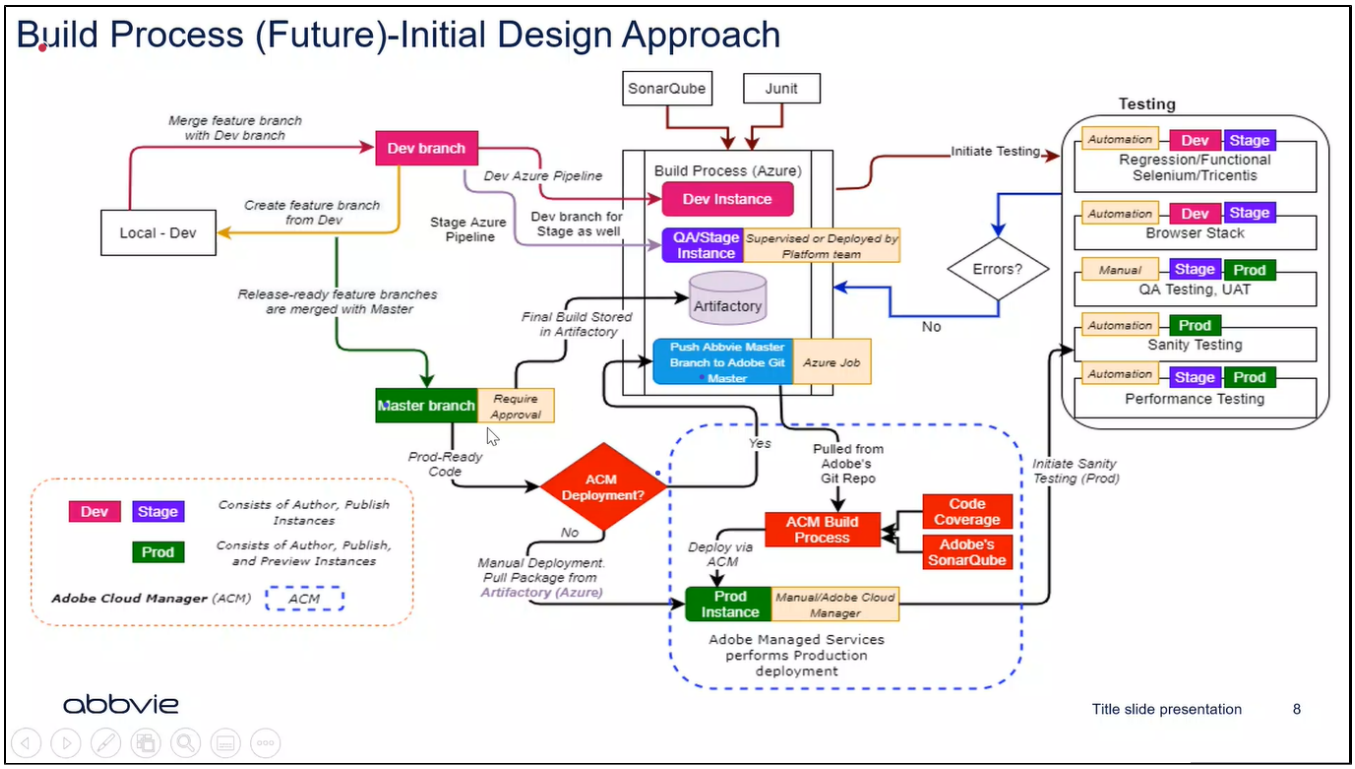
Implementation of AWCM - Consolidated

> *The below document contains the information/requirements gathered from OUS & US AEM Teams in different conversations happened during Weekly calls/ Working Sessions and the developments made by OUS-DevOps Team in different stages of the pipeline migration accordingly.*
>
> *All the developments in the pipeline are made as per the agreement with the AEM team.*
>
> *The Planning and Design related documents have all recorded information about the Interactions with the Dev team, Requirement gathering and the Plan Developed to progress to Implementation Stage*

Preface: AWCM Migration Project is to move the Implementation of the Adobe Experience Manager(AEM) pipeline in Jenkins to Azure DevOps. The purpose of the Azure Pipeline is to merge the latest code on the Azure Repos into AEM (Adobe Experience Manager) Repositories

The architecture below depicts the Build Process in Jenkins.



The plan is to replicate the above implementation in Jenkins to Azure DevOps

As a first step of migration, the entire structure is divided into three parts and named the three stages as Phase I, II & III. All the pipelines are manual pipelines(CI Disabled).

