

1. DevSecOps Modernization	2
1.1 Important Migration Information	2
1.2 CIO Enablement Package for Migration to Azure DevOps Services & GitHub Enterprise Cloud	5
1.2.1 DevSecOps Environment	5
1.2.1.1 Azure DevOps Services	5
1.2.1.1.1 Azure DevOps Organization	6
1.2.1.1.2 Azure DevOps Team Project	6
1.2.1.1.3 Azure DevOps Team Project Access Types	6
1.2.1.1.4 Available Pipeline Agent Capabilities	7
1.2.1.1.5 Azure DevOps Training Material	8
1.2.1.2 GitHub Enterprise Cloud	8
1.2.1.2.1 GitHub Enterprise Managed Users (EMU)	9
1.2.1.2.2 GitHub Organization	9
1.2.1.2.3 GitHub Restriction within BMO	9
1.2.1.2.4 GitHub Repo Access Type	9
1.2.1.2.5 GitHub Training Material	10
1.2.2 Migrate to DevSecOps	10
1.2.2.1 GitHub Team	11
1.2.2.2 Migration Consideration: Active vs Inactive	11
1.2.2.3 Migration of AWS Workload	11
1.2.2.4 Pipeline Migration	11
1.2.2.4.1 Migrate to Azure Pipeline	12
1.2.2.5 Repository Migration	59
1.2.2.5.1 BitBucket to GitHub Migration	59
1.2.2.5.2 Pre-Migration Checklist	60
1.2.2.5.3 Source Code Migration Types	61
1.2.2.5.4 TFS (Azure DevOps Servers) to GitHub Migration	61
1.2.2.5.5 GitHub Onboarding Process	62
1.2.2.6 GitHub PAT Token	63
1.2.3 Appendix	64
1.2.3.1 git-filter-repo Instructions	64
1.2.3.2 Migration FAQs	68
1.2.3.3 Service Account and Token for GitHub Connection	68
1.2.3.4 Mono-Repo vs. Multi-Repo Comparison	69
1.2.3.5 Azure AD vs GitHub Teams Discussion	69
1.2.3.6 git-sizer instructions	70
1.2.3.7 Branching Strategies: Git Flow vs. GitHub Flow Comparison	72
1.2.3.8 DLP scan for GitHub Migration	75
1.2.3.9 Intake requests on GitHub and ADO	75
1.2.3.10 Connecting to GitHub of BMO Organization from Ansible and OpenShift	76
1.2.3.11 Onboarding CD Tools (Ansible, Openshift, Artifactory)	76
1.3 Public Announcements	76
1.3.1 Security Best Practices for Jira and Confluence	76

# DevSecOps Modernization

DevSecOps Modernization is BMO's progressive activity of migration from legacy and on-prem platforms to cloud based platforms for the below segments-

- **Source Code Management Tool:**  
Repos from Bitbucket or TFS/Azure DevOps Servers will be migrated to GitHub Enterprise Cloud.
- **Build Orchestration Platform:**  
Bamboo pipelines will be migrated to Azure Pipelines (on Azure DevOps Services).

The following document set provides comprehensive details about DevSecOps Modernization:

<https://bmo01.atlassian.net/wiki/spaces/DSOM/pages/20775480/Migration+Engagement+Model> gives important information on the migration process.

[CIO Enablement Package for Migration to Azure DevOps Services & GitHub Enterprise Cloud](#) deep dives into the enablement package for migration.

## Recently updated

You'll see the 5 most recently updated pages that you and your team create.

-  [Visual Studio Build](#)  
Jul 13, 2023 • contributed by Prachi Gulihar
-  [Ant Build](#)  
Jul 13, 2023 • contributed by Prachi Gulihar
-  [NPM Build](#)  
Jul 13, 2023 • contributed by Prachi Gulihar
-  [Mobile Template Build](#)  
Jul 13, 2023 • contributed by Prachi Gulihar
-  [Maven Build](#)  
Jul 13, 2023 • contributed by Prachi Gulihar

## Important Migration Information

### Enterprise Migration Kickoff:

- Presentation Deck: [DevSecOps Modernization Enterprise Migration Kick off](#)
- Presentation Recording: [ADO\\_GHEC Migration Kickoff Sept 13\\_22 Video Recording.mp4](#)
- Cloud Deployment - GitHub Actions KT Session: [Migration to GitHub- GitHub Actions Demo](#)

### CIO Enablement Package and Migration Playbook:

<https://bmo01.atlassian.net/l/cp/FXtpPYnA>

### Migration Tracker: CIO Team Primary Contact Info and Repo List:

- [Engineering and EAI](#)
- [TREO](#) - Technology Resiliency and Experience Operation
- [NA P&BB](#) - Personal and Business Banking and Wealth Management
- [BMO Capital Markets](#)
- [Data & Analytics](#)
- [NA Commercial Banking](#)
- [Corporate Areas Technology and Operation](#)
- [Business Managed Application & Misc.](#)

## Overall Repo migration Status:

[CIO-LTO migration status report](#)

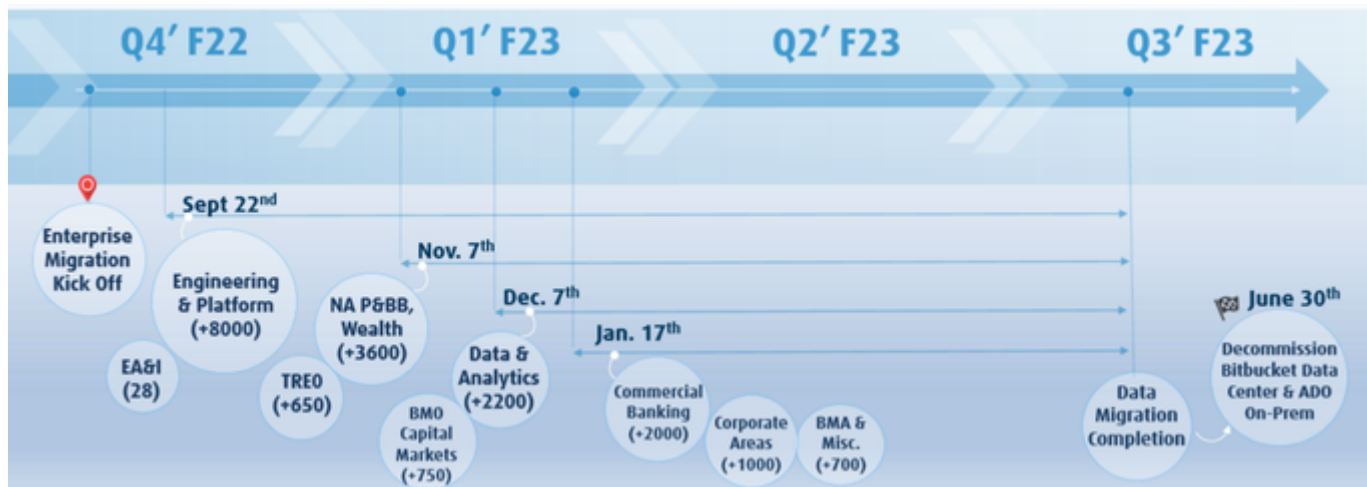
## Migration Engagement Model:

- Each CIO team will be assigned a dedicated DevSecOps Technical Lead throughout the migration.
- Each CIO team is required to assign a **primary developer lead/contact** for each repository/project. The primary contact must:
  - Act as STO/Owner representative.
  - Must have admin rights for the repository and build plans.
  - **Must have full authority on any migration decision or changing data for repository.**
  - Must have CI/CD expertise.
  - Only primary contact will engage DevSecOps Lead for any migration inquiries and issues.
- Overall Migration Lead: Godwin Wong
- Overall DevSecOps Architect: Pang Chung Tsang

CIOO	LOB (CIO level)
<b>Engineering and Platforms - Graeme Whittington</b>	
	Wang, Jon
	Maffioli, Christopher
	Movva, Anantha
<b>BMO Capital Markets - Victor Tung</b>	
	Jeraj, Fera
	Ross, Raymond
	Prado, Kim
	Stewart, Drew
<b>Corporate Areas – Susan White</b>	
	Mackintosh, Jeff
	Rajaram, Sriram
	Sloan, Randall
	Gomes, Carl
<b>Data and Analytics – Sandip Sahota</b>	
	Christofilos, Lisa
	Sahota, Sandip
<b>Enterprise Architect and Innovation – Lawrence Wan</b>	
	Shen, Grace Hui
	O'Sullivan, Sharon
	Zhang, Simon
<b>NA Commercial Banking - Sidney Deloatch</b>	
	Daga, Alok
	Sidhu, Gurreet
	Kalra, Harsh
<b>NA Personal and Business Banking and Wealth Management - Bojan Pavlovic</b>	
	Jawor, Mariusz
	Komor, Voytek
	Jovanovic, Sladjana
	Phillips, Eric
	Verma, Atul K
	Alvar Thiruvengadathan, Bindhu
<b>Technology Resiliency and Experience Operation - Angela Sim</b>	
	Wong, Alan1
	Wintle, Michael
	Anderson, Leslie
	Keeton-Curtis, Pauline
<b>Business and Others</b>	

### Migration Timeline:

- Each LOB group will be engaged for smaller kick off/preparation sessions based on the following schedules.
- Each LOB group will require to work with PM and DevSecOps Lead to confirm their migration path and dates after their LOB kick off.
- All migrations are target to complete by June 28<sup>th</sup>, 2023, as the vendor licenses will expire by then.



## CIO Enablement Package for Migration to Azure DevOps Services & GitHub Enterprise Cloud

This Enablement Package has been created to aid CIO teams in migrating to GitHub Enterprise Cloud and Azure Pipelines (on Azure DevOps Services) depending on whether the starting project is in Bitbucket/Bamboo or TFS/Azure DevOps Servers.

The documentation has been broken down into three sections:

- The package begins by explaining the features offered in the [DevSecOps Environment](#) section.
- The next section, [Migrate to DevSecOps](#) is divided into 2 categories: Azure DevOps Services and GitHub Enterprise Cloud - each category covers the migrating process and types of user access.
- The last section, [Appendix](#) has the supplementing information.

### Audience

These documents have been developed to help support the CIO development teams.

### Usage

These documents are living documents. As the Service Catalogue evolves and services are expanded these pages will be updated by the DevSecOps team to keep the CIO teams informed about the latest developments. It is suggested that the CIO teams bookmark these pages for use as they will need to be referenced as each repository is migrated to GitHub Enterprise Cloud and the associated Pipeline recreated in Azure DevOps.

### DevSecOps Environment

BMO underwent through assessment and PoC and based on the cloud first strategy, BMO has selected Azure DevOps Services and GitHub Enterprise Cloud as the next generation CI/CD orchestration tools and source code repositories.

This is going to be an evolving environment. So, make sure to keep checking often to see enhancements that will be made. The following sections explain the features offered by GitHub Enterprise Cloud and Azure DevOps Service.

- [Azure DevOps Services](#)
- [GitHub Enterprise Cloud](#)

For migration related FAQs please refer [Migration FAQs](#)

### Azure DevOps Services

Azure DevOps Services is a hosted version of Azure DevOps Server which provides a scalable, reliable, and globally available service. Its main feature is Pipelines which provides build and release services to support continuous integration and delivery of applications and it replaces the function of Bamboo as build orchestration solution.

Enterprise pipeline template has been created for reusable patterns which promote process standardization.

[Pipeline Recreation](#)

### Pipeline Agents:

A pool of agents in 3 OS (Linux, Windows, MacOS) are available for enterprise use.

There agents are self-hosted in BMO environment with pre-installed capabilities.

Microsoft-hosted agent has not been authorized and should not be used.

For available pipeline agent capabilities, refer to [Available Pipeline Agent Capabilities](#)

*Note: The following services of Azure DevOps Services are not offered:*

Azure DevOps Service	Enterprise Tool for Corresponding Function
Azure Boards	Jira Cloud
Azure Repos	GitHub Enterprise Cloud
Azure Test Plan	ALM QC
Azure Artifacts	JFrog Artifactory

#### Guideline

- [Guideline for Azure DevOps Services](#)

#### Training Material

- [Azure DevOps Training Material](#)

## Azure DevOps Organization

Azure DevOps organization creation is restricted. BMO offers two organizations- **BMO-Prod** and **BMO-Sandbox**. BMO-Prod is targeted to handle all the CI/CD pipeline to create testing/deployment artifacts and end-to-end pipeline used by the development team. BMO-Sandbox is targeted for doing experiment and proof of concept (PoC).

Please refer to the [Azure DevOps Guideline](#) for more information.

## Azure DevOps Team Project

A Team Project in Azure DevOps organization is where users plan, track progress, and collaborate on building software solutions. A project represents a fundamental container where the data is stored when adding to Azure DevOps.

On BMO establishment, a Team Project is expected to be tied to one or more application development. Each application development should be associated with an AppCAT.

**Note:** *One AppCAT is typically expected to be associated with only one team project on BMO Azure DevOps organization.*

By convention, a neutral name (based on island name) is automatically selected by the onboarding process as the Team Project Name and this can not be changed.

Team project creation is being handled by the onboarding process. The below information is mandatory for onboarding request.

1. CIO
2. LTO
3. STO
4. Teams Admin(s)
5. AppCAT ID
6. Is Work Item (Boards) required or not?

If yes, then please select the type of board methodology required from the below options:

- a. Agile
- b. Basic
- c. CMMI
- d. Scrum (By Default)

For Azure DevOps project creation process, pls refer to [Azure DevOps Project Creation](#)

**Note:**

1. Three services namely Repos, Test Plans and Artifacts will be disabled, the only option user can enable/disable is Boards.
2. The team project name is neutrally based on island name which is automatically selected team during the onboarding process.

## Azure DevOps Team Project Access Types

The below are the access types depending on which team the person is added in the specific Azure DevOps team project, all three teams described below have the add, modify, and delete items permissions within the team project and can create service connections. The *Admin* team and *User* team automatically will be a part of the default team *Team*.

1. The default access type of Azure DevOps project is team *Team*.

Total 3		New Team	
Name ↓	Description		Members
UA Team Name Admin	Members of this group can add, remove members from group Team Name Admin and Team Name User.		5
UT Team Name Team	Default The default project team.		3
UU Team Name User			2

2. When the user requests a new project to be created in Azure DevOps, then the DevSecOps team will add the person(people) mentioned in the request ticket to the *Admin* team. Members of *Admin* team can add, remove members from *Admin* team and *User* team.

Total 3		New Team	
Name ↓	Description		Members
UA Team Name Admin	Members of this group can add, remove members from group Team Name Admin and Team Name User.		5
UT Team Name Team	Default The default project team.		3
UU Team Name User			2

3. The people within *Admin* team can then add new members to the *User* team and the *Team* team or can remove current members from the *User* team and the *Team* team.

Total 3		New Team	
Name ↓	Description		Members
UA Team Name Admin	Members of this group can add, remove members from group Team Name Admin and Team Name User.		5
UT Team Name Team	Default The default project team.		3
UU Team Name User			2

Available Pipeline Agent Capabilities

These are standard capabilities pre-installed on BMO agent pool.

Preinstalled capabilities	Versions/values	Agent.OS	Available in
JDK	8, 11, 17	<ul style="list-style-type: none"><li>Linux</li><li>Windows_NT</li></ul>	<ul style="list-style-type: none"><li>Pipeline template - Gradle, Maven, NPM</li></ul>
	11	<ul style="list-style-type: none"><li>Darwin</li></ul>	<ul style="list-style-type: none"><li>Pipeline template - mobile</li></ul>
Gradle	7.4.2	<ul style="list-style-type: none"><li>Linux</li><li>Windows_NT</li><li>Darwin</li></ul>	<ul style="list-style-type: none"><li>Pipeline template - Gradle</li></ul>

Maven	3.8	<ul style="list-style-type: none"> <li>Linux</li> <li>Windows_NT</li> <li>Darwin</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - Maven</li> </ul>
NodeJs	14, 16	<ul style="list-style-type: none"> <li>Linux</li> <li>Windows_NT</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - NPM</li> </ul>
	12 (temporary), 14, 16	<ul style="list-style-type: none"> <li>Darwin</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - mobile</li> </ul>
MsBuild Visual Studio Build Tools	'15.0' - to select Visual Studio 2017  '16.0' - to select Visual Studio 2019  '17.0' - to select Visual Studio 2022	<ul style="list-style-type: none"> <li>Windows_NT</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - msbuild</li> </ul>
NuGet	'4.x'	<ul style="list-style-type: none"> <li>Windows_NT</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - msbuild</li> </ul>
Python	3.x	<ul style="list-style-type: none"> <li>Linux</li> <li>Windows_NT</li> <li>Darwin</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - All</li> </ul>
SonarScanner		<ul style="list-style-type: none"> <li>Linux</li> <li>Windows_NT</li> <li>Darwin</li> </ul>	<ul style="list-style-type: none"> <li>All Pipeline template - All</li> </ul>
Xcode	12, 13	<ul style="list-style-type: none"> <li>Darwin</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - mobile builds</li> </ul>
Android SDK Platform	SDK version 19-31	<ul style="list-style-type: none"> <li>Darwin</li> </ul>	<ul style="list-style-type: none"> <li>Pipeline template - mobile builds</li> </ul>

## Azure DevOps Training Material

Below are some of the online learnings available.

A full list of all online learning for Azure DevOps is available [here](#).

The learning path to help prepare for [Exam AZ-400: Designing and Implementing Microsoft DevOps Solutions](#).

<a href="#">Getting Started with Azure DevOps</a>
<a href="#">Introduction to Azure DevOps</a>
<a href="#">Microsoft Azure Fundamentals: Describe core solutions and management tools on Azure</a>
<a href="#">Explore Azure Automation with DevOps</a>
<a href="#">Work with Azure Repos and GitHub</a>
<a href="#">Manage Git repositories</a>
<a href="#">Explore Azure Pipelines</a>
<a href="#">Describe pipelines and concurrency</a>
<a href="#">Manage Agile software delivery plans across teams</a>

## GitHub Enterprise Cloud

GitHub Enterprise Cloud (GHEC) is being selected as the source code repository for BMO and it is going to replace Bitbucket and TFS (Azure DevOps Servers) for source code management. All source codes on Bitbucket and TFS (Azure DevOps Servers) need to be migrated to GitHub Enterprise Cloud.

The following GitHub features are not available to GitHub BMO organizations:

- Personal Access Token creation is restricted in BMO environment.
- GitHub Action is a restricted feature, Azure DevOps Pipeline should be used.
- Fork repo is not allowed due to potential of bypassing access control.

Note: BMO EMU user cannot access non-BMO repos and BMO only allows EMU user to use BMO repos. Please refer to the below guidelines for GitHub usage within BMO.



- [GitHub Security Guidelines](#)

Training Material


- [GitHub Training Material](#)


## GitHub Enterprise Managed Users (EMU)

GitHub Enterprise Managed Users is a feature offered by GitHub to Enterprises. With Enterprise Managed Users, the control of user accounts of GitHub enterprise members will be handled by enterprise identity provider (IdP). Users assigned to the GitHub Enterprise Managed User application in your enterprise IdP are provisioned as new user accounts on GitHub.

The usernames of your enterprise's managed user accounts and their profile information, such as display names and email addresses, are set by through your enterprise IdP and cannot be changed by the users themselves.

In BMO, the enterprise identity provider is **BMO Azure AD**. BMO Azure AD is configured to use SCIM protocol for mapping users from the application access group on BMO Azure AD to GitHub. This synchronization is scheduled to happen regularly, but does not take place instantly.

 For any BMO ID (*user account or service account*) to be available on GitHub, it is a pre-requisite for it to be also available on BMO Azure AD.

 Each user must make sure to have their primary account associated with their email address for correct mapping to GitHub.

Service account needs to be email-enabled before it is synchronized with Azure AD. There is a BMO process to get service account email-enabled, this process comprises of initiating a service request for adding email to an existing service account to make it email-enabled.

### *Abilities and restrictions of managed user accounts*

Managed user accounts can only contribute to private and internal repositories within their enterprise and private repositories owned by their specific user account. Managed user accounts have read-only access to the rest of the GitHub community. These visibility restrictions on users and access restrictions on content apply to all requests, including API requests.

- Managed user accounts cannot be invited to organizations or repositories outside of the enterprise, nor they can be invited to other enterprises.
- Outside collaborators are not supported by Enterprise Managed Users.
- Managed user accounts cannot create issues or pull requests in, comment or add reactions to, nor star, watch, or fork repositories outside of the enterprise.
- Managed user accounts can view all public repositories on [GitHub.com](#), but cannot push code to repositories outside of the enterprise.
- Managed user accounts and the content they create is only visible to other members of the enterprise.
- Managed user accounts can not follow users outside of the enterprise.
- Managed user accounts cannot create gists or comment on gists.
- Managed user accounts cannot install GitHub Apps on their user accounts.
- Other GitHub users cannot see, mention, or invite a managed user account to collaborate.
- Managed user accounts cannot fork repositories from outside of the enterprise and internal repositories.
- Only private and internal repositories can be created in organizations owned by an enterprise with managed users, depending on organization and enterprise's repository visibility settings.

## GitHub Organization

Three organizations are currently offered by BMO, namely **BMO-Prod**, **BMO-Sandbox** and **BMO-Community**. BMO-Prod is to store all the BMO production source code including scripts and code being used in testing. BMO-Sandbox is for doing experiment and proof of concept. BMO-Community is owned by BMO Enterprise Architect and is used for collaboration and hosting enterprise pattern.

Please refers to the [GitHub Security Guideline](#) for more information.

### GitHub Restriction within BMO

Source code is classified as CONFIDENTIAL based on BMO data classification. There are security requirements for the platform to be complied. Information has been provided in the [GitHub Security Guideline](#). It is important for users to understand the security requirement to reduce the risk exposure.

GitHub BMO organization are only be accessible from approved location and it is expected to be accessed from BMO premises. Any external access is subject to assessment and approval.

## GitHub Repo Access Type

GitHub repo is being provisioned by local team within GitHub.

The below are the access types depending on which team the person is added.

--	--	--

Team name	Level of access	purpose
TeamName(Parent Team)	Read	Read and clone repositories. Open and comment on issues and pull requests.
TeamName_User	Write	Triage permissions plus read, clone and push to repositories.
TeamName_Admin	Advance maintain	Write permissions plus manage issues, pulls requests, repository settings and view secret scanning alerts/Dependabot alerts

**Note:** Due to security concerns, permanent full repo admin is not offered.

## GitHub Training Material

Below are some of the online learning currently available for GitHub.

A full list of all online learnings for GitHub is available [here](#).

<a href="#">Introduction to GitHub</a>
<a href="#">Introduction to Git</a>
<a href="#">Introduction to GitHub's Products</a>
<a href="#">Get started with Git and GitHub in Visual Studio</a>
<a href="#">Manage Git branches and workflows</a>
<a href="#">Structure Your Git Repo</a>
<a href="#">Explore Git Hooks</a>
<a href="#">Migrate Your Repository by Using GitHub Best Practices</a>

## Migrate to DevSecOps

There are following types of migrations for transitioning to DevSecOps:

### 1. TFS/Azure DevOps Server Project Migration

- a) In this type of migration, the latest commit of the source code from selected branches will be migrated.
  - Code will be copied to Bitbucket for DLP scanning.
    - DevSecOps team will assist the process.
- b) Artifact storage will be moved to Artifactory (Separate onboarding process with CD team).
- c) CIO Team will recreate pipeline manually by making use of pipeline template. Please refer to [Pipeline Templates](#) for using templates.

### 2. AWS On-demand pipeline Migration

Repo (Bitbucket) will be migrated based on DLP clearance (full history or latest commits of selected branches).

Once the repository is migrated and verified, DevSecOps team will enable AWS CDK workflows on GitHub for the repo.

### 3. Bitbucket/Bamboo Project Migration

Repo will be migrated based on DLP clearance (full history or latest commits of selected branches).

CIO Teams will recreate their build pipelines manually on their Azure DevOps projects by making use of pipeline template. Please refer to [Pipeline Templates](#) for using templates.

### 4. Repository only Migration

Active repositories that have no corresponding build plan and inactive repositories (untouched since 2021/11/01) will be migrated in batch based on DLP clearance (full history or latest commits of selected branches).

Source code migration type:

### a) Full Repository Migration-

Bitbucket source code repositories can be migrated with history and pull requests.

- DLP team clearance is required for allowing the repository to be migrated in full.
- Existing Bitbucket access groups of the repository will be carried over to GitHub.
- Due to setup limitation, DevSecOps Team will handle the migration process

### b) Latest Commit Only without history-

This is the type of TFS project migration. Also for Bitbucket repository that receives limited DLP clearance which only latest commits can be migrated.

- Only source code of selected branches will be migrated.
- DevSecOps Team will provide instruction and assistance for CIO Teams to migrate on their own.

#### **Important points-**

1. Any content to be copied to Azure DevOps Services and GitHub Enterprise Cloud cannot contain any secrets.
2. All source code migration would require clearance from DLP team(Contact info: [dlp.program@bmo.com](mailto:dlp.program@bmo.com)).  
-Source code repo that contains secrets may not be allowed to be migrated with history.
3. GitHub Enterprise Cloud has size limitations on individual file and overall repo.

