Contents

[Version history 2](#_Toc65682576)

[How to install Azure DevOps ALM Octane extension from Marketplace 3](#_Toc65682577)

[How to add an API Access Key in Octane for Azure Service Connection 6](#_Toc65682578)

[How to create a new service connection 7](#_Toc65682579)

[Create a new pipeline with the Octane start and end tasks through YAML editing (implicit Azure job) 11](#_Toc65682580)

[Create a new pipeline with the Octane start and end tasks through classic editor (implicit Azure job) 19](#_Toc65682581)

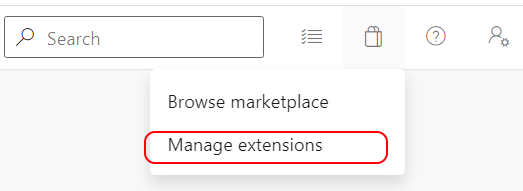
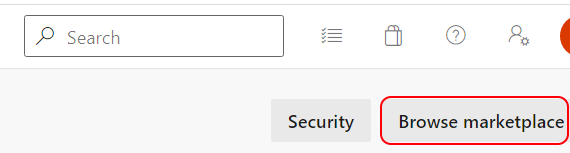
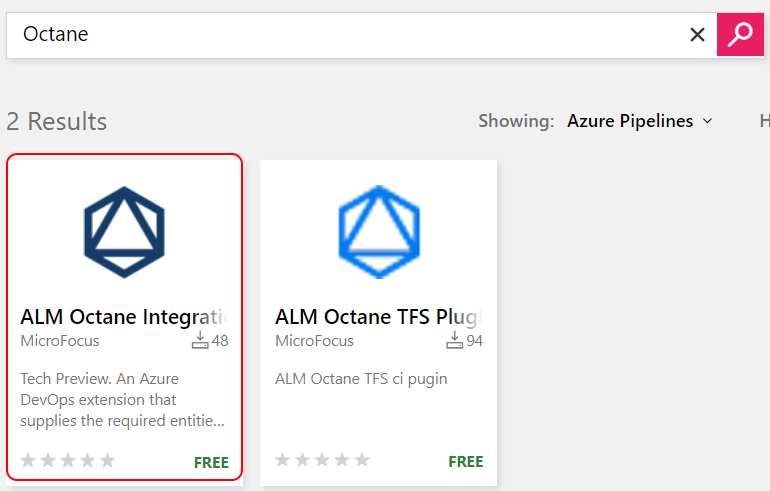
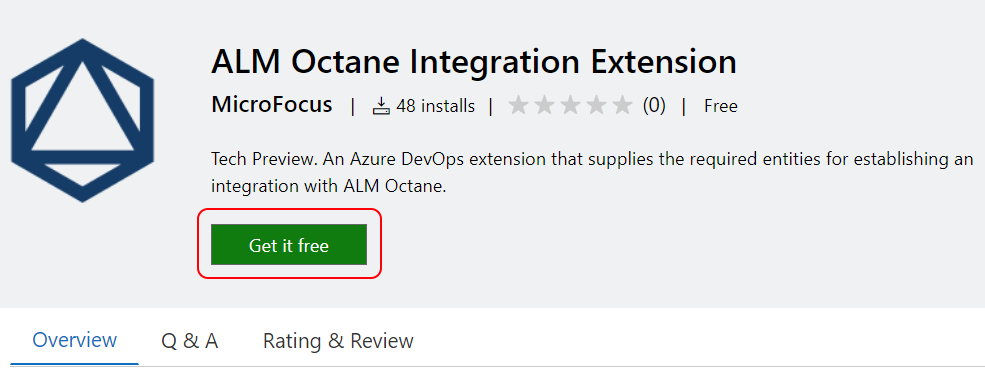
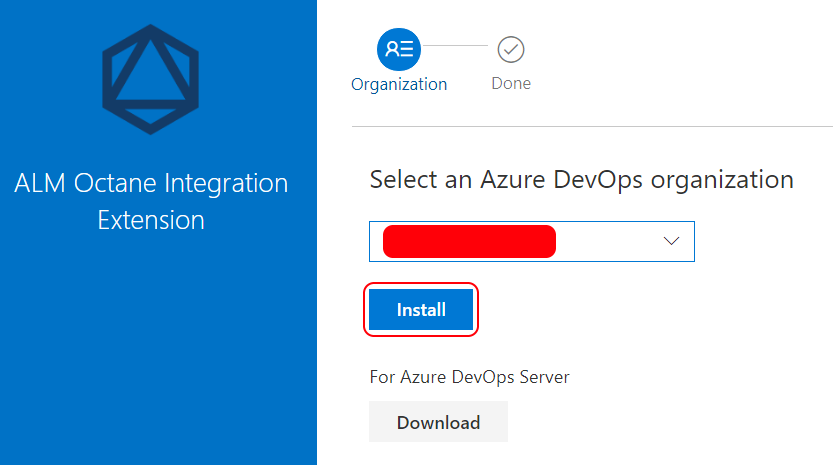
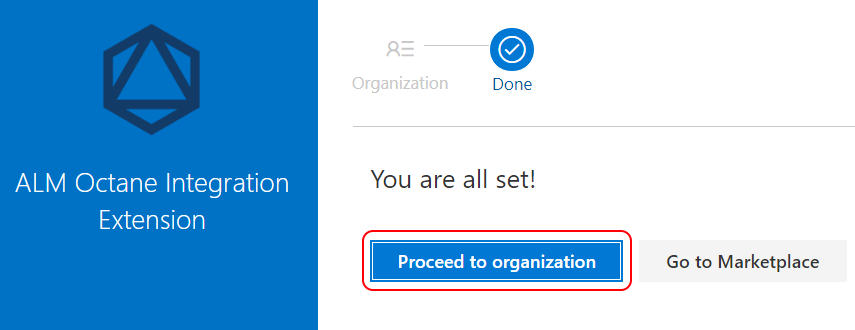
[Create a new pipeline with the Octane start and end tasks through YAML editing (explicit Azure jobs) 28](#_Toc65682582)

[Known issues and limitations 32](#_Toc65682583)

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Change |
| 19/02/2021 | 1.0.0 | Andrei Busmach\*u | Initial version of the document. |
| 03/03/2021 | 1.1.0 | Andrei Busmach\*u | Added YAML chapter on the creation of a pipeline with explicit/independent Azure Jobs for the Octane Start Task and Octane End Task. Added additional limitations and issues. |

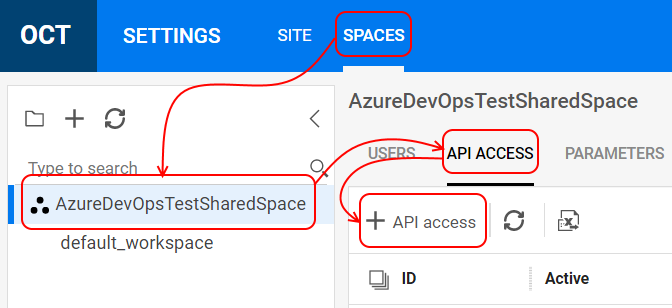
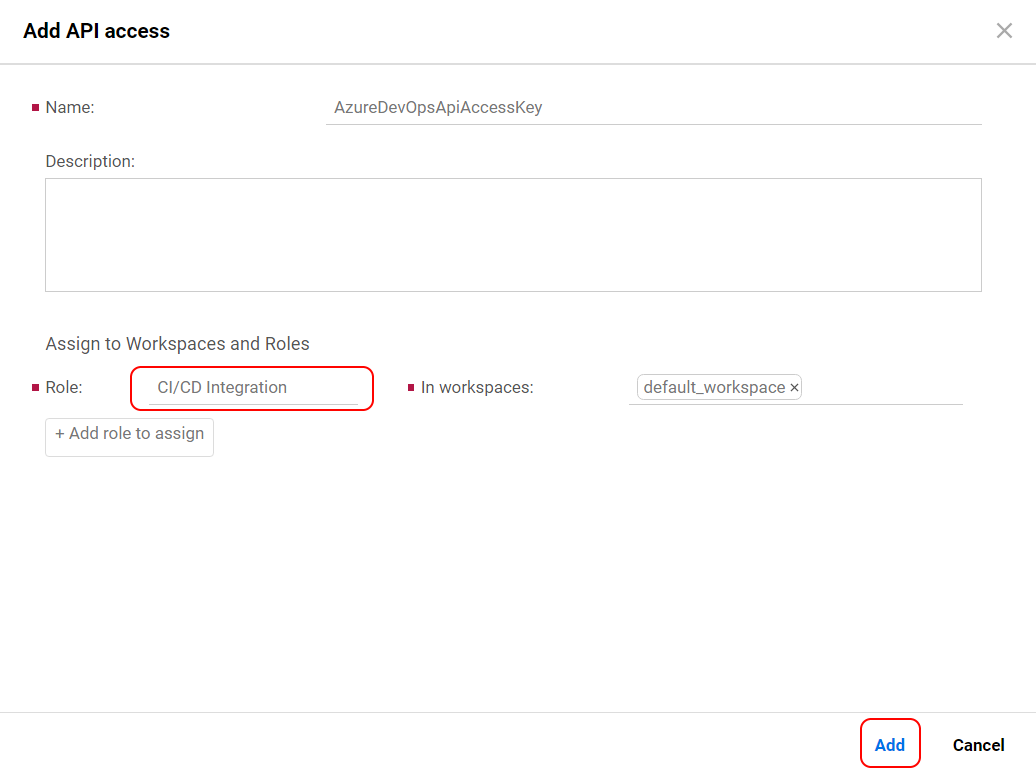
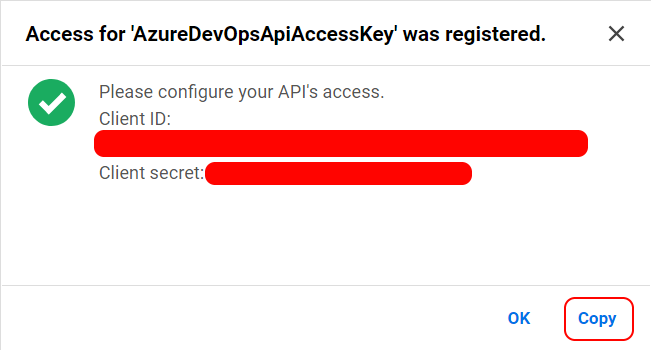
# Version history

# How to install Azure DevOps ALM Octane extension from Marketplace

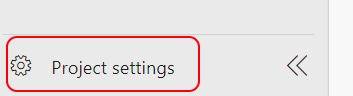
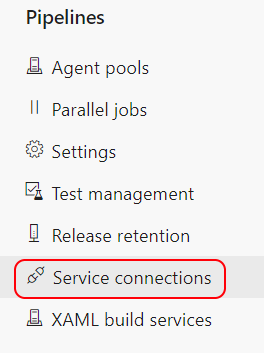
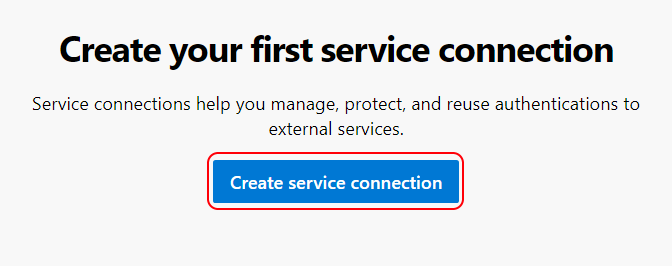
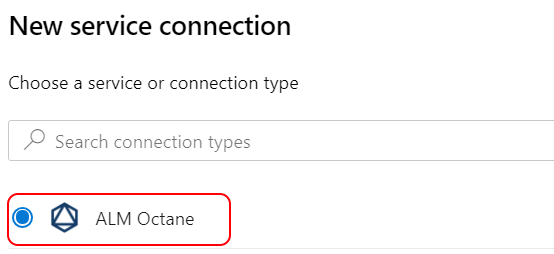
1. Click on “Manage extensions”:
2. Click on “Browse marketplace”:
3. Search for “Octane” and filter by “Azure Pipelines”: 
4. Click on “Get it free”: 
5. Choose the Azure DevOps organization and press “Install”: 
6. Go back to your organization:

