Nikon Azure DevOps Configurations

# 1 Purpose

## This document aims to capture a high-level design for the CI/CD solution for Nikon Mulesoft applications. This design will be used to set up and trigger the pipelines.

## 1.1 Audience

* Mulesoft Architects, Developers and API designers
* DevOps team responsible for deployment and management of MuleSoft applications

# 2 Branches to Mule Environments

|  |  |  |
| --- | --- | --- |
| Branch | Mule Environment | Remarks |
| develop | DEV-OneView | Development |
| release | UAT-OneView | User Acceptance Testing |
| release | Prod-OneView | Production |
| main | Production | Copy of Production Code    After Prod Go-live code needs to merge from release to main |

Note:

The main branch must be created for each application which hold the production equivalent code/configuration.

For Each Feature development : feature/<<TICKET-ID>> or feature/<<TASK-NAME>>

For Development : develop

For UAT, Production : release

Note:

* feature/<<branch-name>> will be merged using PR request only to develop branch.
* develop will be merged with release branch using PR request only. [once develop branch is stable and ready for release]
* Without feature confirmation, feature/<<branch-name>> won’t be merged into develop branch

# 3 Configuration

## 3.1 MuleSoft Application Projects

Project Root Files

Pom.xml - refer doc <https://docs.mulesoft.com/mule-runtime/latest/deploy-to-cloudhub-2>

[ Please refer helloworld-sys-api branch : develop , pom.xml ]

Source Code : [helloworld-sys-app - Repos](https://dev.azure.com/NikonAustraliaMuleSoft/MuleSoft%20Migration/_git/helloworld-sys-app)

Global Properties

* mule.env [ dev, uat, prod ]
* secure.key [ Non-Prod and Prod will have a separate value for this secure key ]

## 3.2 Azure DevOps Variable Groups

The variable groups used for Mulesoft CICD are:

mule.{env}.vars.group

* mule.dev.vars.group (for DEV)
* mule.uat.vars.group (for UAT)
* mule.prod.vars.group (for PROD)

Note: It might be possible that only 3 environments will be there Ex: DEV, UAT and PROD ]

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Note:

* Only 3 Envs are available [ Ex: DEV, UAT and PROD ]
* SECURE\_KEY will be same for DEV and UAT, PROD will have a separate securekey
* TARGET\_NAME will be shared space or private space

|  |  |  |
| --- | --- | --- |
| Variables | Value | Remarks |
| AP\_CLIENT\_ID | <AP\_CLIENT\_ID> | Environment Client ID for DEV, UAT or PROD |
| AP\_CLIENT\_SECRET | <AP\_CLIENT\_SECRET> | Environment Client Secret for DEV, UAT or PROD |
| BG\_ID | <BG\_ID> | Business Group ID |
| ENV | XXX | Anypoint Environments:  DEV-OneView  UAT-OneView  Prod-OneView |
| MULE\_ENV | xxx | Environment parameter used for the application to decide the property file:  dev  uat  prod |
| MULE\_VERSION | 4.6.6 | The runtime version as it appears on the Anypoint platform |
| SECURE\_KEY | < SECURE \_KEY> | The key value to encrypt and decrypt the secret property values for DEV, UAT and PROD |
| TARGET\_NAME | <TARGET\_NAME> | The shared space or private space name should be provided.  Ex: Australia Region  Shared space: Cloudhub-AP-Southeast-2  Private space: TBD |

## 3.3 Azure DevOps Secure Files

The secure files used for MuleSoft CICD are:

settings.xml (the permission needs to be granted to all build pipelines)