Phase I: The pipeline on Dev Branch (non-prod) **to deploy on Dev and Stage** Environments.

Phase II: The pipeline on the **Master Branch to generate an artifact** and to save it in Azure Artifacts.

Phase III: To **merge the latest code** on Master branch of the Azure Repos to their respective Repositories in Adobe. To update the Multimodule Repo at Adobe

Implementation Design of AWCM Phase I

> ***Link to the AWCM Pipeline Phase I***
>
> *Pipeline to deploy the code with different AWCM Profiles in different Environments*

Tools/Utilities:

- Maven (Version - 3.6.0)
- SonarQube
- Azure yml Pipelines

Pool used:

- vmss-ubu-1804-agentpool

AWCM Profiles for Dev Environment:

- dev-author-deploy
- dev-publish-deploy

AWCM Profiles for Stage Environment:

- stage-author1-deploy
- stage-publish1-deploy
- stage-publish2-deploy
- stage-publish3-deploy

All the possible combinations of above profiles are also enabled as parameters

Pipeline Stages (Dev Branch):

There are three stages in the pipeline. Build to compile the code. Deploy to Dev Environment & Deploy to Stage Environment.

- Build

In this stage, the code gets compiled using Maven task and runs the SonarQube analysis. SonarQube Build breaker task is added to break the build in case of Quality gate failure

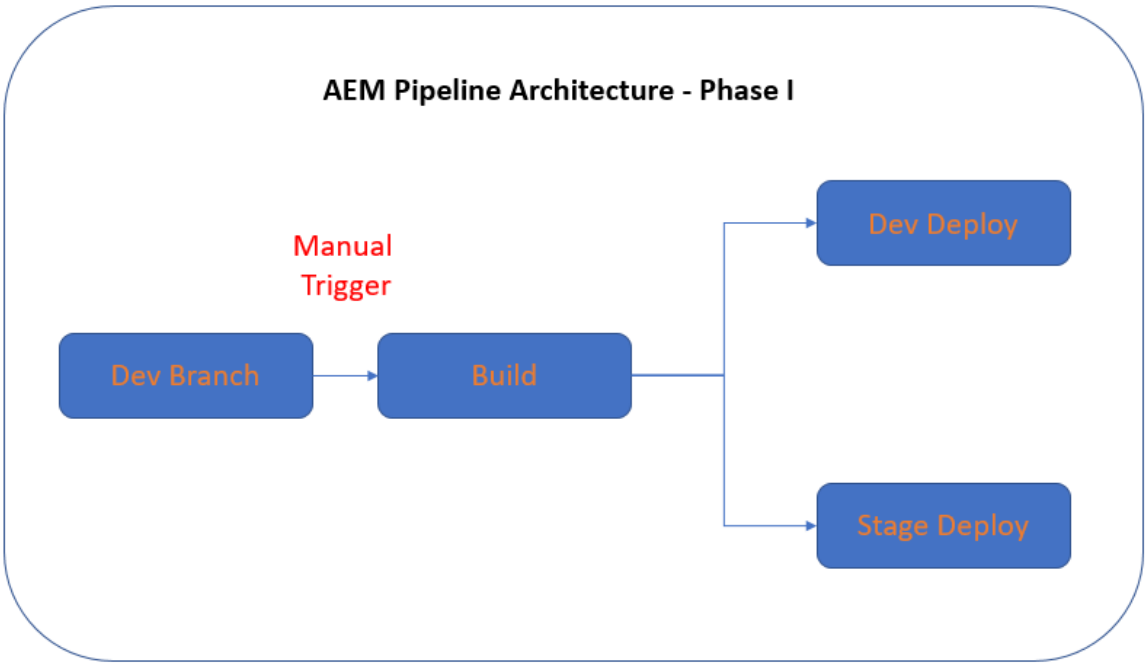
- Deploy to Dev

In this stage, AWCM Profiles for Dev deploys the code to Dev Environment (Once Approved)

- Deploy to Stage

In this stage, AWCM Profiles for Stage deploys the code to Stage Environment (Once Approved)

Workflow:



Specifications on the Pipeline:

1. The pipeline has been enhanced with a feature to select the profiles before running the pipeline. The parameters are set in such a way that multiple profiles can also be selected at once. Hence, CI (Continuous Integration) has been disabled.
2. Sonar Toggle has been created during the pipeline configuration in agreement with AEM team to avoid conflicts in build as the existing code was not set according to Azure SonarQube Quality gates. As the target environment has code scan mechanism, AEM team doesn't want the Sonar Analysis at Azure. Will remove this feature in future when it is confirmed by OUS AEM Team.

