Implementation of Azure-Qlik Integration POC

Objective:

The objective of this POC is to automate the deployments happening in Qlik Management Console through Azure Pipeline.

Roadmap of Azure-Qlik POC:

The plan of action has two major tasks involved:

Task-1: To test the automated deployment through Scripts from Local machine

Phase I:

- [x] a) To acquire the necessary permissions to authenticate with Qlik Server from Local machine
- [x] b) To customize the PowerShell/.NET modules according to our requirements
- [x] c) Run the script and test the automated (Export/Publish/Import) deployments
- [x] d) Authentication through Apigee - Client certs through Proxy

Phase II:

- [x] a) Grouping individual Applications under Application Groups. As Grouping enables, multiple qvfs are stored under App files folder in the Azure Repo. In order to publish multiple qvfs and with different names, below steps to be implemented:
- [x] i) To add foreach loop in .NET code to publish multiple qvfs at a single run
- [x] ii) To find a way to publish each qvf with different names in Qlik by enhancing the script
- [x] b) To work on Mashups and find a way to publish these in Qlik through Azure.
- [x] c) To know the technical details of file copies happening across the QA/PROD file servers in Qlik
- [x] i) Based on the inputs, to maintain the files in Sharepoint
- [x] Download of OV QA QFVs and Rename adding SS)
- [x] Commit SS QVFs to the Repo inside AppFiles sub-folder (rename Repo to OneView)
- [x] Commit OneView.ZIP Mashup to Mashups sub-folder
- [x] Rename Pipeline from POC to OneView and Run it
- [x] Ensure Files are Uploaded to Streams and Mashups (test again for overwrite)
- [x] Create a new QVF from scratch and add XCOPY to its Script (to be added to Repo as new SS QFV App in OneView) - call it Sasi-Reload - Put XCOPY test in script
- [x] Try Manually with Load Data

Task-2: To automate the Qlik manual deployments via Azure Pipeline

Phase I:

- [x] a) Create Azure Repo and Pipeline
- [x] b) Connect to Qlik through Azure Pipeline (Authentication)
- [x] i) An AD user TST-QLIKSS has been created and given access to Qlik Sense to test the authentication
- [x] c) Export .qvf from Qlik Dev to Azure Pipeline Workspace
- [x] d) Copy the .qvf file from Pipeline.Workspace to Azure Repo (Dev Branch)
- [x] e) Merge to the Azure QA Branch (Implemented in Dev for POC)
- [x] f) Publish/Import the .qvf file to the Qlik QA Branch (Implemented in Dev for POC)
- [x] g) Pull the .qvf file from Azure Repo and repeat the above steps through Azure Pipeline
- [x] h) Demo to the Qlik Team on the Phase I

Phase II:

- [x] a) To test and publish multiple qvfs with different names in the Qlik Sense through pipeline.
- [x] b) Qlik team deploys files in different formats to different environments in Qlik. These files are located in different file servers. To automate the copy files task through Pipeline below sub-tasks have to be completed:

- [x] c) Qlik team provided access to Dev Hub File Server **\\wa01238d\sense\data\Global\Central DataMart .** Created DevOps folder with sub-folders Dev, QA & PROD for Azure POC purposes
- [x] d) To modify the Architecture Diagram according to the enhancements we have made in Phase II
- [x] e) Integration of the Pipeline on "Load data" through the Program.cs

A Presentation on Azure - Qlik Integration POC (Click Here): Phase 1 presentation to Qlik Team