A screenshot of a phone

Description automatically generated

## 3.4 Azure DevOps Build Pipelines

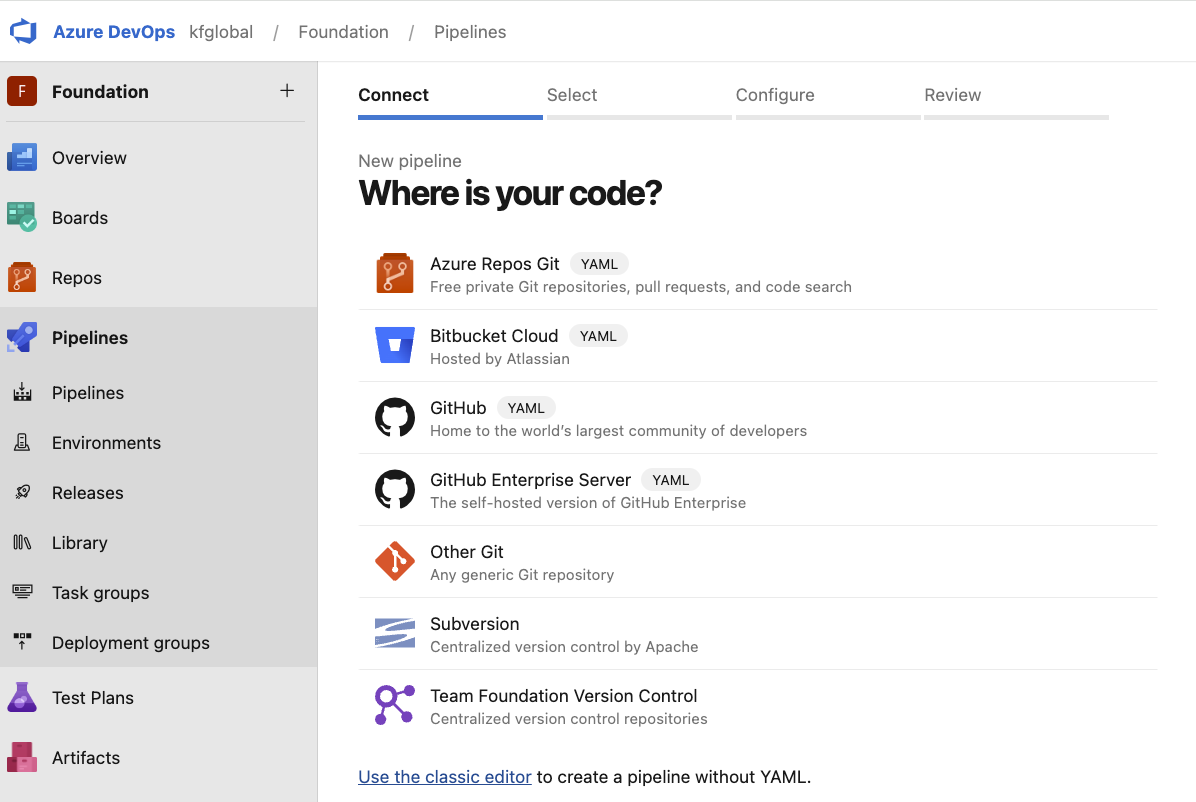
Prerequisites:

1. Have set-up variable groups for each environment (dev, uat, prod) and secure files(settings.xml) under Pipeline > Library.
2. Create three branches: develop, release, main

|  |  |  |  |
| --- | --- | --- | --- |
| **Branch** | **Type of Pipeline** | **Pipeline Naming** | **Pipeline Name** |
| develop | Build Pipeline | <<app-name>>-<<branch-name>>-package | notification-sys-api-develop-package |
| develop | Release Pipeline | <<app-name>>-<<branch-name>>-deploy | notification-sys-api-develop-deploy |
|  |  |  |  |
| release | Build Pipeline | <<app-name>>-<<branch-name>>-package | notification-sys-api-release-package |
| release | Release Pipeline | <<app-name>>-<<branch-name>>-deploy | notification-sys-api-release-deploy |
|  |  |  |  |
| **Note:** | Build Pipeline | <<app-name>>-<<branch-name>>-package | This build pipeline will be used to generate artifact version, munit, package and deploy an artifact to AP Exchange |
|  | Release Pipeline | <<app-name>>-<<branch-name>>-deploy | This release pipeline will be used to deploy an artifact to respective environments. (Ex: DEV, UAT and PROD ) |
|  | Release Pipeline | The release pipeline will have two stages:   1. UAT (MULE UAT deployment) 2. PROD (MULE PROD deployment) |  |