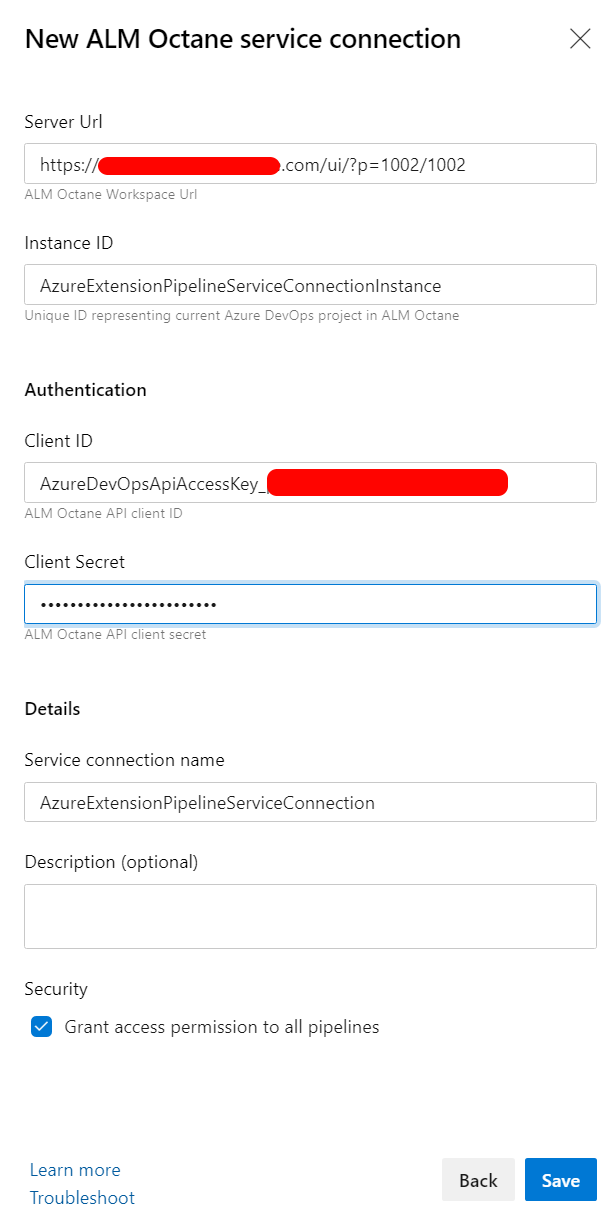
# How to add an API Access Key in Octane for Azure Service Connection

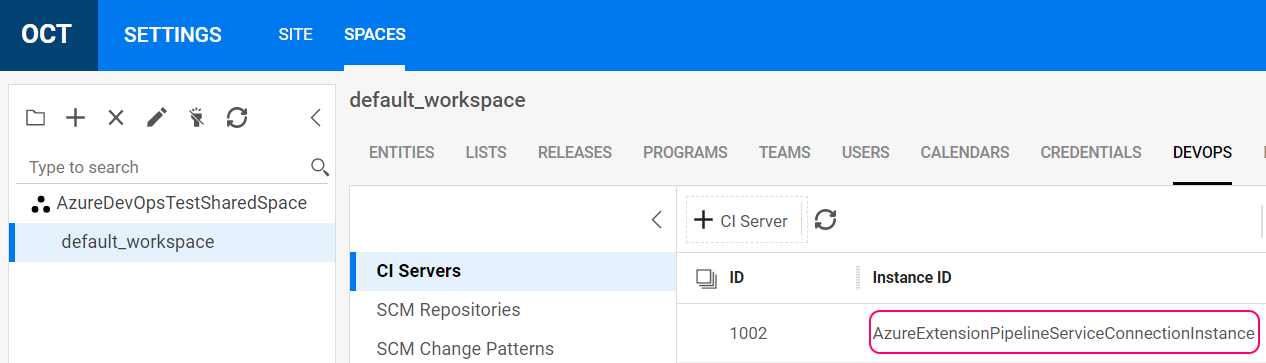
Before you can add a new service connection, please make sure you have a valid API ACCESS key and secret set in Octane. You can following the steps below to create one:

1. Go to the space and in API ACCESS tab press the API access button
2. Fill in all required data, like name, and don’t forget to select the role for CI/CD Integration in the required workspace.
3. In the displayed popup, press copy and save your newly created API Access. 

# How to create a new service connection

1. Go to “Project settings”:   
   
2. Click on “Service connections” available in the “Pipelines”: 
3. Press the “Create service connection” button: 
4. Select the ALM Octane connection type: 
5. Fill in all fields required in the form which appears, as follows:

  
  
**Server Url** - the URL of the Octane the service connection will point to. Make sure to include the sharedspace/workspace query parameter (p=1002/1002)

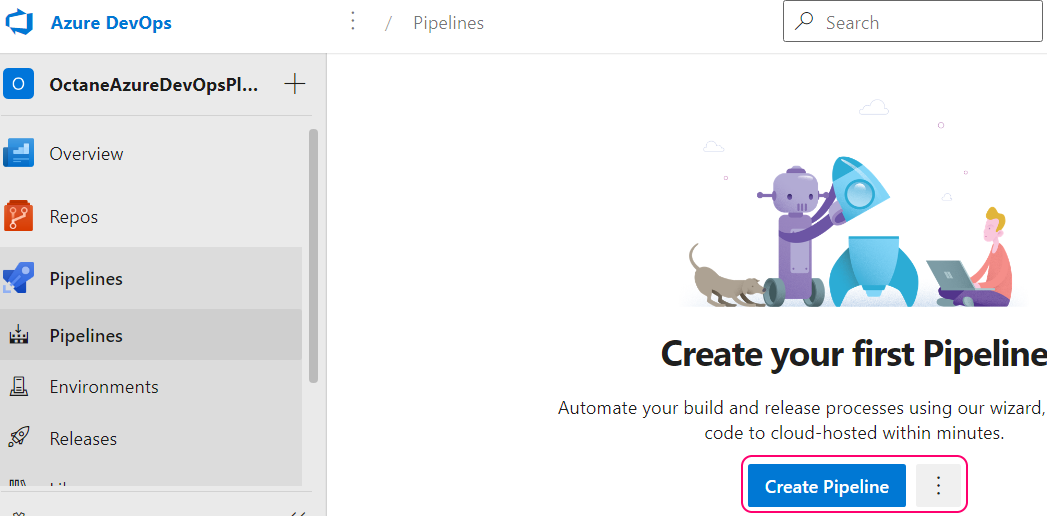
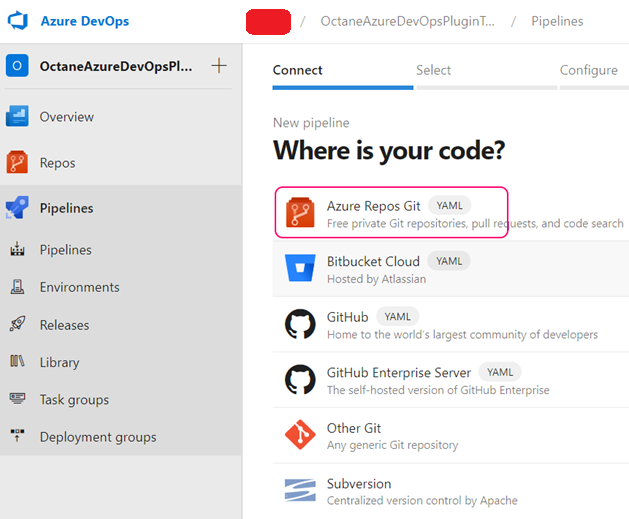
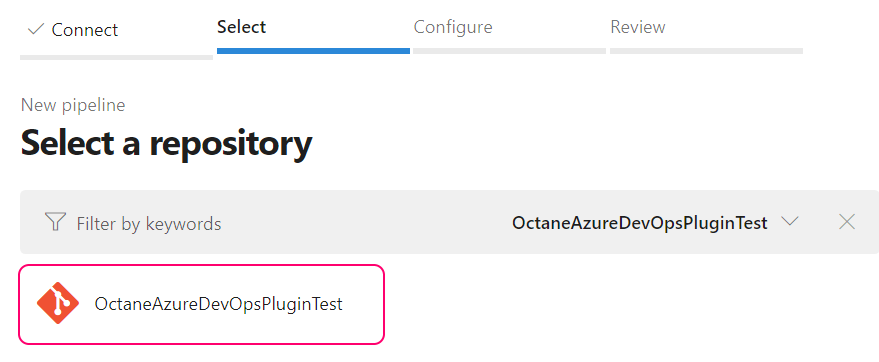
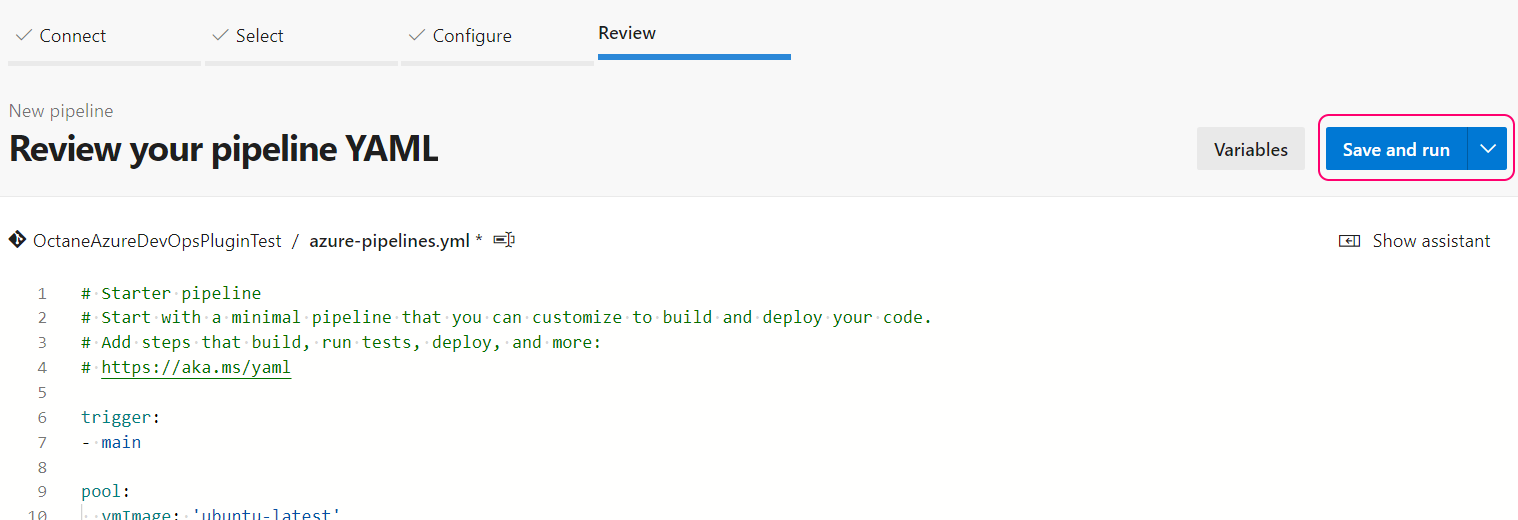
**Instance ID** - the name you expect to see in CI Servers grid in DEVOPS tab for a specific space, like in the example: 

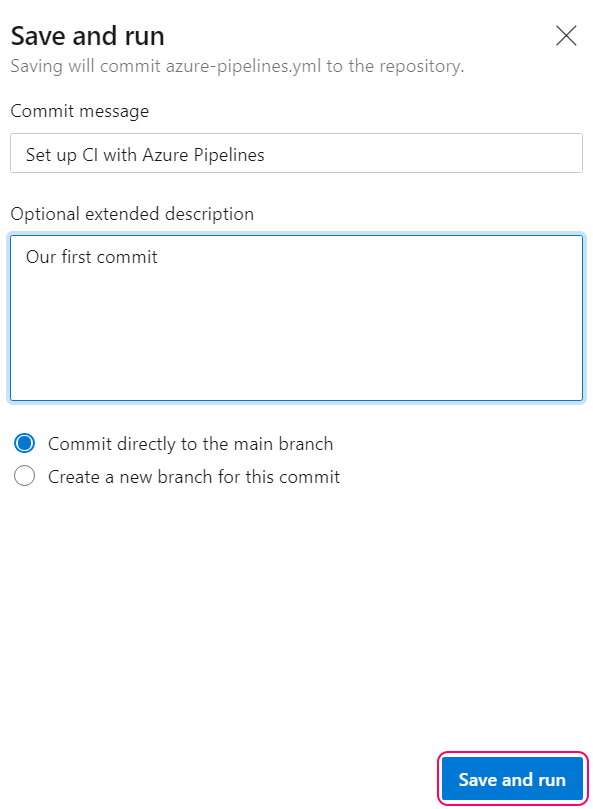
**Authentication** - The API Access key and secret which are created inside the sharedspace, the creation of which was presented earlier in the guide.