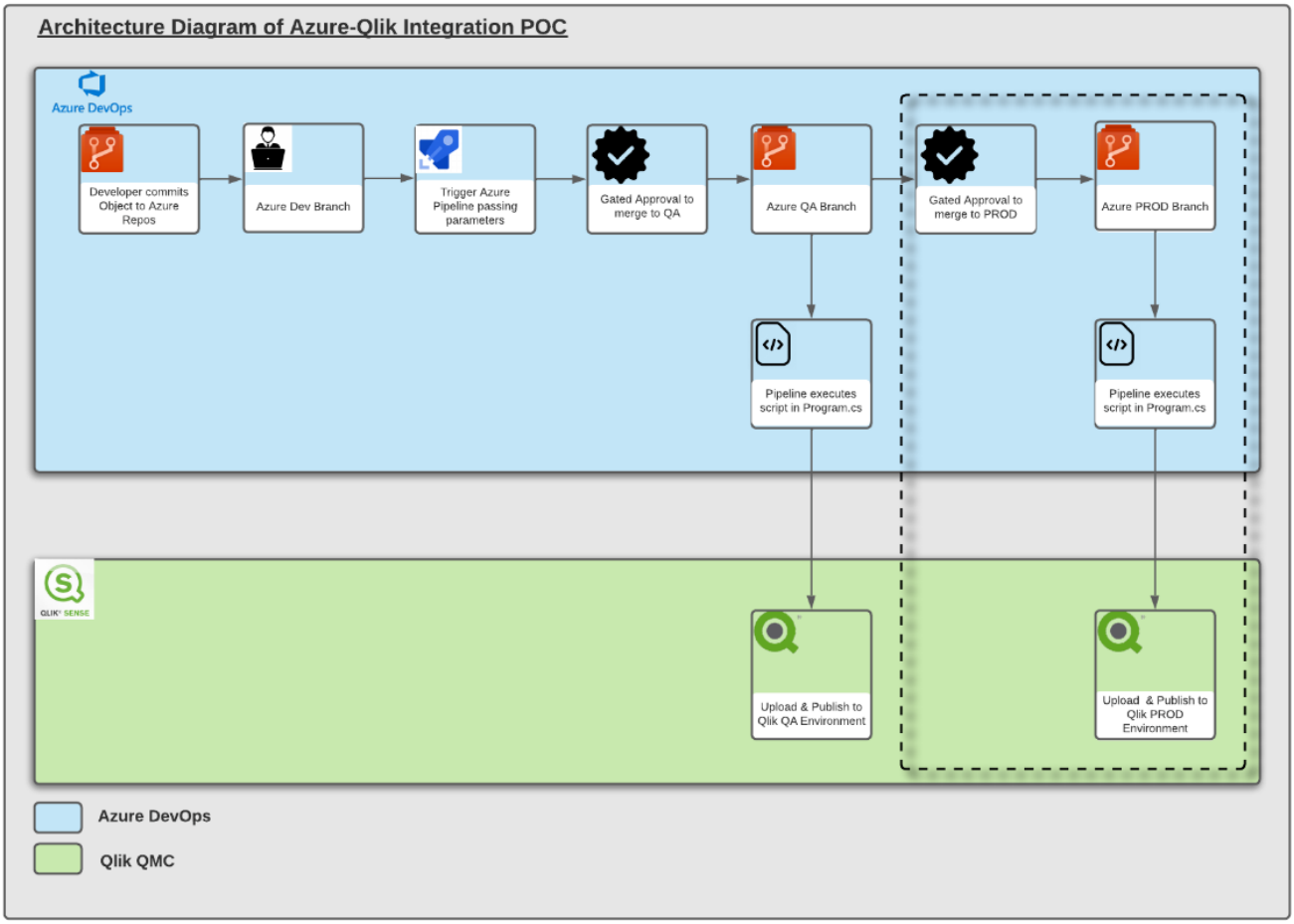
---

Below table is created as per the Improvements on the Roadmap: (Latest)

Review Call by Nicolas: 25th Jan 2022

1	Developer commits the object to Azure Repos (Dev Branch)	**Developer Commit --> Azure Dev Branch**
2	In order to have the selected commits merged to Azure QA Branch, Pull Requests have to be approved	**PR Approval --> Merge to Azure QA**
3	Once approved, object merges from Azure Dev Branch to Azure QA Branch	**Azure Dev --> AZ QA**
4	With the commit in Azure QA Branch, the Azure pipeline can be triggered passing the parameters like App ID, name, stream ID, etc.., to upload and publish the object to Qlik QA after successful authentication	**Azure QA --> Qlik QA**
5	In order to have the selected commits merged to Azure Prod Branch, Pull Requests have to be approved	**PR Approval --> Merge to Azure PROD**
6	Once approved, object merges from Azure QA Branch to Azure Prod Branch	**Azure QA --> Azure PROD**
7	With the commit in Azure Prod branch, the Azure pipeline can be triggered passing the parameters like App ID, name, stream ID, etc.., to upload and publish the object to Qlik Prod after successful authentication	**Azure PROD --> Qlik PROD**

> *The below diagram interprets the process of automated deployments:*



Initial Approach & Roadmap:

Environments: Dev - QA - Prod

MKS: Replaced with Azure Repo

Azure Pipeline (Dev to QA)