Set up to **Build pipelines**

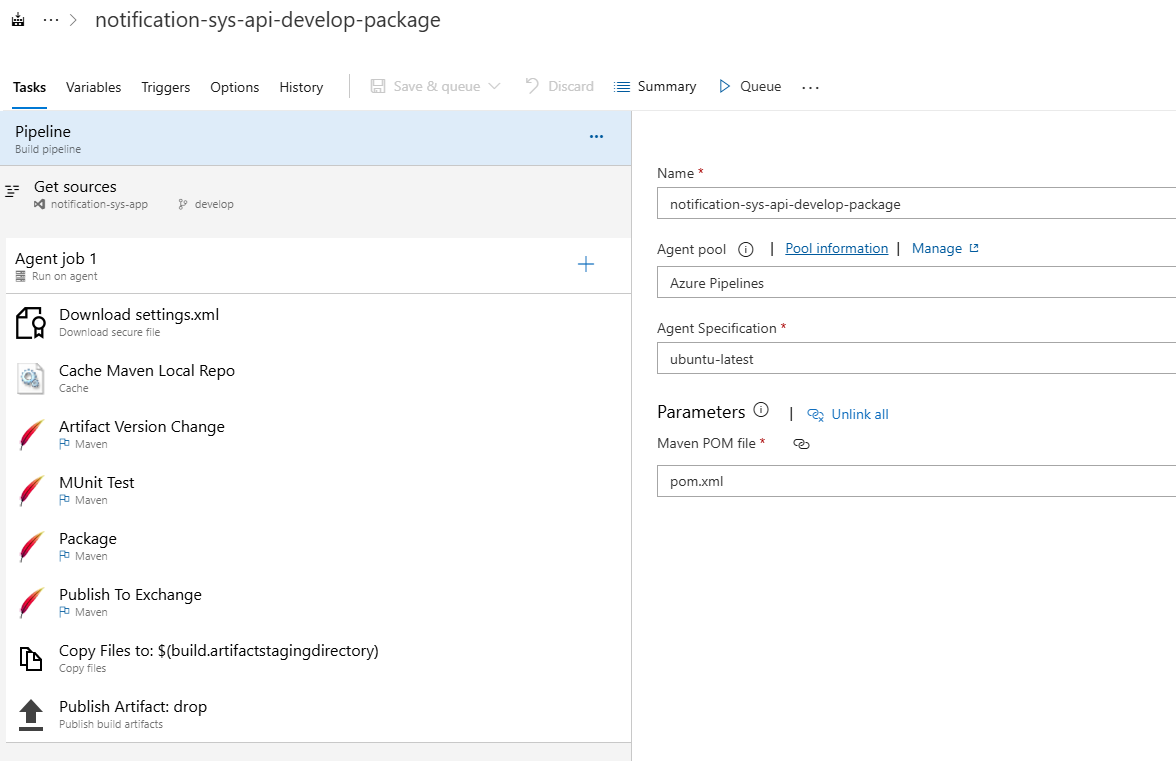
1. Click ‘New pipeline’ to create pipeline
2. Select “Use the classic editor to create a pipeline without YAML” > AzureRepos Git > Maven



A screenshot of a computer

Description automatically generated

1. Configure - Pipeline to: [ Branch : develop and Action : package ]



1. Configure - Agent Job 1 > Agent Pool and Specification

A screenshot of a computer

Description automatically generated

1. Add and configure a new task - Download Secure file

A screenshot of a computer

Description automatically generated

1. Add and configure a new task - Cache

Key: maven | "$(Agent.OS)" | $(Build.SourcesDirectory)/pom.xml

Path: $(MAVEN\_CACHE\_FOLDER)

Additional restore Key prefixes:

maven | "$(Agent.OS)"

maven

A screenshot of a computer

Description automatically generated

1. Configure existing task - Maven
2. Artifact Version Change

A screenshot of a computer

Description automatically generated

Set Goal(s), Option(s) and Remove Junit Test Results

Goal(s)

build-helper:parse-version -DgenerateBackupPoms=false versions:set -DnewVersion=$(NEW\_BUILD\_VERSION)

Option(s):

-s $(mvnSettings.secureFilePath) $(MAVEN\_OPTS)

Note:

* Validate Pipeline variables are configured properly.

