**AZURE DEVOPS TECHNICAL DOCUMENT FOR IOS APPLICATION**

**Goals**

The goal is to implement Continuous Integration and Continuous Deployment/Continuous Delivery pipeline on Azure DevOps for IOS application (HeartMate Touch App**)**

**Requirement**

To implement the Continuous Integration and Continuous Deployment pipeline for IOS Application on Azure DevOps we consider the following requirement.

* Need access for Azure DevOps Account.
* Need access for Bitbucket repository (OnPrem).
* Need Dedicated Mac-OS Agent Machine with Xcode installed (Version 10.15: "Catalina").
* Need access for Artifact Repository.
* Need Apple Account to get a Provisioning profile and p12 (Apple Certificate) to sign iOS application.

**Out of scope**

* Automatic build process not supported as Bitbucket is On-premises. Release team should trigger build manually.
* Android Build and Deployment not supported, It can only support IOS Application for Build and Deployment.
* CI/CD can only support IOS Application Deployment into App Store Connect.
* It can’t support Microsoft Hosted Mac Agent since Bitbucket is On-Premises, it can only support Self-Hosted Mac Agent (It should be there in Abbott Network).
* The process (CI/CD) can’t store Build Artifacts (IPA) inside Azure Devops.

**Approach**

To implement the Continuous Integration and Continuous Deployment pipeline for IOS Application bellow approach has been followed

Below approach are tried in local machine, not on the Abbott infrastructure

**Pipeline with Windows Self-Hosted Agent**: The testing for the IOS Application Build pipeline with Window Self-Hosted Agent got failed due to the build of IOS application Xcode is required. In the Windows it is unable to install Xcode. Moreover, Xcode doesn’t have Windows version. It can’t Build the IOS Application with Windows Agent.

**Pipeline with Ubuntu Self-Hosted Agent**: The testing for the IOS Application Build pipeline with Ubuntu (Self-Hosted Agent) got failed because to build the IOS application Xcode is required. In the Ubuntu, it is unable to install Xcode because Xcode doesn’t have Linux version. It can’t Build the IOS Application with Ubuntu (Linux) Agent.

**Pipeline with Mac Microsoft Hosted Agent**: The testing for the IOS Application Build pipeline with Mac (Microsoft-Hosted Agent) it got succusses with sample code (code in Azure Repos) by default Microsoft Hosted Agent have Xcode.

To Build the Pipeline with Microsoft Hosted Mac Agent, and the code is in On-Premises Bitbucket, the Build got failed.

Due to Bitbucket is On-Premises which is not accessible and the firewall policy does not allowed to clone the repository for Microsoft Hosted Mac agent.

**Conclusion:**

To Build and Deploy successful IOS Application, if the Bitbucket is in On-Premises then the Mac Agent should also in the Same network else it cannot allow to clone the repository from On-Premises Bitbucket.

Mac agent should have Xcode installed.

**Security and Privacy**

* To login into Azure Devops account, the release team requires valid authentication.
* The valid credentials are required for Artifact Repository to Deploy the Artifact (IPA).

**Test Plan**

Xcode Test

Xcode Test will be using the XCTest framework to write unit tests for Xcode projects that integrate seamlessly with Xcode's testing workflow.

Tests assert that certain conditions are satisfied during code execution, and record test failures (with optional messages) if those conditions aren’t satisfied.

Tests can also measure the performance of blocks of code to check for performance regressions, and can interact with an application's UI to validate user interaction flows.

**Deployment**

When the Continuous Integration pipeline succeed, it will create the Artifact (IPA) and then Deployment pipeline will start.

Continuous Deployment pipeline will take the Artifact generated by the CI pipeline and it will deploy to the App Store Connect.

**Rollbacks**

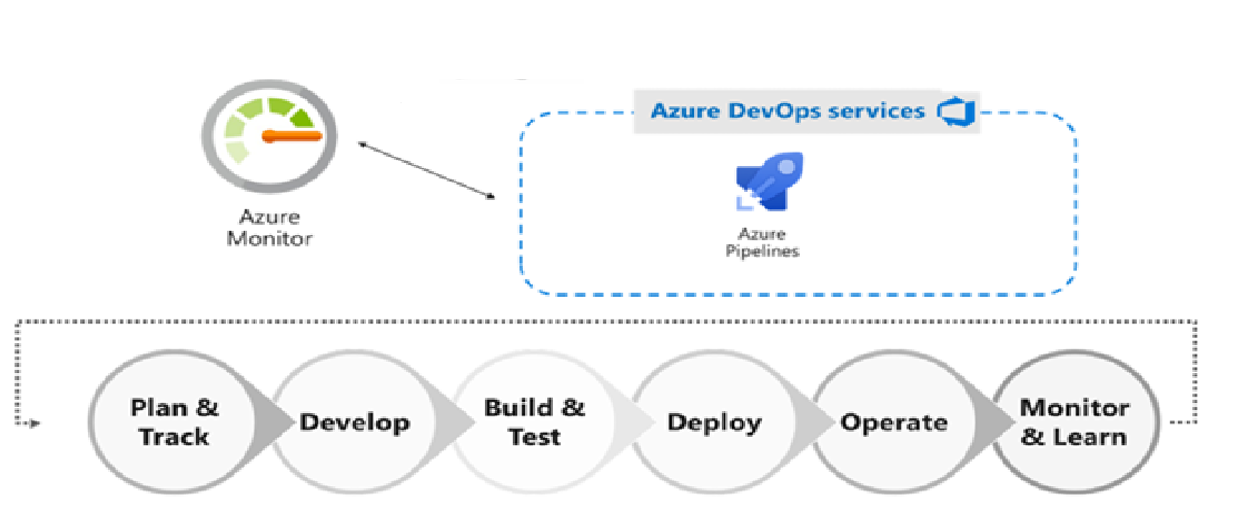
When a deployment fails, it is likely to leave an environment in an unhealthy state. Then we can set a rollback strategy to get the environment back to a healthy state.

we can consider below options as a rollback strategy.