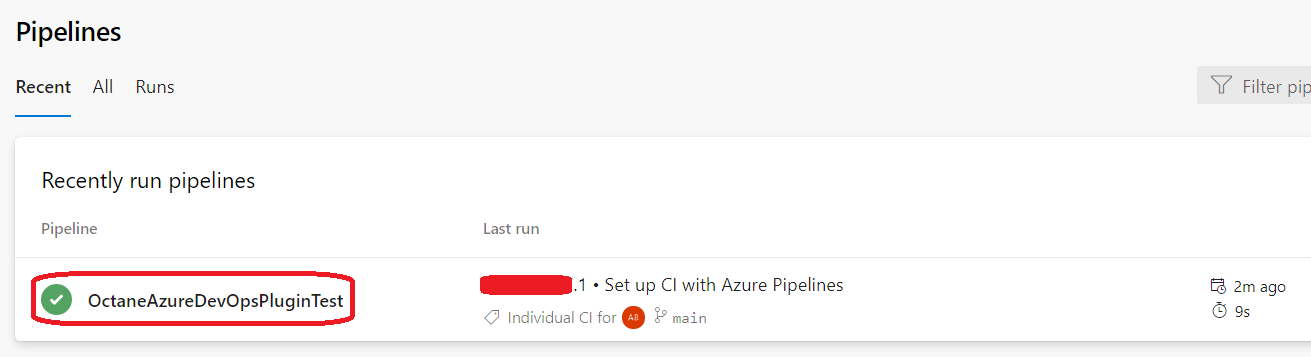
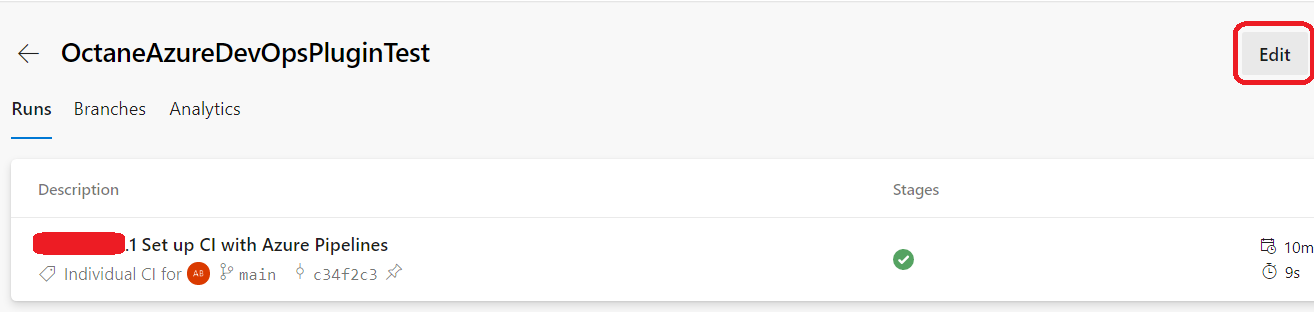
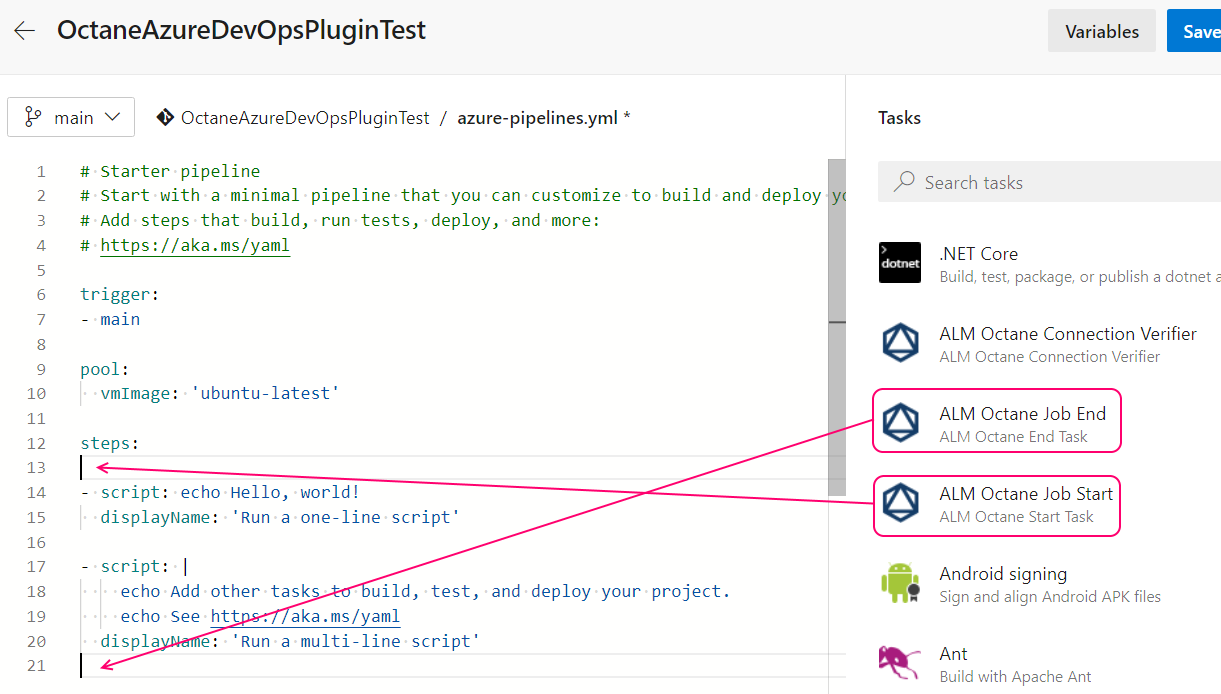
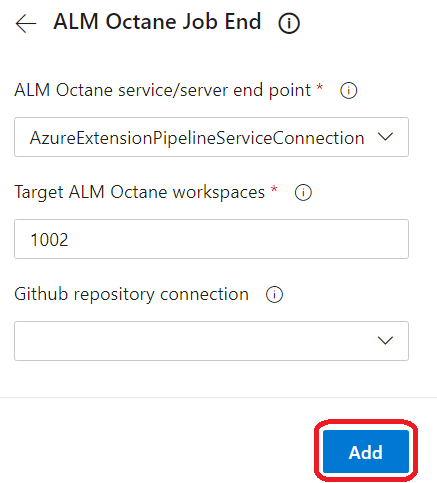
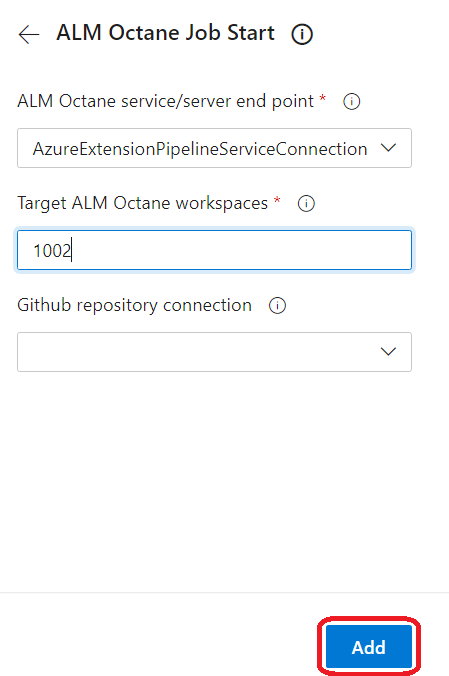
**Service connection name** - The name to be used anywhere inside Azure DevOps to reference this service connection.

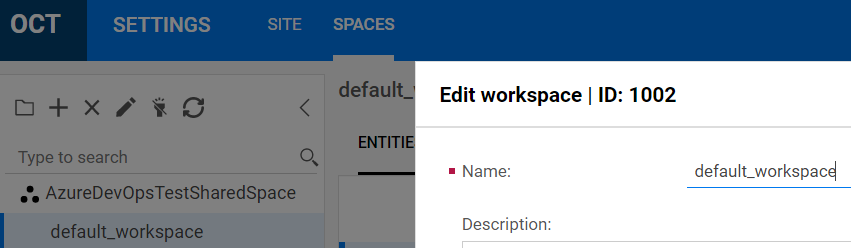
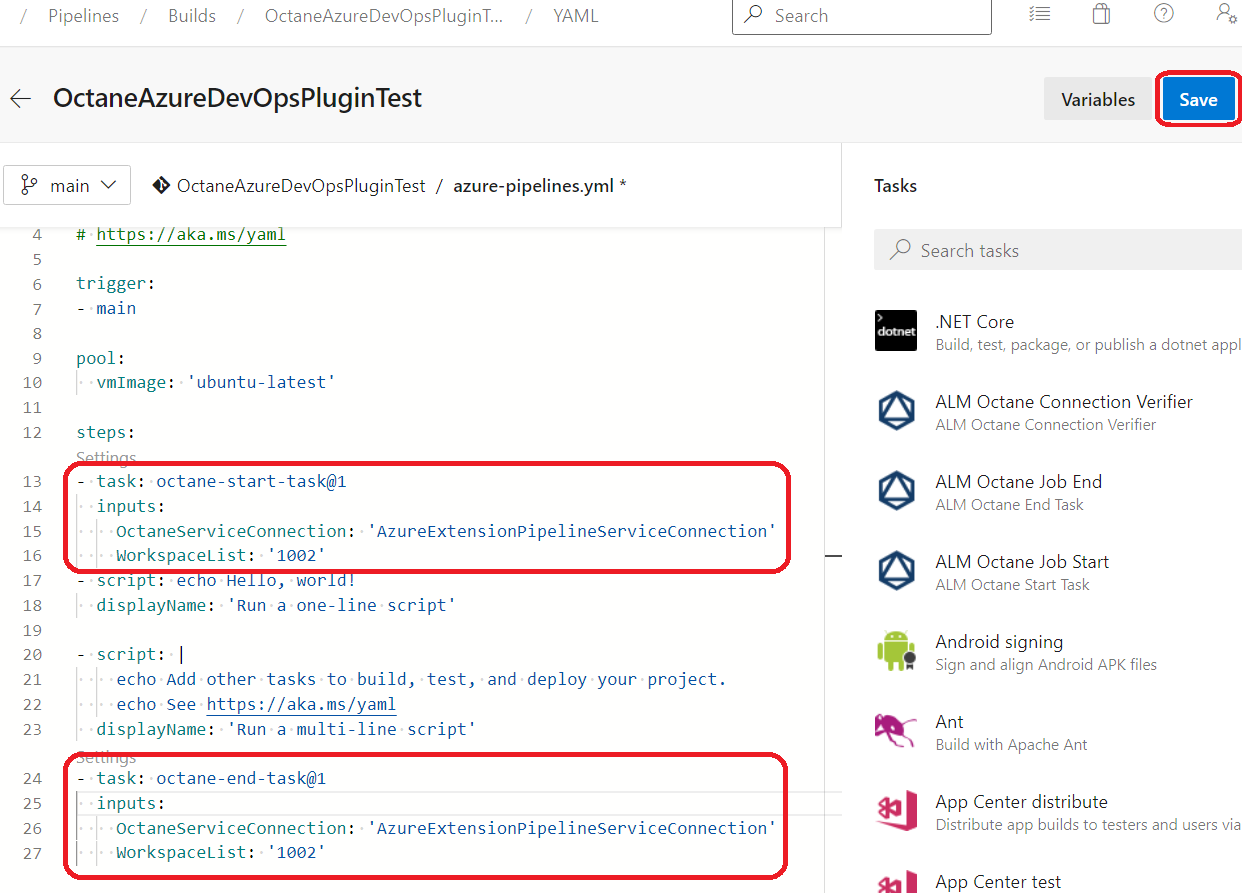
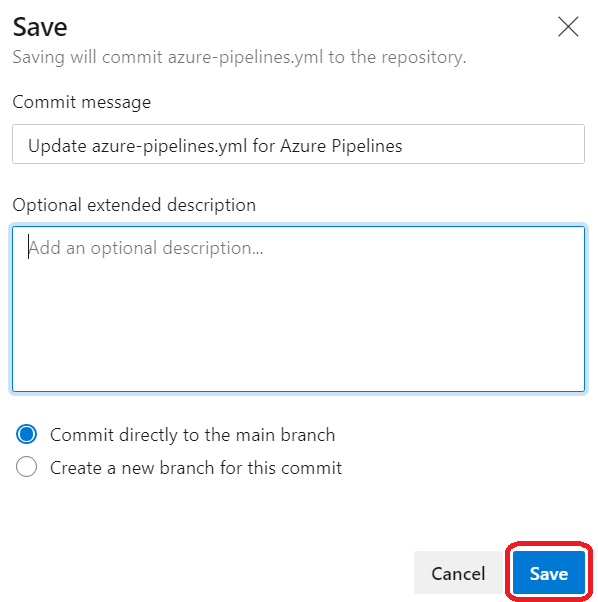
**Grant access permission to all pipelines** - Selecting this checkbox will make this service connection available for all pipelines for usage.