A screenshot of a computer

Description automatically generated

* Validate Variable groups are associated with dev groups

A screenshot of a computer

Description automatically generated

1. MUnit Test

Set Goal(s), Option(s) and Junit Test Results

Goal(s)

clean test

Option(s):

-s $(mvnSettings.secureFilePath) $(MAVEN\_OPTS) -Dsecure.key=$(SECURE\_KEY) -Dmule.env=$(MULE\_ENV)

A screenshot of a computer

Description automatically generated

1. Package

Set Goal(s), Option(s) and Remove Junit Test Results

Goal(s)

clean package -DskipTests

Option(s)

-s $(mvnSettings.secureFilePath) $(MAVEN\_OPTS)

A screenshot of a computer

Description automatically generated

1. Deploy To Exchange

Set Goal(s), Option(s) and Remove Junit Test Results

Goal(s) : clean deploy -DskipTests

Option(s): -s $(mvnSettings.secureFilePath) $(MAVEN\_OPTS)

A screenshot of a computer

Description automatically generated

Note: MAVEN\_OPTS and MAVEN\_CACHE\_FOLDER are pipeline variables

1. Configure existing task - Copy Files to: $(build.artifactstagingdirectory)

Contents: \*/\*.jar

\*\*/pom.xml

A screenshot of a computer

Description automatically generated

1. Under Variables > Variable groups > Link Variable Group > select variable group depending on corresponding environment (In this case develop branch is DEV environment)

A screenshot of a computer

Description automatically generated

1. Enable continuous integration through customizing ‘Triggers’

A screenshot of a computer

Description automatically generated

1. Click Save and leave an appropriate comment regarding your changes
2. Repeat the steps for other branches respectively ( if needed ).

## 3.5 Azure DevOps Release Pipelines

Setup for release pipelines

1. Create a new release pipeline: New > New release Pipeline > Start with an “Empty Job” > Stage name: <environment-name> and change name of release pipeline at the top of page.

A screenshot of a computer

Description automatically generated

1. Add an Artifact  
    2.1 Configure - select matched build pipeline

A screenshot of a computer

Description automatically generated

1. Add **continuous deployment trigger** (For DEV branch ONLY): Click on the lightning icon on the top right of the artifact and enable: “Continuous deployment trigger”.

A screenshot of a computer

Description automatically generated

1. Add ‘Pre-deployment conditions’ for some special environments (like DEV)

Define the trigger that will start deployment for DEV stage

A screenshot of a computer

Description automatically generated

Select the users who can approve or reject deployments to this stage

A screenshot of a computer

Description automatically generated

1. Click on Tasks and start to add the relevant tasks
   1. Configure an existing task - Agent Job:

A screenshot of a computer

Description automatically generated

* 1. Configure an existing task - Download:

A screenshot of a computer

Description automatically generated

* 1. Add and configure a new task - Bash

cd $(Release.PrimaryArtifactSourceAlias)/drop/target/

ls -la

echo "##vso[task.setvariable variable=ARTIFACT]`ls | grep \*.jar`"