Please see [Pre-Migration Checklist](#) on additional details on preparing Bitbucket repos for migration.

#### **GitHub Team**

Teams are groups of organization members that reflect your company or group's structure along with cascading access permissions and mentions.

- Organization owners and team maintainers can grant teams admin, read, or write access to organization repositories.
- Organization members can send a notification to an entire team by mentioning the team's name.
- Organization members can also send a notification to an entire team by requesting a review from that team.
- Organization members can request reviews from specific teams with read access to the repository where the pull request is opened.
- Teams can be designated as owners of certain types or areas of code in a CODEOWNERS file.

**Note:** There is a conflict on using Azure AD Group vs GitHub Team. Due to the lack of Azure AD security group governance today, GitHub Team is being used for permission provisioning. Neither Azure AD security group nor AD group is not available for use on GitHub. Please refer to [Azure AD vs GitHub Teams Discussion](#) for further information.

#### **Migration Consideration: Active vs Inactive**

Definition:

**Active Repository:** Repository that has commit within F22 (on or after Nov 1, 2021 00:00 EST) is considered as active.

**Inactive Repository:** Repository that the last commit is before F22 (before Nov 1, 2021 00:00 EST) is considered as inactive.

Recommended Approach:

Pipeline migration requires development effort to recreate the pipeline. In order to minimize the effort of recreating pipeline, it is recommended not to do any pipeline recreation for build plan that is related to inactive repository. The DevSecOps team will upload inactive repositories in batch and notify the owners when upload is completed.

#### **Migration of AWS Workload**

AWS CDK(Cloud Development Kit) workload is supported by the DevSecOps Environment. The migration from Bitbucket AWS on-demand pipeline to AWS Workload of DevSecOps Environment is handled on the repo migration level. Before the repo migration, this migration path needs to be clearly identified by CIO team and DevSecOps team will enable the AWS workload for the repo after the repo migration.

Please refer to <https://bmo.atlassian.net/wiki/spaces/AUTOMATE/pages/378175524/CloudFormation+CFN+Workflow+Guide+for+Github+Actions> guide for CFN workflow and <https://bmo.atlassian.net/wiki/spaces/AUTOMATE/pages/243272458/CDK-v2+Guide+for+Github+Actions> for CDK workload guide on how to use the AWS workflow on GitHub.



For any issues with AWS workflow or deployment please raise a ServiceNow request to cloud platform team by following <https://bmo.atlassian.net/wiki/spaces/HCPSUP/pages/392359038/Cloud+Platform+Support+Engagement+--+SNOW+Process> .

#### **Pipeline Migration**

Azure Pipelines is the primary pipeline orchestration platform in the DevSecOps environment and Pipeline as code is the orientation in BMO DevSecOps framework. A YAML file in the repository would be used to host the pipeline definition. There are the following benefits in using YAML file:

- The pipeline is versioned with source code. It follows the same branching structure. You get validation of your changes through code reviews in pull requests and branch build policies.
- Every branch you use can modify the pipeline by modifying the YAML file.
- A change to the build process might cause a break or result in an unexpected outcome. Because the change is in version control with the rest of your codebase, you can more easily identify the issue.

Unfortunately, there is no direct migration path from Bamboo pipeline to Azure Pipeline YAML file. All the existing Bamboo build needs to be recreated in the new DevSecOps environment. Template will be provided to facilitate reusable pattern and simplify migration efforts.

In the below section, the pipeline migration will be discussed.

## Migrate to Azure Pipeline

*Azure DevOps projects is a prerequisite for pipeline migration.*

The following sections covers the onboarding, access management and pipeline creation with Azure Pipelines.

- [Azure DevOps Project](#)
- [Pipeline Recreation](#)

Since pipeline code (YAML file) is part of the source code and GitHub is the only authorized source code repository in this environment, so *repository migration is also a prerequisite of the pipeline migration.*

Azure DevOps Project

It is important to have Azure DevOps project setup before the pipeline migration and this section walks through some important aspects of Azure DevOps Project setup.

Azure DevOps Project created by the DevSecOps team will be based on the below received information from the client.

1. CIO
2. LTO
3. STO
4. Teams Admin(s)
5. AppCAT ID
6. Is Work Item (Boards) required or not?

If yes, then please select the type of board methodology required from the below options:

- a. Agile
- b. Basic
- c. CMMI
- d. Scrum (By Default)

### **Note-**

1. Three services namely Repos, Test Plans and Artifacts will be disabled, the only option user can enable/disable is Boards.
2. The team project name is neutrally based on island name which is automatically selected team during the onboarding process.

## Azure DevOps Project Creation

The user can request new Azure DevOps project through onboarding process. Below is the information on using DevOps Tools Onboarding Portal ([DTOP Portal](#)) to onboard an empty project to Azure DevOps Services projects, kindly refer to the link.

[How to Onboard Azure DevOps Services \(ADO\) Project Using DTOP](#)

## Managing User Access for Azure DevOps Projects

This section explains how to manage user access in Azure DevOps Services. Project Admin should check if Team member belongs to the Security Group "ROL-AAD-PR-Azuredevops-dev-users" (under My Apps > My Groups) using the below link before adding user to Azure DevOps Project <https://account.activedirectory.windowsazure.com/r/#/groups>

If they are a member of the group then the Project Admin can follow [How to add user to Azure DevOps project](#) to add the user to Team's Project.

If they are not a member of the group then the Project Admin/user who wants to be added to Team's Project can submit a ServiceNow request by referring to [How to submit an INC ticket to add users to local ADO project team](#) .

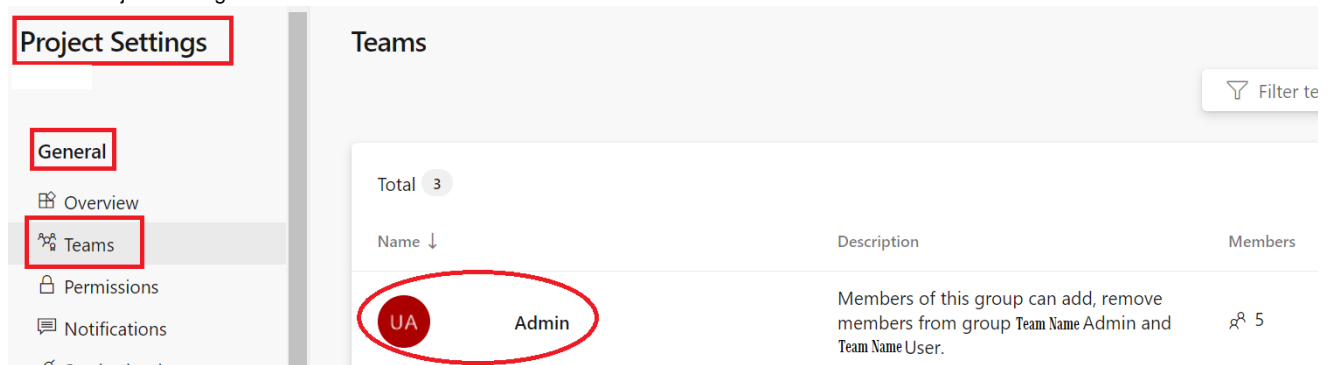
How to add user to Azure DevOps project

*The user having access type as "Admin" only can onboard other members in Azure DevOps project.*

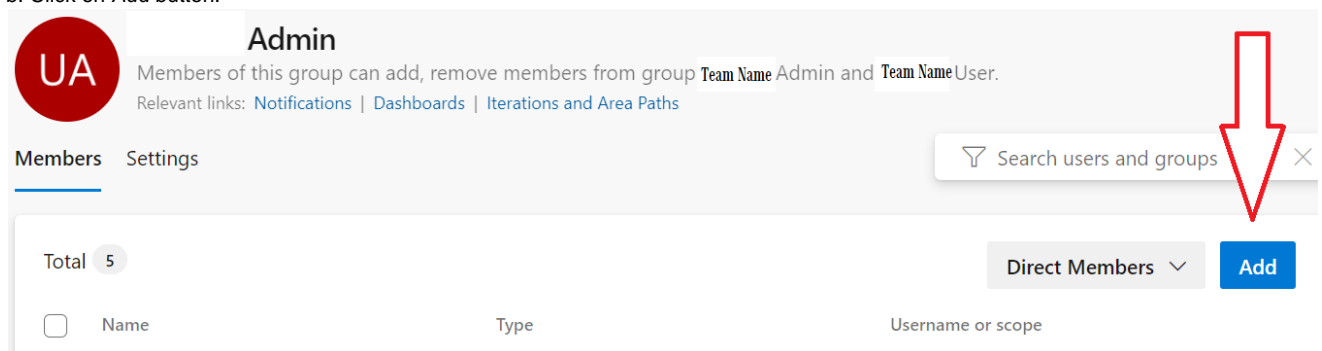
The process for onboarding other members in Azure DevOps project is as below:

### 1. Adding user for Admin access type

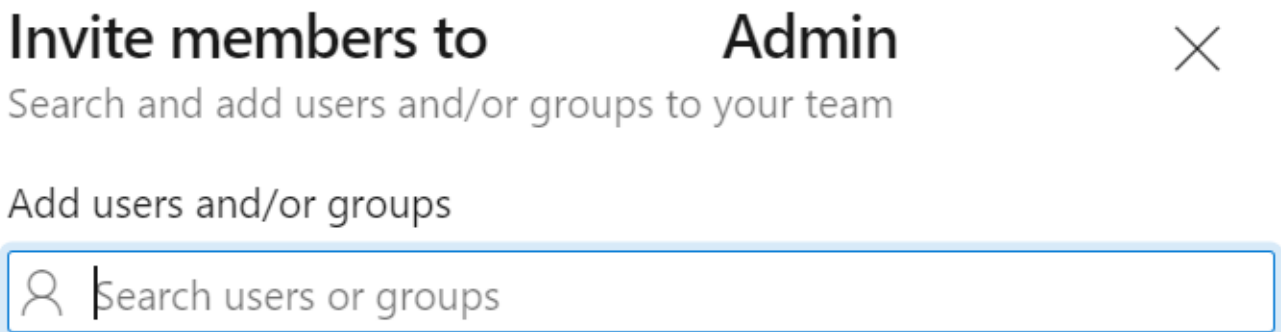
a. Go to Project Settings and select *Admin* under *Teams* section of *General* tab.



b. Click on *Add* button.



c. Search the *e-mail* of the member you want to add and click *Save*.



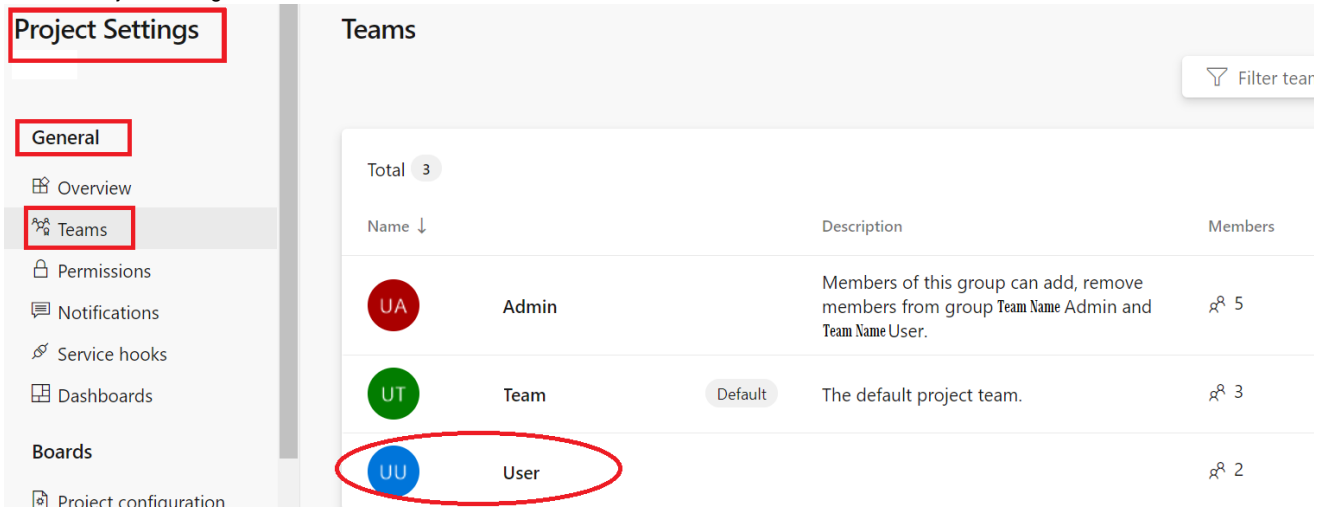
SEARCH BY  
E-MAIL

Cancel

Save

## 2. Adding user for User access type

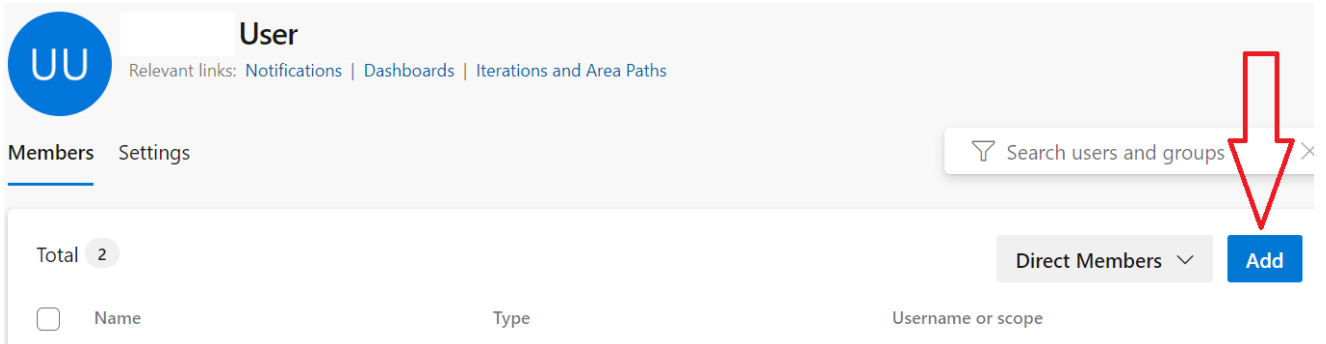
a. Go to Project Settings and select *User* under *Teams* section of *General* tab.



The screenshot shows the 'Project Settings' interface. On the left, a sidebar contains a 'General' tab and a 'Teams' section. The 'Teams' section is expanded, showing a list of teams. The 'User' team is highlighted with a red circle. The 'Teams' section also includes a 'Filter team' button and a 'Total' count of 3.

Name ↓	Description	Members
UA Admin	Members of this group can add, remove members from group Team Name Admin and Team Name User.	5
UT Team	The default project team.	3
UU User		2

b. Click on *Add* button.



The screenshot shows the 'User' profile page. The 'User' profile is displayed with a blue circle icon containing 'UU'. Below the profile, there are tabs for 'Members' and 'Settings'. The 'Members' tab is selected, showing a list of members. The 'Add' button is highlighted with a red arrow.

Relevant links: [Notifications](#) | [Dashboards](#) | [Iterations and Area Paths](#)

Search users and groups

Total 2

Direct Members


Add

c. Search the *e-mail* of the member you want to add and click *Save*.

Invite members to User

Search and add users and/or groups to your team

## Add users and/or groups

 Search users or groups



SEARCH BY  
E-MAIL

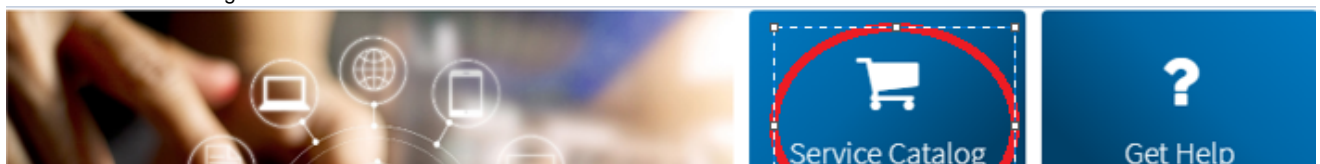
Cancel

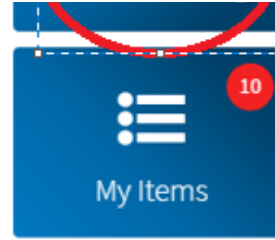
Save

How to submit a ServiceNow request to add users to local ADO project team

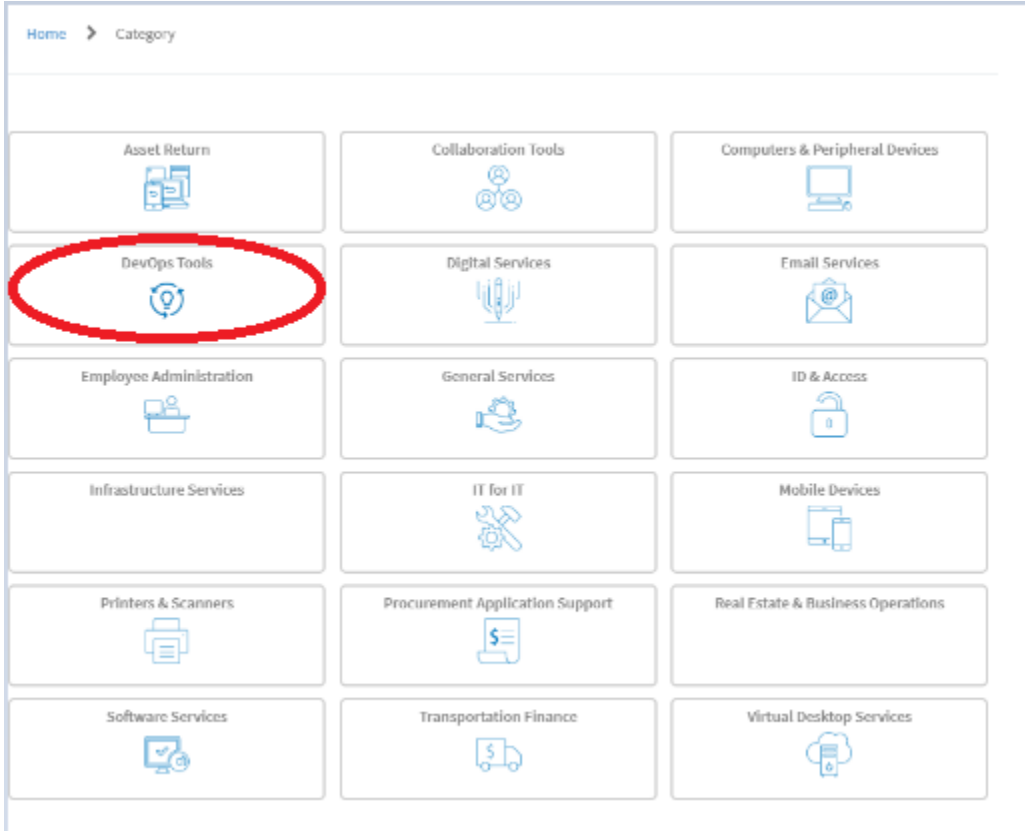
In the event where the Project Admin is unable to add team members to team's project using [How to add user to Azure DevOps project](#) , alternatively the Project Admin/user who wants to be added to Team's Project can submit a Service Now request following the below steps:

1. Launch the ServiceNow Catalog homepage, [https://bmogc.service-now.com/sp?id=sc\\_category](https://bmogc.service-now.com/sp?id=sc_category).
2. Click on Service Catalog.





3. Select DevOps Tools from the Icon list.

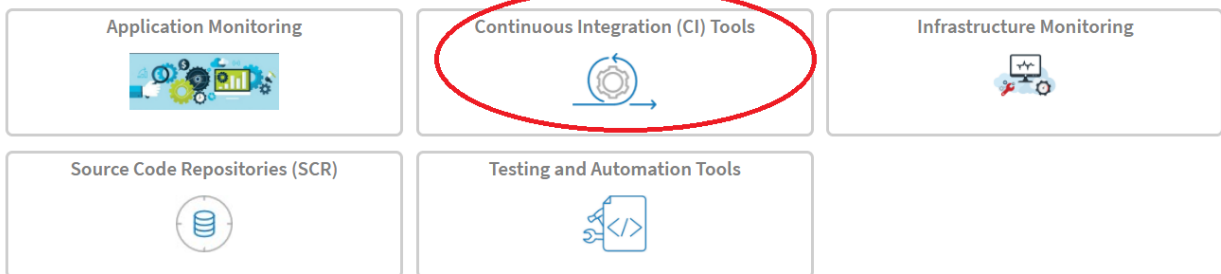


4. Select Continuous Integration (CI) Tools from the list.

[Home](#) > [Service Catalog](#) > DevOps Tools

## DevOps Tools

Request support for Enterprise DevOps Tools.



5. Select "Azure DevOps Services (ADO)" from the drop-down list and fill in the requested information to submit the ticket.

Section	Field(s)	Details	Notes
Employee Information	Requested For	Mandatory – prefilled from user profile	



	Contact Number	Mandatory – enter contact number if not prefilled	
	Enterprise ID	Prefilled	
	Email	Mandatory – prefilled from user profile	
	Department/Business Group	Mandatory – prefilled from user profile	
	PR Code	Mandatory – provide a PR code associated to the request	
	Job Title	Mandatory - prefilled from user profile	
Request Information	Please select application name	Azure DevOps Services (ADO)	
	Request Type	General Request	
	Summary of the request	Mandatory -	
	Detailed description of the request	Mandatory - Email Addresses of Team Members - Team's Project Name - Team's Group - (Admin , User, Team)	
Add Attachment		If "Requester" is not Project Admin, email from Project Admin approving the request is required to be attached	

6. Upon completion - submit the request, a service request ticket will be generated.

7. If you have multiple request items, please submit multiple ServiceNow RITM tickets.

**Please note that our team is supporting one item per ticket. If you are seeking support for multiple items, please create multiple RITM tickets.**

- An item is defined as a simple request for one project. i.e. add Jira custom fields and update workflow should have two RITM tickets created.

**Here is how:**

1. After fill in the first request item, click "Add to Cart" instead of using "Order Now".

The screenshot shows the ServiceNow request form for Continuous Integration (CI) Tools. The form includes fields for Employee Information, Requested For (Grace Yu), Contact Number (+1416-513-5514), Enterprise ID (office\gyu08), Email (Grace1.Yu@bmo.com), Department / Business Group, and PR Code. On the right side, there is a summary box with a dropdown menu showing '1' and two buttons: 'Add to Cart' (circled in red) and 'Order Now'.

2. You will see the green highlighted information bar together with all the filled-in information from the first requested item.

The screenshot shows the same ServiceNow request form, but now a green confirmation bar is visible at the top, stating: "Your item has been added to your Cart. To make changes to the items in your cart, click View Cart". The 'Add to Cart' button in the summary box is also circled in red.

3. Modify the pre-filled information and "Add to Cart" again for your 2nd requested item.

4. Keep adding requested items into your cart until all the items are added. You can view all added items by clicking "View Cart".

5. Click "Order Now".

6. You will get one REQ number containing multiple RITM tickets which can be followed up separately.

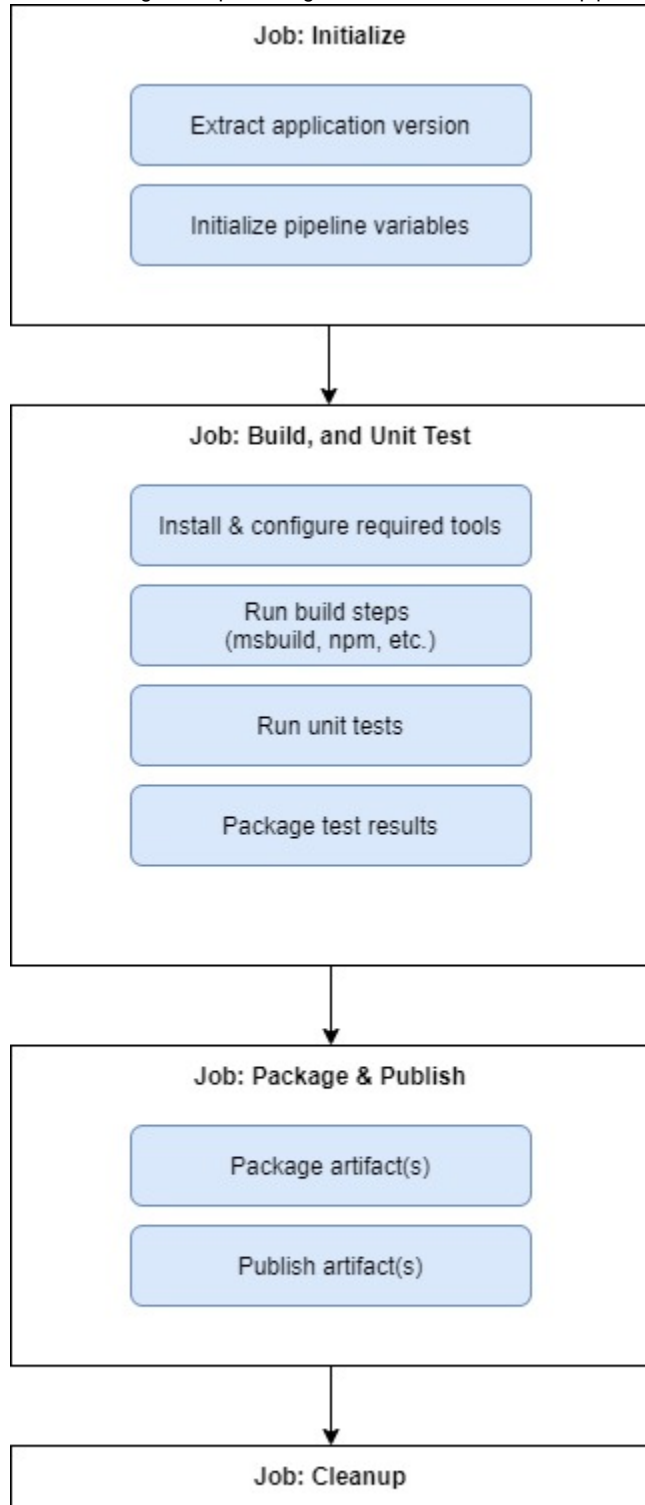
7. Each RITM ticket will get its own work effort estimation/charge against the PR code provided.