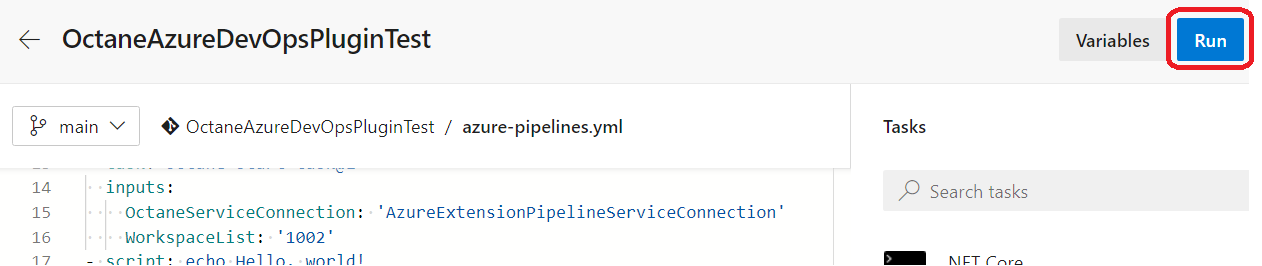
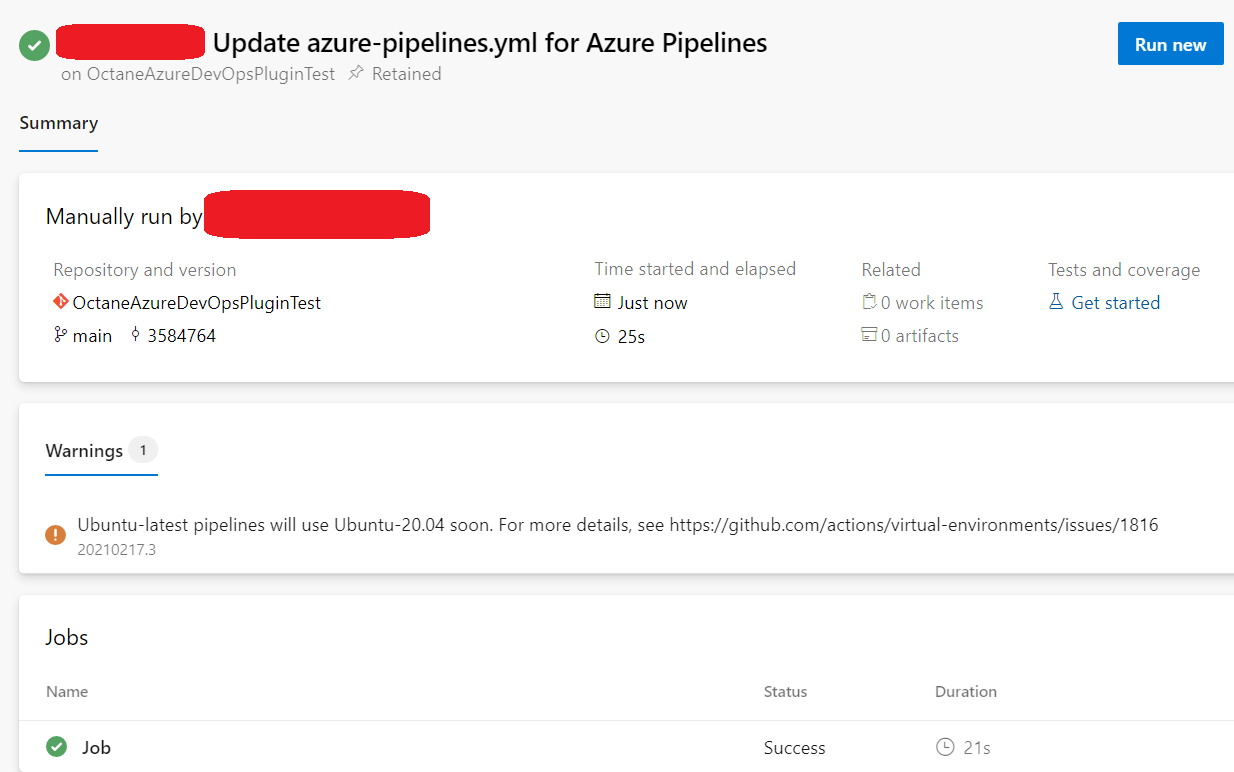
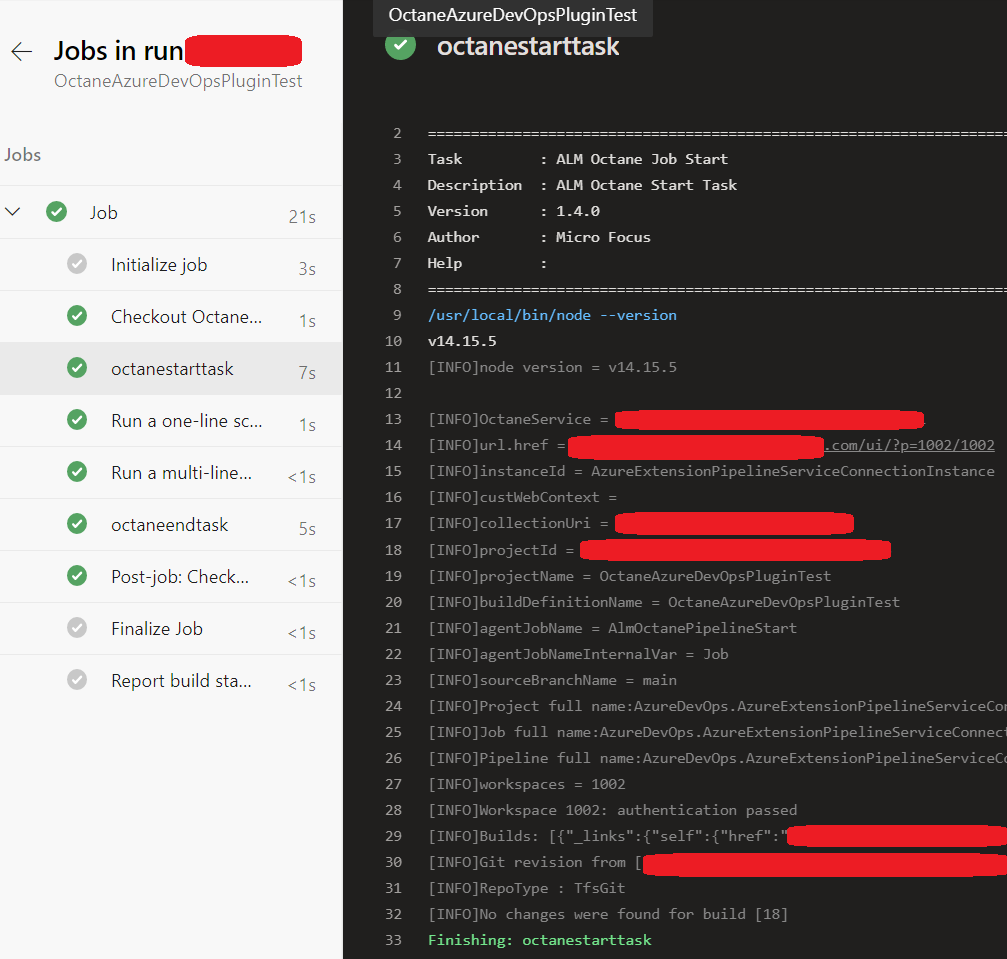
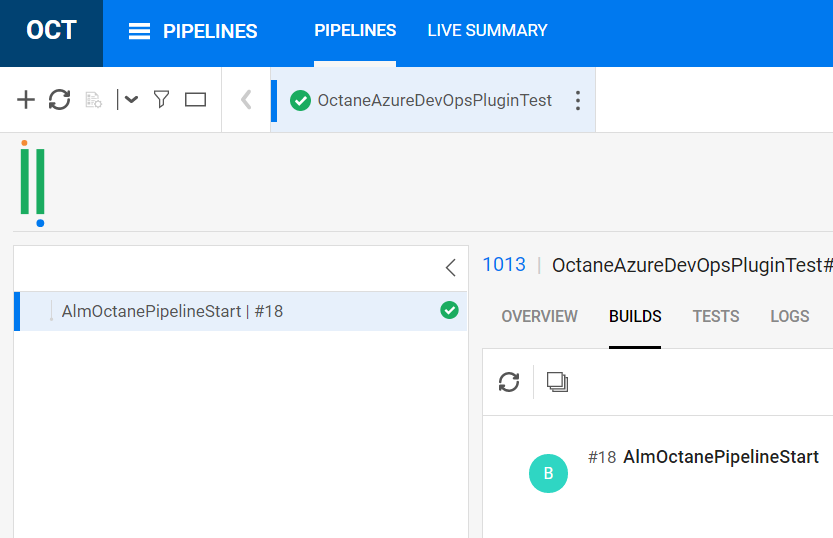
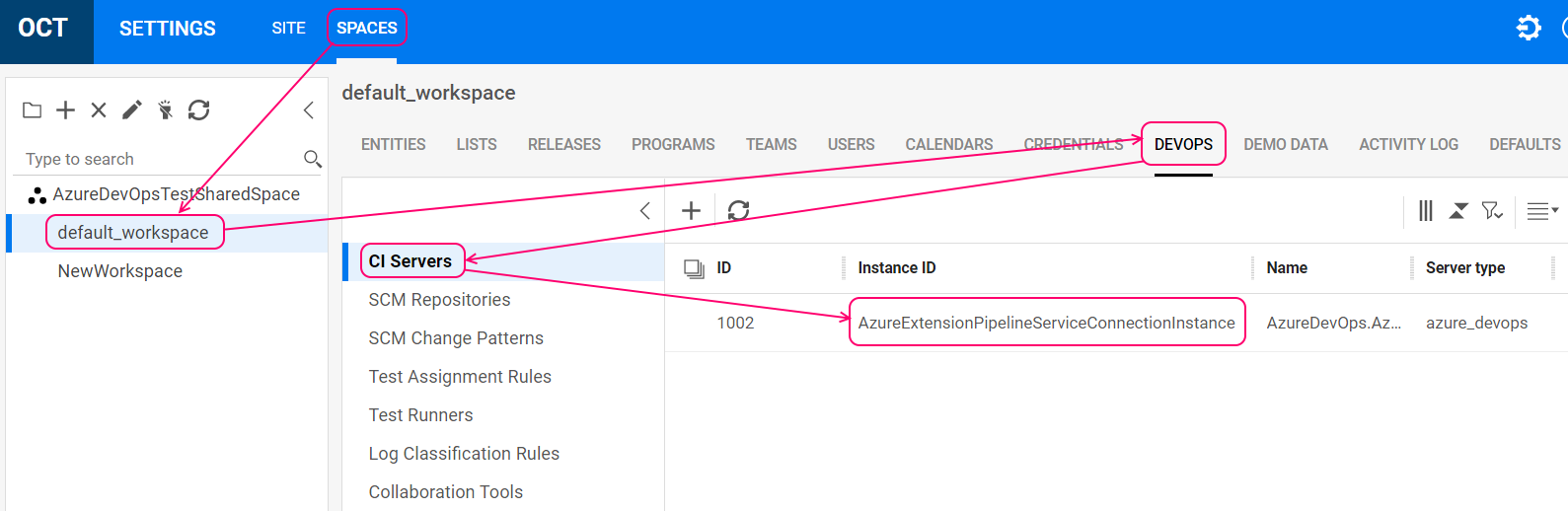
# Create a new pipeline with the Octane start and end tasks through YAML editing (implicit Azure job)

1. Go to Pipelines and press Create Pipeline:
2. Select source code repository type you use, in this case we will use the local Azure Repos Git:   
   Note that depending on the repository type you select, there might be different steps to execute!
3. Select the actual repository: 
4. Edit your YML if you require, and press Save and Run: 