1. Pull from Dev Qlik Management Console the QVF File - API Script
2. Push QVF from Qlik to Azure Repo QA Branch
3. Push QVF from Azure Repo QA Branch to QA Qlik Management Console (QMC)

    a. POC Qlik AzureRepo with Dev and QA branches

    b. Commit manually a QVF file to Dev Branch (simulate Dev action)

    c. Script (API) and test a copy of QVF from Azure Repo to QA QMC

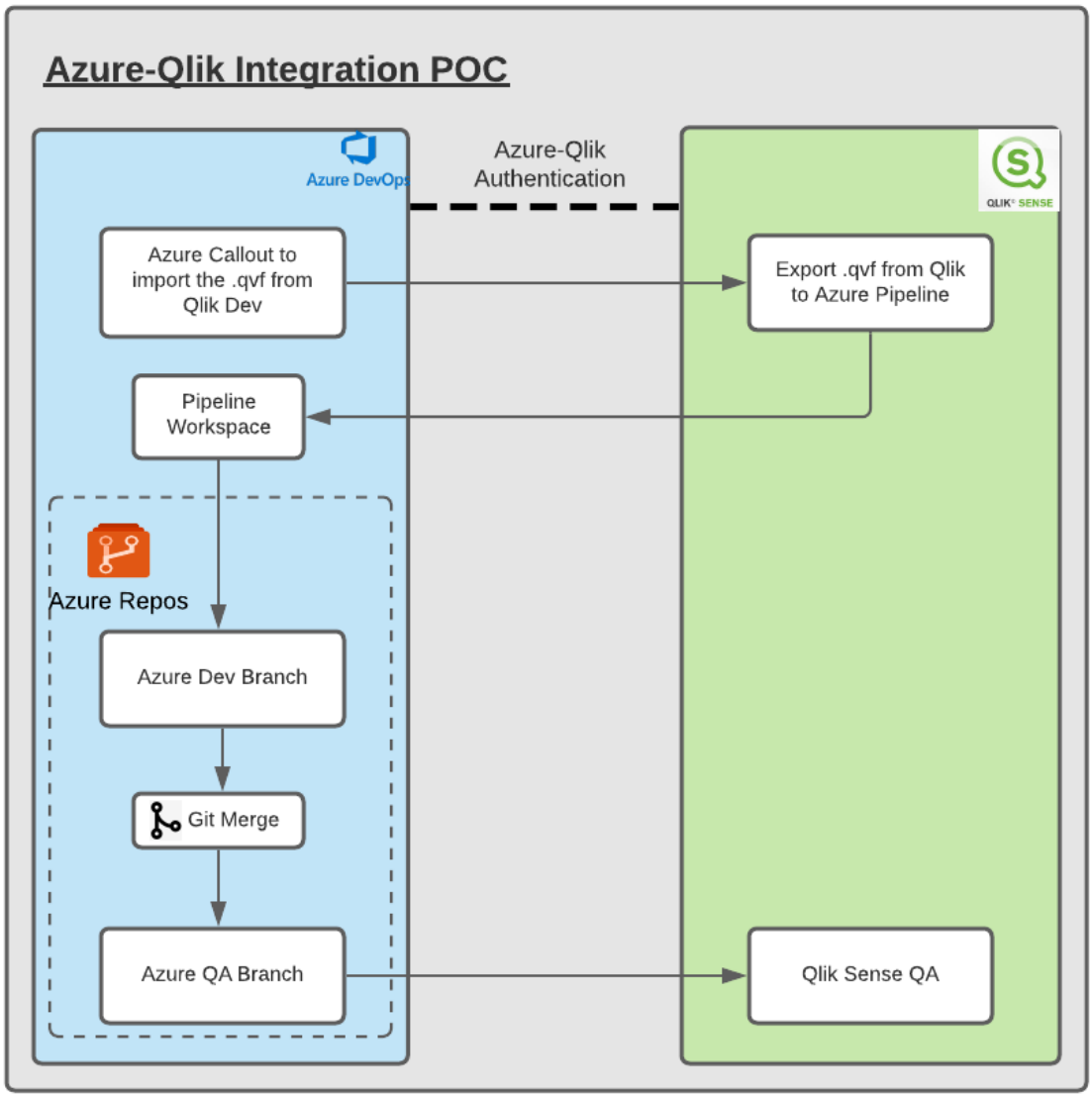
        i. Language the code the API calls? Python? Powershell?

        ii. Server to host the API script? (reachable by Azure Pipeline) Azure Agent

        iii. Basic API Script authenticating on QMC using Powershell via Azure Agent

    d. Have API script called by Azure Pipeline

> *The below diagram interprets the process of automated deployments:*


Azure-Qlik Integration POC
Azure DevOps
Azure-Qlik Authentication
QLIK SENSE
Azure Callout to import the .qvf from Qlik Dev
Export .qvf from Qlik to Azure Pipeline
Pipeline Workspace
Azure Repos
Azure Dev Branch
Git Merge
Azure QA Branch
Qlik Sense QA