ALM Octane Integration with Azure DevOps Services

General information:

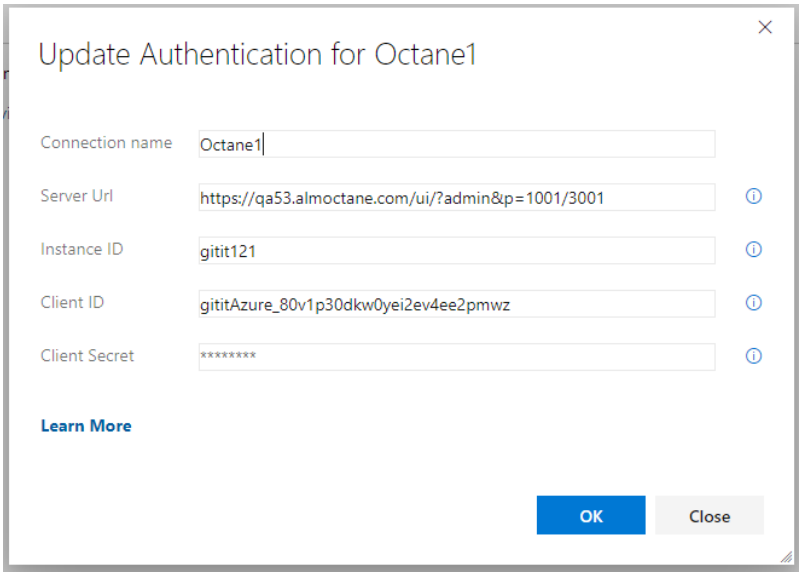
1. An Azure DevOps Services account normally implies an organization level.
2. Every account (organization) may have one or more projects.
3. Every project may have one or more pipelines.
4. Every pipeline consists of one or more jobs that may run in parallel.
5. Every job is a sequence of one or more tasks.
6. Every task is an activity executed during the run of the job containing it. A task can be written in a variety of programming/scripting languages and can be a part of either the Azure DevOps Services framework or an extension.
7. The only unit of an executable code that an extension can contribute to the Azure DevOps Services framework is a task that is normally executed during a pipeline run.

Pipeline decorators (currently a "Preview Feature" of the Azure DevOps Services):

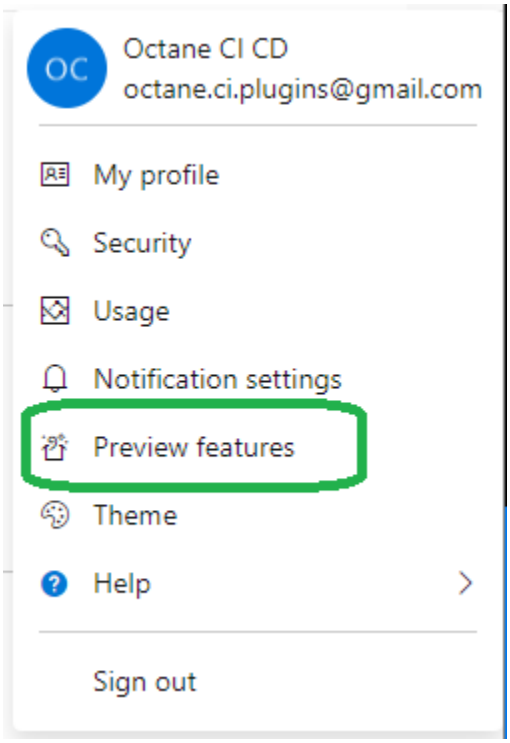
1. A pipeline decorator is a mechanism for injecting permanent tasks to the beginning and/or the end of every job.
2. A typical use case for pipeline decorators is injecting a "virus scan" task to the beginning of every job in a pipeline.
3. Pipeline decorators as a feature can be toggled on/off for the whole organization (there is no per-project control over toggling on/off the pipeline decorators).
4. Pipeline decorators can use conditions (for example values of pipeline variables etc.) to decide whether to inject their tasks or not.
5. Azure DevOps Services Extensions support contributing new pipeline decorators.