Mail from AEM on SonarQube

Initial implementation of Phase I : RE_ Initial Azure Pipeline for AWCM completed.msg

The above mail has the information about the first build that was successful and the Architecture diagram that was designed for Phase I and the approval from the Stakeholder Narava, Hema Sandeep

---

Implementation Design of AWCM Phase II

OUS-DevOps team, in agreement with the OUS AEM team has designed the Phase II pipeline in congruence with the Jenkins setup. The Phase II pipeline is built specifically to create artifact without deployment stage in it.

Stories that are qualified by the Dev team in Phase I after their Testing and Analyzing, the feature branch of each qualified stories is merged to their respective 'master' branches one after another. After the merge is complete with approval, Dev Lead will manually trigger the master pipeline which stores the Artifact in Azure Feed; which means the Dev Lead has the permission to approve the Pull Request raised by any developer

Link to the AWCM Pipeline Phase II

> *Pipeline on master to generate artifacts for future deployments on Phase III*

Tools/Utilities:

- Maven (Version - 3.6.0)
- SonarQube
- Azure Artifacts
- Azure yml Pipelines

Pool used:

- vmss-ubu-1804-agentpool

Pipeline Stages (Master Branch):

This pipeline runs on the **Master** Branch has a single Build stage with different tasks to publish the artifact.

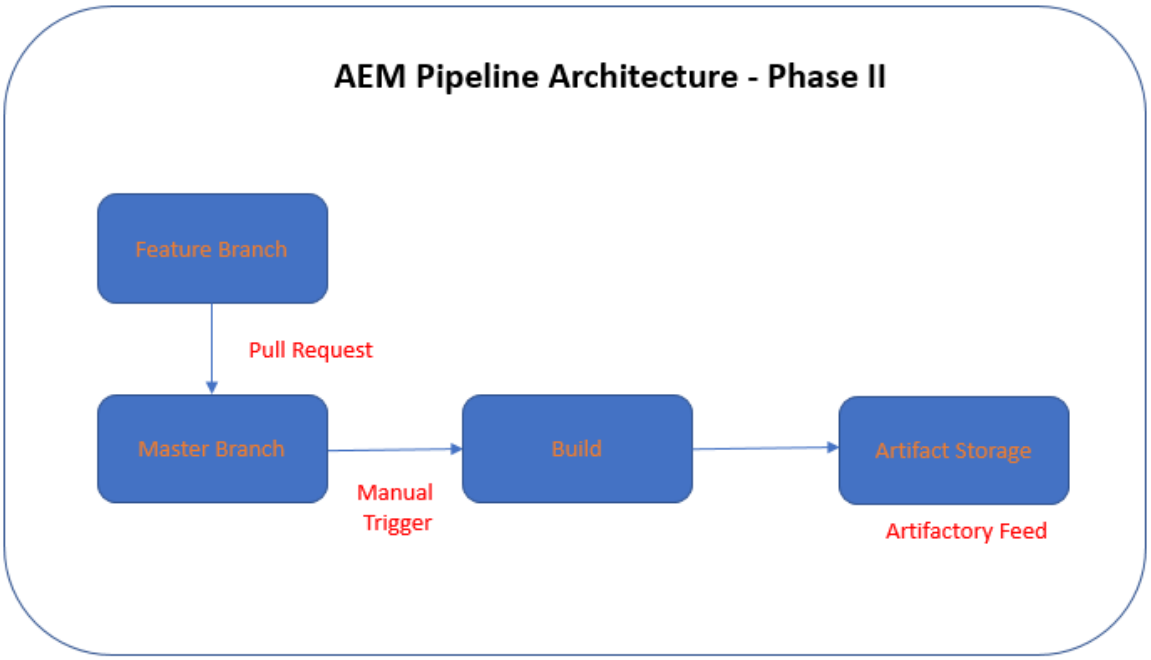
- Build

In this stage, the code gets compiled using Maven task and runs the SonarQube analysis. SonarQube Build breaker task is added to break the build in case of Quality gate failure. Copy tasks to copy the files - ZIP & JAR and to publish them on Azure Artifacts (awcm-ous-prod-artifactory) using Maven tasks

> ***Link to Azure Artifacts***
>
> *The artifacts published are in jar and zip formats*

Dev team may download these Artifacts published in Azure Artifacts and manually merge to Adobe which does Production deployment on the Adobe Cloud Manager.