A screenshot of a computer

Description automatically generated

* 1. Add and configure a new task - Maven

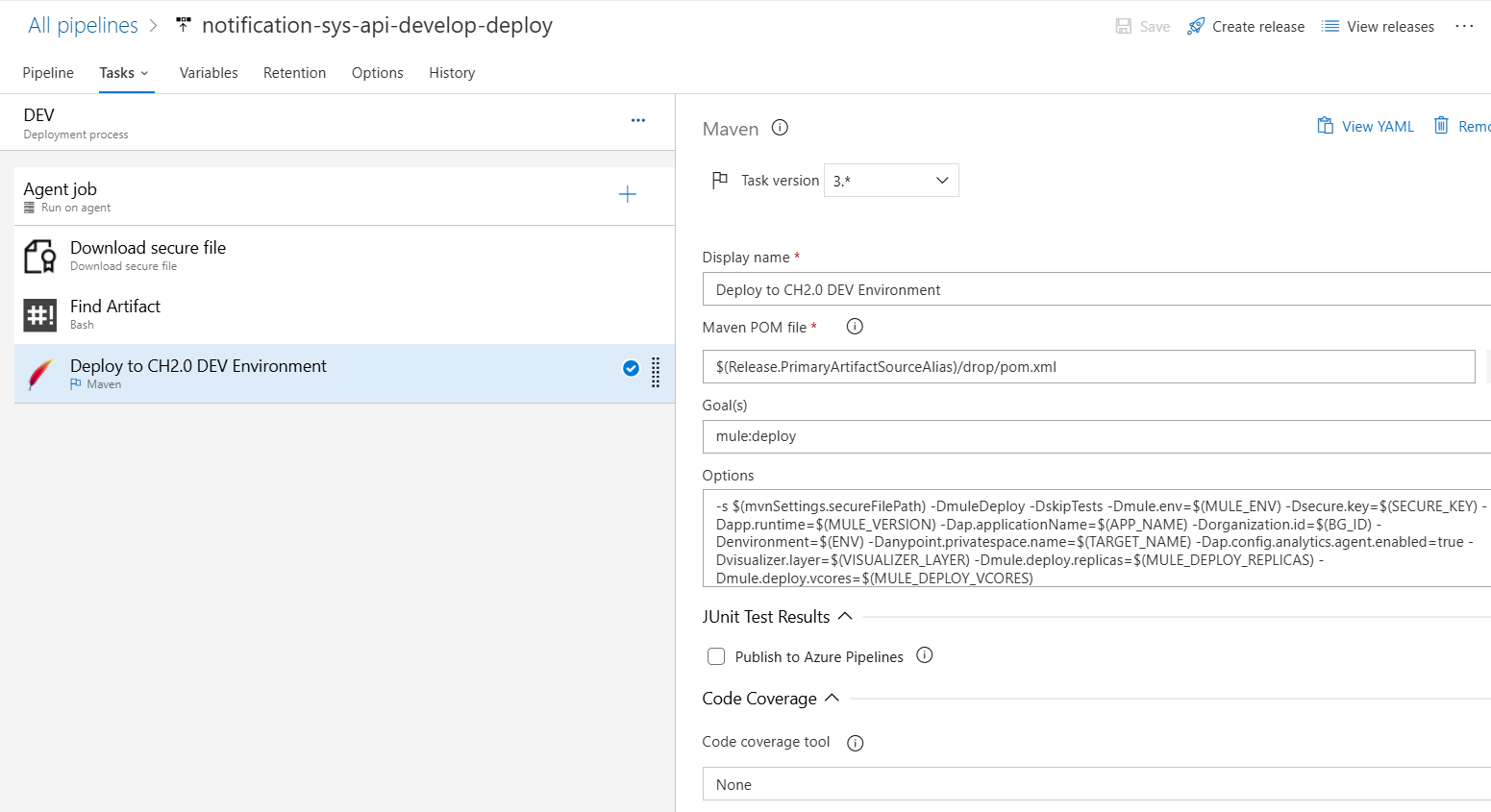
Maven POM file: $(Release.PrimaryArtifactSourceAlias)/drop/pom.xml

Goal(s): mule:deploy

Option(s):

-s $(mvnSettings.secureFilePath) -DmuleDeploy -DskipTests -Dmule.env=$(MULE\_ENV) -Dsecure.key=$(SECURE\_KEY) -Dapp.runtime=$(MULE\_VERSION) -Dap.applicationName=$(APP\_NAME) -Dorganization.id=$(BG\_ID) -Denvironment=$(ENV) -Danypoint.privatespace.name=$(TARGET\_NAME) -Dap.config.analytics.agent.enabled=true -Dvisualizer.layer=$(VISUALIZER\_LAYER) -Dmule.deploy.replicas=$(MULE\_DEPLOY\_REPLICAS) -Dmule.deploy.vcores=$(MULE\_DEPLOY\_VCORES) -Danypoint.platform.client\_id=$(AP\_CLIENT\_ID) -Danypoint.platform.client\_secret=$(AP\_CLIENT\_SECRET)

Note: Uncheck the JUnit Test Results:



* 1. Add Pipeline variables (APP\_NAME)

1. Configure: Pipeline variables for each environment(DEV)

A screenshot of a computer

Description automatically generated

* 1. Add Variable Groups (Select variable groups corresponding to current environment deploying to)

A screenshot of a computer

Description automatically generated

* 1. Click Save and leave an appropriate comment regarding your changes.
  2. Repeat steps for release branch respectively.

# 4. How to use these pipelines

## 4.1 Build pipeline

* Build pipeline will automatically be triggered once code merge into branch
* Or can manually run the pipeline by click ‘Run pipeline’ button