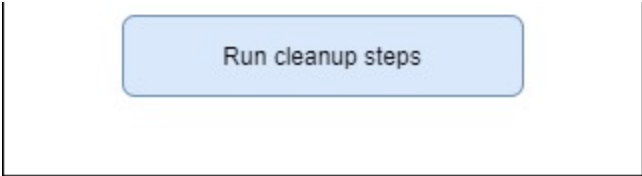
Azure Pipelines implementation will be leveraging templates which allows to define reusable content, logic, and parameters. With this approach, we will develop build pipeline template framework that is extensible and easy to consume.

Build pipeline template framework will consist of :

- Core build modules that are required to run with every pipeline and can not be overwritten by the developers. For example: pipeline initialization steps or certain required security checks.
- Technology stack module that will be inserted into the pipeline depending on user specified parameters. For example: if buildType is set to msbuild, then pipeline will run Visual Studio msbuild steps.
- Pipeline template files will be stored in a separate GitHub repo with branch protection rules enforced.

The below diagram depicts a high level behavior of the build pipelines:





#### Consuming pipeline templates

To consume pipeline templates, users need to add a reference to repository resource where pipeline template files reside. Users will also be required to pass parameters to the pipeline templates to configure the pipeline as needed.

```
resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo
      endpoint: github.com_sa-onboarding_bmogc-Test_Island
      ref: refs/tags/v0.4-alpha
  stages:
    - template: pipeline_templates/build.yml@templates
      parameters:
```

#### Onboarding new technology stack

Request to DevSecOps Modernization project should be made for new technology requirement. (Please refers to [link](#)). When a request is made to support new technology stack by the pipelines, the following steps are taken by DevSecOps team:

- Review hardware and software requirements to ensure that support of requested technology stack is possible by the pipeline.
- Make changes to pipeline agents, if needed, to onboard new technology.
- Stage pipeline template changes to onboard new technology.
- Test changes in staged pipeline template.
- Create pull request to merge changes to master branch.
- Once PR changes reviewed and approved, merge changes to the master branch.

#### Downstream System Connections

It is a prerequisite that the downstream connections should be setup as *service connections* (when available) in Azure DevOps team project for pipeline usage. The project team can contact DevSecOps team for any assistance required.

The below is the list of downstream connections which will use service connections to connect.

1. GitHub
2. Artifactory
3. SonarQube
4. OpenShift
5. Azure Resource Manager

The following systems will be connected by methods other than service connections.

1. Ansible

#### NOTE:

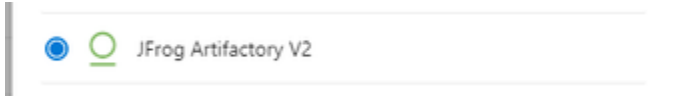
- **API token is recommended when creating service connection. Username/Password should be avoided.**
- **CIO team is responsible to have proper setup on their downstream connectivity. DevSecOps platform team will not provide any default credential on the platform level for any downstream connectivity.**

#### GitHub

Shared service connection `github.com_sa-onboarding_bmogc-<project name>` is created during project onboarding and should be used for all pipeline connections. There is no need to create another service connection for GitHub.

#### Artifactory

Jfrog Azure DevOps extension is available on Azure DevOps BMO organization. When creating service service connection JFrog Artifactory v2 should be selected.



**NOTE:**

- Azure DevOps Services does not have direct connectivity to Artifactory QA instance which is located on-prem. Connectivity can not be verified during service connection setup.
- It is recommended not to specify any repo name on the Artifactory URL when service connection is being created. Repo name should be specified on the task level and the service connection can be used for any type of Artifactory connection of the project as long as the credential can access all the necessary repos.

**SonarQube**

SonarQube Azure DevOps extension is available on Azure DevOps BMO organization. Onboarding SonarQube Enterprise instance is a prerequisite for DevSecOps pipeline using SonarQube.

**OpenShift**

Opensift Azure DevOps extension is available on Azure DevOps BMO organization. OCP4 cluster should be used with Azure DevOps pipeline.

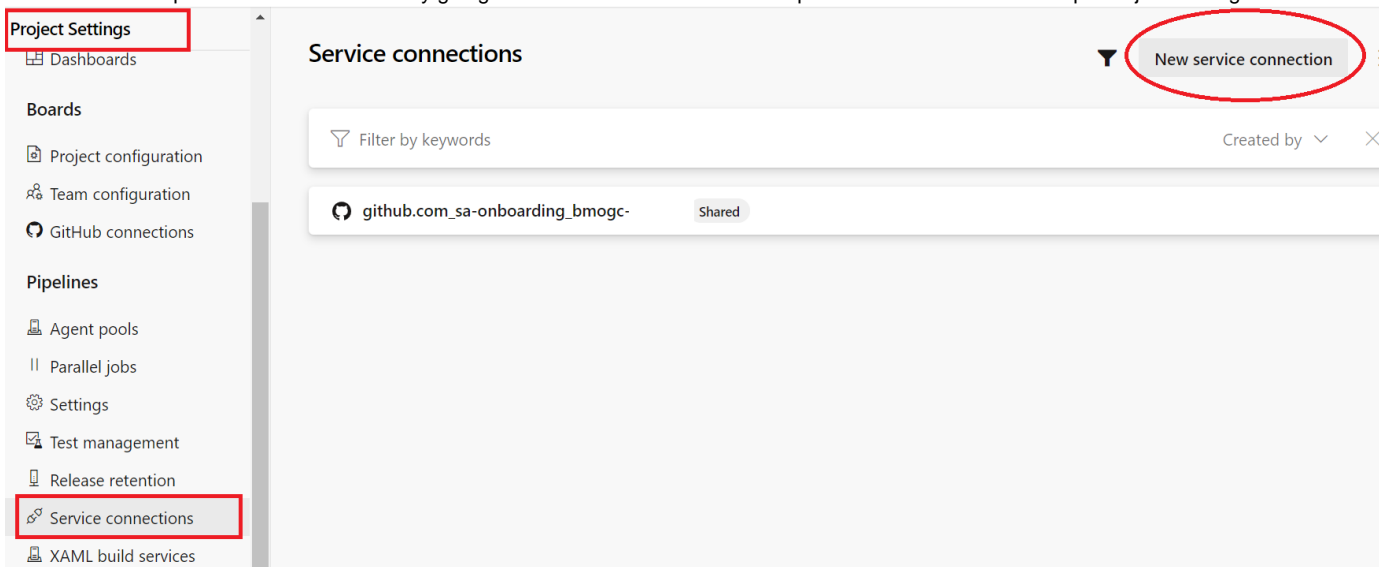
**Azure Resource Manager**

Azure Resources Manager is required for any connection to Azure resources. The key to connect to Azure resources does not get shared with CIO team. The CIO team needs to consult DevSecOps leads for creating connection to Azure resources.

**Service Connection Creation in Azure DevOps**

*Note- The Azure DevOps project user should avoid using personal account's credentials, instead the user should use service account credentials.*


The user can setup the service connection by going to Service Connections in the Pipeline section of Azure DevOps Project Settings.



Click on **New service connection** button to select from a list of connection types click **Next** button. And the new connection will appear under the **Service Connections**.


## New service connection

Choose a service or connection type




Search connection types


☒

 Azure Classic


☐

 Azure Repos/Team Foundation Server


☐

 Azure Resource Manager


☐

 Azure Service Bus


☐

 Bitbucket Cloud

☐

 Chef

☐

 Docker Host

### How to create a New Pipeline

Follow the below steps to create a new pipeline leveraging pipeline template:

1. Browse to [Azure DevOps organization](#), then to your Azure DevOps team project.
2. Click on **Pipelines**.



B

BMO-Prod

New organization

You have been assigned Stakeholder access and will experience limited

BMO-Prod


Projects

My work items

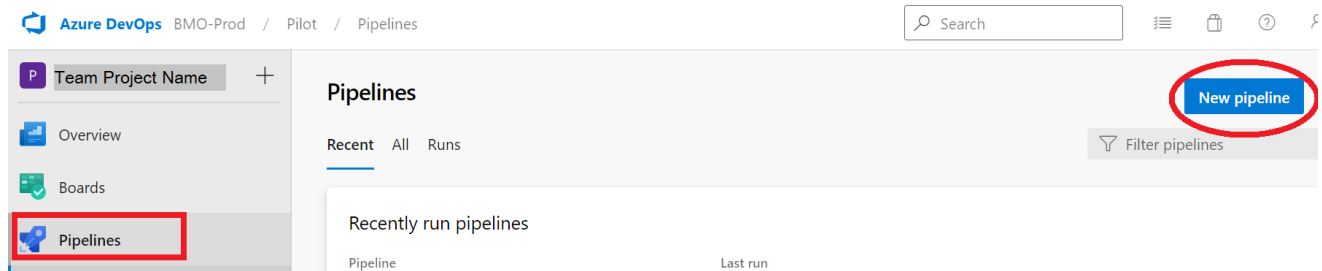
My pull requests

J

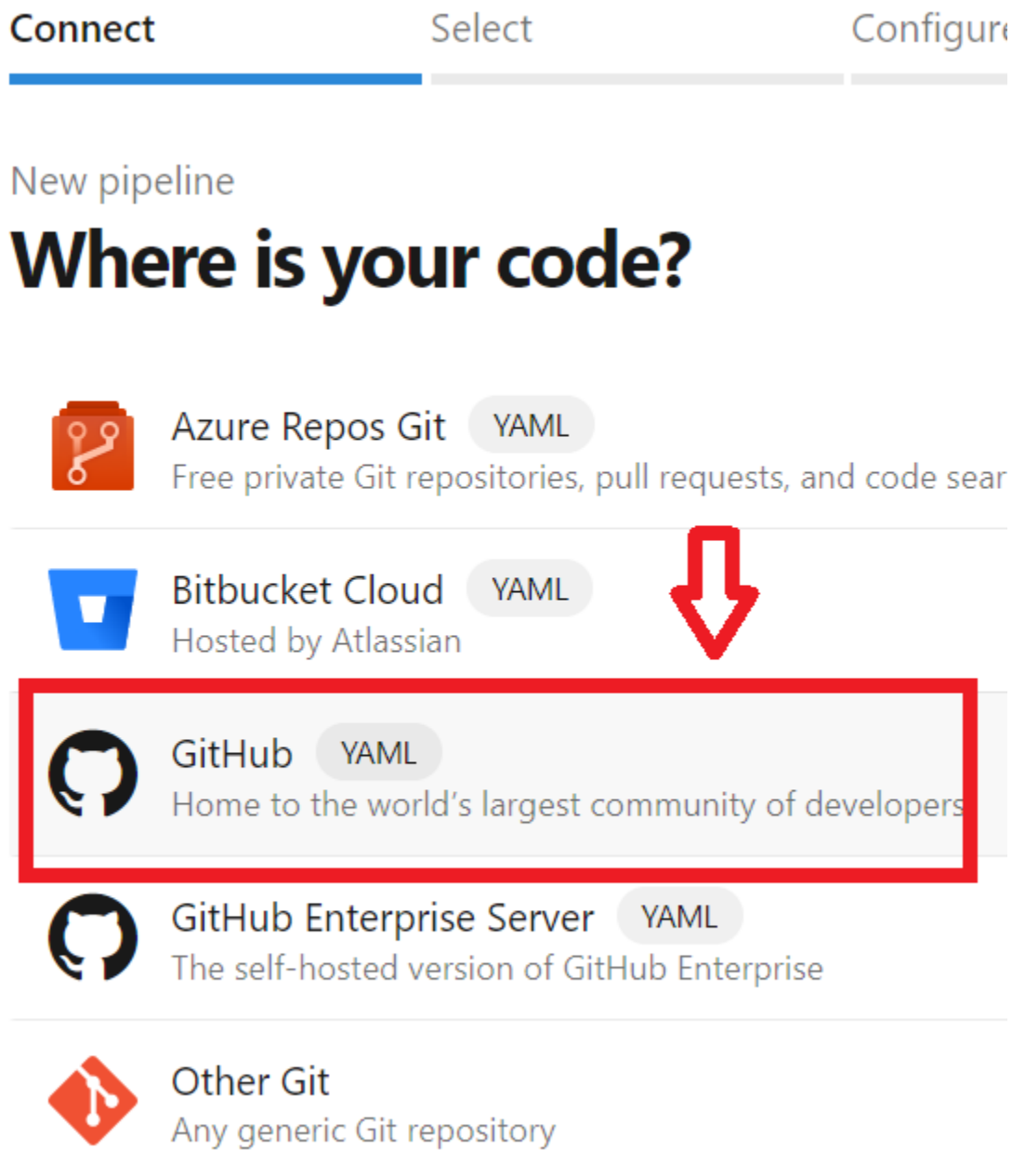
Team Project Name



3. Click on **New pipelines**.



4. On "Where is your code?" page, click on **GitHub**.



5. On "Select a repository" page, click on your GitHub repository.
6. On "Configure your pipeline" page, click on Starter pipeline or pick Existing Azure Pipelines YAML file, if you already have YAML file to pick. The contents of the YAML file will have to be replaced.
7. Next, replace the contents of the YAML file using one of the samples provided. Make sure your pipeline file has reference to pipeline template repository. Add a reference to pipeline template file and pass parameters to that template. Please refer to the complete samples of the pipeline template based on the type of the build on [Pipeline Templates](#).

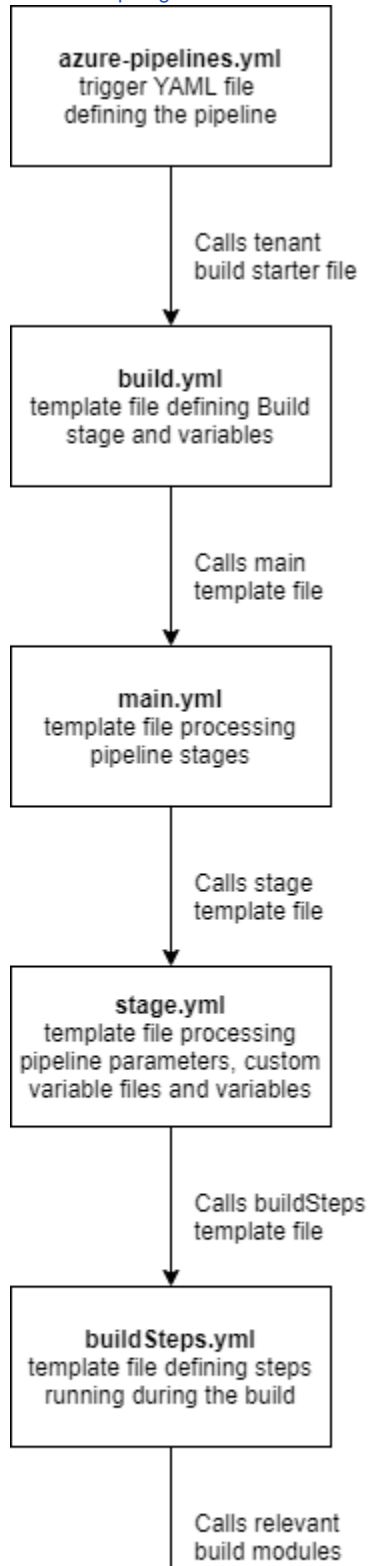
8. Save the pipeline changes and run the pipeline.

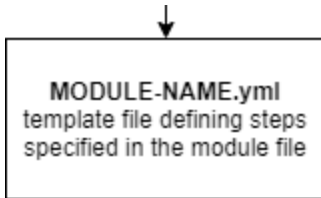
### Pipeline Templates

Latest release of template ref: `refs/tags/v1.1.0`

Please refer to [Release Notes](#) for list of changes

Azure Pipeline templates let you define reusable content, logic, and parameters to control what is allowed in a pipeline. Azure Pipeline templates reside in [https://github.com/BMO-Prod/template\\_repo](https://github.com/BMO-Prod/template_repo). The pipeline templates consists of the following structure:





#### Build pipeline steps

buildSteps.yml file that defines the steps run during build pipeline:

- Get source code (application code and pipeline templates).
- Load pipeline secrets from Global Azure KeyVault containing secrets that are common for all Azure Pipelines.
- Load pipeline secrets from core Azure KeyVault containing secrets managed by DevSecOps for the application.
- Load pipeline secrets from application Azure KeyVault containing secrets managed by application developers for the application.
- Run pre-build steps, if defined by the application pipeline.
- Run build prerequisite steps defined by "buildType" parameter specified in application pipeline.
- Run build steps defined by "buildType" parameter specified in application pipeline.
- Run post-build steps, if defined by the application pipeline.
- Run pre-publish steps, if defined by the application pipeline.
- Run steps that create release notes, if defined "buildReleaseNotes" parameter is set to True and build was not triggered by the pull request.
- Run steps that publish artifacts to Azure DevOps, if defined "publishAzureDevOps" parameter is set to True and build was not triggered by the pull request.
- Run steps that publish artifacts to Artifactory, if defined "publishArtifactory" parameter is set to True and build was not triggered by the pull request.
- Run post-publish steps, if defined by the application pipeline.
- Run nDepend steps, if defined "buildRunNDepend" parameter is set to True.

#### Build modules

Azure pipeline templates supports the following build modules:

- **Gradle** module defining how Gradle builds. See [Gradle Build](#) for details.
- **Maven** module defining how Maven builds. See [Maven Build](#) for details.
- **Mobile Template** module defining how mobile application (both iOS and Android) builds. See [Mobile Template Build](#) for details.
- **NPM** module defining how NPM builds. See [NPM Build](#) for details.
- **Ant** module defining how Ant builds. See [Ant Build](#) for details.
- **VSBuild** module defining how Visual Studio or MSBuild builds. See [Visual Studio Build](#) for details.

Other references: [Ansible Deploy](#)  
[Additional Template Steps](#)

#### Ansible Deploy

- [Template Snippets](#)
  - [TemplateReference](#)
  - [PipelineFileReference](#)
- [Steps](#)
- [Template Parameters](#)
- [Sample Pipeline](#)
  - [Sample Pipeline Code](#)

#### Template Snippets

To use Ansible deploy, you will need to create pipeline YAML file. Pipeline YAML file will first need to include the reference to the Git repository hosting pipeline template that it is using:

**i** In Azure DevOps (*Project Settings->Service connections*), a GitHub service connection is created by default during the onboarding process. This service connection (*github.com\_sa-onboarding\_bmogc-ProjectName*) will be used for pipeline endpoint.

#### TemplateReference

```
resources:
  repositories:
    - repository: templates # alias for the template reference
      type: github
```



```

    name: BMO-Prod/template_repo # name of the template repository
    endpoint: <Github Service Connection Name> # Replace with the
name of the GitHub service connection used to connect to template
GitHub repository
    ref: refs/tags/<tag_name> # name of the Git Tag. The latest
release tag is documented in page https://bmo01.atlassian.net/wiki
/spaces/DSOM/pages/18710536/Pipeline+Templates

```

After the pipeline template reference, you will need to include the YAML file from above mentioned GitHub repo using the below:  
**PipelineFileReference**

```

stages:
- template: pipeline_templates/release.yml@templates
  parameters:

```

## Steps

Ansible pipeline consists of the following steps:

1. Checkout the pipeline templates repo to *templates* directory.
2. [Use Python version task](#) to set Python version to v3.
3. Pipeline runs steps configured in *insertPreDeploySteps* parameter, this parameter is used by the users to run steps they need to run before running the deploy steps.
4. Pipeline runs step to call the values stored in the Variable groups (*defaultVariables* and *customVariables*).
5. Pipeline runs Ansible playbook by executing *autodeploy* script.
6. Pipeline runs steps configured in *insertPostDeploySteps* parameter, this parameter is used by the users to run steps they need to run after running the deploy steps.
7. Pipeline runs steps configured in *insertPreTestSteps* parameter, this parameter is used by the users to run steps they need to run before running the Test steps.
8. Pipeline runs steps configured in *insertPostTestSteps* parameter, this parameter is used by the users to run steps they need to run after running the Test steps.

## Template Parameters

Pipeline runs the above mentioned steps using configurable pipeline parameters prescribing how auto deploy pipeline should run:

Parameter	Required	Supported values	Description
agentPoolName	Yes	Prod	Specify the agent pool to use during the pipeline run.
agentOS	Yes	Linux	Specify the operating system the pipeline run requires. Default to "Linux".
jobTimeoutInMinutes	No		Specify the job timeout duration. Default value is 60 minutes.
defaultVariables:			Required for establishing connection to tower.
__default__:			
ansibleUsername	Yes		Specify Ansible Tower username.
ansiblePassword	Yes		Specify Ansible Tower user password.
ansibleInventory	Yes		Specify Ansible Tower Inventory name.
ansibleJobTemplate	No		Specify the name of the Job or Workflow Template to update and launch. Otherwise, name will be <i>ProjectName_InventoryName_autodeploy</i> .
ansibleProject	Yes		Specify Ansible Tower Project name.
ansibleRelease	Yes		Specify app release version. It must not contain any spaces.
ansibleApiHost	Yes		Example: <a href="http://xxx.bmoggc.net">xxx.bmoggc.net</a>
envName	Yes		Environment on the Team Project
displayName	No		Display name to be shown for this stage on Azure DevOps

customVariables:			Required to specify application team's specific operations.
Deploy:			Specify stage name which can be development, testing, production.
additionalAnsibleArguments	No		Specify Ansible Arguments specific to user setup.
		--extra_var_name	Specify any extra-vars to be set in the Job Template. Example: --installDir=/opt/deliverables/app.
		--tags	Comma-separated list of Playbook Task Tags to run. A null or value of all clears the JOB TAGS field in the Job Template and sets default value to all.
		--skip-tags	Comma-separated list of Playbook Task Tags to skip. Can be included along with the --tags arg (they are not mutually exclusive). A null or value of never clears the SKIP TAGS field in the Job Template and sets a default value to null.
		--monjob-stdout	Job/Workflow stdout will be copied to the ADO log. Argument --monjob must also be specified.

#### Sample Pipeline

A sample pipeline code is given below for reference:

#### Sample Pipeline Code

```
resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo
      endpoint: github.com_sa-onboarding_bmogc
      ref: refs/tags/v1.1.0

variables:
- group: AnsibleVar

stages:
- template: pipeline_templates/release.yml@templates
  parameters:
    agentPoolName: 'Prod'
    agentOS: 'Linux'
    jobTimeoutInMinutes: 180
  defaultVariables:
    __default__:
      envName: Dev
      ansibleUsername : $(tower_user)
      ansiblePassword : $(tower_password)
      ansibleInventory: 'DevOpsCITools_DTOP_Dev'
      ansibleJobTemplate: 'DevOpsCITools_SampleApp_Install'
      ansibleProject: 'DevOpsCITools_DTOP_Dev'
      ansibleRelease: 'Bamboo_Test_Release_Value'
      ansibleApiHost: ansible.bmogc.net
  customVariables:
    Deploy:
      additionalAnsibleArguments: '--install_host="cmtoldsdlcapp14.dev.bmocm.com" --install_dir="/apps/sdlctools/webdesign/testAnsible" --install_binary="DevOpsCITools-Virtual/DTOP/BMOAgileApp-5.0.2+DTOP+1.
```


```
zip"'
```

## Gradle Build

- [Template Snippets](#)
  - [TemplateReference](#)
  - [PipelineFileReference](#)
- [Steps](#)
- [Template Parameters](#)
- [Sample Pipeline](#)
  - [Sample Pipeline Code](#)

### Template Snippets

To create Gradle build, you will need to create pipeline YAML file. Pipeline YAML file will first need to include the reference to the Git repository hosting pipeline template that it is using:

 In Azure DevOps (*Project Settings->Service connections*), a GitHub service connection is created by default during the onboarding process. This service connection (*github.com\_sa-onboarding\_bmogc-ProjectName*) will be used for pipeline endpoint.