Workflow:



Implementation Design of AWCM Phase III

Pipeline Name: abbvie-ous-awcm-merge

This phase has two episodes:

- **Standalone Merge**

  Standalone Merge is to merge the latest code from Azure Repos to their respective Repositories in Adobe

- **Multimodule Merge**

  Multimodule Merge is to update the abbvie-cm (Adobe Multimodule Repo) in Azure Workspace and push back to Adobe.

How does Standalone Merge happen?

Once the Phase II completes i.e, Pull request to Master Branch is approved, the developer can manually trigger the pipeline by selecting the required branch and the Merge type(Standalone) to push the latest code from Azure Repo to its respective Adobe Repo. Before triggering the pipeline, the required Repo Name has to be uncommented in the pom.xml file.

Link to refer Standalone Merge

> *Console Output of a Standalone Merge*

How does Multimodule Merge happen?

First implementation was a Z-Model Setup which is similar to current base setup but has a local cloning. This was later changed as AEM team do not prefer the .gitmodules change

Second Implementation was a parallel syncing model which was later altered as US AEM Team looking for a completely self-managed pipeline

Current Implementation ON-HOLD is again a Z-Model with complete change in the mechanisms like Workspace model (no local space), Self-managed Mappings which is merging the latest code to Target Environment without changes in .gitmodules file

Reasons to put the work on-hold:

Due to frequent changes in AEM Leads and Consumers and their On-spot requirements, Phase III implementation has become challenging.

The 95% of the pipeline configuration has been done in the presence of OUS & US AEM teams during the Working sessions.

- This has been put On-hold as there are fresh requirements of uncommenting pom.xml file before merging Multimodule code.
- Learnt that there is another requirement from a recent call to automatically modify the submodules through pipeline.
- OUS-DevOps Lead has sent a mail asking for a justification in the above matter. Awaiting a reply with justification to resume the work.

Mail sent by DevOps Lead

Link to refer Multimodule Merge

> *Console Output of a Multimodule Merge*

Tools/Utilities:

Pool used:

- vmss-ubu-1804-agentpool

Pipeline Stages:

It has three stages with different tasks in it.

- Load Mappings
- Standalone Merge
- Multimodule Merge

Load Mappings: The mappings mechanism is set in order to map the respective Azure & Adboe URLs, Repo Names & .xml files.

Standalone Merge: Based on the mappings in the previous stage, pipeline pulls the latest code from Azure Repo and merges into the Adobe environment to update the Adobe Repo with the latest.

Multimodule Merge: This can be done only after the Standalone Merge happens. This stage involves different steps:

Step-1: Clones the Adobe Multimodule Repo (abbvie-cm) into Azure Workspace

Step-2: Pulls the submodules recursively using special combination of git and linux commands with the aid of .gitmodules file (The altered .gitmodules will be reset before merging)

Step-3: Once the abbive-cm repo is completely cloned in the Azure Worksapce, the repo will be updated with using update commands