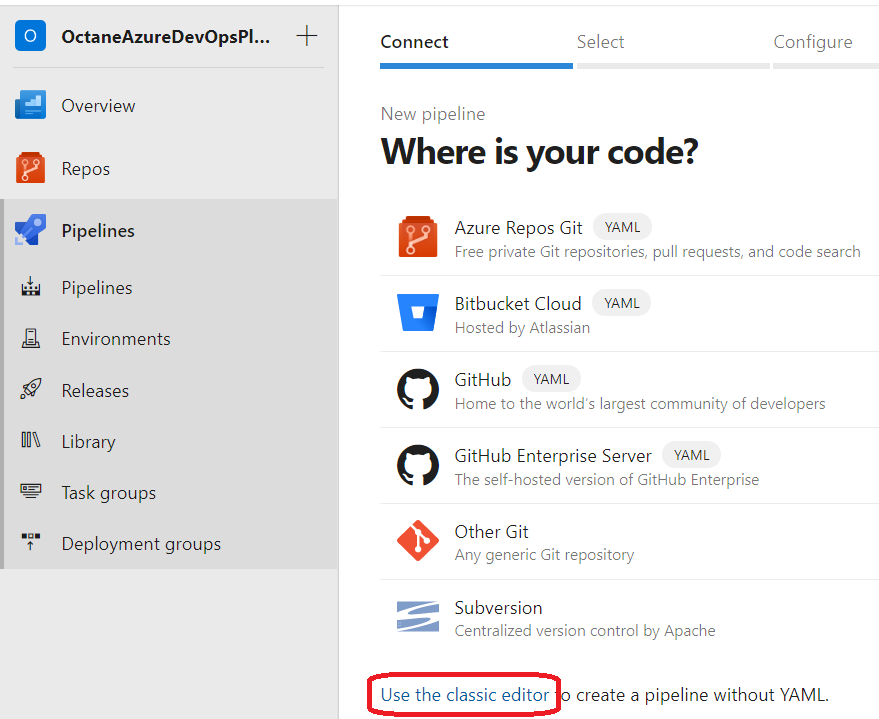
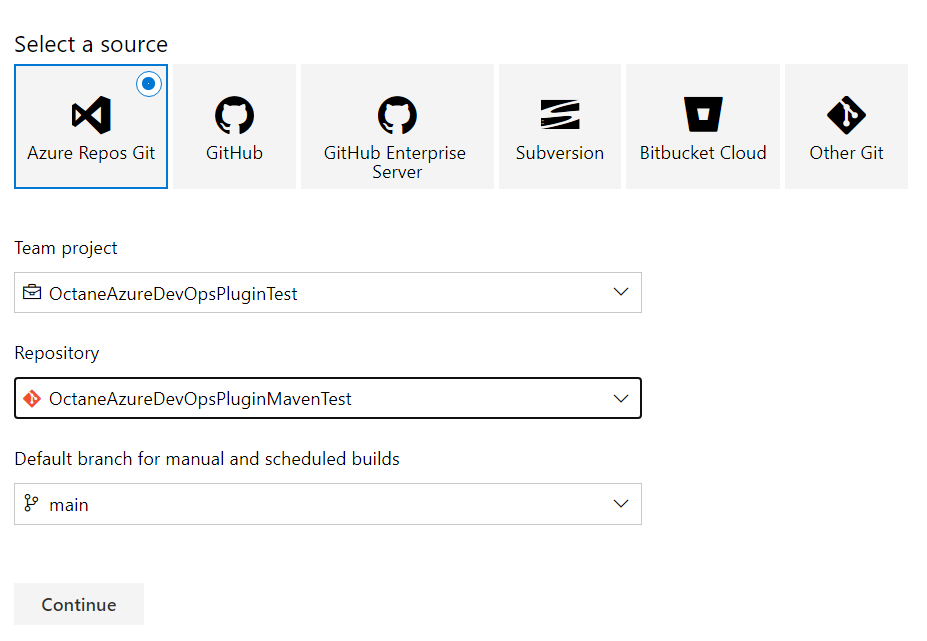
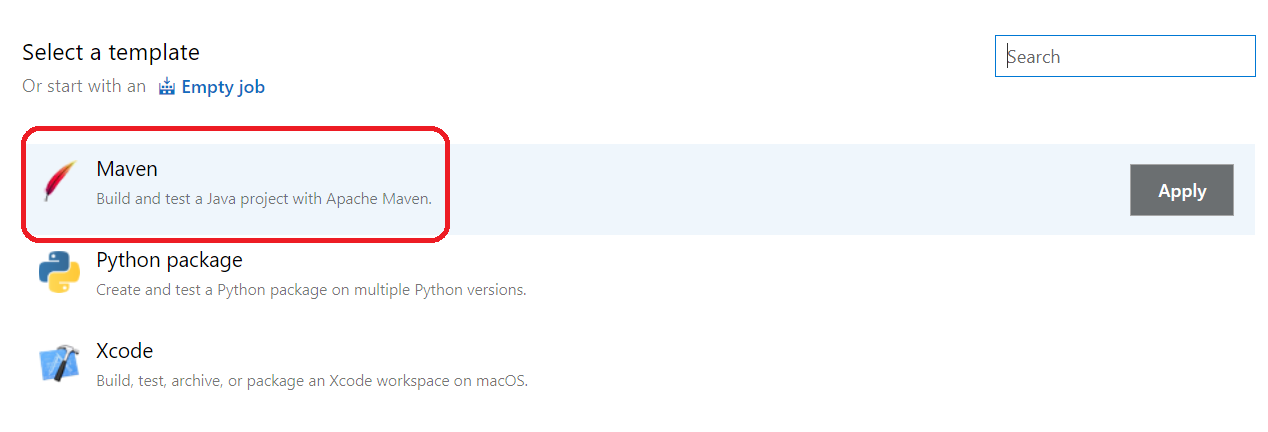
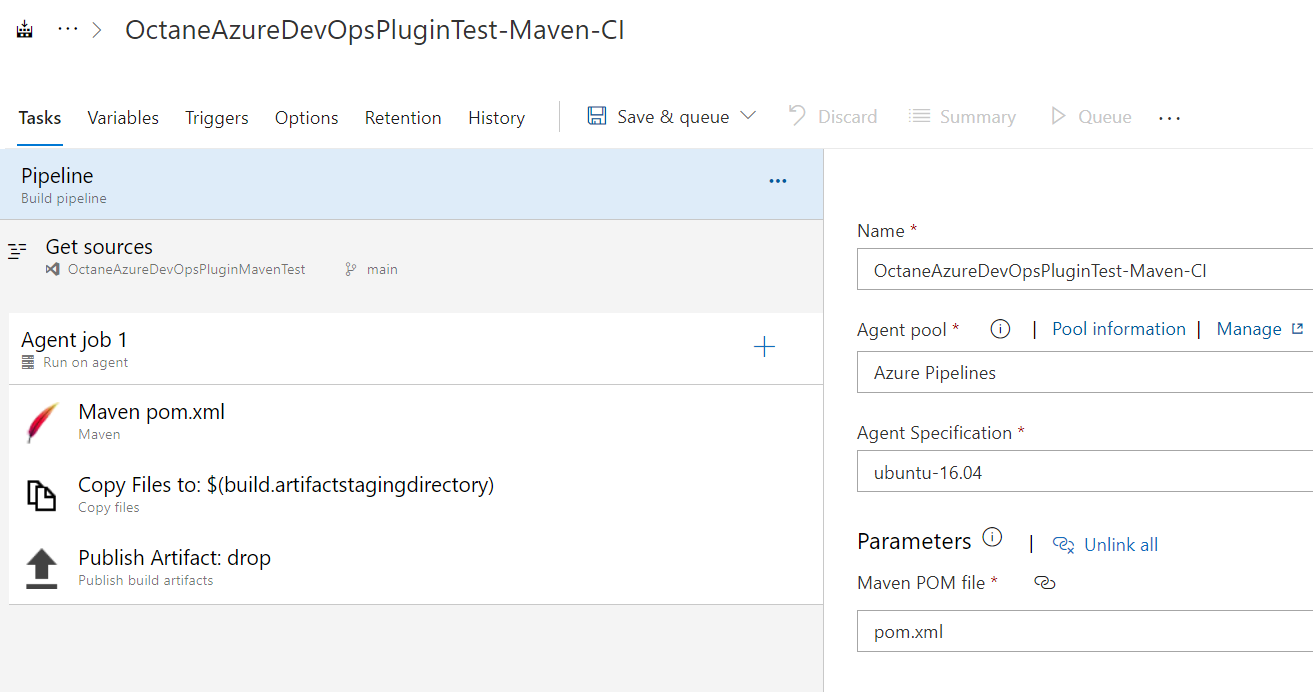
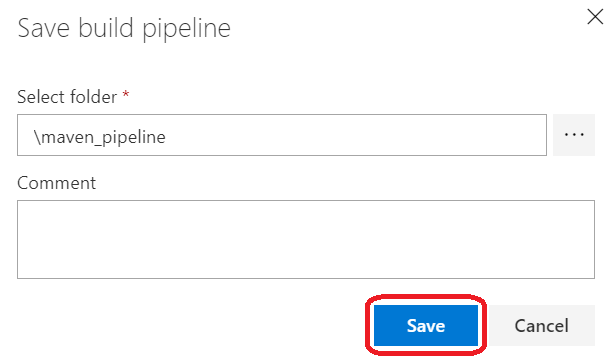
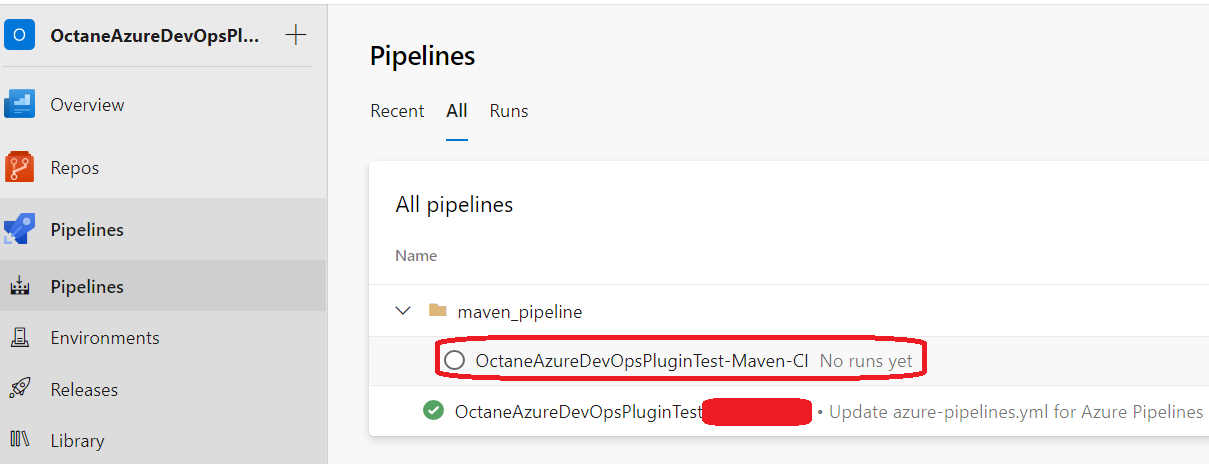
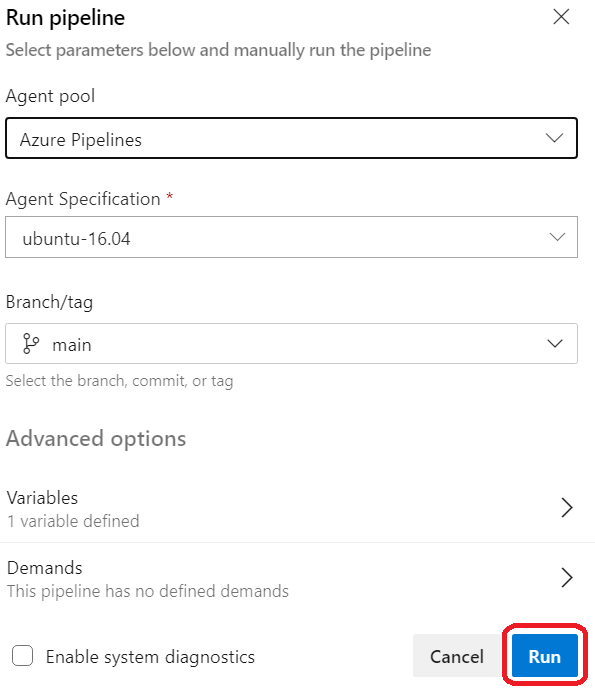