### TemplateReference

```
resources:
  repositories:
    - repository: templates # alias for the template reference
      type: github
      name: BMO-Prod/template_repo # name of the template repository
      endpoint: <Github Service Connection Name> # Replace with the
name of the GitHub service connection used to connect to template
GitHub repository
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest
release tag is documented in page https://bmo01.atlassian.net/wiki
/spaces/DSOM/pages/18710536/Pipeline+Templates
```

After the pipeline template reference, you will need to include the YAML file from above mentioned GitHub repo using the below:

### PipelineFileReference

```
stages:
- template: pipeline_templates/build.yml@templates
  parameters:
```

## Steps

Gradle pipeline consists of the following steps:

1. Checkout the pipeline templates repo to "templates" directory
2. Checkout the application Git repository to "s" directory
3. [Use Python version task](#) to set Python version to v3
4. If sonarqube parameter is set to true, pipeline runs Python script that set SonarQube project name and project key values to the name of the application Git repository
5. If sonarqube parameter is set to true, pipeline runs [SonarQube Prepare Analysis Configuration task](#) to set all the required SonarQube settings prior to executing a build

6. Pipeline runs steps configured in insertPreBuildSteps parameter. This parameter can be used by the users to run steps they need to run before the build steps run
7. If buildCache parameter is set to true, pipeline runs [Cache task](#) that allows you to improve build performance by caching files, like dependencies, between pipeline runs
8. [JFrog Gradle task](#) runs Gradle build steps.
9. If publishTestResults parameter set to true, pipeline runs [Publish test results task](#)
10. If publishCoverageResults parameter set to true, pipeline runs [Publish code coverage results task](#)
11. Pipeline runs steps configured in insertPostBuildSteps parameter. This parameter can be used by the users to run steps they need to run after the build steps run
12. If sonarqube parameter is set to true, pipeline runs [SonarQube Code Analysis task](#) to execute source code analysis
13. If pipeline was not triggered via pull request, pipeline runs steps configured in insertPrePublishSteps parameter. This parameter can be used by the users to run steps they need to run before the publish steps run
14. If pipeline was not triggered via pull request and gradlePublish parameter is set to true, pipeline runs [JFrog Gradle task](#) with gradlePublishTasks set to "artifactoryPublish", by default. This step publishes Gradle artifact to Gradle repository in Artifactory
15. If pipeline was not triggered via pull request and gradlePublish parameter is set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish Gradle artifacts
16. If pipeline was not triggered via pull request and publishAzureDevOps parameter or publishArtifactory parameter set to true, pipeline runs [Copy Files task](#) that stages specified files using match patterns
17. If pipeline was not triggered via pull request and publishAzureDevOps parameter or set to true, pipeline runs [Publish Pipeline Artifacts task](#) that specified artifacts to Azure DevOps or specified file share
18. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [Archive Files task](#) that creates an archive file from specified source folder
19. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Generic Artifacts task](#) that upload specified zip file(s) to Artifactory
20. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish Gradle artifacts
21. Pipeline runs steps configured in insertPostPublishSteps parameter. This parameter can be used by the users to run steps they need to run after the publish steps run

#### Template Parameters

Pipeline runs the above mentioned steps using configurable pipeline parameters prescribing how build pipeline should run:

Parameter	Required	Supported values	Description
buildType	Yes	gradle	Parameter prescribing the pipeline to run Gradle build type.
agentPoolName	Yes	'Prod'	Specify the agent pool to use during the pipeline run.
agentOS	Yes	'Windows_NT' - to select Windows based agents  'Linux' - to select Linux based agents	Specify the operating system the pipeline run requires.
agentAdditionalDemands	No		Specify the extra capability demands on the agents for specific use cases. Please consult DevSecOps team on the value to be used.
jobTimeoutInMinutes	No		Specify the job timeout duration. Default value is 60 minutes.
jdkVersion	Yes	8, 11, 17	Specify the jdk version to use with gradle
gradlePublish	No	true/false	Specify whether the pipeline should run the publish steps
gradleFilePath	Yes		Specify the full path to build.gradle
gradleWorkingDirectory	No		Specify the directory to run gradle in
gradleBuildOptions	No		Specify additional gradle command arguments
gradleBuildTasks	Yes		Specify gradle build task(s).
gradlePublishOptions	No		Specify arguments to use when publishing
gradlePublishTasks	No		default value is "artifactoryPublish"
gradleUseArtifactoryPlugin	No	true/false	Specify whether the the artifactory plugin is already applied in the build script. Default is false
gradleUseWrapper	No	true/false	Specify whether to use the gradle wrapper. Default is false.
gradleCollectBuildInfo	No	true/false	Specify whether to collect build Information. Default is true.
gradleDeployIvyDescription	No	true/false	Specify whether to use Ivy descriptors. Default is false
gradleIvyDescriptionPattern	No		Specify Ivy Description Pattern
gradleDeployIvyArtifactsPattern	No		Specify Ivy deployment artifact pattern
packageFeedResolveServiceConnection	Yes		Specify the Artifactory service connection to be used to resolve the packages

packageFeedReleaseResolveRepo	Yes		Specify Artifactory release resolve repository
packageFeedSnapshotResolveRepo	Yes		Specify Artifactory snapshot resolve repository
packageFeedPublishServiceConnection	Yes		Specify the Artifactory service connection to be used to publish the packages
packageFeedReleasePublishRepo	No		Specify Artifactory release publish repository
packageFeedSnapshotPublishRepo	No		Specify Artifactory snapshot publish repository
publishTestResults	No	true/false	Specify whether the pipeline should publish test results. Default value is set to true.
publishTestResultsFormat	No	JUnit, NUnit, VSTest, xUnit, cTest	Specify the format in which to publish test results. Default value is "JUnit"
publishTestResultsFiles	No		Specify path to test result files. More Info: <a href="#">Publish Test Results task - Azure Pipelines</a>   <a href="#">Microsoft Docs</a>
publishCoverageResults	No	true/false	Specify whether the pipeline should publish code coverage test results. . Default value is set to true.
publishCoverageTool	No	'JaCoCo', 'Cobertura'	Specify Code Coverage tool to use.
publishCoverageReportDirectory	No		Specify the path of the code coverage HTML report directory. The report directory is published for later viewing as an artifact of the build. The value may contain minimatch patterns
publishCoverageResultsSummaryFileLocation	No		Specify the path of the summary file containing code coverage statistics, such as line, method, and class coverage. Multiple summary files will be merged into a single report
publishAzureDevOps	No	true/false	Specify whether pipeline should publish Azure DevOps artifacts
publishAzureDevOpsTargetPath	No		Specify the path to be published in Azure DevOps
publishAzureDevOpsLocation	No	pipeline/filepath	Specify whether the pipeline should publish artifacts to file share or to Azure DevOps
publishAzureDevOpsFileShare	No		Specify the file share path where Azure DevOps artifacts should be published (if AzureDevOpsLocation option is set to filepath)
publishAzureDevOpsArtifactName	No		Specify the name of the Azure Pipeline artifact. Default value is drop.
buildCache	No	true/false	Specify whether dependency cache should be used. Default value is false
sonarqube	No	true/false	Specify whether the pipeline should run SonarQube steps
sonarqubeFailOnError	No	true/false	Specify whether the pipeline should fail if SonarQube gate fails
sonarqubeServiceConnection	No		Specify the name of the SonarQube service connection
sonarqubeProjectKey	No		Specify the SonarQube project key
sonarqubeProjectName	No		Specify the SonarQube project name
sonarqubeProjectVersion	No		Specify the SonarQube project version
sonarqubeExclusions	No		Specify the SonarQube exclusion pattern
sonarqubeExtraProperties	No	multi-line sonar properties in key=value pair	Specifies <a href="#">additional properties</a> (except exclusion pattern) to be passed to the scanner. Specify each key=value pair on a new line.

### Sample Pipeline

A sample pipeline code is given below for reference:

#### Sample Pipeline Code

```
resources:
  repositories:
```

```

- repository: templates
  type: github
  name: BMO-Prod/template_repo
  endpoint: Pilot
  ref: refs/tags/<tag_name> # name of the Git Tag. The latest tag
can be retrieved from: https://github.com/BMO-Prod/template_repo/tags
stages:
- template: pipeline_templates/build.yml@templates
  parameters:
    agentPoolName: 'Pilot'
    agentOS: 'Linux'
    jobTimeoutInMinutes: 180
    buildCache: true
    buildType: 'gradle'
    jdkVersion: '8'
    gradlePublish: true
    # Gradle related variables
    gradleFilePath: '$(System.DefaultWorkingDirectory)/RestThemeSvcEAR
/build.gradle'
    gradleWorkingDirectory: '$(System.DefaultWorkingDirectory)'
    gradleBuildOptions: '--debug --init-script=init.gradle -
PBRANCH_NAME=master -PBUILD_NUMBER=$(Build.BuildNumber) -
DIGNORE_FAILURES=true -Dsubset=$(Build.Repository.Name) -
DCommonLibVersion=8.7.1-1.7-129 --refresh-dependencies -Dfile.
encoding=utf-8'
    gradleBuildTasks: 'ear'
    gradlePublishOptions: '--debug --init-script=init.gradle -
PBRANCH_NAME=master -PBUILD_NUMBER=$(Build.BuildNumber) -
DIGNORE_FAILURES=true -Dsubset=$(Build.Repository.Name) -
DCommonLibVersion=8.7.1-1.7-129 --refresh-dependencies -Dfile.
encoding=utf-8'
    gradlePublishTasks: 'artifactoryPublish'
    gradleUseArtifactoryPlugin: false
    gradleUseWrapper: false
    gradleCollectBuildInfo: true
    gradleDeployIvyDescription: false
    gradleIvyDescriptionPattern: ''
    gradleDeployIvyArtifactsPattern: ''
    # End Gradle related variables
    packageFeedResolveServiceConnection: 'Artifactory-Read-Shared-v2-
Pilot'
    packageFeedReleaseResolveRepo: 'ODS-Maven-Virtual'
    packageFeedSnapshotResolveRepo: 'ODS-Maven-Virtual'
    packageFeedPublishServiceConnection: 'JFrog-DTOP-Snapshot'
    packageFeedReleasePublishRepo: 'DTOP-Snapshots/Gradle-sample'
    packageFeedSnapshotPublishRepo: 'DTOP-Snapshots/Gradle-sample'
    publishTestResults: true
    publishTestResultsFiles: '**/TEST-*.xml'
    publishCoverageResults: true
    publishCoverageReportDirectory: '$(System.DefaultWorkingDirectory)

```

```

/target/jacoco'
  publishCoverageResultsSummaryFileLocation: '$(System.
DefaultWorkingDirectory)/target/jacoco/jacoco.xml'
  sonarqube: true
  sonarqubeScannerMode: 'Other'
  sonarqubeProjectKey: '$(Build.Repository.Name)'
  sonarqubeProjectName: '$(Build.Repository.Name)'
  sonarqubeProjectVersion: '1.0'
  sonarqubeServiceConnection: 'SonarQube-Enterprise-Dev-SA_SonarQube'
  sonarqubeExclusions: ''
  sonarqubeExtraProperties: |
    sonar.sources=src/main/java/
    sonar.language=java
    sonar.java.binaries=target/classes


```

## Maven Build

- [Template Snippets](#)
  - [TemplateReference](#)
  - [PipelineFileReference](#)
- [Pipeline Steps](#)
- [Steps for using OCP](#)
- [Template Parameters](#)
- [Sample Pipeline](#)
  - [Sample Pipeline Code](#)

### Template Snippets

To create Maven build, you will need to create pipeline YAML file. Pipeline YAML file will first need to include the reference to the Git repository hosting pipeline template that it is using:

 In Azure DevOps (*Project Settings->Service connections*), a GitHub service connection is created by default during the onboarding process. This service connection (*github.com\_sa-onboarding\_bmogc-ProjectName*) will be used for pipeline endpoint.

### TemplateReference

```

resources:
  repositories:
    - repository: templates # alias for the template reference
      type: github
      name: BMO-Prod/template_repo # name of the template repository
      endpoint: <Github Service Connection Name> # Replace with the
name of the GitHub service connection used to connect to template
GitHub repository
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest
release tag is documented in page https://bmo01.atlassian.net/wiki
/spaces/DSOM/pages/18710536/Pipeline+Templates

```

After the pipeline template reference, you will need to include the YAML file from above mentioned GitHub repo using the below:

### PipelineFileReference

```

stages:
- template: pipeline_templates/build.yml@templates
  parameters:

```

## Pipeline Steps

Maven pipeline consists of the following steps:

1. Checkout the pipeline templates repo to "templates" directory.
  2. Checkout the application Git repository to "s" directory.
  3. [Use Python version task](#) to set Python version to v3.
  4. If sonarqube parameter is set to true, pipeline runs Python script that set SonarQube project name and project key values to the name of the application Git repository.
  5. If sonarqube parameter is set to true, pipeline runs [SonarQube Prepare Analysis Configuration task](#) to set all the required SonarQube settings prior to executing a build.
  6. Pipeline runs steps configured in insertPreBuildSteps parameter. This parameter can be used by the users to run steps they need to run before the build steps run.
  7. If buildCache parameter is set to true, pipeline runs [Cache task](#) that allows you to improve build performance by caching files, like dependencies, between pipeline runs.
  8. [JFrog Maven task](#) runs Maven build steps. Default Maven build goals are set to "clean package". Since package goal already includes test goal, the unit tests will be also executed (see more information about [Maven goals and phases](#).) Maven build task automatically includes SonarQube Code Analysis.
  9. If publishTestResults parameter set to true, pipeline runs [Publish test results task](#).
  10. If publishCoverageResults parameter set to true, pipeline runs [Publish code coverage results task](#).
  11. Pipeline runs steps configured in insertPostBuildSteps parameter. This parameter can be used by the users to run steps they need to run after the build steps run.
  12. If sonarqube parameter is set to true, pipeline runs [SonarQube Code Analysis task](#) to execute source code analysis.
  13. If pipeline was not triggered via pull request, pipeline runs steps configured in insertPrePublishSteps parameter. This parameter can be used by the users to run steps they need to run before running the publish steps.
  14. If pipeline was not triggered via pull request and mavenDeploy parameter is set to true, pipeline runs [JFrog Maven task](#) with Maven goals set to "deploy", by default. This step publishes Maven artifact to Maven repository in Artifactory.
  15. If pipeline was not triggered via pull request and mavenDeploy parameter is set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish Maven artifacts.
  16. If pipeline was not triggered via pull request and publishAzureDevOps parameter or publishArtifactory parameter set to true, pipeline runs [Copy Files task](#) that stages specified files using match patterns.
  17. If pipeline was not triggered via pull request and publishAzureDevOps parameter or set to true, pipeline runs [Publish Pipeline Artifacts task](#) that specified artifacts to Azure DevOps or specified file share.
  18. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [Archive Files task](#) that creates an archive file from specified source folder.
  19. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Generic Artifacts task](#) that upload specified zip file(s) to Artifactory.
  20. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish Maven artifacts.
  21. Pipeline runs steps configured in insertPostPublishSteps parameter. This parameter can be used by the users to run steps they need to run after running the publish steps.
- ✓ Click here if you want to use OpenShift Container Platform (OCP).

### Steps for using OCP

Openshift build steps build docker image in Openshift using shared/common dockerfile. This dockerfile pulls the build artifacts published by the Maven build in the same pipeline, and deploys the docker image to the target Openshift dev environment.

### Pre-requisites:

1. Setup BuildConfigs object in OpenShift (can also be done through Argo CD).
2. Setup DeploymentConfigs in OpenShift (can also be done through Argo CD).
3. Request four service accounts, one for each- Proxy Server, GitHub, JFrog, ServiceNow integration.
4. Dockerfile which is referenced in the BuildConfigs (pre-requisite 1).
5. Variable group created in Azure DevOps by the name **project-common**. The following variables are required in this variable group.

Name	Value	Secret /Masked	Description
docker-snapshots-repo	For example, chsb-docker-snapshots	No	A project/team should have its own repository.
artifactory_sa		No	The service account used to upload docker image to artifactory.
artifactory_apikey		Yes	The API key generated for the service account above



springboot_base_image_name	For example, introscope-tomcat-openjdk15	No	The base image of spring boot image, obtained from the Spring Boot dockerfile.
springboot_base_image_version	For example, v1.3	No	The Spring Boot base image version.
nodejs_base_image_name	For example, rhel8/nodejs-16	No	The base image of NodeJS image, obtained from the NodeJS dockerfile.
nodejs_base_image_version	For example, latest	No	The NodeJS base image version.
servicenow_register_image_container_url	<a href="https://bmogcqa02.service-now.com/api/x_baom_icm/register_image_container">https://bmogcqa02.service-now.com/api/x_baom_icm/register_image_container</a>	No	The ServiceNow REST API service endpoint.
servicenow_basicauth_token		Yes	The digest authorization token used for making ServiceNow REST API call.

Post completing the regular build tasks for Maven, OpenShift build consists of the following tasks:

1. Image tag is created based on the build information, in the form of <ado build number>-<artifact-base-version>-<short git commit sha>.
2. Openshift client is installed and logged in.
3. Namespace is selected.
4. BuildConfigs is patched.
5. Build is started.
6. Task to register image container in ServiceNow is performed.
7. Patch DeploymentConfigs (only when deploy parameter is set to true), this results in a rolling update of the deployment.
8. Logout.

#### Template Parameters

Pipeline runs the above mentioned steps using configurable pipeline parameters prescribing how build pipeline should run:

Parameter	Required	Supported values	Description
buildType	Yes	maven	Parameter prescribing the pipeline to run Maven build type.
agentPoolName	Yes	'Prod'	Specify the agent pool to use during the pipeline run
agentOS	Yes	'Windows_NT' - to select Windows based agents  'Linux' - to select Linux based agents	Specify the operating system the pipeline run requires
agentAdditionalDemands	No		Specify the extra capability demands on the agents for specific use cases. Please consult DevSecOps team on the value to be used.
jobTimeoutInMinutes	No		Specify the job timeout duration. Default value is 60 minutes.
jdkVersion	Yes	8, 11, 17	Specify the jdk version to use with maven
mavenBuildPomFilePath	Yes		Specify the full path to pom.xml
mavenDeployPomFilePath	Yes		Specify the full path to pom.xml
mavenBuildGoals	Yes		Specify maven build goal(s). Default value is 'clean build'
mavenBuildOptions	No		Specify additional maven command arguments
mavenDeploy	No	true/false	Specify whether the pipeline should run maven deploy goal(s)
mavenDeployOptions	No		Specify additional maven command arguments
mavenDeployGoals	No		Specify maven deploy goal(s). Default value is 'deploy'
packageFeedResolveServiceConnection	Yes		Specify the Artifactory service connection to be used to resolve the packages
packageFeedReleaseResolveRepo	Yes		Specify Artifactory release resolve repository
packageFeedSnapshotResolveRepo	Yes		Specify Artifactory snapshot resolve repository

packageFeedPublishServiceConnection	Yes		Specify the Artifactory service connection to be used to publish the packages
packageFeedReleasePublishRepo	No		Specify Artifactory release publish repository
packageFeedSnapshotPublishRepo	No		Specify Artifactory snapshot publish repository
publishTestResults	No	true/false	Specify whether the pipeline should publish test results. Default value is set to true.
publishTestResultsFormat	No	JUnit, NUnit, VSTest, xUnit, cTest	Specify the format in which to publish test results. Default value is "JUnit"
publishTestResultsFiles	No		Specify path to test result files. More Info: <a href="#">Publish Test Results task - Azure Pipelines   Microsoft Docs</a>
publishCoverageResults	No	true/false	Specify whether the pipeline should publish code coverage test results. . Default value is set to true.
publishCoverageTool	No	'JaCoCo', 'Cobertura'	Specify Code Coverage tool to use.
publishCoverageReportDirectory	No		Specify the path of the code coverage HTML report directory. The report directory is published for later viewing as an artifact of the build. The value may contain minimatch patterns
publishCoverageResultsSummaryFileLocation	No		Specify the path of the summary file containing code coverage statistics, such as line, method, and class coverage. Multiple summary files will be merged into a single report
publishAzureDevOps	No	true/false	Specify whether pipeline should publish Azure DevOps artifacts
publishAzureDevOpsTargetPath	No		Specify the path to be published in Azure DevOps
publishAzureDevOpsLocation	No	pipeline/filepath	Specify whether the pipeline should publish artifacts to file share or to Azure DevOps
publishAzureDevOpsFileShare	No		Specify the file share path where Azure DevOps artifacts should be published (if AzureDevOpsLocation option is set to filepath)
publishAzureDevOpsArtifactName	No		Specify the name of the Azure Pipeline artifact. Default value is drop.
buildCache	No	true/false	Specify whether dependency cache should be used. Default value is false
sonarqube	No	true/false	Specify whether the pipeline should run SonarQube steps
sonarqubeFailOnError	No	true/false	Specify whether the pipeline should fail if SonarQube gate fails
sonarqubeServiceConnection	No		Specify the name of the SonarQube service connection
sonarqubeProjectKey	No		Specify the SonarQube project key
sonarqubeProjectName	No		Specify the SonarQube project name
sonarqubeProjectVersion	No		Specify the SonarQube project version
sonarqubeExclusions	No		Specify the SonarQube exclusion pattern
sonarqubeExtraProperties	No	multi-line sonar properties in key=value pair	Specifies <a href="#">additional properties</a> (except exclusion pattern) to be passed to the scanner. Specify each key=value pair on a new line.
openshift	No, default to false.	true/false	Setting it to true will perform OpenShift build.

openshiftServiceConnection	Yes, if openshift parameter above is set to true.		Specify the service connection name for the OpenShift cluster.
ocProject	Yes, if openshift parameter above is set to true.		Specify the OpenShift project/namespace name.
ocServiceName	Yes, if openshiftparameter above is set to true.		Specify the deployment service name.
ocDockerRepo	Yes, if openshift parameter above is set to true.		A project/team should have its own repository.
ocBaseImageName	Yes, if openshift parameter above is set to true.		The base image of Spring Boot image which is obtained from the Spring Boot dockerfile.
ocBaseImageVersion	Yes, if openshift parameter above is set to true.		The Spring Boot base image version.
ocSkipDeploy	No, default to false.	true/false	Setting it to false will deploy new docker image to the given environment.
ocArtifactExtension	Yes, if openshift parameter above is set to true.	For example, jar, war, tgz, zip	Specify the artifact extension used for the docker build to lookup and download the build artifact from JFrog.  i.e. artifact name = <code>\${artifact_id}-\${artifact_version}-\${artifact_classifier}.\${artifact_extension}</code>
ocArtifactClassifier	Yes, if openshift parameter above is set to true.	For example, -exec for jar file or empty for war or zip files	Specify the artifact classifier used for the docker build to lookup and download the build artifact from JFrog.  i.e. artifact name = <code>\${artifact_id}-\${artifact_version}-\${artifact_classifier}.\${artifact_extension}</code>
ocArtifactPath	Yes, if openshift parameter above is set to true.		Location of artifact.
ocArtifactorySa	Yes, if openshift parameter above is set to true.		The service account used to upload docker image to Artifactory.
ocArtifactoryApiKey	Yes, if openshift parameter above is set to true.		The API key generated for the service account above.

#### Sample Pipeline

A sample pipeline code is given below for reference:

#### Sample Pipeline Code

```
resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo # name of the GitHub repository
      endpoint: Pilot
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest
tag can be retrieved from: https://github.com/BMO-Prod/template_repo
/tags
stages:
- template: pipeline_templates/build.yml@templates
  parameters:
    buildType: 'maven'
    agentPoolName: 'Pilot'
    agentOS: 'Linux'
    jobTimeoutInMinutes: 180
    jdkVersion: '11'
```

```

mavenBuildPomFilePath: $(Build.Repository.LocalPath)/pom.xml
mavenDeployPomFilePath: $(Build.Repository.LocalPath)/pom.xml
mavenBuildGoals: 'package'
    mavenBuildOptions: '-DXmx512M'
mavenDeploy: true
mavenDeployOptions: '-DXmx512M'
mavenDeployGoals: 'deploy'
packageFeedResolveServiceConnection: 'Artifactory-Read-Shared-v2-
Pilot'
    packageFeedReleaseResolveRepo: 'ODS-Maven-Virtual'
    packageFeedSnapshotResolveRepo: 'ODS-Maven-Virtual'
    packageFeedPublishServiceConnection: 'JFrog-DTOP-Snapshot'
    packageFeedReleasePublishRepo: 'DTOP-Snapshots/Maven-sample'
    packageFeedSnapshotPublishRepo: 'DTOP-Snapshots/Maven-sample'
    publishTestResults: true
    #<publishTestResultsFormat>
    publishTestResultsFiles: '**/TEST-*.xml'
    publishCoverageResults: true
    #<publishCoverageTool>
    publishCoverageReportDirectory: '$(System.DefaultWorkingDirectory)
/target/jacoco'
    publishCoverageResultsSummaryFileLocation: '$(System.
DefaultWorkingDirectory)/target/jacoco/jacoco.xml'
    sonarqube: true
    sonarqubeServiceConnection: 'SonarQube-Enterprise-Dev-SA_SonarQube'
    sonarqubeProjectKey: '$(Build.Repository.Name)'
    sonarqubeScannerMode: 'Other'
    sonarqubeProjectName: '$(Build.Repository.Name)'
    sonarqubeProjectVersion: '1.0'
    sonarqubeExclusions: ''
    sonarqubeExtraProperties: |
        sonar.sources=src/main/java/
        sonar.language=java
        sonar.java.binaries=target/classes

    #*****The below parameters are to be added if you want to use
OpenShift Container Platform (OCP)*****
    openshift: true
    openshiftServiceConnection: 'ocp4-green-connection'
    ocProject: 'cde-cde-24076-dv1'
    ocServiceName: 'channels-ms-partyv1'
    ocDockerRepo: 'docker-snapshots-repo'
    ocBaseImageName: 'redhat-openjdk-18/openjdk18-openshift'
    ocBaseImageVersion: 'latest'
    ocSkipDeploy: false
    ocArtifactExtension: 'war'
    ocArtifactClassifier: '-exec'
    ocArtifactPath: 'com/bmo/channels'
    ocArtifactorySa: 'svc_bwa_dev01'
    ocArtifactoryApiKey: ''


```

## Mobile Template Build

- [Template Snippets](#)
  - [TemplateReference](#)
  - [PipelineFileReference](#)
- [Steps](#)
- [Template Parameters](#)
- [Sample Pipeline](#)
  - [Sample Pipeline Code](#)

### Template Snippets

To create Mobile Template build, you will need to create pipeline YAML file. Pipeline YAML file will first need to include the reference to the Git repository hosting pipeline template that it is using:

 In Azure DevOps (*Project Settings->Service connections*), a GitHub service connection is created by default during the onboarding process. This service connection (*github.com\_sa-onboarding\_bmogc-ProjectName*) will be used for pipeline endpoint.