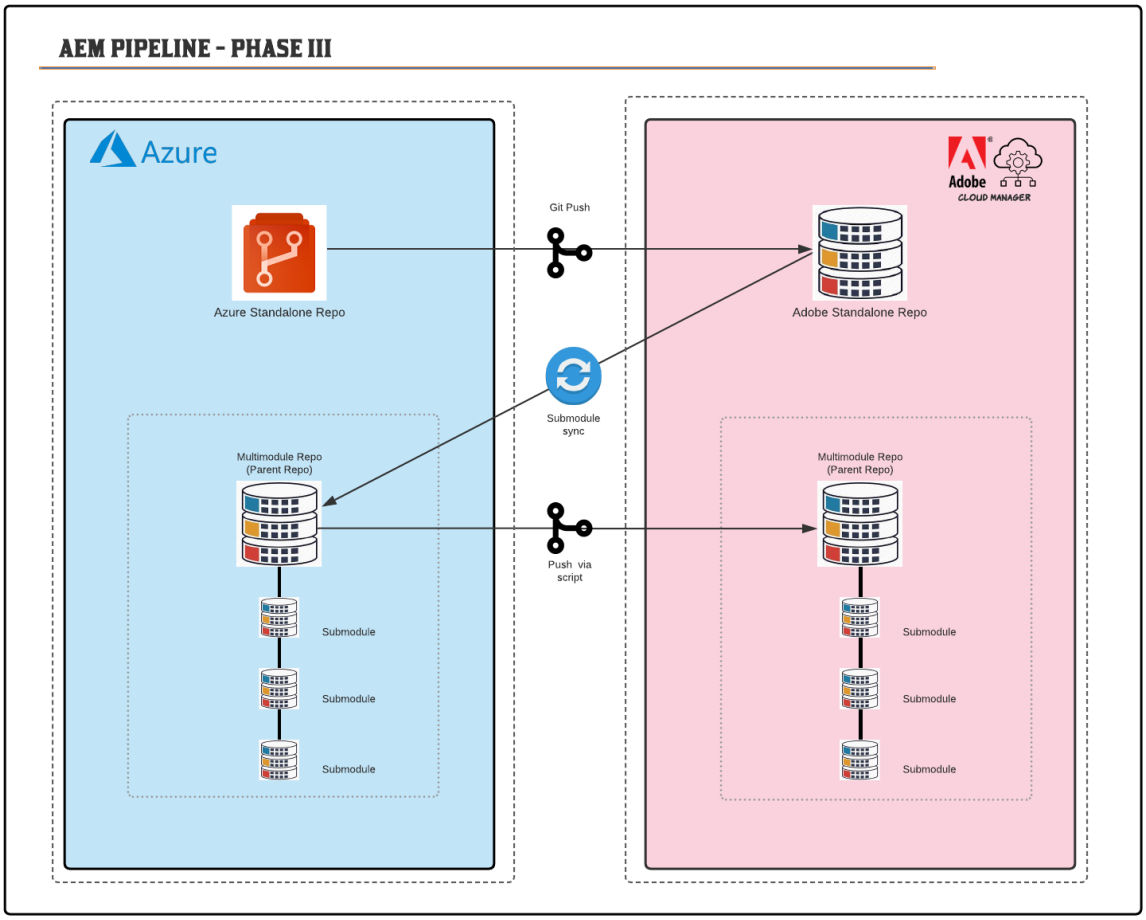
Step-4: The updated Multimodule Repo will be pushed back to the Adobe CM

To consolidate, the stage Multimodule Merge pulls the updated code from Adobe Standalone Repo to update the abbvie-cm at Azure and pushes back the updated abbvie-cm to Adobe

Link to the Phase III Pipeline

> *This Pipeline has both Standalone & Multimodule Stages (Phase III)*

Workflow:



Specifications on the Pipeline:

1. The pipeline has been enhanced with a feature to select the target branch and the type of merge to happen before running the pipeline. Hence, CI (Continuous Integration) has been disabled.
2. The submodules of Adobe Multimodule Repo is recursively cloned using .gitmodules file into Azure Workspace. However, the change in the .gitmodules is reverted before the merge happens.

Validation of the Pipeline Implementation AWCM Phase I & II

Validation of the these pipelines are completed and approved by the Stakeholders Narava, Hema Sandeep & Suresh Madham

A dry run of these pipelines is done and handed over these pipelines to the OUS AEM Team in the mid of the May' 2021

The working of these pipelines has been confirmed and Phase I & II are completely functional

Mail confirming the Phase I & II functionality

Validation of the Pipeline Implementation AWCM Phase III

The Standalone Merge is successful and has been validated by the developer, Asad during a test run.

The only task to be finalized is the Multimodule Merge. This stage has been configured and in working condition as per the previous requirements. As the AEM team has other requirements, the process has been on-hold to get the crystal clear requirements and to get the justification in Abbvie terms for those requirements. The process resumes once both the teams (OUS DevOps & OUS AEM) agree upon the justification sent.

Ad hoc Requests from AWCM Dev Team:

- [x] To provide a Restricted Environment for a developer on a PR (Pull Request) from Dev to Master Branch - To assign the task to a reviewer (Kancherla, Rishi Kumar) before the merge happens to Master Branch
- [x] To rename the Artifacts and the Azure Repos

- [x] To rename Environments
- [x] To have the mappings in .xml file instead of pipeline
- [x] To have a self-managed pipeline
- [x] To restructure the AWCM Folders
- [x] Not to change the .gitmodules file before merge
- [x] Permissions on the pipelines
- [ ] To have the root pom.xml edited before Multimodule merge happens
- [ ] To have the submodules creation/deletion automatically