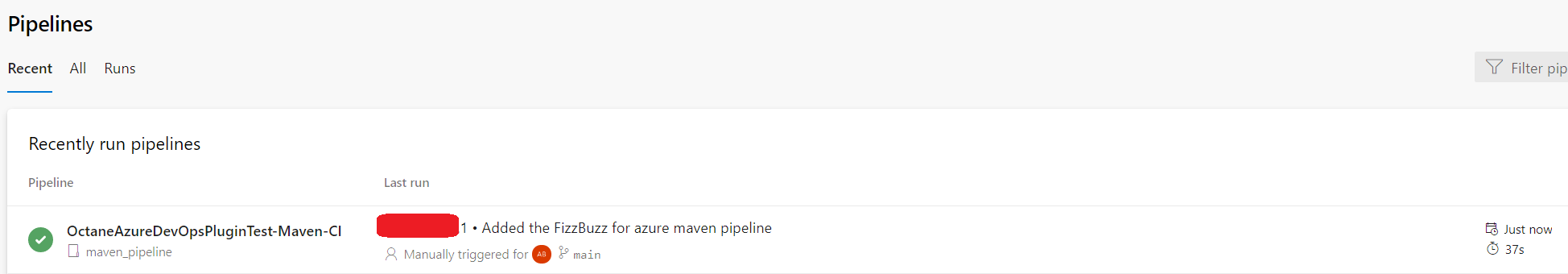
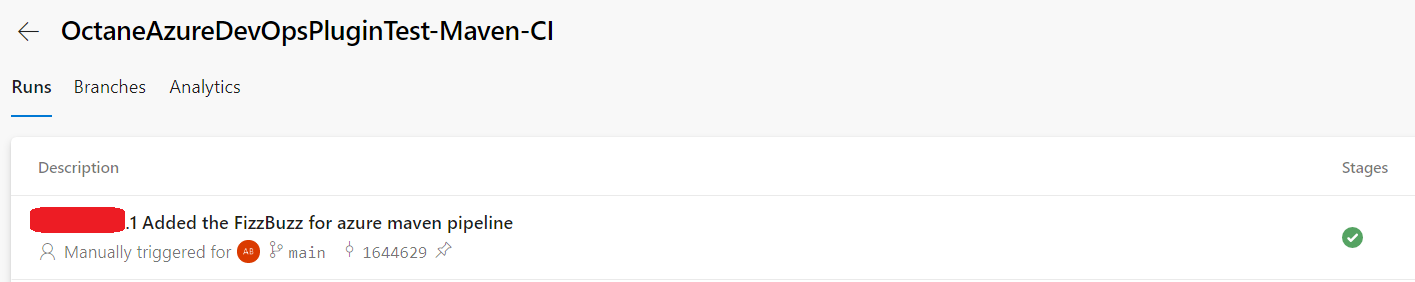
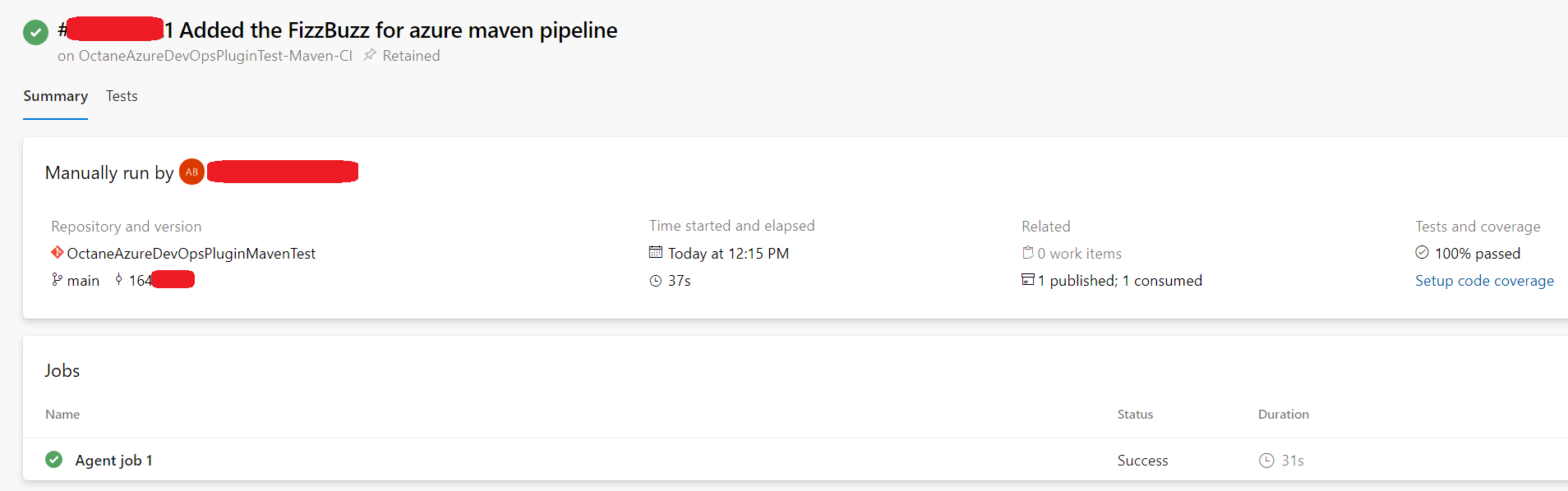
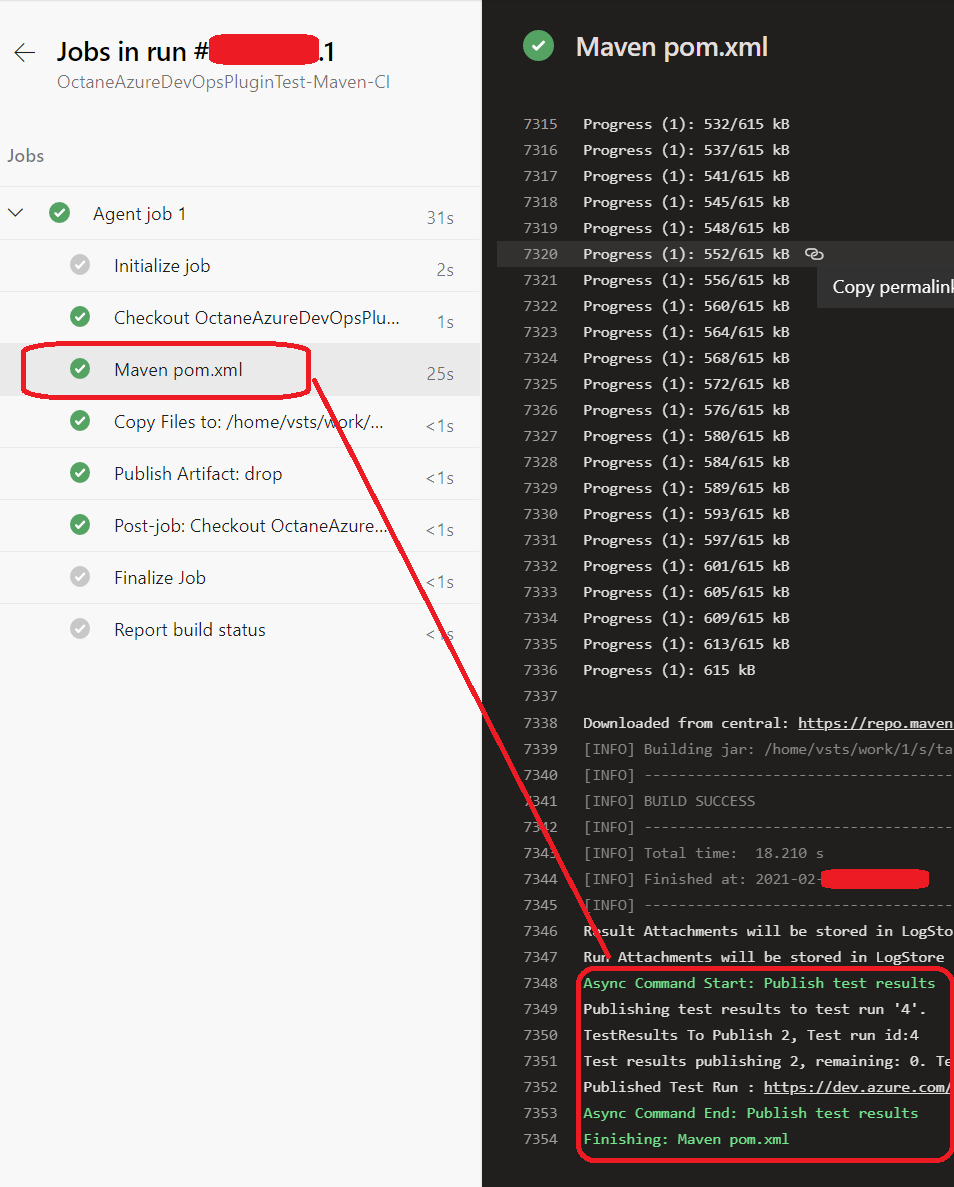
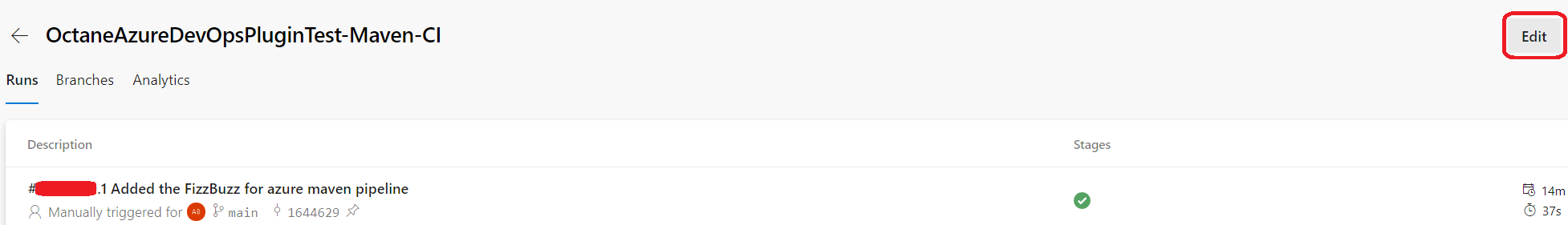
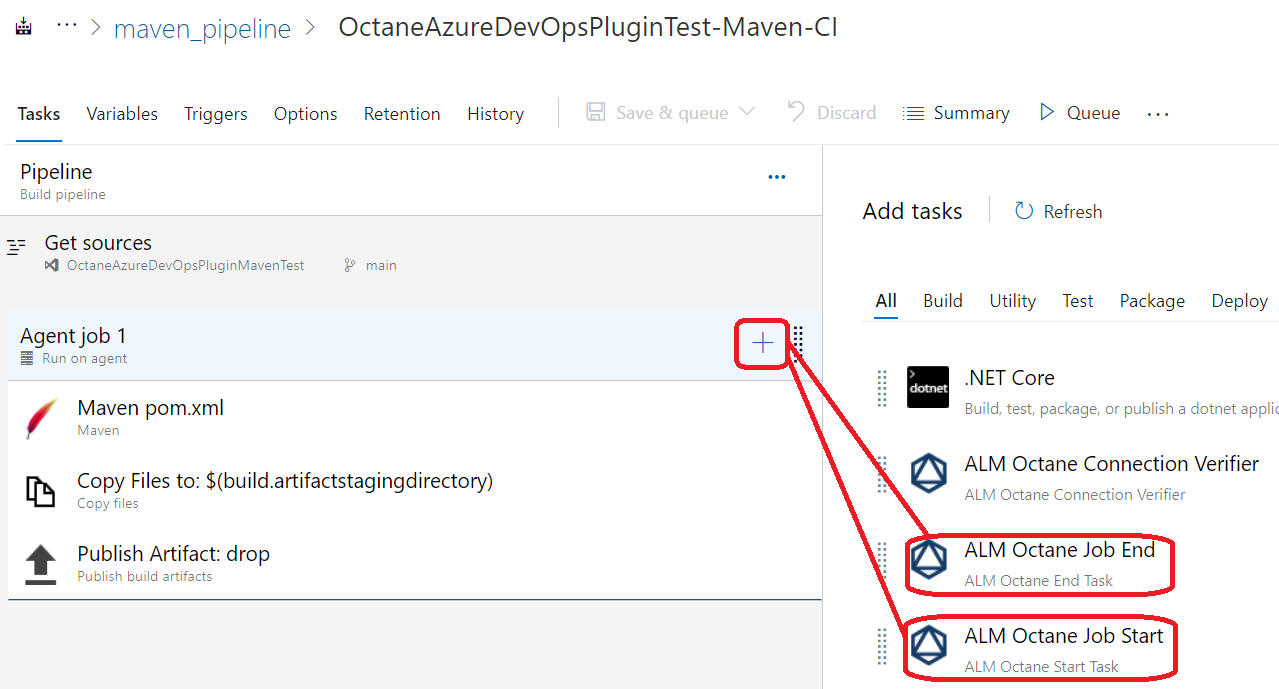
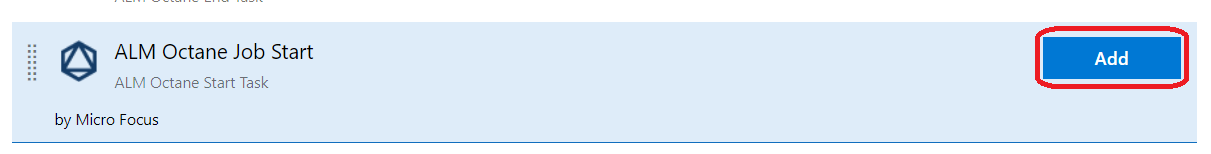