### TemplateReference

```
resources:
  repositories:
    - repository: templates # alias for the template reference
      type: github
      name: BMO-Prod/template_repo # name of the template repository
      endpoint: <Github Service Connection Name> # Replace with the
name of the GitHub service connection used to connect to template
GitHub repository
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest
release tag is documented in page https://bmo01.atlassian.net/wiki
/spaces/DSOM/pages/18710536/Pipeline+Templates
```

After the pipeline template reference, you will need to include the YAML file from above mentioned GitHub repo using the below:

### PipelineFileReference

```
stages:
- template: pipeline_templates/build.yml@templates
  parameters:
```

### Steps

Mobile Template pipeline consists of the following steps:

1. Checkout the pipeline templates repo to "templates" directory.
2. Checkout the application Git repository to "s" directory.
3. [Use Python version task](#) to set Python version to v3.
4. If sonarqube parameter is set to true, pipeline runs Python script that set SonarQube project name and project key values to the name of the application Git repository.
5. If sonarqube parameter is set to true, pipeline runs [SonarQube Prepare Analysis Configuration task](#) to set all the required SonarQube settings prior to executing a build.
6. Pipeline runs steps configured in insertPreBuildSteps parameter. This parameter can be used by the users to run steps they need to run before running the build steps.

7. If buildCache parameter is set to true, pipeline runs [Cache task](#) that allows you to improve build performance by caching files, like dependencies, between pipeline runs.
8. The Build Cordova template contains the following steps:
  - a. Identify the version of xcode to use and set relevant environment variables.
  - b. Install Node - This step will check to see if the identified Nodejs version is installed. If it is, it will prepend it to the path. If not, it will attempt to download the identified version and error out because the agents don't have internet access by design.
    - i. Use the [JFrog NPM task](#) to install all required node modules.
  - c. If buildCache parameter is set to true, pipeline runs [Cache task](#) that allows you to improve build performance by caching files, like dependencies, between pipeline runs.
  - d. [JFrog Gradle task](#) runs Gradle build steps.
  - e. Publishes the build log using the build-publish-azdo.yml template.
  - f. Publishes the build artifact using the build-publish-azdo.yml template.
9. Pipeline runs steps configured in insertPostBuildSteps parameter. This parameter can be used by the users to run steps they need to run after running the build steps.
10. If sonarqube parameter is set to true, pipeline runs [SonarQube Code Analysis task](#) to execute source code analysis.
11. If pipeline was not triggered via pull request, pipeline runs steps configured in insertPrePublishSteps parameter. This parameter can be used by the users to run steps they need to run before running the publish steps.
12. Pipeline runs steps configured in insertPostPublishSteps parameter. This parameter can be used by the users to run steps they need to run after running the publish steps.

## Template Parameters

Pipeline runs the above mentioned steps using configurable pipeline parameters prescribing how build pipeline should run:

Parameter	Required	Supported values	Description
buildType	Yes	cordova	Parameter prescribing the pipeline to run mobile template builds
agentPoolName	Yes	'Prod'	Specify the agent pool to use during the pipeline run
agentOS	Yes	'Darwin' - to select Linux based agents	Specify the operating system the pipeline run requires
agentAdditionalDemands	No		Specify the extra capability demands on the agents for specific use cases. Please consult DevSecOps team on the value to be used.
jobTimeoutInMinutes	No		Specify the job timeout duration. Default value is 60 minutes.
xcodeVersion	Yes	12,13	Specify the XCode version to use.
nodejsVersion	Yes	14.x, 16.x	Specify a supported version of nodejs that is available on the agent.
npmWorkingFolder	No		Specify the full path to the package.json. Default is the base directory for the repository.
npmInstallArgs	No		Specify any arguments that should be used while installing node modules. Default is ""
npmCollectBuildInfo	No	true/false	Specify whether the pipeline should collect the build info. Default is true
npmInstallThreads	No		Specify number of parallel threads to use when installing modules.
gradleFilePath	Yes		Specify the full path to build.gradle
gradleWorkingDirectory	No		Specify the directory to run gradle in
gradleBuildOptions	No		Specify additional gradle command arguments
gradleBuildTasks	Yes		Specify gradle build task(s).
gradleUseWrapper	No	true/false	Specify whether to use the gradle wrapper. Default is false.
gradleCollectBuildInfo	No	true/false	Specify whether to collect build Information. Default is true.
packageFeedResolveServiceConnection	Yes		

			Specify the Artifactory service connection to be used to resolve the packages
packageFeedReleaseResolveRepo	Yes		Specify Artifactory release resolve repository
packageFeedGenericRepo	Yes		Specify Artifactory resolve repository
packageFeedGenericServiceConnection	Yes		Specify the Artifactory service connection to be used to retrieve the packages
buildCache	No	true/false	Specify whether dependency cache should be used. If package-lock.json does not exist, this variable must be false. Default value is false.
sonarqube	No	true/false	Specify whether the pipeline should run SonarQube steps
sonarqubeFailOnError	No	true/false	Specify whether the pipeline should fail if SonarQube gate fails
sonarqubeServiceConnection	No		Specify the name of the SonarQube service connection
sonarqubeProjectKey	No		Specify the SonarQube project key
sonarqubeProjectName	No		Specify the SonarQube project name
sonarqubeProjectVersion	No		Specify the SonarQube project version
sonarqubeExclusions	No		Specify the SonarQube exclusion pattern
sonarqubeExtraProperties	No	multi-line sonar properties in key=value pair	Specifies <a href="#">additional properties</a> (except exclusion pattern) to be passed to the scanner. Specify each key=value pair on a new line.

#### Sample Pipeline

A sample pipeline code is given below for reference:

#### Sample Pipeline Code

```
resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo
      endpoint: Pilot
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest tag
can be retrieved from: https://github.com/BMO-Prod/template_repo/tags
parameters:
- name: gradleTask
  displayName: Gradle task
  type: string
  default: 'buildAll'
- name: buildEnvironment
  displayName: Build environment
  type: string
  default: qa2
values:
- perf1
- prod
```

- qa1
- qa2
- qa3
- qa4
- qa5
- qa6
- qa7
- qa8
- name: platformVersion
  - displayName: Platform version
  - type: string
  - default: '1251'
- name: appVersion
  - displayName: App version
  - type: string
  - default: '5.35.0'
- name: buildMode
  - displayName: Build mode
  - type: string
  - default: 'release'
  - values:
    - debug
    - release
- name: appDistribution
  - displayName: App distribution
  - type: string
  - default: 'enterprise'
  - values:
    - appstore
    - enterprise
- name: environmentSwitcher
  - displayName: Switch environments?
  - type: boolean
  - default: true

stages:

- template: pipeline\_templates/build.yml@templates
  - parameters:
    - agentPoolName: 'Pilot'
    - agentOS: 'Darwin'
    - buildType: 'cordova'
    - nodeJsVersion: '12.x'
    - npmWorkingFolder: '\$(System.DefaultWorkingDirectory)'
    - # Gradle related variables
    - gradleFilePath: '\$(System.DefaultWorkingDirectory)/build.gradle'
    - gradleWorkingDirectory: '\$(System.DefaultWorkingDirectory)'
    - gradleBuildOptions: '-Dorg.gradle.project.BUILD\_NUMBER="\$(Build.BuildNumber)" -Dorg.gradle.project.buildForEnv="\$\${ parameters.buildEnvironment }}" -Dorg.gradle.project.platformVersion="\$\${ parameters.platformVersion }}" -Dorg.gradle.project.appVersion="\$\${



```

parameters.appVersion }}" -Dorg.gradle.project.buildMode="${{
parameters.buildMode }}" -Dorg.gradle.project.distribution="${{
parameters.appDistribution }}" -Dorg.gradle.project.
environmentSwitcher="${{ parameters.environmentSwitcher }}" -Dorg.
gradle.project.mavenUser="$(mavenUser)" -Dorg.gradle.project.
mavenPwd="$(mavenPwd)" -Dorg.gradle.project.
USERIOSPWD="$(keychainPassword)" --debug --stacktrace --info'
  gradleBuildTasks: '${{ parameters.gradleTask }}'
  gradleUseWrapper: true
  gradleCollectBuildInfo: true
  xcodeVersion: '13'
  packageFeedGenericServiceConnection: 'Artifactory-Read-Shared-v2-
Pilot'
  packageFeedGenericRepo: 'npm-virtual'
  packageFeedResolveServiceConnection: 'JFrog-DTOP-Snapshot'
  packageFeedReleaseResolveRepo: 'DTOP-Snapshots'
  sonarqube: false
  sonarqubeScannerMode: 'Other'
  sonarqubeProjectKey: '${(Build.Repository.Name)}'
  sonarqubeProjectName: '${(Build.Repository.Name)}'
  sonarqubeProjectVersion: '${(Build.BuildNumber)}'
  sonarqubeServiceConnection: 'SonarQube-Enterprise-Dev-SA_SonarQube'
  sonarqubeExclusions: ''
  sonarqubeExtraProperties: |
    sonar.sources=src/main/java/
    sonar.language=java
    sonar.java.binaries=target/classes


```

## NPM Build

- [Template Snippets](#)
  - [TemplateReference](#)
  - [PipelineFileReference](#)
- [Pipeline Steps](#)
- [Steps for using OCP](#)
- [Template Parameters](#)
- [Sample Pipeline](#)
  - [Sample Pipeline Code](#)

### Template Snippets

To create NPM build, you will need to create pipeline YAML file. Pipeline YAML file will first need to include the reference to the Git repository hosting pipeline template that it is using:

 In Azure DevOps (*Project Settings->Service connections*), a GitHub service connection is created by default during the onboarding process. This service connection (*github.com\_sa-onboarding\_bmogc-ProjectName*) will be used for pipeline endpoint.

### TemplateReference

```

resources:
  repositories:
    - repository: templates # alias for the template reference

```

```

    type: github
    name: BMO-Prod/template_repo # name of the template repository
    endpoint: <Github Service Connection Name> # Replace with the
name of the GitHub service connection used to connect to template
GitHub repository
    ref: refs/tags/<tag_name> # name of the Git Tag. The latest
release tag is documented in page https://bmo01.atlassian.net/wiki
/spaces/DSOM/pages/18710536/Pipeline+Templates

```

After the pipeline template reference, you will need to include the YAML file from above mentioned GitHub repo using the below:  
**PipelineFileReference**

```

stages:
- template: pipeline_templates/build.yml@templates
  parameters:

```

## Pipeline Steps

NPM pipeline consists of the following steps:

1. Checkout the pipeline templates repo to "templates" directory.
2. Checkout the application Git repository to "s" directory.
3. [Use Python version task](#) to set Python version to v3.
4. If sonarqube parameter is set to true, pipeline runs Python script that set SonarQube project name and project key values to the name of the application Git repository.
5. If sonarqube parameter is set to true, pipeline runs [SonarQube Prepare Analysis Configuration task](#) to set all the required SonarQube settings prior to executing a build.
6. Pipeline runs steps configured in insertPreBuildSteps parameter. This parameter can be used by the users to run steps they need to run before running the build steps.
7. If buildCache parameter is set to true, pipeline runs [Cache task](#) that allows you to improve build performance by caching files, like dependencies, between pipeline runs.
8. The NPM Build template contains the following steps:
  - a. Install Node - This step will check to see if the identified NodeJs version is installed. If it is, it will prepend it to the path. If not, it will attempt to download the identified version and error out because the agents don't have internet access by design.
  - b. Use the [JFrog NPM task](#) to install all required node modules .
  - c. Use the [JFrog NPM task](#) to run "ci" task.
  - d. Use the [JFrog NPM task](#) to run "custom" build tasks as defined in the variables.
9. If publishTestResults parameter set to true, pipeline runs [Publish test results task](#).
10. If publishCoverageResults parameter set to true, pipeline runs [Publish code coverage results task](#).
11. Pipeline runs steps configured in insertPostBuildSteps parameter. This parameter can be used by the users to run steps they need to run after running the build steps.
12. If sonarqube parameter is set to true, pipeline runs [SonarQube Code Analysis task](#) to execute source code analysis.
13. If pipeline was not triggered via pull request, pipeline runs steps configured in insertPrePublishSteps parameter. This parameter can be used by the users to run steps they need to run before running the publish steps.
14. If pipeline was not triggered via pull request and npmDeploy parameter is set to true, pipeline runs [JFrog NPM task](#) with npmDeployArgs by default. This step publishes NPM artifact to NPM repository in Artifactory.
15. If pipeline was not triggered via pull request and npmDeploy parameter is set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish NPM artifacts.
16. If pipeline was not triggered via pull request and publishAzureDevOps parameter or publishArtifactory parameter set to true, pipeline runs [Copy Files task](#) that stages specified files using match patterns.
17. If pipeline was not triggered via pull request and publishAzureDevOps parameter or set to true, pipeline runs [Publish Pipeline Artifacts task](#) that specified artifacts to Azure DevOps or specified file share.
18. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [Archive Files task](#) that creates an archive file from specified source folder.
19. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Generic Artifacts task](#) that upload specified zip file(s) to Artifactory.
20. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish npm artifacts.

21. Pipeline runs steps configured in insertPostPublishSteps parameter. This parameter can be used by the users to run steps they need to run after running the publish steps.

✓ Click here if you want to use OpenShift Container Platform (OCP).

Steps for using OCP

Openshift build steps build docker image in Openshift using shared/common dockerfile. This dockerfile pulls the build artifacts published by the NPM build in the same pipeline, and deploys the docker image to the target Openshift dev environment.

#### Pre-requisites:

1. Setup BuildConfigs object in OpenShift (can also be done through Argo CD).
2. Setup DeploymentConfigs in OpenShift (can also be done through Argo CD).
3. Request four service accounts, one for each- Proxy Server, GitHub, JFrog, ServiceNow integration.
4. Dockerfile which is referenced in the BuildConfigs (pre-requisite 1).
5. Variable group created in Azure DevOps by the name **project-common**. The following variables are required in this variable group.

Name	Value	Secret /Masked	Description
docker-snapshots-repo	For example, chsb-docker-snapshots	No	A project/team should have its own repository.
artifactory_sa		No	The service account used to upload docker image to artifactory.
artifactory_apikey		Yes	The API key generated for the service account above
springboot_base_image_name	For example, introscope-tomcat-openjdk15	No	The base image of spring boot image, obtained from the Spring Boot dockerfile.
springboot_base_image_version	For example, v1.3	No	The Spring Boot base image version.
nodejs_base_image_name	For example, rhel8/nodejs-16	No	The base image of NodeJs image, obtained from the NodeJs dockerfile.
nodejs_base_image_version	For example, latest	No	The NodeJs base image version.
servicenow_register_image_container_url	<a href="https://bmogcqa02.service-now.com/api/x_baom_icm/register_image_container">https://bmogcqa02.service-now.com/api/x_baom_icm/register_image_container</a>	No	The ServiceNow REST API service endpoint.
servicenow_basicauth_token		Yes	The digest authorization token used for making ServiceNow REST API call.

Post completing the regular build tasks for NPM, OpenShift build consists of the following tasks:

1. Image tag is created based on the build information, in the form of <ado build number>-<artifact-base-version>-<short git commit sha>.
2. Openshift client is installed and logged in.
3. Namespace is selected.
4. BuildConfigs is patched.
5. Build is started.
6. Task to register image container in ServiceNow is performed.
7. Patch DeploymentConfigs (only when deploy parameter is set to true), this results in a rolling update of the deployment.
8. Logout.

#### Template Parameters

Pipeline runs the above mentioned steps using configurable pipeline parameters prescribing how build pipeline should run:

Parameter	Required	Supported values	Description
buildType	Yes	npm	Parameter prescribing the pipeline to run npm builds.
agentPoolName	Yes	'Prod'	Specify the agent pool to use during the pipeline run
agentOS	Yes	'Windows_NT' - to select Windows based agents 'Linux' - to select Linux based agents	Specify the operating system the pipeline run requires
agentAdditionalDemands	No		Specify the extra capability demands on the agents for specific use cases. Please consult DevSecOps team on the value to be used.
jobTimeoutInMinutes	No		Specify the job timeout duration. Default value is 60 minutes.

nodejsVersion	Yes	14.x, 16.x	Specify a supported version of NodeJs that is available on the agent.
npmWorkingFolder	No		Specify the full path to the package.json. Default is the base directory for the repository.
npmInstallArgs	No		Specify any arguments that should be used while installing node modules. Default is "
npmCollectBuildInfo	No	true/false	Specify whether the pipeline should collect the build info. Default is true
npmInstallThreads	No		Specify number of parallel threads to use when installing modules.
npmBuildArgs	Yes		Specify what commands to run when building. If you would like to run multiple tasks as part of your build process, consider bundling them in package.json. EG:  "scripts": { "task1": "\lcp -fR ../android/nativeResources/" .wl/android-nobuild/", "task2": "rimraf wl/android-nobuild/assets/www/default/ && node --max_old_space_size=8192 . /node_modules/@angular/cli/bin/ng build --configuration android", "task3": "npm run task1 && npm run task2" },
npmTest	No	true/false	Specify whether the artifact should be tested. Default is false. Also affects whether test results are published as well as code coverage tests are done.
npmTestArgs	No		specify what commands to run while testing. default is "test"
npmPublish	No	true/false	Specify whether the artifact should be published. Default is false
npmPublishArgs	No		Specify publishing arguments
packageFeedResolveServiceConnection	Yes		Specify the Artifactory service connection to be used to resolve the packages. The service connection needs to be pointing top the base artifactory url and not the repository.
packageFeedReleaseResolveRepo	Yes		Specify Artifactory release resolve repository
packageFeedSnapshotResolveRepo	Yes		Specify Artifactory snapshot resolve repository
packageFeedPublishServiceConnection	Yes		Specify the Artifactory service connection to be used to publish the packages. The service connection needs to be pointing top the base artifactory url and not the repository.
packageFeedReleasePublishRepo	No		Specify Artifactory release publish repository
packageFeedSnapshotPublishRepo	No		Specify Artifactory snapshot publish repository
publishTestResults	No	true/false	Specify whether the pipeline should publish test results. Default value is set to true.
publishTestResultsFormat	No	JUnit, NUnit, VSTest, xUnit, cTest	Specify the format in which to publish test results. Default value is "JUnit"
publishTestResultsFiles	No		

			Specify path to test result files. More Info: <a href="#">Publish Test Results task - Azure Pipelines   Microsoft Docs</a>
publishCoverageResults	No	true/false	Specify whether the pipeline should publish code coverage test results. . Default value is set to true.
publishCoverageTool	No	'JaCoCo', 'Cobertura'	Specify Code Coverage tool to use.
publishCoverageReportDirectory	No		Specify the path of the code coverage HTML report directory. The report directory is published for later viewing as an artifact of the build. The value may contain minimatch patterns
publishCoverageResultsSummaryFile Location	No		Specify the path of the summary file containing code coverage statistics, such as line, method, and class coverage. Multiple summary files will be merged into a single report
publishAzureDevOps	No	true/false	Specify whether pipeline should publish Azure DevOps artifacts
publishAzureDevOpsTargetPath	No		Specify the path to be published in Azure DevOps
publishAzureDevOpsLocation	No	pipeline/filepath	Specify whether the pipeline should publish artifacts to file share or to Azure DevOps
publishAzureDevOpsArtifactName	No		Specify the name of the Azure Pipeline artifact. Default value is drop.
buildCache	No	true/false	Specify whether dependency cache should be used. If package-lock.json does not exist, this variable must be false. Default value is false.
sonarqube	No	true/false	Specify whether the pipeline should run SonarQube steps
sonarqubeFailOnError	No	true/false	Specify whether the pipeline should fail if SonarQube gate fails
sonarqubeServiceConnection	No		Specify the name of the SonarQube service connection
sonarqubeProjectKey	No		Specify the SonarQube project key
sonarqubeProjectName	No		Specify the SonarQube project name
sonarqubeProjectVersion	No		Specify the SonarQube project version
sonarqubeExclusions	No		Specify the SonarQube exclusion pattern
sonarqubeExtraProperties	No	multi-line sonar properties in key=value pair	Specifies <a href="#">additional properties</a> (except exclusion pattern) to be passed to the scanner. Specify each key=value pair on a new line.
openshift	No, default to false.	true/false	Setting it to true will perform OpenShift build.
openshiftServiceConnection	Yes, if openshift parameter above is set to true.		Specify the service connection name for the OpenShift cluster.
ocProject	Yes, if openshift parameter above is set to true.		Specify the OpenShift project /namespace name.
ocServiceName	Yes, if openshiftparameter above is set to true.		Specify the deployment service name.
ocDockerRepo	Yes, if openshift parameter above is set to true.		A project/team should have its own repository.
ocBaseImageName	Yes, if openshift parameter above is set to true.		The base image of NodeJs image, obtained from the NodeJs dockerfile.
ocBaseImageVersion			The NodeJs base image version.

	Yes, if openshift parameter above is set to true.		
ocSkipDeploy	No, default to false.	true/false	Set it to false will deploy new docker image to the given environment.
ocArtifactExtension	Yes, if openshift parameter above is set to true.	For example, jar, war, tgz, zip	Specify the artifact extension used for the docker build to lookup and download the build artifact from JFrog.  i.e. artifact name = <code>\${artifact_id}-\${artifact_version}-\${artifact_classifier}.\${artifact_extension}</code>
ocArtifactClassifier	Yes, if openshift parameter above is set to true.	For example, -exec for jar file or empty for war or zip files	Specify the artifact classifier used for the docker build to lookup and download the build artifact from JFrog.  i.e. artifact name = <code>\${artifact_id}-\${artifact_version}-\${artifact_classifier}.\${artifact_extension}</code>
ocArtifactPath	Yes, if openshift parameter above is set to true.		Location of Artifact.
ocArtifactorySa	Yes, if openshift parameter above is set to true.		The service account used to upload docker image to Artifactory.
ocArtifactoryApiKey	Yes, if openshift parameter above is set to true.		The API key generated for the service account above.