The integration:

1. The current implementation supports only one direction – an Azure DevOps pipeline and its jobs can notify ALM Octane about their progress and statuses, as well as send information about the source control commits and test results relevant to them. However running and stopping the pipeline from ALM Octane, is not supported.
2. The integration was implemented by developing an Azure DevOps Services extension with the following contributions:
    a. A new service connection type (ALM Octane Service Connection). By creating an a service connection of this type the user provides the properties of an ALM Octane Workspace (Connection name, URL, instance ID, shared space ID, workspace ID, API client ID and secret). This instance is then used by Azure DevOps pipelines for notifying the workspace with the events and data specified above. See the figure below for illustration:

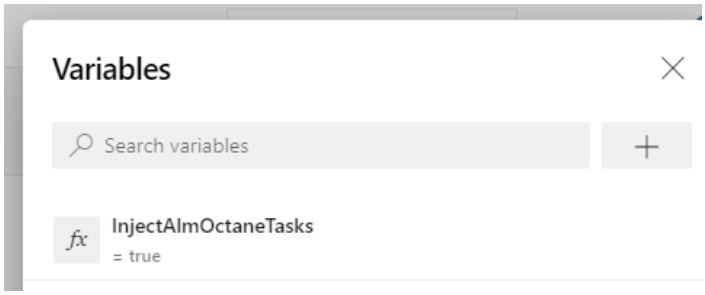


b. 2 new task types (“ALM Octane Job Start” and “ALM Octane Job End”), instances of which are auto-injected to every job in the pipeline at the appropriate locations – the start and the end points of every job. The Pipeline Decorators Preview feature must be on.



The auto-injection occurs only if there is a pipeline variable **InjectAlmOctaneTasks** set to **true** (for example in the pipeline YAML itself or via the pipeline variables UI as shown in the figures below):

```
variables:
  InjectAlmOctaneTasks: 'true'
```



Below is the workflow for establishing a successful integration between an ALM Octane workspace and an Azure DevOps Services project:

1. In the Azure DevOps organization:
    a. Install the ALM Octane Integration Extension. (For the Alpha and Beta versions a special request must be sent to ALM Octane Dev for sharing the extension with the target organization. For the GA release, the extension will be published to the Microsoft Azure DevOps Marketplace).
    b. Enable the "Pipeline Decorators" preview feature.
2. In the ALM Octane shared space: Create an API client ID and secret with the "CI/CD Integration" role over the target ALM Octane workspace.
3. In the target Azure DevOps project:
    a. Create a new ALM Octane Service Connection by providing all the required settings. Let's assume the connection name is OctaneCon.
    b. Set the following pipeline variables:

```
variables:
  InjectAlmOctaneTasks: 'true'
  OctaneConnectionName: 'OctaneCon'
```

    c. Let's assume your pipeline consists of the following jobs: A, B, C and D. You must add 2 extra jobs to the pipeline: one to the pipeline beginning and the other to the pipeline end. These jobs must have the following names: **AlmOctanePipelineStart** and **AlmOctanePipelineEnd**. They must be set to always run (independently of the status of any other jobs). Moreover, the **AlmOctanePipelineEnd** must wait until all the other jobs will finish, even if the other jobs run in parallel. The best way to ensure this is by setting it as dependent on the other jobs. Thus the final version of our **AlmOctanePipelineEnd** job will be:

```
- job: AlmOctanePipelineEnd
  condition: always()
  dependsOn:
  - AlmOctanePipelineStart
  - A
  - B
  - C
  - D
```

On the other hand, we want to let the **AlmOctanePipelineStart** to always run first. Therefore we need to force the other jobs to depend on it:

```
- job: A
  dependsOn: AlmOctanePipelineStart
  steps:
  - bash: echo "A"
```

The final version of our pipeline should be similar to the figure below.
After it runs, the appropriate “CI server” and “Pipeline” entries will be created in ALM Octane (if they don’t already exist), and the pipeline UI will reflect all the regular properties related to the pipeline run: its progress, topology, status, related source control commits, and test results.

```
variables:
  InjectAlmOctaneTasks: 'true'
  OctaneConnectionName: 'OctaneCon'

jobs:
- job: AlmOctanePipelineStart
  condition: always()

- job: A
  dependsOn: AlmOctanePipelineStart
  steps:
  - bash: echo "A"

- job: B
  dependsOn: AlmOctanePipelineStart
  steps:
  - bash: echo "B"

- job: C
  dependsOn: AlmOctanePipelineStart
  pool:
```

```
    vmImage: 'ubuntu-latest'
  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: D
  dependsOn: AlmOctanePipelineStart
  pool:
    vmImage: 'ubuntu-latest'
  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: AlmOctanePipelineEnd
  condition: always()
  dependsOn:
  - AlmOctanePipelineStart
  - A
  - B
  - C
  - D
```