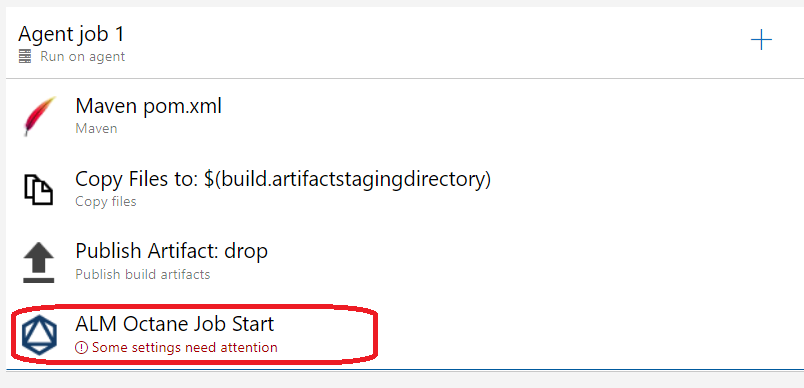
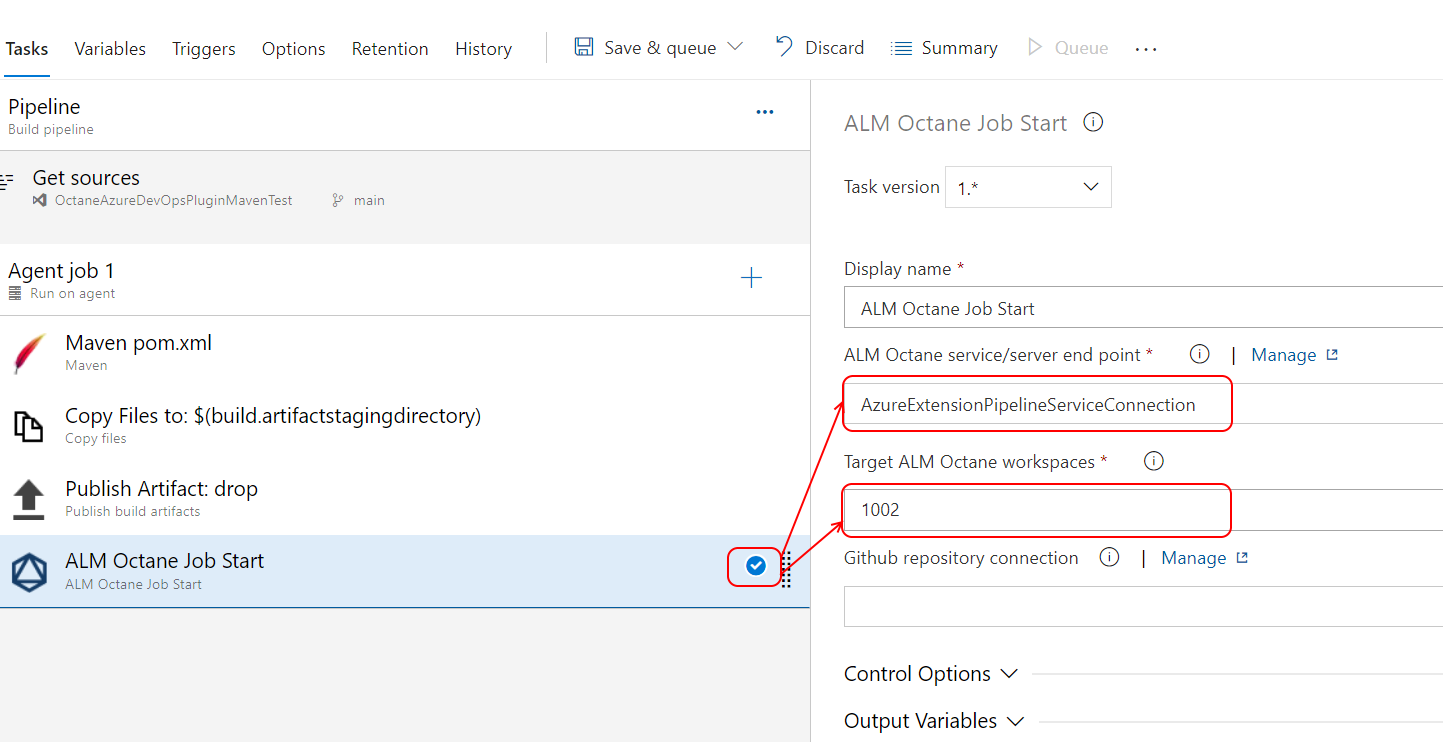
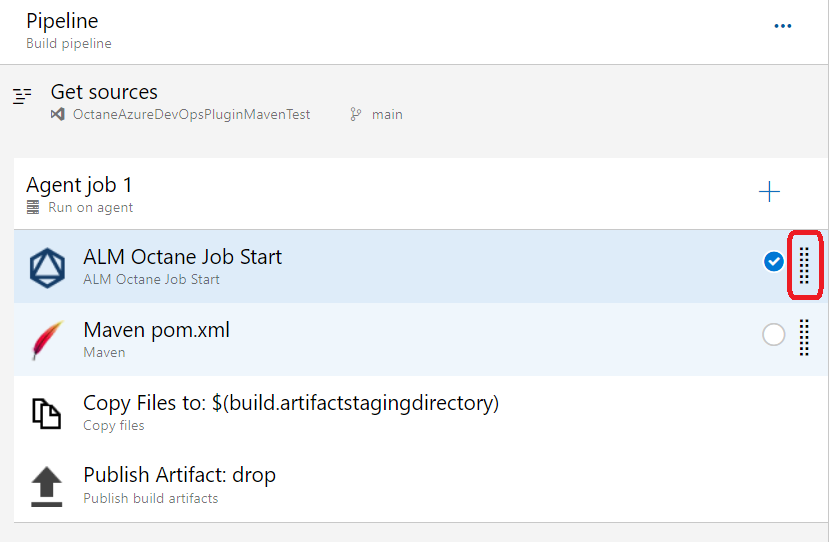
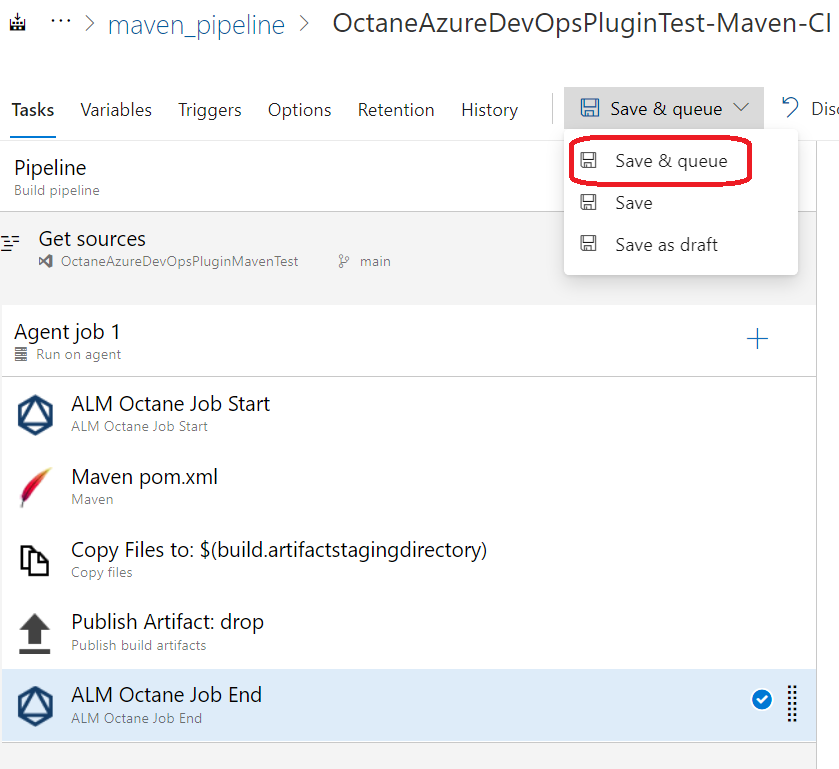
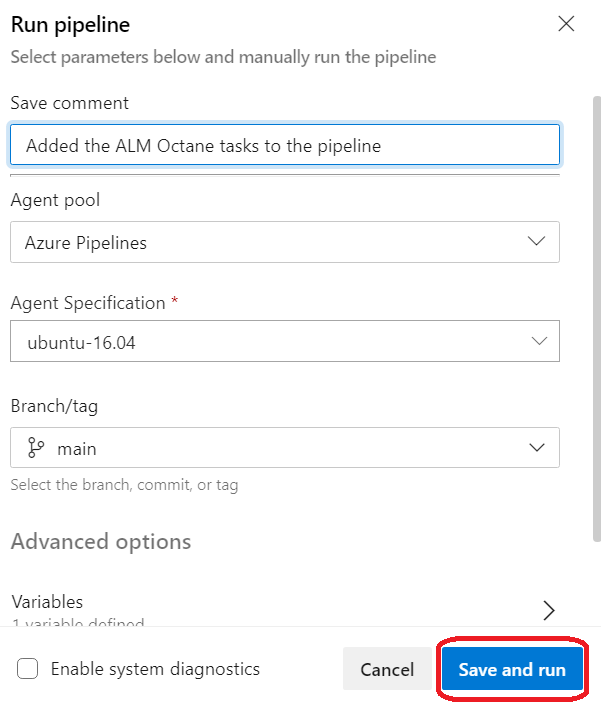
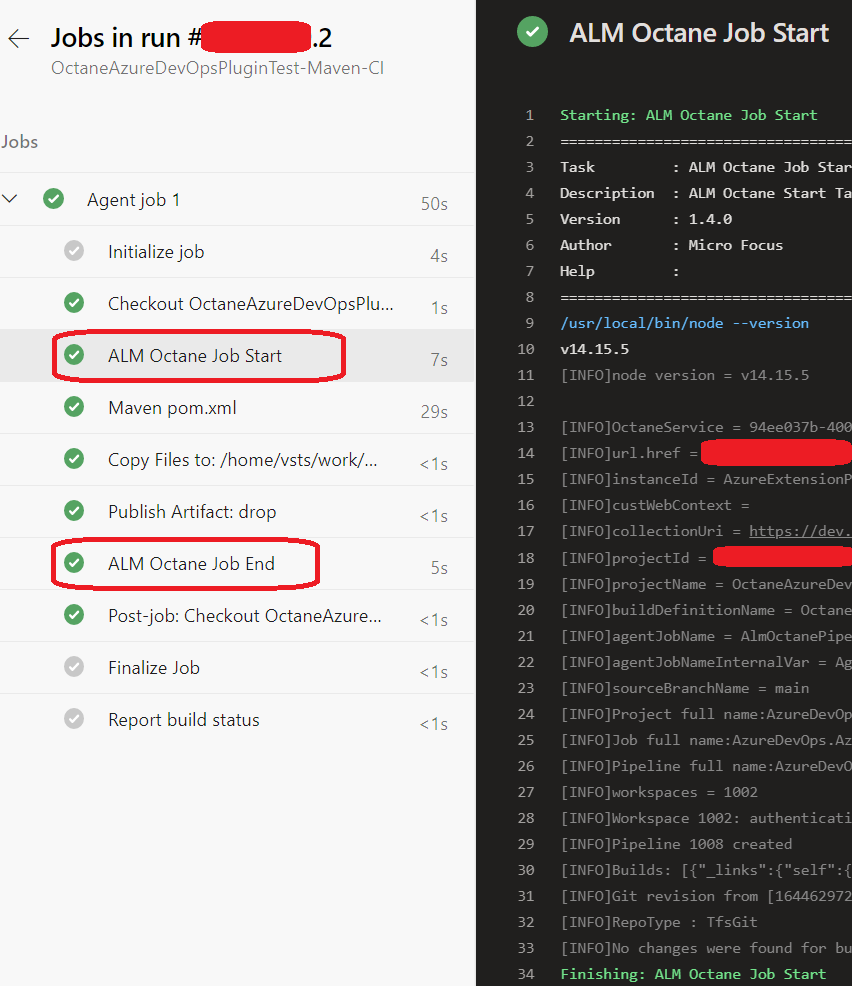
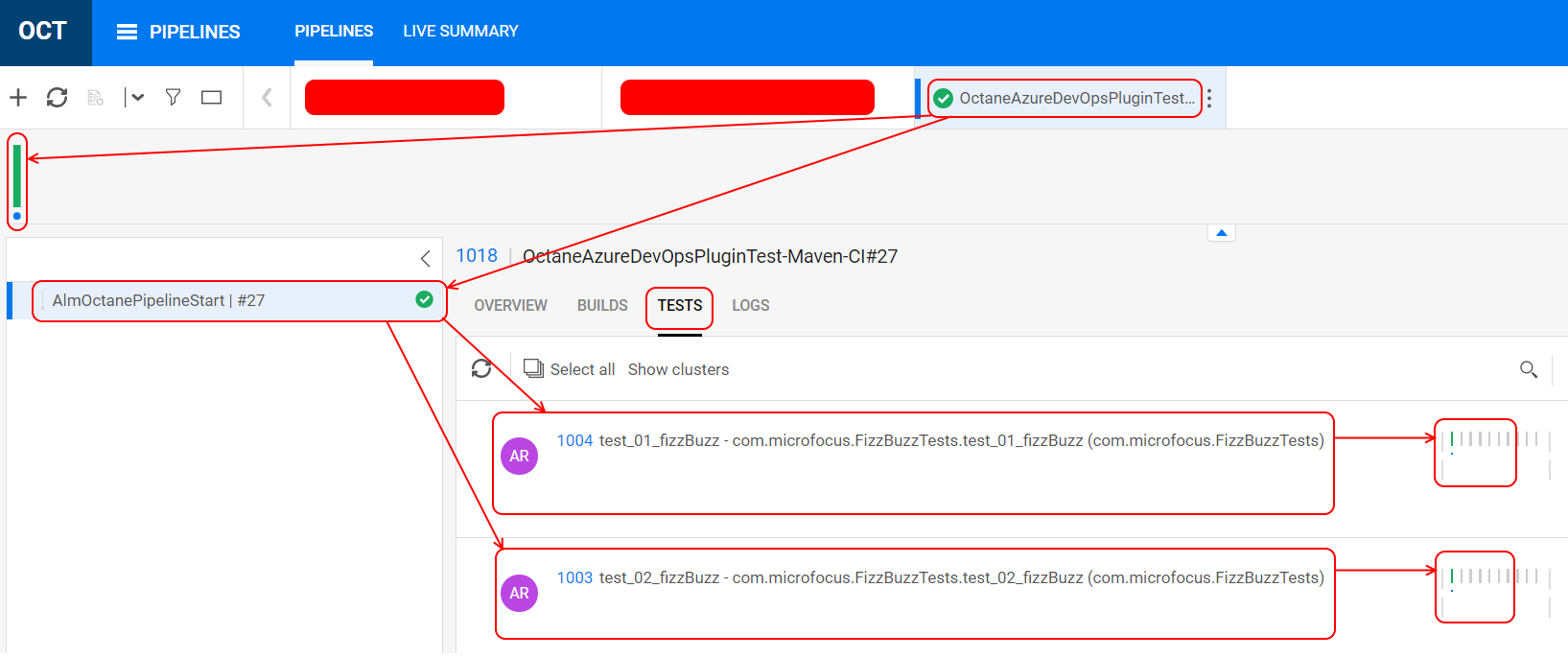