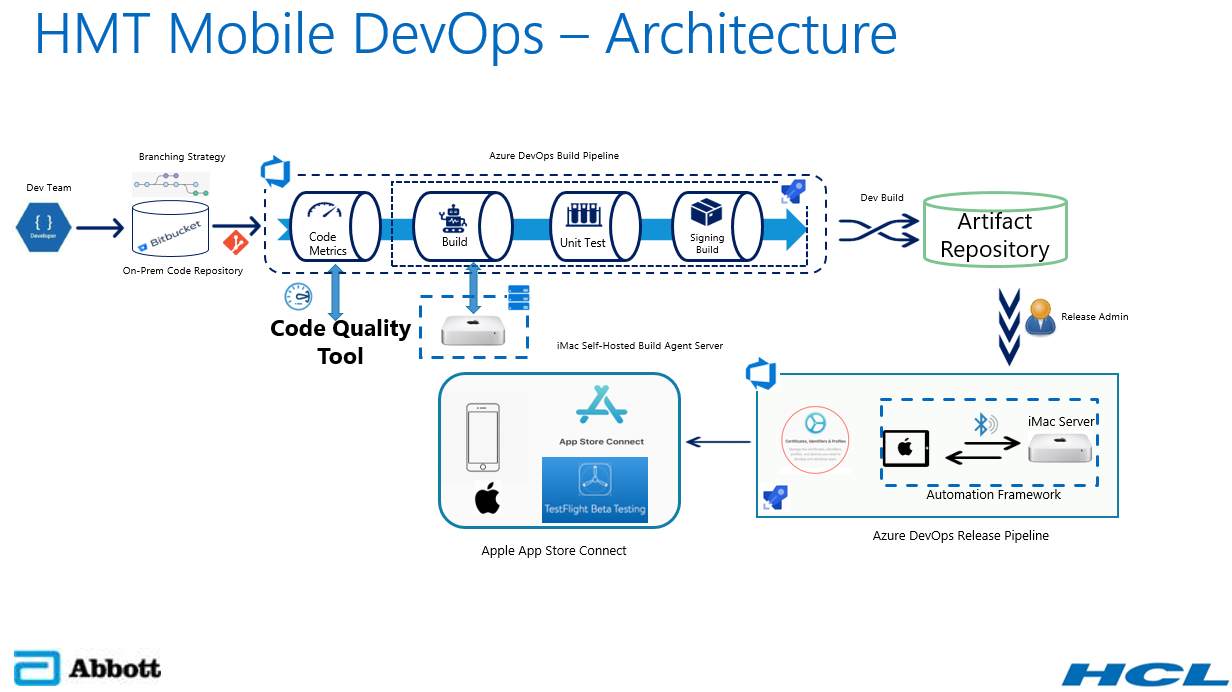
Undo the change by redeploying the previous release version. This options to simply overwrite current unsuccessful redeploy the previously successful release.

**Monitoring**

For monitoring CI/CD pipeline Azure Monitor will use which can help end-to-end visibility across the resources, drill down to the most probable root cause of a problem, even to actual lines of code, fix the issue in the app or infrastructure, and re-deploy in a matter of minutes. Azure Monitor should be able to find the errors and fix issues way before it starts impacting the customers.



**Technical Design Diagram**



Developer develops application with team and save your code in a Bitbucket repository. Each time developer adds a new feature and it will add the Unit Tests associated. Code will be part of a Pull request that one or two developers of team will validate. Release team will trigger the Pipeline manually for Build and Deployment process.

Start running Unit Tests to validate that everything is still working. Then apply an environment configuration, and build application on a Mac Self Hosted agent with Azure DevOps.

When build succeed we will move the artifact (IPA) to Artifact Repository. Once CI is success Continuous Deployment pipeline will start and it will distribute application to App Store Connect for Apple review. Once Apple review is success it will move to App Store.

**Technical Specifications**

Provide detailed technical specifications for required infrastructure as outlined below.

**Servers**

Under the heading list of a brief description and function for each server that is needed, and include the following information:

**Servers Requirement**

In this Azure Devops we required Mac Machine (Self-Hosted Agent) for Build and Release the IOS Application

Mac Machine Specification

* Mac Machine Version 10.15: "Catalina"
* RAM-
* Hard Disk-

**Artifact Repository**

Nexus or Jfrog:

Are the artifact repository which is essential for rapid releases and tighter security. Being able to manage binary artifacts makes it easier to identify and incorporate the correct versions of them.

**App Store Connect**

App Store Connect is a suite of web-based tools for managing apps sold on the App Store for iPhone, iPad, Mac, Apple Watch, Apple TV, and iMessage.

**Access Requirements**

* Need the Azure Devops access for Release Team to trigger the pipeline
* Need access for Bitbucket (ON-PREMISES)

**Prerequisites**

* The following Dependencies are required for IOS Application CI/CD.
* Azure DevOps account for Build and Release pipeline.
* Bitbucket (On-Premises) -Source Code Repository.
* Mac Dedicated Machine as a Self-Hosted Agent.
* Artifact Repository.
* Install Xcode on Mac Machine.
* Access to an Apple Account to get a Provisioning profile and p12 to sign your iOS application.
* App Store Connect Account.