#### Sample Pipeline

A sample pipeline code is given below for reference:

#### Sample Pipeline Code

```
resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo
      endpoint: Pilot
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest tag
can be retrieved from: https://github.com/BMO-Prod/template_repo/tags
stages:
- template: pipeline_templates/build.yml@templates
  parameters:
    buildType: 'npm'
    agentPoolName: 'Pilot'
    agentOS: 'Linux'
    jobTimeoutInMinutes: 180
    nodeJsVersion: '14.x'
    npmWorkingFolder: ''
    npmInstallArgs: ''
    npmCollectBuildInfo: true
    npmInstallThreads: '3'
    # NPM related parameters
    packageFeedSnapshotProjectKey: 'DTOP-Snapshots'
    npmBuildArgs: 'run build'
    npmTest: false
```

```

    npmTestArgs: 'test'
    npmPublish: true
    npmPublishArgs: ''
    # End NPM related parameters
    packageFeedResolveServiceConnection: 'Artifactory-Read-Shared-v2-
Pilot'
    packageFeedReleaseResolveRepo: 'npm-virtual'
    packageFeedSnapshotResolveRepo: 'ODS-Maven-Virtual'
    packageFeedPublishServiceConnection: 'JFrog-DTOP-Snapshot'
    #<packageFeedReleasePublishRepo>
    packageFeedSnapshotPublishRepo: 'DTOP-Snapshots'
    publishTestResults: true
    publishTestResultsFormat: 'NUnit3'
    publishTestResultsFiles: '**/TEST-*.xml'
    publishCoverageResults: true
    publishCoverageTool: 'Jacoco'
    publishCoverageReportDirectory: '$(System.DefaultWorkingDirectory)
/target/jacoco'
    publishCoverageResultsSummaryFileLocation: '$(System.
DefaultWorkingDirectory)/target/jacoco/jacoco.xml'
    #<publishAzureDevOps>
    #<publishAzureDevOpsTargetPath>
    #<publishAzureDevOpsLocation>
    #<publishAzureDevOpsArtifactName>
    buildCache: false
    sonarqube: false
    #<sonarqubeFailOnError>
    sonarqubeServiceConnection: 'SonarQube-Enterprise-Dev-SA_SonarQube'
    sonarqubeProjectKey: '$(Build.Repository.Name)'
    sonarqubeScannerMode: 'Other'
    sonarqubeProjectName: '$(Build.Repository.Name)'
    sonarqubeProjectVersion: '1.0'
    sonarqubeExclusions: ''
    sonarqubeExtraProperties: |
        sonar.sources=src/main/java/
        sonar.language=java
        sonar.java.binaries=target/classes

    #*****The below parameters are to be added if you want to use
OpenShift Container Platform (OCP)*****
    openshift: true
    openshiftServiceConnection: 'ocp4-green-connection'
    ocProject: 'cde-cde-24076-dv1'
    ocServiceName: 'channels-ms-partyv1'
    ocDockerRepo: 'docker-snapshots-repo'
    ocBaseImageName: 'redhat-openjdk-18/openjdk18-openshift'
    ocBaseImageVersion: 'latest'
    ocSkipDeploy: false
    ocArtifactExtension: 'war'
    ocArtifactClassifier: '-exec'

```


```
ocArtifactPath: 'com/bmo/channels'
ocArtifactorySa: 'svc_bwa_dev01'
ocArtifactoryApiKey: ''
```

## Ant Build

- [Template Snippets](#)
  - [TemplateReference](#)
  - [PipelineFileReference](#)
- [Steps](#)
- [Template Parameters](#)
- [Sample Pipeline](#)
  - [Sample Pipeline Code](#)

### Template Snippets

To create Ant build, you will need to create pipeline YAML file. Pipeline YAML file will first need to include the reference to the Git repository hosting pipeline template that it is using:

 In Azure DevOps (*Project Settings->Service connections*), a GitHub service connection is created by default during the onboarding process. This service connection (*github.com\_sa-onboarding\_bmogc-ProjectName*) will be used for pipeline endpoint.

### TemplateReference

```
resources:
  repositories:
    - repository: templates # alias for the template reference
      type: github
      name: BMO-Prod/template_repo # name of the template repository
      endpoint: <Github Service Connection Name> # Replace with the
name of the GitHub service connection used to connect to template
GitHub repository
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest
release tag is documented in page https://bmo01.atlassian.net/wiki
/spaces/DSOM/pages/18710536/Pipeline+Templates
```

After the pipeline template reference, you will need to include the YAML file from above mentioned GitHub repo using the below:

### PipelineFileReference

```
stages:
- template: pipeline_templates/build.yml@templates
  parameters:
```

## Steps

Ant pipeline consists of the following steps:

1. Checkout the pipeline templates repo to "templates" directory.
2. Checkout the application Git repository to "s" directory.
3. [Use Python version task](#) to set Python version to v3.



4. If sonarqube parameter is set to true, pipeline runs Python script that set SonarQube project name and project key values to the name of the application Git repository.
5. If sonarqube parameter is set to true, pipeline runs [SonarQube Prepare Analysis Configuration task](#) to set all the required SonarQube settings prior to executing a build.
6. Pipeline runs steps configured in insertPreBuildSteps parameter. This parameter can be used by the users to run steps they need to run before running the build steps.
7. If buildCache parameter is set to true, pipeline runs [Cache task](#) that allows you to improve build performance by caching files, like dependencies, between pipeline runs.
8. If packageArtifactoryDownload parameter is set to true, run [JFrog Managing Generic Artifacts](#) to download necessary dependencies.
9. [Ant task](#) runs Ant build steps.
10. If publishTestResults parameter set to true, pipeline runs [Publish test results task](#).
11. If publishCoverageResults parameter set to true, pipeline runs [Publish code coverage results task](#).
12. Pipeline runs steps configured in insertPostBuildSteps parameter. This parameter can be used by the users to run steps they need to run after the build steps run
13. If sonarqube parameter is set to true, pipeline runs [SonarQube Code Analysis task](#) to execute source code analysis.
14. If pipeline was not triggered via pull request, pipeline runs steps configured in insertPrePublishSteps parameter. This parameter can be used by the users to run steps they need to run before running the publish steps.
15. If pipeline was not triggered via pull request and publishAzureDevOps parameter or publishArtifactory parameter set to true, pipeline runs [Copy Files task](#) that stages specified files using match patterns.
16. If pipeline was not triggered via pull request and publishAzureDevOps parameter or set to true, pipeline runs [Publish Pipeline Artifacts task](#) that specified artifacts to Azure DevOps or specified file share.
17. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [Archive Files task](#) that creates an archive file from specified source folder.
18. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Generic Artifacts task](#) that upload specified zip file(s) to Artifactory.
19. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish npm artifacts.
20. Pipeline runs steps configured in insertPostPublishSteps parameter. This parameter can be used by the users to run steps they need to run after running the publish steps.

#### Template Parameters

Pipeline runs the above mentioned steps using configurable pipeline parameters prescribing how build pipeline should run:

Parameter	Required	Supported values	Description
buildType	Yes	ant	Parameter prescribing the pipeline to run ant build type.
agentPoolName	Yes	'Prod'	Specify the agent pool to use during the pipeline run.
agentOS	Yes	'Windows_NT' - to select Windows based agents 'Linux' - to select Linux based agents	Specify the operating system the pipeline run requires.
agentAdditionalDemands	No		Specify the extra capability demands on the agents for specific use cases. Please consult DevSecOps team on the value to be used.
jobTimeoutInMinutes	No		Specify the job timeout duration. Default value is 60 minutes.
jdkVersion	Yes	8, 11, 17	Specify the jdk version to use with ant.
antBuildFile	Yes		Specify the full path to build.xml
antBuildTargets	Yes		Specify Build Targets.
antBuildOptions	No		Specify additional ant command arguments.
packageArtifactoryDownload	No	true/false	Specify if download from Artifactory is required.
packageFeedResolveServiceConnection	Yes, if packageArtifactoryDownload is true.		Specify the Artifactory service connection to be used to resolve the packages.
packageArtifactoryDownloadFilePattern	Yes, if packageArtifactoryDownload is true.		Specify Artifactory download path.
packageArtifactoryDownloadTarget	Yes, if packageArtifactoryDownload is true.		Specify Artifactory download file.

publishArtifactory	No	true/false	Specify whether the pipeline should publish artifacts to Artifactory. Default is false.
packageFeedPublishServiceConnection	Yes, if publishArtifactory is true.		Specify the Artifactory service connection to be used to publish the packages.
publishArtifactoryUploadFilePattern	Yes, if publishArtifactory is true.		Specify Artifactory upload path.
publishArtifactoryUploadTarget	Yes, if publishArtifactory is true.		Specify Artifactory upload file.
publishTestResults	No	true/false	Specify whether the pipeline should publish test results. Default value is set to true.
publishTestResultsFormat	No	JUnit, NUnit, VSTest, xUnit, cTest	Specify the format in which to publish test results. Default value is "JUnit".
publishTestResultsFiles	No		Specify path to test result files. More Info: <a href="#">Publish Test Results task - Azure Pipelines   Microsoft Docs</a> .
publishCoverageResults	No	true/false	Specify whether the pipeline should publish code coverage test results. . Default value is set to true.
publishCoverageTool	No	'None', 'JaCoCo', 'Cobertura'	Specify Code Coverage tool to use.
publishCoverageReportDirectory	No		Specify the path of the code coverage HTML report directory. The report directory is published for later viewing as an artifact of the build. The value may contain minimatch patterns.
publishCoverageResultsSummaryFileLocation	No		Specify the path of the summary file containing code coverage statistics, such as line, method, and class coverage. Multiple summary files will be merged into a single report.
publishAzureDevOps	No	true/false	Specify whether pipeline should publish Azure DevOps artifacts .
publishAzureDevOpsTargetPath	No		Specify the path to be published in Azure DevOps.
publishAzureDevOpsLocation	No	pipeline/filepath	Specify whether the pipeline should publish artifacts to file share or to Azure DevOps.
publishAzureDevOpsArtifactName	No		Specify the name of the Azure Pipeline artifact. Default value is drop.
buildCache	No	true/false	Specify whether dependency cache should be used. Default value is false.
sonarqube	No	true/false	Specify whether the pipeline should run SonarQube steps.
sonarqubeFailOnError	No	true/false	Specify whether the pipeline should fail if SonarQube gate fails.
sonarqubeServiceConnection	No		Specify the name of the SonarQube service connection.
sonarqubeProjectKey	No		Specify the SonarQube project key.
sonarqubeProjectName	No		Specify the SonarQube project name.
sonarqubeProjectVersion	No		Specify the SonarQube project version.
sonarqubeExclusions	No		Specify the SonarQube exclusion pattern.
sonarqubeExtraProperties	No	multi-line sonar properties in key=value pair	Specifies <a href="#">additional properties</a> (except exclusion pattern) to be passed to the scanner. Specify each key=value pair on a new line.

## Sample Pipeline

A sample pipeline code is given below for reference:

### Sample Pipeline Code

```
resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo # name of the GitHub repository
      endpoint: Pilot
      ref: refs/tags/v0.9-alpha

stages:
- template: pipeline_templates/build.yml@templates
  parameters:
    agentPoolName: 'Prod'
    agentOS: 'Linux'
    jobTimeoutInMinutes: 180
    buildType: 'ant'
    jdkVersion: '8'
    antBuildFile: 'build.xml'
    antBuildTargets: 'clean init compile dist'
    antBuildOptions: '-Dbuild.no.label="$(Build.BuildNumber)" -Drelease.
number="1.0.0"'
    packageFeedResolveServiceConnection: 'Artifactory-Read-Shared-v2-
Pilot'
    packageArtifactoryDownload: true
    packageArtifactoryDownloadFilePattern: 'DevOps_Tools/ADO2020/CS
/README.txt'
    packageArtifactoryDownloadTarget: 'temp/'
    publishTestResults: false
    publishTestResultsFiles: '**/TEST-*.xml'
    publishCoverageTool: 'JaCoCo'
    publishCoverageClassFilesDirectories: '.'
    publishCoverageClassFilter: ''
    publishCoverageSourceDirectories: 'src'
    publishCoverageResults: false
    publishCoverageReportDirectory: '$(System.DefaultWorkingDirectory)
/target/jacoco'
    publishCoverageResultsSummaryFileLocation: '$(System.
DefaultWorkingDirectory)/target/jacoco/jacoco.xml'
    publishArtifactory: true
    publishAzureDevOpsLocation: '$(System.DefaultWorkingDirectory)/bin'
    publishArtifactoryIncludeRootFolder: true
    publishArtifactoryTargetArchivePath: 'archive/$(Build.BuildId).tar'
    publishArtifactoryTargetArchiveType: 'tar'
    packageFeedPublishServiceConnection: 'JFrog-DTOP-Snapshot'
    publishArtifactoryUploadFilePattern: 'archive/$(Build.BuildId).tar'
    publishArtifactoryUploadTarget: 'DTOP-Snapshots/Ant-sample-new
/$(Build.BuildId).tar'
```

```

sonarqube: true
sonarqubeScannerMode: 'Other'
sonarqubeProjectKey: '$(Build.Repository.Name)'
sonarqubeProjectName: '$(Build.Repository.Name)'
sonarqubeProjectVersion: '1.0'
sonarqubeServiceConnection: 'SonarQube-Enterprise-Dev-SA_SonarQube'
sonarqubeExclusions: ''
sonarqubeExtraProperties: |
    sonar.sources=src/com/mkyong/core/utils/
    sonar.language=java
    sonar.java.binaries=bin/com/mkyong/core/utils/


```

#### Visual Studio Build

- [Template Snippets](#)
  - [TemplateReference](#)
  - [PipelineFileReference](#)
- [Steps](#)
- [Template Parameters](#)
- [Sample Pipeline](#)
  - [Sample Pipeline Code](#)

#### Template Snippets

To create Visual Studio build, you will need to create pipeline YAML file. Pipeline YAML file will first need to include the reference to the Git repository hosting pipeline template that it is using:

 In Azure DevOps (*Project Settings->Service connections*), a GitHub service connection is created by default during the onboarding process. This service connection (*github.com\_sa-onboarding\_bmogc-ProjectName*) will be used for pipeline endpoint.

#### TemplateReference

```

resources:
  repositories:
    - repository: templates # alias for the template reference
      type: github
      name: BMO-Prod/template_repo # name of the template repository
      endpoint: <Github Service Connection Name> # Replace with the
name of the GitHub service connection used to connect to template
GitHub repository
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest
release tag is documented in page https://bmo01.atlassian.net/wiki
/spaces/DSOM/pages/18710536/Pipeline+Templates

```

After the pipeline template reference, you will need to include the YAML file from above mentioned GitHub repo using the below:

#### PipelineFileReference

```

stages:
- template: pipeline_templates/build.yml@templates
  parameters:

```

## Steps

Visual Studio pipeline consists of the following steps:

1. Checkout the pipeline templates repo to "templates" directory.
2. Checkout the application Git repository to "s" directory.
3. [Use Python version task](#) to set Python version to v3.
4. If sonarqube parameter is set to true, pipeline runs Python script that set SonarQube project name and project key values to the name of the application Git repository.
5. If sonarqube parameter is set to true, pipeline runs [SonarQube Prepare Analysis Configuration task](#) to set all the required SonarQube settings prior to executing a build.
6. Pipeline runs steps configured in insertPreBuildSteps parameter. This parameter can be used by the users to run steps they need to run before running the build steps.
7. If buildCache parameter is set to true, pipeline runs [Cache task](#) that allows you to improve build performance by caching files, like dependencies, between pipeline runs.
8. Pipeline runs Python script that delete all Visual Studio solution files except for the one specified to be built by this pipeline run. This step is necessary because otherwise "nuget restore" step below fails.
9. [JFrog Nuget task](#) runs "nuget restore" command for the specified Visual Studio solution.
10. [Visual Studio build task](#) builds specified Visual Studio solution with specified version of msbuild.
11. If vsTest parameter is set to true, pipeline runs [Visual Studio Test task](#) to execute the unit tests.
12. If vsTest parameter is set to true and publishTestResults parameter set to true, pipeline runs [Publish test results task](#).
13. If vsTest parameter is set to true and publishCoverageResults parameter set to true, pipeline runs [Publish code coverage results task](#).
14. If vsBuildPublishSymbols parameter is set to true, pipeline runs [Index Sources and Publish Symbols task](#).
15. Pipeline runs steps configured in insertPostBuildSteps parameter. This parameter can be used by the users to run steps they need to run after running the build steps.
16. If sonarqube parameter is set to true, pipeline runs [SonarQube Code Analysis task](#) to execute source code analysis.
17. If pipeline was not triggered via pull request, pipeline runs steps configured in insertPrePublishSteps parameter. This parameter can be used by the users to run steps they need to run before running the publish steps.
18. If pipeline was not triggered via pull request and publishAzureDevOps parameter or publishArtifactory parameter set to true, pipeline runs [Copy Files task](#) that stages specified files using match patterns.
19. If pipeline was not triggered via pull request and publishAzureDevOps parameter or set to true, pipeline runs [Publish Pipeline Artifacts task](#) that specified artifacts to Azure DevOps or specified file share.
20. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [Archive Files task](#) that creates an archive file from specified source folder.
21. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Generic Artifacts task](#) that upload specified zip file(s) to Artifactory.
22. If pipeline was not triggered via pull request and publishArtifactory parameter set to true, pipeline runs [JFrog Publish Build Info task](#) to associate build information with publish Maven artifacts.
23. Pipeline runs steps configured in insertPostPublishSteps parameter. This parameter can be used by the users to run steps they need to run after running the publish steps.

## Template Parameters

Pipeline runs the above mentioned steps using configurable pipeline parameters prescribing how build pipeline should run:

Parameter	Required	Supported values	Description
buildType	Yes	vsbuild	Parameter prescribing the pipeline to run Visual Studio build type
agentPoolName	Yes	'Prod'	Specify the agent pool to use during the pipeline run
agentOS	Yes	'Windows_NT'	Specify the operating system the pipeline run requires
agentAdditionalDemands	No		Specify the extra capability demands on the agents for specific use cases. Please consult DevSecOps team on the value to be used.
jobTimeoutInMinutes	No		Specify the job timeout duration. Default value is 60 minutes.
vsbuildSolutionPath	Yes	.sln files	Specify the full path to the Visual Studio solution file. If you want to build multiple solutions, specify search criteria. You can use a single-folder wildcard (*) and recursive wildcards (**). For example, '**.sln' searches for all .sln files in all subdirectories.
vsbuildSolutionFileName	Yes	.sln files	

			Specify the name of the Visual Studio solution file. Use '*.sln' or '**.sln' when building multiple solutions.
vsBuildArgs	No		Specify msbuild arguments to be used during the pipeline run
vsVersion	Yes	'16.0' - to select Visual Studio 2019 '17.0' - to select Visual Studio 2022	Specify what version of Visual Studio should be used during the pipeline run
vsbuildSolutionPlatform	Yes		Specify the platform you want to build such as Win32, x86, x64 or any cpu.
vsbuildSolutionConfiguration	Yes		Specify the configuration you want to build such as debug or release.
vsBuildClean	No	true/false	Specify whether the pipeline should clean the staging folders before compiling the Visual Studio solution. Default value is true.
vsBuildLogProjectEvents	No		Specify whether the pipeline should record timeline details for each project.
vsTest	No	true/false	Specify whether the pipeline should run Visual Studio tests
publishTestResults	No	true/false	Specify whether the pipeline should publish test results
publishTestResultsFormat	No	JUnit, NUnit, VSTest, xUnit, cTest	Specify the format in which to publish test results. Default value is "JUnit"
publishTestResultsFiles	No		Specify path to test result files. More Info: <a href="#">Publish Test Results task - Azure Pipelines   Microsoft Docs</a>
publishCoverageResults	No	true/false	Specify whether the pipeline should publish code coverage test results
publishCoverageTool	No	'JaCoCo', 'Cobertura'	Specify Code Coverage tool to use.
publishCoverageReportDirectory	No		Specify the path of the code coverage HTML report directory. The report directory is published for later viewing as an artifact of the build. The value may contain minimatch patterns
publishCoverageResultsSummaryFile Location	No		Specify the path of the summary file containing code coverage statistics, such as line, method, and class coverage. Multiple summary files will be merged into a single report
vsBuildCreateLogFile	No		Specify whether the pipeline should create a log file (Windows only).
packageFeedResolveServiceConnection	Yes		Specify the Artifactory service connection to be used to resolve nuget packages
packageFeedResolveRepo	Yes		Specify the name of the repository used to resolve nuget packages
packagesDirectory	Yes		Specify the path used to store downloaded nuget packages
useNuget	No	true or false	Specify whether to use nuget restore. Default is true.
nugetVersion	No	6	Specify to use NuGet 6. Other input values will default to the base version 4.
nugetProtocolVersion	Yes	'2.x' or '3.x'	Specify the version of nuget client
publishArtifactory	No	true/false	Specify whether the pipeline should publish artifacts to Artifactory. Default is false.
packageFeedPublishServiceConnection	Yes		Specify the Artifactory service connection to be used to publish the packages

publishArtifactoryTargetArchivePath	No		Specify the name of the archive file to create
publishArtifactoryTargetArchiveType	No	zip, 7z, tar, wim	Specify what archive type should be created during the pipeline run. Default value is zip
packageFeedPublishRepo			Specify whether the pipeline should publish artifacts to Artifactory
publishArtifactoryUploadFilePattern	Yes		Specify the file pattern to be uploaded
publishArtifactoryUploadTarget	Yes		Specify the the upload target path in Artifactory
publishArtifactoryIncludeRootFolder	No	true/false	Prepends the root folder name to file paths in the archive. Otherwise, all file paths will start one level lower. Default value: true.
publishAzureDevOps	No	true/false	Specify whether pipeline should publish Azure DevOps artifacts
publishAzureDevOpsTargetPath	No		Specify the path to be published in Azure DevOps
publishAzureDevOpsLocation	No	pipeline/filepath	Specify whether the pipeline should publish artifacts to file share or to Azure DevOps
publishAzureDevOpsFileShare	No		Specify the file share path where Azure DevOps artifacts should be published (if AzureDevOpsLocation option is set to filepath)
publishAzureDevOpsArtifactName	No		Specify the name of the Azure Pipeline artifact. Default value is drop.
buildCache	No	true/false	Specify whether dependency cache should be used. Default value is false
buildReleaseNotes	No	true/false	Specify whether the script should be run to generate release notes
buildOutputDir	No		Specify where the build output is expected
buildReleaseNotesArguments	No		Specify PowerShell script arguments allowed by the script
vsBuildAssemblySerializerGenerator	No	true/false	Specify whether the pipeline should run assembly serializer generator
vsBuildScriptIndex	No	true/false	Specify whether the pipeline should run build script index steps
vsBuildCodeSign	No	true/false	Specify whether the pipeline should sign code
vsBuildPublishSymbols	No	true/false	Specify whether the pipeline should publish symbols
buildRunNDepend	No	true/false	Specify whether the pipeline should run ndepend steps
sonarqube	No	true/false	Specify whether the pipeline should run SonarQube steps
sonarqubeFailOnError	No	true/false	Specify whether the pipeline should fail if SonarQube gate fails
sonarqubeServiceConnection	No		Specify the name of the SonarQube service connection
sonarqubeProjectKey	No		Specify the SonarQube project key
sonarqubeProjectName	No		Specify the SonarQube project name
sonarqubeProjectVersion	No		Specify the SonarQube project version
sonarqubeExclusions	No		Specify the SonarQube exclusion pattern
sonarqubeExtraProperties	No		

	multi-line sonar properties in key=value pair	Specifies <a href="#">additional properties</a> (except exclusion pattern) to be passed to the scanner. Specify each key=value pair on a new line.
--	---	--

## Sample Pipeline

A sample pipeline code is given below for reference:

### Sample Pipeline Code

```
resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo
      endpoint: Pilot
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest tag
can be retrieved from: https://github.com/BMO-Prod/template_repo/tags
stages:
- template: pipeline_templates/build.yml@templates
  parameters:
    agentPoolName: 'Pilot'
    agentOS: 'Windows_NT'
    jobTimeoutInMinutes: 180
    buildType: 'vsbuild'
    buildCache: false
    buildOutputDir: '$(Build.Repository.LocalPath)\Output'
    buildReleaseNotes: false
    buildReleaseNotesArguments:
' "$(majorVersion).$(minorVersion).$(patchVersion)",
'$(patchVersion)"$(patchVersion)", "BranchChannelSolutions-
ReleaseCandidates/CusC/D$(majorVersion).$(minorVersion)
/QF$(patchVersion)"$(patchVersion)", "0", "0", "0", "Yan, Min",
'$(Build.BuildNumber)", "WorkItems.xlsx", "Paula Richardson", "May
2022", " " -buildDirectory "\\CCBSBCCWDVAPP33\Drops_CC\$(Build.
DefinitionName)\$(Build.BuildNumber)" '
    vsbuildSolutionFileName: 'custMock.sln'
    vsbuildSolutionPath: '$(Build.Repository.LocalPath)\custMock.sln'
    vsVersion: '16.0'
    vsbuildSolutionPlatform: 'Any CPU'
    vsbuildSolutionConfiguration: 'Release'
    publishAzureDevOps: false
    publishAzureDevOpsContents: '**\bin\Release\**'
    publishAzureDevOpsLocation: 'filepath'
    publishAzureDevOpsArtifactName: 'msbuild-sample'
    publishAzureDevOpsFileShare: ''
    packageFeedResolveRepo: 'BranchChannelSolutions-NuGet-Virtual'
    packageFeedPublishServiceConnection: 'JFrog-DTOP-Snapshot'
    publishArtifactoryUploadFilePattern: '$(Build.BinariesDirectory)\*.
zip'
    publishArtifactoryUploadTarget: 'DTOP-Snapshots/TFS/2018/'
    packageFeedPublishRepo: 'DTOP-Snapshots'
```



```

packageFeedPublishRelativePath: 'TFS/2018'
publishArtifactory: true
packageFeedResolveServiceConnection: 'Artifactory-Read-Shared-v2-
Pilot'
packagesDirectory: $(Build.Repository.LocalPath)\packages
sonarqube: true
sonarqubeProjectKey: '$(Build.Repository.Name)'
sonarqubeProjectName: '$(Build.Repository.Name)'
sonarqubeProjectVersion: '1.0'
sonarqubeServiceConnection: 'SonarQube-Enterprise-Dev-SA_SonarQube'
sonarqubeExclusions: '**/*.bin'

```

#### Additional Template Steps

Occasionally, teams may have a need to insert steps both before and after the main template runs. These could include tasks that have not been accounted for by the template or cleanup activities after the templates have run. These steps can be inserted within the repository pipeline and can include any task that the agent supports. For example, you will not be able to run a bash task on a Windows agent. The following code snippet shows how to invoke pre and post build steps.

```

insertPreBuildSteps:
- script: |
  dir
  displayName: 'custom pre-build step 1'
- script: |
  env
  displayName: 'custom pre-build step 2'

insertPostBuildSteps:DTOP-Snapshots
- script: |
  dir
  displayName: 'custom post-build steps'