1. Go to Pipelines and you should see your new pipeline and it should be marked with Green, as the job succeeded:
2. Press on the pipeline and edit it:
3. Add the **ALM Octane Job Start** and ALM Octane Job End. You must make sure to add them together because otherwise the pipelines will not show properly in Octane. Job start is required in order to create the CI Server in Octane, the Pipeline and mark it as running. Job end is required in order to let Octane know that the pipeline ended. It really should be the last task inside your Azure DevOps pipeline. Make sure you put your cursor in the right location before adding the task as it will generate a YAML task entry directly in the position where your cursor was before you click on the task.  
     
   You need to know your workspace ID(s) within Octane where you want your pipeline to be created and updated. You can find it in 2 ways:  
   1. From the URL of the Octane instance you access, usually is the second integer, marked with RED: https://yourOctaneDomain.com/ui/?p=1002/1002

2. If you have sharedspace admin rights, going to Spaces within Octane and pressing the Edit Workspace button:  
In the end you should have in your YAML file something like below, and you should press save.  


1. Execute the pipeline by pressing Run:
2. You should observe that the Job was successful, like below
3. If you will click on the Job, you can see all its tasks and if you press, for example, on the octanestarttask, you can observe logs indicating that the connection to Octane was successful:
4. Back in Octane, you can check 2 things that were created successfully:
   1. That the pipeline actually got created
   2. If you have shared space admin role, that we have a new entry in the CI Servers grid, with the Instance ID matching the one you have previously configured: 

# Create a new pipeline with the Octane start and end tasks through classic editor (implicit Azure job)

1. Go to the Pipelines and find the hyperlink as below:
2. In the next step, choose the right team project, repository and branch
3. Choose the right template(Let’s suppose you want a Maven configuration to build Java code):
4. Fill in all required information, if the default values are not satisfactory for you and select Save:
5. Save the pipeline in a folder which you want:
6. Go to Pipelines and find your new pipeline and try to run it. It is important to understand that you need a valid project in that repository with a valid pom.xml and surefire plugin configured so that the actual tests are triggered and the resultant report to be published to Octane.  
   
7. Observe the execution results:  
   If you will go inside the pipeline, you can observe the following:  
   Clicking on the run, you should see a view similar to the one below:  
   Going inside “Agent job 1”, pressing on Maven pom.xml and scrolling down, you can observe:
8. Now, next step is to add the ALM Octane tasks. Go back and press “Edit” in order to edit the pipeline and add the Octane tasks.
9. In the newly displayed view, press the plus sign on the right of the “Agent Job 1” in order to reveal all possible tasks to be added:
10. Add the start task:
11. The task should be added to the end of the list, and now you should position it before all other tasks. Additionally, you need to configure it:
12. Click on the radio button on the right and configure the task: 
13. Drag and drop the ALM Octane Job Start and position it first in the list:
14. Do the same thing for the ALM Octane Job End, so that you end up with the following pipeline. Press “Save & queue”:  
    
15. Add a relevant comment to the save, and press the “Save and run” button:
16. Wait for the pipeline to run, and afterwards go inside the “Agent Job 1” and observe that you have 2 additional ALM Octane tasks that were executed:
17. Now you can go to Octane and verify that the pipeline got created and actually shows the results:

# Create a new pipeline with the Octane start and end tasks through YAML editing (explicit Azure jobs)

Previous chapters focused on the demonstration how to create pipelines with Octane tasks inside them. This is ok if you do not need complex pipelines with multiple jobs inside them, or you are just testing how the extension might fit your needs. For more complex scenarios where multiple jobs are used and the pipelines already exist, tasks under existing jobs might not be a solution.

Let’s suppose you have the following YAML, which contains a simple task of building a maven project specified under an unnamed job (in theory you might have many jobs here, but for the sake of simplicity, we will work with 1):

# Maven

# Build your Java project and run tests with Apache Maven.

# Add steps that analyze code, save build artifacts, deploy, and more:

# https://docs.microsoft.com/azure/devops/pipelines/languages/java

trigger:

- main

jobs:

- job:

  pool: 'Default'

  steps:

  - task: Maven@3

    inputs:

      mavenPomFile: 'pom.xml'

      mavenOptions: '-Xmx3072m'

      javaHomeOption: 'JDKVersion'

      jdkVersionOption: '1.8'

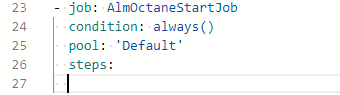
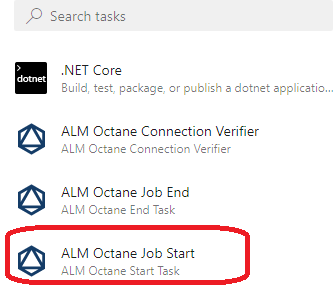
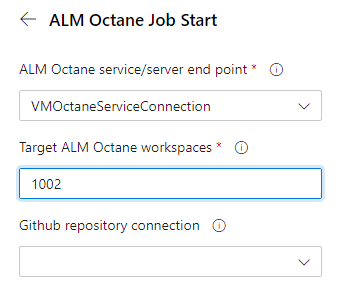
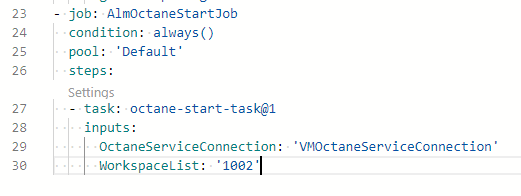
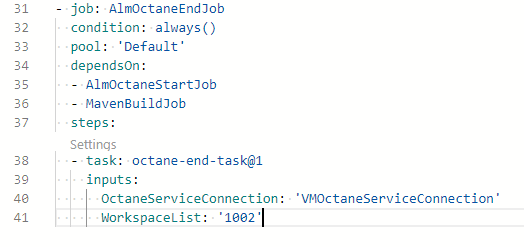
      jdkArchitectureOption: 'x64'

      publishJUnitResults: true

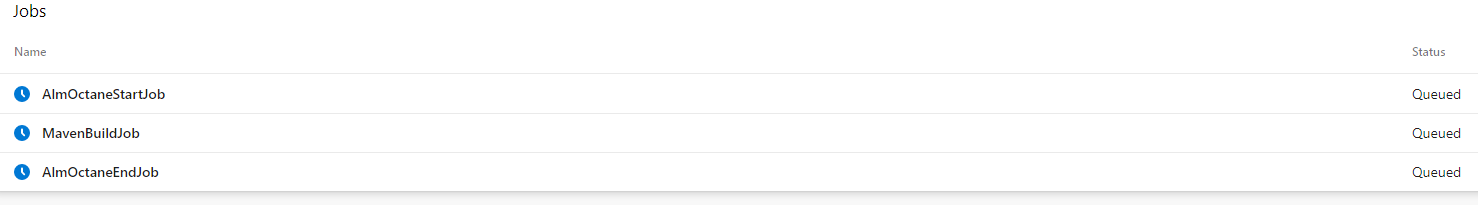
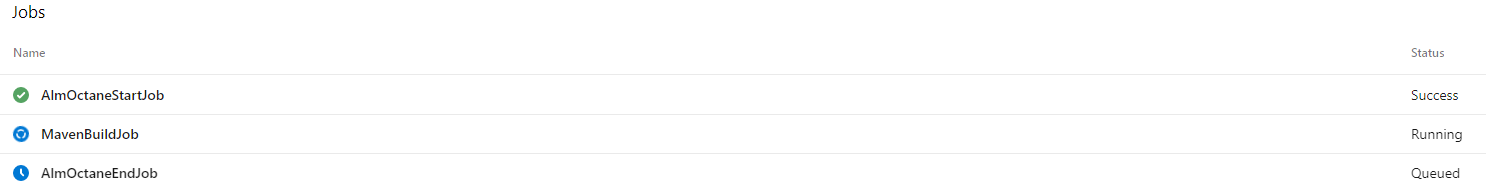
      testResultsFiles: '\*\*/surefire-reports/TEST-\*.xml'

      goals: 'package'

If you wish ALM Octane Start Task and End Task to be included as separate jobs, then you must do the following:

1. Give a name to your existing job(or jobs, if multiple)
2. Add, at the same level as your existing job, a new job and place your cursor on the next line after “steps:”:  
   
3. Press on the ALM Octane Job Start to be added:  
   
4. Fill in the required fields(Please refer to previous chapters for pre-requisites, explanation and examples):  
   
5. You should end up with the following block of code:  
   
6. Add the ALM Octane Job End in the same way as for start, with the difference that you need to have the “dependsOn” property set to the list of all existing jobs, including Octane Start Job. This is required because jobs might run in parallel, and in order to make sure that End job runs last, we need to set the dependencies on all other jobs:  
   
7. As a last step, you need to set “dependsOn” property of your jobs to depend on the AlmOctaneStartJob (or whatever name you’ve assigned to it), as follows:  
   
8. In the end, you should have the YAML file similar to the one below:

|  |
| --- |
| # Maven  # Build your Java project and run tests with Apache Maven.  # Add steps that analyze code, save build artifacts, deploy, and more:  # https://docs.microsoft.com/azure/devops/pipelines/languages/java  trigger:  - main  jobs:  - job: MavenBuildJob    pool: 'Default'    dependsOn:    - AlmOctaneStartJob    steps:    - task: Maven@3      inputs:        mavenPomFile: 'pom.xml'        mavenOptions: '-Xmx3072m'        javaHomeOption: 'JDKVersion'        jdkVersionOption: '1.8'        jdkArchitectureOption: 'x64'        publishJUnitResults: true        testResultsFiles: '\*\*/surefire-reports/TEST-\*.xml'        goals: 'package'  - job: AlmOctaneStartJob    condition: always()    pool: 'Default'    steps:    - task: octane-start-task@1      inputs:        OctaneServiceConnection: 'VMOctaneServiceConnection'        WorkspaceList: '1002'  - job: AlmOctaneEndJob    condition: always()    pool: 'Default'    dependsOn:    - AlmOctaneStartJob    - MavenBuildJob    steps:    - task: octane-end-task@1      inputs:        OctaneServiceConnection: 'VMOctaneServiceConnection'        WorkspaceList: '1002' |

1. Save and Run the pipeline, and you should see something similar to below:  
   
2. Now, because we specified the dependencies between the jobs, AlmOctaneStartJob should run first, MavenBuildJob second and AlmOctaneEndJob should be last:  
   
3. Now you are able to create complex scenarios with different jobs. Just make sure you understand the dependencies and current limitations of the Azure DevOps pipeline and the extension.

# Known issues and limitations

1. ALM Octane Connection Verifier is non-functional. Will be removed in a future version.
2. When creating the pipeline with YAML and adding the ALM Octane tasks, the label is not displayed properly (octanestarttask)
3. The ALM Octane tasks might show as GREEN even if these have errors, like:
   1. if you specify a wrong URL, for example, like in the case <http://192.168.1.129:9090/?p=1001/1002>, meaning skipping /ui/ part)
   2. If Octane Server is down, you might see in the octane start task log: “[ERROR]{"code":"ECONNREFUSED","errno":"ECONNREFUSED","syscall":"connect","address":"192.168.1.129","port":9090}”
   3. if you specified wrong credentials or the API key was revoked in Octane
4. All tests which are running with surefire plugin, for example, regardless of their nature, will be published to Octane as Automated runs. We currently do not support Gherkin tests execution injection into Octane or other type of tests.
5. In comparison with Jenkins, for example, currently the extension does not support injecting events of jobs/subjobs that are running in Azure, meaning that you will have only one job injected in Octane which will be the Octane Start task and which will show as completed with the related status when the pipeline ends with the Octane End Task. This behavior is limited because of the way Azure DevOps Pipelines currently work.
6. YAML is based on TABs, and as such, if you miss a TAB you might end up with a wrongly formatted YAML file and as such the pipeline will not work.