retention Options History

Artifacts | + Add

backend.release

Schedule not set

Stages | + Add

105D  
1 job, 1 task

UAT 105D  
1 job, 1 task

Pre-deployment conditions

UAT\_SILICA\_SCI05D

Triggers  
Define the trigger that will start deployment to this stage

Pre-deployment approvals  
Select the users who can approve or reject deployments to this stage

Approvers  
Ram Kumar X Search users and groups for approvers

Timeout  
30 Days

Approval policies


- ☐ The user requesting a release or deployment should not approve it
- ☐ Revalidate identity of approver before completing the approval.
- ☐ Skip approval if the same approver approved the previous stage

Gates  
Define gates to evaluate before the deployment.

Deployment queue settings  
Define behavior when multiple releases are queued for deployment

Blow screenshot shows how to find trigger option of a build pipeline, in case of trigger override.

Continuous integration

 .backend.releases  
Enabled

Scheduled


+ Add

No builds scheduled

Build completion

+ Add

Build when another build completes

 .backend.releases

☒ Override the YAML continuous integration trigger from here

☐ Disable continuous integration

☒ Enable continuous integration


☐ Batch changes while a build is in progress

Branch filters

Type

Include

Branch specification

 main

Path filters

+ Add