```

A more complete pipeline example is shown below:

```

resources:
  repositories:
    - repository: templates
      type: github
      name: BMO-Prod/template_repo
      endpoint: Pilot
      ref: refs/tags/<tag_name> # name of the Git Tag. The latest tag
can be retrieved from: https://github.com/BMO-Prod/template_repo/tags
  stages:
    - template: pipeline_templates/build.yml@templates
  parameters:
    agentPoolName: 'Pilot'

```

```

agentOS: 'Windows_NT'
jobTimeoutInMinutes: 180
buildType: 'vsbuild'
buildCache: false
#useNuget: false
buildOutputDir: '$(Build.Repository.LocalPath)\Output'
buildReleaseNotes: false
buildReleaseNotesArguments:
'"$(majorVersion).$(minorVersion).$(patchVersion)",
$(patchVersion)"$(patchVersion)", "BranchChannelSolutions-
ReleaseCandidates/Cusc/D$(majorVersion).$(minorVersion)
/QF$(patchVersion)"$(patchVersion)", "0", "0", "0", "Yan, Min",
$(Build.BuildNumber)", "WorkItems.xlsx", "Paula Richardson", "May
2022", "" -buildDirectory "\\CCBSBCCWDVAPP33\Drops_CC\$(Build.
DefinitionName)\$(Build.BuildNumber)"'
    vsbuildSolutionFileName: 'custMock.sln'
    vsbuildSolutionPath: '$(Build.Repository.LocalPath)\custMock.sln'
    # vsBuildArgs: '/p:TargetFrameworkSDKToolsDirectory="C:\Program
Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.7.2 Tools" /p:
CscToolPath="C:\Program Files (x86)\Microsoft Visual
Studio\2019\Enterprise\MSBuild\Current\Bin\Roslyn"'
    vsVersion: '16.0'
    vsbuildSolutionPlatform: 'Any CPU'
    vsbuildSolutionConfiguration: 'Release'
    # msbuildExePath: 'C:\Program Files (x86)\Microsoft Visual
Studio\2019\Enterprise\MSBuild\Current\Bin\MSBuild.exe'
    publishAzureDevOps: false
    publishAzureDevOpsContents: '**\bin\Release\**'
    publishAzureDevOpsLocation: 'filepath'
    publishAzureDevOpsArtifactName: 'msbuild-sample'
    publishAzureDevOpsFileShare: '\\CCBSBCCWDVAPP33\Drops_CC\$(Build.
DefinitionName)\$(Build.BuildNumber)'
    packageFeedResolveRepo: 'BranchChannelSolutions-NuGet-Virtual'
    packageFeedPublishServiceConnection: 'JFrog-DTOP-Snapshot'
    publishArtifactoryUploadFilePattern: '$(Build.BinariesDirectory)\*.
zip'
    publishArtifactoryUploadTarget: 'DTOP-Snapshots/TFS/2018/'
    packageFeedPublishRepo: 'DTOP-Snapshots'
    packageFeedPublishRelativePath: 'TFS/2018'
    publishArtifactory: true
    packageFeedResolveServiceConnection: 'Artifactory-Read-Shared-v2-
Pilot'
    packagesDirectory: $(Build.Repository.LocalPath)\packages
    sonarqube: true
    sonarqubeProjectKey: '$(Build.Repository.Name)'
    sonarqubeProjectName: '$(Build.Repository.Name)'
    sonarqubeProjectVersion: '1.0'
    sonarqubeServiceConnection: 'SonarQube-Enterprise-Dev-SA_SonarQube'
    sonarqubeExclusions: '**/*.bin'
    insertPreBuildSteps:

```

```

- script: |
    dir
    displayName: 'custom pre-build steps'
insertPostBuildSteps:DTOP-Snapshots
- script: |
    dir
    displayName: 'custom post-build steps'

```

## Pipeline Variables

### 1. Non-secret Variables

Non-secret pipeline variables are defined in variable files and/or in variables section of YAML. Variables are loaded in the following order during the pipeline execution:

1. [Pipeline system variables](#)
2. [Pipeline common variables](#)
3. Variables specified in variable YAML files in application repository and passed to build using customVariableFiles parameter in application pipeline YAML file
4. Variables specified using customVariables parameter in application pipeline YAML file

### 2. Secret Variables

Secret pipeline variables are defined in Azure KeyVault(s) and loaded during pipeline execution. There are 3 types of Azure KeyVault loaded during the pipeline execution in the order specified below:

1. Global Azure KeyVault which contains secrets common for all tenants. This Azure KeyVault managed by OperationsTeam.
2. Tenant Core Azure KeyVault which contains secrets specific tenant/project. These Azure KeyVaults managed by OperationsTeam.
3. Tenant Application Azure KeyVault which contain secrets specific tenant/project. This Azure KeyVault managed by Application Developers.

## Repository Migration

BMO is currently using Bitbucket and TFS (Azure DevOps Servers) for Source Code Repository. All the source code repositories are required to migrate to GitHub Enterprise Cloud in the DevSecOps environment. Bitbucket and TFS (Azure DevOps Servers) will be decommissioned after the repository migration.

The data in the repositories is owned by the CIO teams.

**NOTE:** The compliance requirement of the repositories is the responsibility of the CIO teams. It is the responsibility of the CIO team to make sure the migration and decommission of Bitbucket and TFS (Azure DevOps Servers) does not impact their compliance requirement.

In this section, the repository migration will be discussed. It is important to go through the [Source Code Migration Type](#) and [Pre-Migration Checklist](#) before the repository migration. No secret will be allowed in repositories. Completion of DLP scan with remediation (if any) is mandatory for repository migration.

## BitBucket to GitHub Migration

To migrate repositories from Bitbucket to GitHub the CIO team can follow the below processes depending on whether repository migration is required without history or full repository migration is required with history.

### Scenario 1: Repo Migration without History

In case of repo migration without history, the application active source code is to be migrated from selected Bitbucket branch to the new GitHub Repo Git repository. If the initial DLP scan reports that no secret is found and remediation is needed, a Bitbucket repo can be migrated with history to GitHub.

- [Bitbucket Migration Playbook \(Without History\)](#)

### Scenario 2: Full Repo Migration with History

Please note that full repository migration with history will be executed by the DevSecOps Team. By default, the repo will be migrated with the following naming convention:

<project key>\_<repo name>\_<App Cat ID>

The main project access groups of the Bitbucket repo will be migrated as Teams to GitHub along with the equivalent access levels. *Other access groups and individual access rights of the repo will not be migrated to GitHub.* Kindly discuss with the assigned DevSecOps lead for your specific migration needs.

#### For any Repository with Bitbucket AWS On-Demand Pipeline:

The repository with Bitbucket AWS on-demand pipeline is still on Bitbucket so the migration process will be the same as per defined in above scenarios. If AWS workload is required, the DevSecOps Team will enable the AWS workflow for the pipeline after the repository migration. The request should be communicated to the assigned DevSecOps lead in advance.

#### Bitbucket Migration Playbook (Without History)

This playbook defines the guidelines how to migrate the latest version of the active Bitbucket source code from on premises instance of BitBucket Server to GitHub Enterprise Cloud instance. The application active source code will be migrated from selected Bitbucket branch to the new GitHub Repo Git repository. To migrate an application active source code from BitBucket Server to GitHub Enterprise Cloud instance:

#### Pre-requisites

- Ensure that you have gone through the [Pre-Migration Checklist](#) to prepare the repository for migration.
- Identify the users that need to be taken over to GitHub.

#### Migration Process for Repositories

1. Prepare BitBucket source code to be migrated
  - a. Identify the folders and files in Bitbucket repository of the selected team project that is related to the application to be migrated.
  - b. Download identified folders and files onto your local computer using Git Bash. For example,
2. Prepare GitHub repository
  - a. GitHub repository and teams created in GitHub Enterprise Cloud instance by the DevSecOps team.
  - b. Permissions granted to appropriate GitHub teams to the GitHub repository
  - c. Identified users added to appropriate teams in GitHub.
  - d. Branch protection rules configured for master/main (and develop branch, if applicable)
  - e. The user clones the GitHub repository onto your local computer. For example,

```
git clone https://github.com/BMO-Prod/<repoName>.git
```

- f. The user creates feature branch

```
git branch <feature branch name>
git checkout <feature branch name>
```

3. Stage GitHub repository source code locally

- a. Clone the BitBucket Repository locally.

```
# Fill in the required values
```

```
git clone --branch <branchname> https://bitbucket.bmogg.net/<projectname>/<reponame>.git /c/repos
/bitbucket/<foldername>
```

- b. Copy source code from local BitBucket folder to appropriate local GitHub folder. For example, from C:\BitBucket\AppName to C:\GitHub\TeamProject\_AppName folder

- c. Re-arrange application files and folders, as needed

- d. Update .gitignore file based on the source code selected, as needed.

- e. Add CODEOWNERS and (optionally) consensus.yml file to be used by Git repository branch protection rules. Instructions on how to modify the CODEOWNERS file can be found here: [About code owners - GitHub Docs](#)

4. Commit Git changes

- a. Add changes in the Git working directory to the staging area

```
git add .
```

- b. Commit staged files into local Git branch



```
git commit -m "First Commit"
```

- c. Push local repository content to a remote repository

```
git push -u origin <feature branch name>
```

## Pre-Migration Checklist

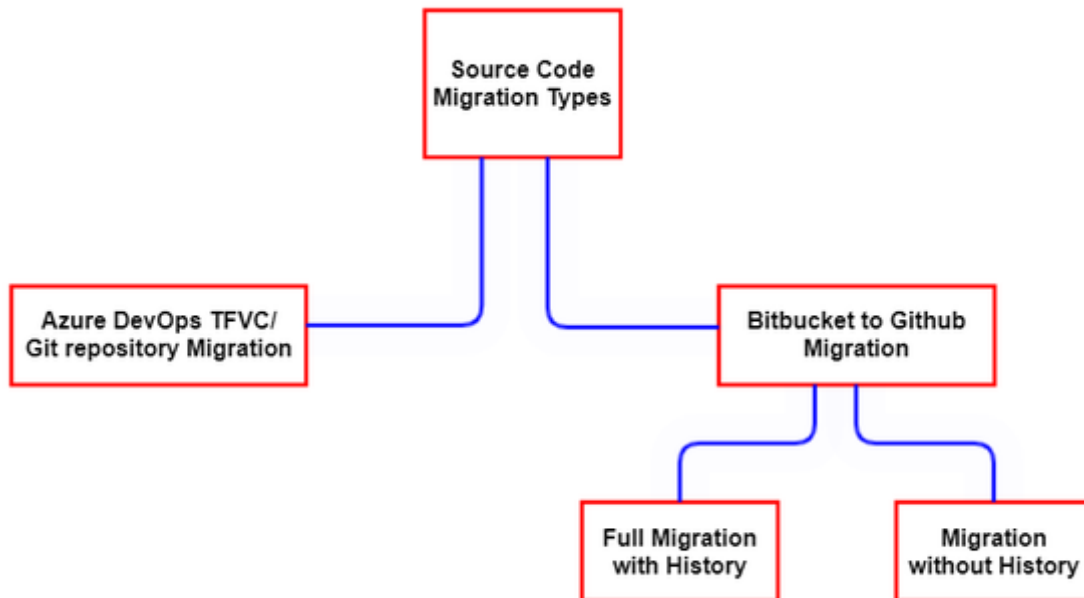
The following is the pre-migration checklist for the requestor to look into:

-  Confirm the list of repositories to be migrated over.
-  Confirm owner and AppCat ID of the repos.

- ☐ Determine if history is being migrated
  - Bitbucket repo history can be migrated if DLP team clearance covers repo history
  - TFVC source code can be migrated with the latest version only.
- 🔧 Clean up repository content
  - Remove unnecessary content and large files from the repo, so the migration process would be less error-prone and more efficient
  - Advised to trim the overall repo size to less than 1GB, so there will be room to grow after migration
  - Use [git-sizer instructions](#) to identify large file and binary files.
  - Remove obsolete branches.
  - Remove files over 100MB
  - Move binary files out from the repository
  - Move dependencies and build artifacts to Artifactory
  - Consider using .gitignore file to ignore files and paths that are not central to the project.
  - Consider splitting the repository if it houses multiple modules of the application ([Mono-Repo vs. Multi-Repo Comparison](#))
  - Consider migrate the latest version of code only if history is not needed.
  - Use [git-filter-repo Instructions](#) to help removing large files and binary files.
- ☐ Create [README.md](#) file for the repo and [CHANGELOG.md](#) file to record changes between versions, if not exist.
- 🔧 Request **Final scan for migration** to [DLP Program](#) when ready and provide the reply as clearance for full repo history or specific branch tip migration

## Source Code Migration Types

There can be the following types of source code migrations:



**NOTE:** “Full Migration with History” and “Migration without History” depends on the DLP scan. Endorsement from DLP team is required for “Full Migration with History”

## TFS (Azure DevOps Servers) to GitHub Migration

### **Note:**

**Completion of Onboarding process(to Team Project and GitHub) is a prerequisite for the below Migration process to begin.**

### *TFS (Azure DevOps Servers) Repository*

There are two types of repository available on TFS (Azure DevOps Servers), namely TFVC and Git. Git repository is compatible with GitHub and TFVC is not compatible with GitHub. Therefore, there will be extra steps in migrating TFVC repo.

### *Migration Approach*

The DLP scan team is using Bitbucket feature to conduct the DLP scan. Since TFS (Azure DevOps Servers) does not offer any native feature to do the DLP scan. Both TFVC and Git repositories on TFS (Azure DevOps Servers) will be moved to Bitbucket to conduct DLP scan before the repos being moved to GitHub.

### *Azure DevOps TFVC/Git repository Migration*

This playbook defines the guidelines how to migrate the latest version of the active TFVC/Git source code from on premises instance of Azure DevOps Server to GitHub Enterprise Cloud instance. The application active source code will be migrated from selected TFVC/Git branch to the new GitHub Repo Git repository.

To migrate an application active source code from Azure DevOps Server to GitHub Enterprise Cloud instance follow the steps in the following section.

#### Migration Process for Repositories

1. Prepare TFVC source code to be migrated
  - a. Identify the folders and files in TFVC/Git repository of the selected team project that is related to the application to be migrated.
  - b. Download identified folders and files onto your local computer using Visual Studio client. For example, C:\TFS\TeamProject\AppName folder
  - c. Review and cleanup downloaded files, where applicable. Please see the [Pre-Migration Checklist](#) for details on how to perform the cleanup.
  - d. Check if Visual Studio solution can be simplified and reduced in size.
2. Ensure that the repository can pass the DLP Scan
  - a. Import the repository into BitBucket to prepare for the DLP Scan.
  - b. Inform the DLP Team by sending e-mail to [dlp.program@bmo.com](mailto:dlp.program@bmo.com) so that they can begin the DLP Scan.
  - c. DLP Team informs the CIO Team (Owner) of the results of the scan.
  - d. If the application fails the test, have the repo owner remediate the issues in TFVC and repeat step 1.
  - e. Ensure that the repository passes the DLP Scan
3. Prepare GitHub repository
  - a. During the onboarding process, you must have created GitHub repository in GitHub Enterprise Cloud instance with Visual Studio .gitignore file, granted permissions to appropriate GitHub teams to the GitHub repository and configured branch protection rules for master/main (and developed branch, if applicable).
  - b. Now, clone GitHub repository onto your local computer.  
For example,  
C:\GitHub\TeamProject\_AppName folder  

```
git clone https://github.com/BMO-Prod/<repoName>.git
```
  - c. Create feature branch  

```
cd <repo name>
git branch <feature branch name>
git checkout <feature branch name>
```
4. Stage GitHub repository source code locally
  - a. Clone the BitBucket Repository locally. This is assuming that the repository was taken into BitBucket for the DLP Scan. Otherwise, the url should reflect the TFVC Server url.  

```
# Fill in the required values
git clone --branch <branchname> https://bitbucket.bmogc.net/<TFSSCAN>/<reponame>.git /c/repos/bitbucket/<foldername>
```
  - b. Copy source code from local BitBucket folder to appropriate local GitHub folder. For example, from C:\BitBucket\AppName to C:\GitHub\TeamProject\_AppName folder
  - c. Re-arrange application files and folders, as needed
  - d. Update .gitignore file based on the source code selected
  - e. Add CODEOWNERS and (optionally) consensus.yml file to be used by Git repository branch protection rules. Instructions on how to modify the CODEOWNERS file can be found here: [About code owners - GitHub Docs](#)
5. Commit Git changes
  - a. Add changes in the Git working directory to the staging area  

```
git add .
```
  - b. Commit staged files into local Git branch  

```
git commit -m "First Commit"
```
  - c. Push local repository content to a remote repository  

```
git push -u origin <feature branch name>
```

## GitHub Onboarding Process

**Note:** For full repo migration, new git repo is not required.

For information on using DevOps Tools Onboarding Portal ([DTOP Portal](#)) to onboard new repos to GitHub Enterprise Cloud, kindly refer to the below link.

[How to Onboard GitHub Enterprise Cloud\(GHEC\) using DTOP](#)



Users who need to upload code to GitHub must submit a [ServiceNow request](#) to be set up in the **office\SEC\_Proxy-Github-Upload-Allow** Active Directory security group (ID & Access -> User Groups -> Add Access to an active directory security group).

**i** Business Users who do not have access to “User Groups” on SNOW, please address the concern to your DevSecOps lead. They will help to raise the SNOW ticket.

### **Note: Onboarding AWS-enabled Repository**

AWS-enabled repository will be onboarded to GitHub Enterprise Cloud via DevOps Tools Onboarding Portal ([DTOP Portal](#)), kindly refer to [How to Onboard AWS-enabled Repository using DTOP](#)

#### Naming Convention for New Repository

The following table describes the basic naming standard for new repository in GitHub

Resource Type	Format	Length	Restrictions	Example
Repository	<i>{purpose}_{appcatid}</i>	1-40 (for purpose)	<ul style="list-style-type: none"><li>Alphanumeric, hyphen (-), period (.) and underscore (_)</li></ul>	edelivery_15103
Team with Read access	<i>{purpose}_{appcatid}</i>	1-255	<ul style="list-style-type: none"><li>Alphanumeric and hyphen (-)</li></ul>	edelivery_abc1234
Team with Privileged access	<i>{purpose}_{appcatid}_Admin</i>	1-255	<ul style="list-style-type: none"><li>Alphanumeric and hyphen (-)</li></ul>	edelivery_abc1234_Admin
Team with Normal access	<i>{purpose}_{appcatid}_User</i>	1-255	<ul style="list-style-type: none"><li>Alphanumeric and hyphen (-)</li></ul>	edelivery_abc1234_User

**For full repos migrating from Bitbucket**, the existing access groups will be directly copied to GitHub as Teams.

Migration repos will have the following naming by default:

<Project Key>\_<Repo Name>\_<AppCat ID>

where <Project Key> is the current Bitbucket project key and <Repo Name> is current Bitbucket repo name.

#### Managing User Access on GitHub

This section includes information about managing user access in GitHub Enterprise Cloud.

- [How to add/remove users in local GitHub team](#)

#### *How to add/remove users in local GitHub team*

For information on using DevOps Tools Onboarding Portal ([DTOP Portal](#)) to onboard new users to GitHub Enterprise Cloud or remove existing users, kindly refer to the below link.

#### [Adding/Removing users in GitHub team](#)

**i** In case of any issue while adding user/in case the Team Member (with Maintainer role) is unable to add other team members to GitHub by the above method, then the Team Member (with Maintainer role)/User who wants to be added to GitHub team can raise a ServiceNow request to the support team stating that this self-serve function is not working.

#### User Requesting for Access to GitHub Repo

For information on using DevOps Tools Onboarding Portal ([DTOP Portal](#)) to request access to a GitHub repository, kindly refer to the below link.

#### [Request For Access to GitHub Repo](#)

#### **GitHub PAT Token**

GitHub Pat token allows connectivity to GitHub bypassing the SSO Login

#### Steps to Obtain a PAT Token

1. To obtain a PAT token we first need to obtain a GitHub Service Account
2. The account must be email enabled and onboarded to the CyberArc
3. The Dom request must be raised with a well defined use case: [Intake requests on GitHub and ADO](#)

Only one of the 3 defined use cases are eligible for the PAT token

1. Connectivity from Ansible Tower
2. Connectivity from OpenShift
3. Connectivity from ArgoCD


*Please note: The PAT token request will be provided only for on of the three Use Cases defined above.*

## Appendix

This section provides the supplementary information on Migration to DevSecOps.

- [git-filter-repo Instructions](#)
- [Migration FAQs](#)
- [Service Account and Token for GitHub Connection](#)
- [Mono-Repo vs. Multi-Repo Comparison](#)
- [Azure AD vs GitHub Teams Discussion](#)
- [git-sizer instructions](#)
- [Branching Strategies: Git Flow vs. GitHub Flow Comparison](#)
- [DLP scan for GitHub Migration](#)
- [Intake requests on GitHub and ADO](#)
- [Connecting to GitHub of BMO Organization from Ansible and OpenShift](#)
- [Onboarding CD Tools \(Ansible, Openshift, Artifactory\)](#)
- [CDK Workload Migration Guide for Github Actions](#)

### git-filter-repo Instructions

 Using git-filter-repo is a destructive operation which should not be used lightly. Please read through the instruction and manual of the tool.

 Please take extra backup as indicated below. **DevOps Tools team does not have capability to restore a particular repo.**

- [Introduction](#)
- [Installation](#)
  - [Python 3 on Windows](#)
    - [After Installation](#)
  - [git-filter-repo](#)
- [General Operations](#)
  - [Basic functions](#)
    - [Analyze the repo](#)
    - [Basic Examples](#)
      - [Replace text strings in file history and commit messages](#)
      - [Remove big blobs](#)

#### Introduction

git-filter-repo is a git extension that can help identify and remove large blobs/files and secrets from git repo history.

Source code can be found here <https://github.com/newren/git-filter-repo>.

User manual can be found here <https://htmlpreview.github.io/?https://github.com/newren/git-filter-repo/blob/docs/html/git-filter-repo.html>

#### Installation

##### Python 3 on Windows

Python 3 is required. For Windows, embeddable package can be used.

##### Steps to Install Python















1. Launch ServiceNow Catalog homepage, [https://bmogc.service-now.com/sp?id=sc\\_category](https://bmogc.service-now.com/sp?id=sc_category).
2. Click **Service Catalog**.





3. Select **Software Services** from Icon list.

Home > Category



IT Operations 	IT Operations Administration 	Mobile Devices 
Network Operations 	People, Process & Change	Printers & Scanners 
Privileged Access Management (PAM) 	Procurement Services 	Real Estate & Business Operations
SmartCore Services 	<b>Software Services</b> 	Technology & Operations Intake 
Technology & Operations Privileged Access 	Telephone Services 	Transportation_Equipment Finance & IA 
Treasury & Payment Solutions (TPS) Support <b>TPS</b>	Virtual Desktop Services 	

4. Select Add & Remove Software Request.

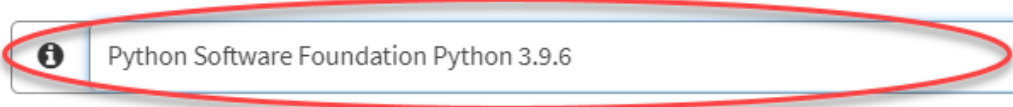
Home > Service Catalog > Software Services

### Software Services

Request installation or removal of software

<b>Add &amp; Remove Software Request</b> 	<b>Non-standard Software</b> 
---	---

5. Fill the requested information as per the instructions.

Requested For	Mandatory– prefilled from user profile.
Location	Mandatory– prefilled from user profile.
Computer Asset Tag	Mandatory– prefilled from user profile.
Contact Number	Mandatory– enter contact number if not prefilled.
Cost Centre	Prefilled from user profile.
OS	Prefilled once Computer Asset Tag is added.
Clarity Code	Provide a PR code associated to the request.
Request Type	Mandatory– choose Install option
Is this a bulk order for at least 50 users?	Select No
Select Software	<p>Mandatory- Add details</p> <p>* Software ?</p> <p>If you are unable to find the software you are searching for please try using a wild card search title or keywords from the title. If the software is part of the BMO approved list it will show u</p> <p> Python Software Foundation Python 3.9.6</p>

6. Upon completion- submit the request, a service request ticket will be generated.

After Installation

1. In the python local directory, copy `python.exe` to `python3.exe`

```
copy python.exe python3.exe
```

2. Add the local directory to system environment variable `PATH`, if it is not already included.
3. Open a new command prompt, execute the following to confirm python3 is executable

```
python3 --version
```

*git-filter-repo*

Copy the file [git-filter-repo](#) (from v2.34.0) to the directory that returned from the following command

```
git --exec-path
```

#### General Operations

git-filter-repo updates the repo including history, which the whole repo would be overwritten on the server.