A screenshot of a computer

Description automatically generated

Select branches name to build and click ‘Run’ button

A screenshot of a computer

Description automatically generated

A screenshot of a computer

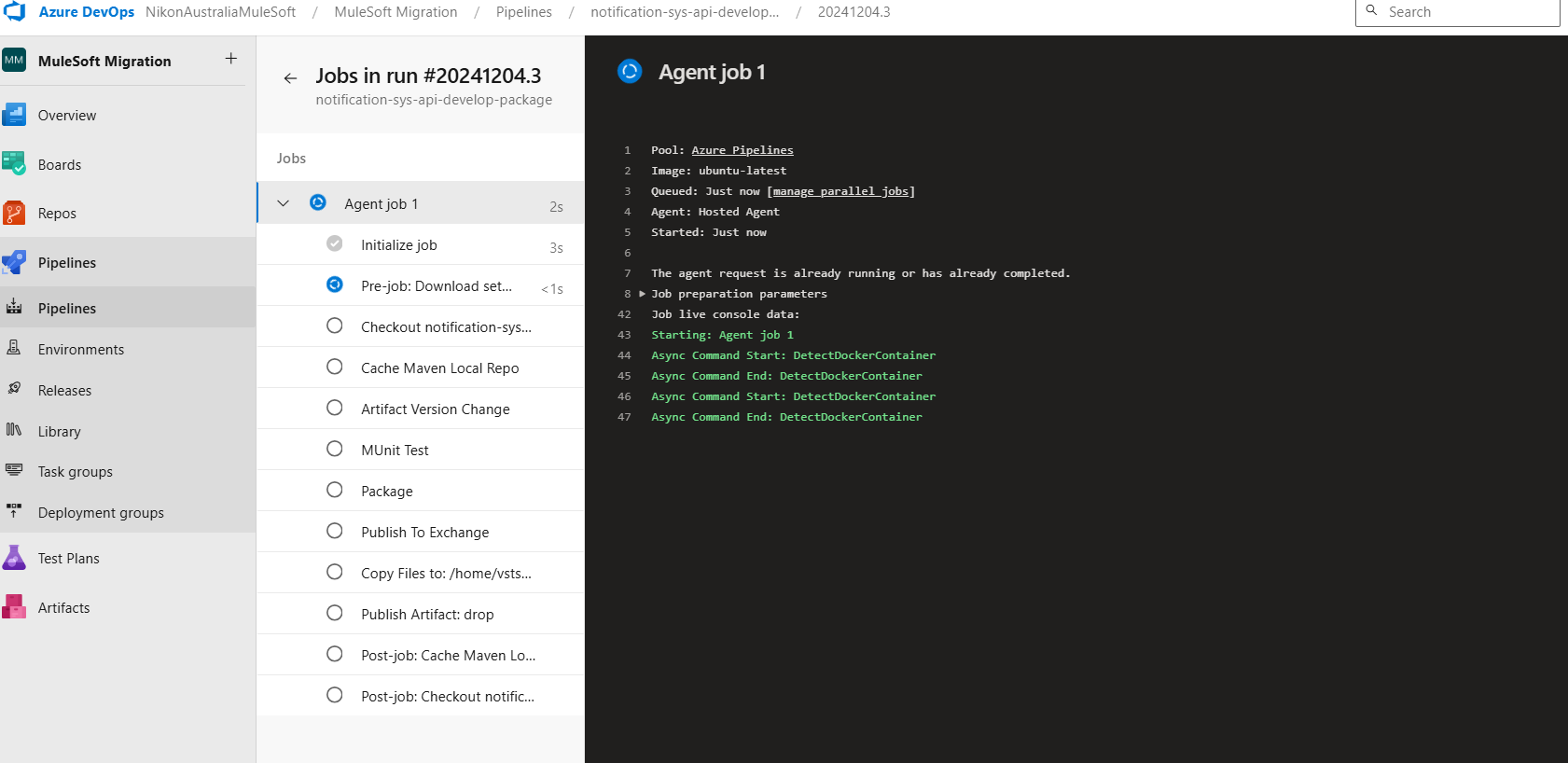
Description automatically generated

Check pipeline logs by click ‘Agent Job 1

A screenshot of a computer

Description automatically generated

It shows logs as below:



Clone existing build pipeline:

A screenshot of a computer

Description automatically generated

## 4.2 Release Pipeline

* Release pipeline will be automatically triggered and created when a new build is available
* Or you can manually create it by click button ‘Create Release’

A screenshot of a computer

Description automatically generated

* Clone existing release pipeline

A screenshot of a computer

Description automatically generated

* Check logs

Select release pipeline

A screenshot of a computer

Description automatically generated

the release you want to check and Stage, then click ‘Logs’ button

A screenshot of a computer

Description automatically generated

Note: Below are URL details for DEV, UAT and PROD environments.

|  |  |  |
| --- | --- | --- |
| ENV | URL | Comments |
| DEV |  |  |
| UAT |  |  |
| PROD |  |  |