***It is advised to take extra copy (another mirror clone) for backup/restore, and freeze any push to the repo server before the operation is completed or the push would be lost.***

Clone a mirror copy of the repo locally to work on

```
git clone --mirror <repo url>
```

***Clone another mirror copy of the repo in a different location as a backup, so it can be used for restoration.***

Once finish update, execute the following to push back to Bitbucket

```
git push --force
```

If a restore is required, run the same command from the backup clone to overwrite the repo on Bitbucket

#### Basic functions

Manual page can be found here: <https://htmlpreview.github.io/?https://github.com/newren/git-filter-repo/blob/docs/html/git-filter-repo.html>

Analyze the repo

```
git filter-repo --analyze
```

#### Basic Examples



Run with `--dry-run` option to review the report before actually applying the changes

#### Replace text strings in file history and commit messages

Put the text string(s) into a file outside of the repo (e.g. password.txt). Multiple strings should be on separate lines. By default, the text string(s) would be replaced by "\*\*\*REMOVED\*\*\*".

```
git filter-repo --replace-text ../passwords.txt --replace-message ..  
/passwords.txt
```

#### Remove big blobs

```
git filter-repo --strip-blobs-bigger-than <size>
```

## Migration FAQs

- ✓ What are the mandatory activities are expected by CIO Teams?  
Security scanning of repositories is mandatory and remediation is required for any secrets/exposure found prior to migration. DLP team will be enforcing strict governance on source code repositories.
- ✓ What is the data migration approach for the source code repository?  
We will be using Bitbucket Exporter to batch migrate the repositories.  
For approx. 1000 repos (1GB), it takes 3 hours.
- ✓ What about the linkages in Jira we have in the Bitbucket repository?  
GitHub will be integrate with Jira SaaS. Once Jira SaaS and GitHub Enterprise is integrated, the linkages will be maintained and available.
- ✓ How do I get onboarded into new tool once is enterprise available?  
Please use our automated onboarding portal : [BMO Agile & DevOps \(bmogc.net\)](https://bmogc.net)  
Service Now onboarding will be in the future phase.
- ✓ Are there training available for Azure DevOps Services and GitHub Enterprise?  
Yes, self-learning for the tools are available.
- ✓ What is the migration approach for Bamboo pipelines?  
There is no direct migration, as this is new pipeline. We will provide re-usable pipeline templates for development team to adopt. The dependencies will be assessed by CIO team and work with DevSecOps team to implement into the new pipeline.
- ✓ Today, the build farm is restricted. Can I have my own build farm environment?  
For initial roll out, we will maintain similar restriction. However, as we iterate the pipeline, we will properly test the use cases/pattern and enable them accordingly. This can be part of pipeline refinement after your migration complete.
- ✓ What build environment will be available?  
Windows, Mac OS, and Linux.
- ✓ Is there any code changes required for migration?  
There will be no code changes required. However, there may be a need to have new framework (such as node, Gradle) task to consolidate multiple tasks for the pipeline template to consume.
- ✓ How are the migration being prioritized?  
Based on repo complexity and size. As part of migration analysis, we will define the approach to minimize CIO delivery impact. We will engage LOB to kick start and provide playbooks and details.
- ✓ How do I get notified about program updates and changes?  
All announcements in regards to Program updates will be published [here](#).
- ✓ How do I add users to my GitHub team?  
To add users to your GitHub team please refer to steps in [How to add a user to the GitHub team](#).
- ✓ How to add user to the DevSecOps tools via DTOP?  
[DTOP portal](#) is to be used to onboard user to DevSecOps tools namely Jira, Confluence, GitHub, Azure DevOps, BitBucket, Bamboo, SonarQube. Please refer to steps in [DevOps Tools Onboarding Portal - Usage](#) for addition of user.

## Service Account and Token for GitHub Connection

As a security requirement, service accounts and tokens should be used for non-interactive integration connections.

- [How to request for a Service Account?](#)
- [How can Service Account be onboarded to GitHub?](#)
- [How to create a GitHub token for Service Account?](#)


### How to request for a Service Account?

Service Account in BMO can be requested via [ServiceNow Requests](#) to Directory Service team. Please mention in the service request that the service account needs to be **available in Azure AD with email enabled** (*Microsoft 356 license may be required*).

**i** If an existing service account doesn't have an email address, please request by creating a [General Request](#) in ServiceNow and assigning it to **RUN\_MSG\_CLIENT** RUN\_MSG\_EXCONLINE (pending confirmation).

**i** If the service account credential will be used in git proxy connection setting, please request to be added to AD group office\SEC\_Proxy-Github-Upload-Allow

Please request by creating a [General Request](#) in ServiceNow, with email approvals from the **owner of the service account** and **Tsang, Pang Chung** [PangChung.Tsang@bmo.com](mailto:PangChung.Tsang@bmo.com) (owner of the group), and assigning it to RUN\_INFROPS\_DIRECTORY

 As per standard setup, Service Account credentials must be stored in CyberArk. After the Service Account is created, the AD team will release the Service Account password in CyberArk safe.

### *How can Service Account be onboarded to GitHub?*

Service Accounts are being onboarded to GitHub in the same way as the users are being added to access GitHub.

Please refer to <https://bmo.atlassian.net/wiki/spaces/EDEVSECOPS/pages/281452084/How+to+add+a+user+to+the+GitHub+team> to onboard the Service Account to GitHub.

### *How to create a GitHub token for Service Account?*

PAT token access to GitHub is limited and monitored.

Please submit the use case detail of the PAT token usage in an [Intake requests on GitHub and ADO](#) for review. Once it is approved, an arrangement will be made with the requester on creating the PAT token.

### **Mono-Repo vs. Multi-Repo Comparison**

	<b>Mono-Repo</b>	<b>Multi-Repo</b>
<b>Definition</b>	One repository publishing multiple artifacts	Separate repository for each artifact used by the application
<b>Size</b>	A single repository creating multiple artifacts can increase the size of the repository	Each repository only contains code that's required to publish a particular artifact.
<b>Separation of concerns</b>	The entire repository must be published again generating multiple artifacts even if only one component has changed.	Each component can be updated individually based on requirements. Referencing components can then be updated to use new version
<b>Dependency Management</b>	Code becomes difficult to manage with integrated dependencies	Artifactory becomes the one source of truth
<b>Code Management</b>	Monolithic code base	Smaller code base to manage for each artifact
<b>Team Management</b>	Multiple people need to collaborate to publish artifacts	Smaller teams makes collaboration easier
<b>Technology Integration</b>	Multiple technologies may need to co-exist within the same repository. For example, a java application with a nodejs frontend.	Only technology needed for the artifact needs to be included.
<b>Code Distribution</b>	All code related to the application is in one repository	Code is distributed across multiple repositories.

### **Azure AD vs GitHub Teams Discussion**

As part of the migration process to GitHub, BMO and MS Team met with GitHub to review how to manage the complexity of syncing users across AD and GitHub Teams. Below are notes from the discussion as well as a sample repo from GitHub of how they have automated the process themselves.

#### July 18 Meeting Notes

- Team sync doesn't work with NESTED teams - no parent child relationships can exist in GitHub
  - Can add them directly in GitHub, but not in AD
  - Authorization repo query within GitHub to have a GitHub Actions that runs through the permissions.
  - ANTHONY: internal repository that acts as an identity access that allows teams to PR into it and get assigned access. Automation built in to GitHub.
    - Access for each of the repo needs to point access to the enterprise level so that the manager can attest the access every year - this would be the entitlements-app framework that is in GitHub.
    - Employee leaves the company - Remove the user from the IDP as part of the removal process will prevent them from logging into GitHub based on the EMU connection.
    - Clean up script that would run in GitHub to make sure that the AD and GitHub teams are remaining in sync. GitHub Actions running in the background to do these clean up activities.

- There are two places to manage the complexity either in GitHub Teams (noted above options) and Azure AD.
- DevOps Architect prefers to use the Azure AD to manage access
  - Given that different teams may create AD groups in different ways. Making it difficult to align them with users that need to be sent over to GitHub.
  - Creating a net new GitHub group then it kind of defeats using the Azure AD group as the starting point. - This is because right now each application may have its own permission groups. And each app asks the project team to create their own AD group. Making it difficult to manage as users migrate from one team/project to another within BMO.
- Potential to setup Authentication = Azure AD level with Authorization = GitHub level.

### Example GitHub Repo

- <https://github.com/github/entitlements-app>

#### Outcome

BMO to review the viability of the sample process GitHub provided for the BMO environment. This page has been created simply to capture the notes from the discussion for future use by the BMO DevSecOps team.

#### git-sizer instructions

- Download git-sizer from [github/git-sizer: Compute various size metrics for a Git repository, flagging those that might cause problems](#)
- Make sure you have git installed and in your PATH
- Open git bash
- Clone the repository that you wish to analyze:

```
git clone --mirror <path to repository>
# eg: git clone --mirror ssh://git@bitbucket.bmogc.net:7999/mpf
/mobile.git
```

- Navigate to the repository you want to analyze and run the following command:

```
cd <path to repo to be analyzed> # eg: cd /c/repos/bitbucket/mobile
<path to git-sizer>/git-sizer --verbose # eg: /c/apps/git-sizer/git-
sizer --verbose
```

- This should give you a report in the following format:

```
Processing blobs: 124210
Processing trees: 280000
Processing commits: 29579
Matching commits to trees: 29579
Processing annotated tags: 74
Processing references: 162
| Name | Value | Level of
concern |
| ----- | ----- |
| Overall repository size |
| * Commits |
| * Count | 29.6 k
```

* Total size		10.9 MiB
* Trees		
* Count		280 k
* Total size		138 MiB
* Total tree entries		3.55 M
* Blobs		
* Count		124 k
* Total size		4.65 GiB
* Annotated tags		
* Count		74
* References		
* Count		162
* Branches		2
* Tags		81
* Remote-tracking refs		79
Biggest objects		
* Commits		
* Maximum size	[1]	7.41 KiB
* Maximum parents	[2]	2
* Trees		
* Maximum entries	[3]	930
* Blobs		
* Maximum size	[4]	202 MiB

\*\*\*\*\*

History structure			
* Maximum history depth		7.04 k	
* Maximum tag depth	[5]	1	
Biggest checkouts			
* Number of directories	[6]	6.17 k	
***			
* Maximum path depth	[7]	22	
**			
* Maximum path length	[8]	223 B	
**			
* Number of files	[9]	36.1 k	
* Total size of files	[10]	1.86 GiB	
*			
* Number of symlinks	[3]	93	
* Number of submodules		0	

```

[1] c3bb4c62497a5ff427978b2b9565c954ab253e01
[2] 8da4db2229d210cbf20c2a4e7c2768abc7f8a8eb (refs/remotes/origin
/feature/RGMT-AUG12)
[3] e393ebda88157ac3cbd359e53472cc45e8fcb148
(e0616a2ce11c23d2069113be4ff79e17e8de4135:BMOMobileBanking/apps
/BMOMobileBanking/hybrid/node_modules)
[4] 471a70429ef8ef403c234b2a0ce5ba9c6c62061f
(bf3ce2f190379a703ea47915c032c7d9b5093623:BMOMobileBanking/apps
/BMOMobileBanking/iphone/nativeResources/MiSnapSDK/OptionalLibraries
/MiSnapCreditCardScanner/libCardIO.a)
[5] 826a671671c07243fc8b5384449dc5140ac8baa6 (refs/tags/1.0.0)
[6] a42582f95071057fd6bfd6ae5472b9b17727c61e8
(32465f0d582e8ec8704fbc1c7aa7e500fc0f4345^{tree})
[7] 6e0702c9a3e37f878441406a285c3172e67fa304
(d57bb7bbe8c7644ebfe3e87e1a6b14d4d2cd5^{tree})
[8] 0e821228c01a6ddacafea052d79b2bb11f2ea0cd
(5d78b29c991e778d37c461a595e9fee716e849f8^{tree})
[9] c092b88dc9c9ec6ca194ed5e2419f819f7f4bdd2
(98626b1077e6e9e6d726e60ee6a0594e38324d2c^{tree})
[10] 198d1a3f009a010567c722f800c305b9439f875f
(1a691543f375abe51d2c460928927154e71a4368^{tree})

```

- Take a look at the level of concern and address those items. eg: We know that GitHub has a limit of 100MB and the Maximum size for a blob is showing as 202MB. The footnotes identify items that are the ones causing the concern.
- These items can then be individually removed from the history (with analysis after each removal), or they can be removed using [git-filter-repo](#)



## Branching Strategies: Git Flow vs. GitHub Flow Comparison

The adoption of each branching strategy depends on several factors like:

- Existing branching strategy if any
- The project team's familiarity with Git and best practices.
- The complexity of the components within the repository.

### Overview

BMO will utilize the two main branching strategies approved by git and the user-base.

- GitHub Flow
- Git Flow

The following table explains the differences between the two strategies.

	GitHub Flow	Git Flow
<b>Pros</b>	<p>Clear and simple collaboration rules.</p> <p>Continuous integration and deployment.</p> <p>Encourages fast feedback loops, making it possible for teams to quickly identify issues and make changes.</p> <p>Continuous deployment is pretty much a must with GitHub Flow. You have no development branch where finished features sit and wait for release. created quickly becomes value delivered.</p> <p>There's less risk of technical debt with this branching strategy. Within Git Flow, one shortcut or sub-optimal piece of code can quickly get buried and become a refactoring nightmare. The emphasis GitHub Flow places on the shallow structures and small changes make it easier in the long run to identify and manage technical debt.</p>	<p>Complete rules with definite responsibilities of branches.</p> <p>Conducive to the distribution of traditional software.</p> <p>Provides very thorough and detailed version control as merges are bundled, clearly labeled, and able to be traced cleanly.</p> <p>Easy to scale out development. It simplifies parallel processing and continuous development by isolating features and ensuring that you never have to freeze either your development or master branch for release preparation.</p> <p>Flexible branching strategy. Easier to jump between parts of the development pipeline (say, between a feature for a later release and the urgent priorities of the current release).</p> <p>The code in the master branch remains very clean and organized, updated only with code that has been thoroughly tested and refined.</p>
<b>Cons</b>	<p>GitHub Flow is not as well-organized as Git Flow. Its speed comes at the cost of comprehensiveness and may make it harder to manage the overall development process.</p> <p>This branching strategy emphasizes constant deployment. While this can work well for some software teams, others will find it limiting as they seek to make larger releases or if they want to test multiple features together before deployment.</p> <p>The master branch can become cluttered more easily since it functions as both the production and development branch. Release preparation and bug fixes both happen in this branch — and require extra attentiveness.</p>	<p>Many branches with complicated rules</p> <p>Heavy maintenance workload for released versions</p> <p>The rigid structure and specific development path of Git Flow conflicts with the iterative approach of the Agile methodology.</p>

**Example 1:** If the user's repository contains a singular main/master branch, and smaller update branches, GitHub Flow should be easier to adopt for them.

**Example 2:** If the user's repository contains a main/master branch as well as multiple other trunk branches (DEV/TEST/QA/PROD), GitFlow should be easier to adopt.

## GITHUB\_FLOW

### Create a Branch

Create a branch in your repository. A short, descriptive branch name enables collaborators to see ongoing work at a glance. Creating a branch, provides a space to work without affecting the default branch and allows collaborators a chance to review the work prior to deployment.

### Make Changes

On your branch, make any desired changes to the repository. Your branch is a safe place to make changes. If you make a mistake, you can revert your changes or push additional changes to fix the mistake. Your changes will not end up on the default branch until you merge your branch. Commit and push your changes to your branch. Give each commit a descriptive message to help you and future contributors understand what changes the commit contains. Ideally, each commit contains an isolated, complete change. This makes it easy to revert your changes if you decide to take a different approach. For example, if you want to rename a variable and add some tests, put the variable rename in one commit and the tests in another commit. Later, if you want to keep the tests but revert the variable rename, you can revert the specific commit that contained the variable rename. If you put the variable rename and tests in the same commit or spread the variable rename across multiple commits, you would spend more effort reverting your changes. By committing and pushing your changes, you back up your work to remote storage. This means that you can access your work from any device. It also means that your collaborators can see your work, answer questions, and make suggestions or contributions. Continue to make, commit, and push changes to your branch until you are ready to ask for feedback.

### Create a pull request

Create a pull request to ask collaborators for feedback on your changes. Pull request review is so valuable that some repositories require an approving review before pull requests can be merged. If you want early feedback or advice before you complete your changes, you can mark your pull request as a draft. When you create a pull request, include a summary of the changes and what problem they solve. You can include images, links, and tables to help convey this information. If your pull request addresses an issue, link the issue so that issue stakeholders are aware of the pull request and vice versa. If you link with a keyword, the issue will close automatically when the pull request merges. In addition to filling out the body of the pull request, you can add comments to specific lines of the pull request to explicitly point something out to the reviewers.

### Merge your pull request

Once your pull request is approved, merge your pull request. This will automatically merge your branch so that your changes appear on the default branch. GitHub retains the history of comments and commits in the pull request to help future contributors understand your changes. GitHub will tell you if your pull request has conflicts that must be resolved before merging. Branch protection settings may block merging if your pull request does not meet certain requirements. For example, you need a certain number of approving reviews or an approving review from a specific team.

### Delete your branch

After you merge your pull request, delete your branch. This indicates that the work on the branch is complete and prevents you or others from accidentally using old branches. Your pull request and commit history will not be deleted. You can always restore your deleted branch or revert your pull request if needed.

## GIT\_FLOW

[GitFlow](#) is a branching model for Git, created by Vincent Driessen. It has attracted a lot of attention because it is very well suited to collaboration and scaling the development team.

One of the great things about GitFlow is that it makes parallel development very easy, by isolating new development from finished work. New development (such as features and non-emergency bug fixes) is done in “*feature*” branches, and is only merged back into main body of code when the developer(s) is happy that the code is ready for release.

### Collaboration

Feature branches also make it easier for two or more developers to collaborate on the same feature, because each feature branch is a sandbox where the only changes are the changes necessary to get the new feature working. That makes it very easy to see and follow what each collaborator is doing.

### Release Staging Area

As new development is completed, it gets merged back into the “*develop*” branch, which is a staging area for all completed features that haven’t yet been released. So when the next release is branched off of the develop branch, it will automatically contain all of the new stuff that has been finished.

### Support For Emergency Fixes

GitFlow supports “*hotfix*” branches - branches made from a tagged release. You can use these to make an emergency change, safe in the knowledge that the hotfix will only contain your emergency fix. There’s no risk that you’ll accidentally merge in new development at the same time.

New development (new features, non-emergency bug fixes) are built in **feature branches**:

- Feature branches are branched off of the **develop branch**, and finished features and fixes are merged back into the **develop branch** when they’re ready for release:
- When it is time to make a release, a **release branch** is created off of **develop**:

- The code in the **release branch** is deployed onto a suitable test environment, tested, and any problems are fixed directly in the release branch. This **deploy -> test -> fix -> redeploy -> retest** cycle continues until you're happy that the release is good enough to release to customers.
- When the release is finished, the **release branch** is merged into **master** and into **develop** too, to make sure that any changes made in the **release branch** aren't accidentally lost by new development.
- The **master branch** tracks released code only. The only commits to **master** are merges from **release branches** and **hotfix branches**.
- **Hotfix branches** are used to create emergency fixes:
- They are branched directly from a tagged release in the **master branch**, and when finished are merged back into both **master** and **develop** to make sure that the hotfix isn't accidentally lost when the next regular release occurs.

## DLP scan for GitHub Migration

1. At the time of CIO migration kick off, the DevSecOps team will request an initial batch scan request on all the repos in the list.
2. Vulnerability results will be sent to respective repo owners (STOs) for remediation.
  - If the scan is clean, the repo owner will not receive any notification.
  - DLP team should be updated for any ownership changes, and a new scan may be required for the latest owners to receive the results.
  - Project teams should request directly to DLP team on [dlp.program@bmo.com](mailto:dlp.program@bmo.com) for rescanning or other ad hoc scanning.
3. Below are the conditions on whether repo history can be migrated.
  - If a repo contains no secret during the initial scan, it can be migrated with history.
  - If a repo is found with secrets, it has to be remediated.
    - If all secrets have been remediated, the repo can be migrated with history.
  - If only select branches can be remediated, then only those branches will be migrated (without history).
4. Final scan is required before the migration.
  - The scan request should indicate that it is for final migration scan by keeping the subject line of the request mail as *"final scan for migration"*. Once requested for the final migration scan, the repo is expected to be frozen (i.e. no further changes in the repo).
  - Repo owner will receive the result as a clearance for migration (see migration condition on point 3 above).

 DevSecOps team will verify the repo during the migration, and it will not be migrated if any changes is found.

## Intake requests on GitHub and ADO

For changes or new capability on GitHub and ADO, please following the steps below.

- [1. User to create](#)
- [2. Please fill in the below details while submitting the new Intake.](#)
- [3. Field Name](#)
- [4. Description](#)

### 1. User to create

Requesters to go to the JIRA intake board:

<https://bmo.atlassian.net/jira/software/c/projects/DOM/boards/24594>

Click "Create" to create new intake ticket

And select intake for Issue Type:

### 2. Please fill in the below details while submitting the new Intake.

All fields with \* are mandatory fields. We will not process the ticket unless the mandatory fields are properly filled.

3. Field Name	4. Description
Project *	Enterprise DevSecOps Modernization (DOM)
Issue Type*	Intake
Request Type*	
Summary*	Provide high level overview of what's required

Description*	Provide detailed description of the requirement. Include attachments if any
Use Cases*	Provide detailed use cases
Desired Date of Fulfillment	Provide desired date for requestor to receive estimate or have resource(s) identified.
Priority	
Estimate Level Engagement*	
Clarity PR #*	Provide Clarity Project Number (PR) to fund this initiative request.
Requester*	Primary contact for this intake
Requesting CIO*	Requesting CIO
LOB*	Requesting LOB
If Other	If Selected "Other" for LOB, please specify Line of Business here.
Lead Solution Architecture	Name of LSA
Accountable STO *	Accountable STO/ Senior manager
Platforms Impacted	Please check all platforms that are getting impacted with this request
Reporter *	Name of Reporting person
Labels	Select Label that applies

### Connecting to GitHub of BMO Organization from Ansible and OpenShift

Please consult the following documents from the respective support teams on connection setup from Ansible and OpenShift to GitHub.

[Using Github from Ansible Tower](#)

[Azure DevOps/GitHub/OpenShift Integration Cheat Sheet](#)

Please first connect with Ansible [Ansible Tower Support](#) or OpenShift [Support \[OpenShift\]](#) support team on issues connecting to GitHub from the respective systems. DevSecOps Tools team will involve if the issue cannot be resolved after the initial investigation.

### Onboarding CD Tools (Ansible, Openshift, Artifactory)

CI and CD tools are working very closely together. The CD tools onboarding process can be found with the links below. If you need help in the onboarding process, please consult the corresponding platform team.

Ansible: [Ansible Tower Onboarding Process - DevOps Deployment Automation - BMO Enterprise Confluence \(atlassian.net\)](#)

Artifactory: [Artifactory Onboarding Guide - DevOps Deployment Automation - BMO Enterprise Confluence \(atlassian.net\)](#)

Openshift: [OpenShift Onboarding Process - DevOps Deployment Automation - BMO Enterprise Confluence \(atlassian.net\)](#)

## Public Announcements

### Security Best Practices for Jira and Confluence

#### To the users of Confluence & Jira:

**It's important we ALL understand that security is everyone's responsibility!** We would like to remind all BMO employees and contractors to safeguard all sensitive information types, including, but not limited to:

- Personally Identifiable Information (PII)
- Payment Card Industry Data Security Standard (PCI DSS)
- Passwords (BMO Credentials)
- API keys/secrets

Practicing proper information security practices will prevent security breaches and uphold BMO's strong Cyber Security posture.

## Best Practices for Jira and Confluence

### Do's:

- Continue to **educate** your team with security best practices
- Adopt to the principle of '**least privilege access**' (e.g., ensure projects are not open for the whole organization to see, but only locked down specifically for those who truly need it)
- Routinely audit activity logs and **remove accounts that are no longer required**

### Don'ts

- **Do not store any sensitive data** within Production OR Non-production environments in any of the file sharing/collaboration/ticketing spaces (e.g., Personally Identifiable Information (PII), Payment Card Industry Data Security Standard (PCI DSS), Passwords (BMO Credentials), and API keys/secrets)
- **Do not grant excessive access** – grant the minimum system resources and authorization that a user needs to perform their daily work functions
- **Do not forget** to regularly review product audit logs including user security events, product access and admin permission

### If your Active Directory password has been compromised

1. **MUST** change it immediately and inform your manager
2. For any technical issues related to Active Directory passwords, please contact your local support or [Help Desk](#)

### Resources: Supporting BMO Standards and Password Tips

1. [Data Classification and Protection Management Standard](#)
2. [Active Directory Management Standard](#)
3. [Enterprise Identity and Access Management Standard](#)
4. <https://bmo.sharepoint.com/sites/classic-TandO/FCU/SecurityCentre/PublishingImages/2.9---Password-Infographic.jpg>

Remaining in compliance with the standards above ensures alignment with [BMO's Code of Conduct](#) and ultimately the **protection of our employees and clients**.

Thank you for your cooperation,

*BMO Information Security – Financial Crimes Unit*

### References

<https://support.atlassian.com/security-and-access-policies/docs/how-to-keep-my-organization-secure/>