ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Informatica Intellegent Cloud Services Code Deployment with Azure DevOps

Naveen Muppa

10494 Red Stone Dr, Collierville, Tennessee, USA

ABSTRACT

Continuous Integration/Continuous Delivery (CI/CD) has become a cornerstone of modern software development, enabling teams to deliver high-quality applications at speed and scale. Informatica Intelligent Cloud Services (IICS) provides a robust platform for data integration and management in the cloud. This abstract explores the implementation of CI/CD practices within IICS, focusing on streamlining the development, testing, and deployment of data integration.

***Corresponding author**

Naveen Muppa, 10494 Red Stone Dr, Collierville, Tennessee, USA.

Received: October 12, 2022; **Accepted:** October 16, 2022; **Published:** October 22, 2022

Keywords: Continuous Delivery and Deployment: Implementing CD pipelines to automate the deployment of data integration workflows from development through to production environments, reducing manual effort and minimizing the risk of errors

Introduction

Azure DevOps is a solution provided by Microsoft that includes Version Control, Reporting, Requirements Management, automated builds, testing, and release management features. The goal provided by the solution is to cover the full product lifecycle and provide Dev Ops practices to an organization. On the version control side, both Git and Team Foundation Version Control (TFVC) could be deployed.

Setup and Configure Azure DevOps with IICS

To utilize DevOps for IICS, a project should be set up and it should include a git-based repository as seen in the following screenshot.

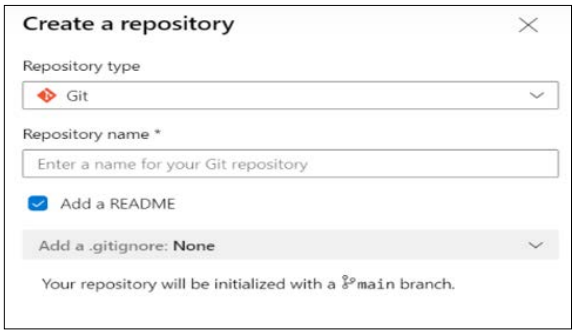

Figure 1: Repository

Additionally, after a repository has been created, setting up user accounts with the correct permissions on the repository is important. By default, DevOps has premade groups to help manage permissions. Permissions could be managed on a branch level of a repository if that level of control is needed. The menu could be found by navigating to the settings for the project and picking the Repositories option.

An example of this menu is as follows:

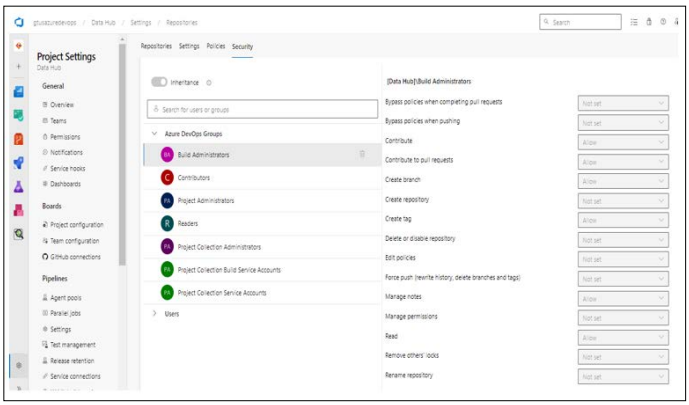

Figure 2: Repository

IICS Configuration and Setup

After the project and repository are set up, the setup in IICS is very similar to within GitHub as follows:

• In the Administrator Panel, navigate to settings and if you are licensed for version control, see the following:

Edit icon is on the top right side. One can see the option Allow Push to GIT. Enable this only in development org and make sure to disable this option from nondevelopment orgs.

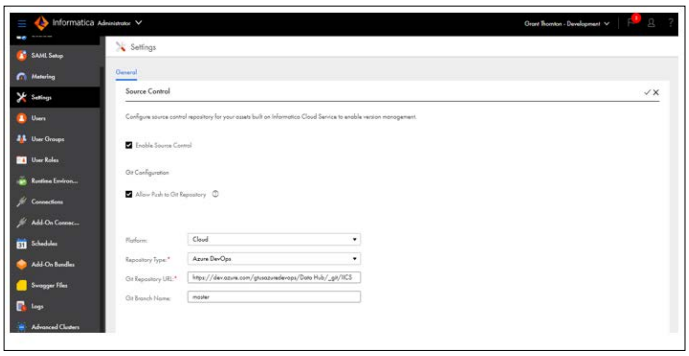

Figure 3: Setup

• Select Enable Source Control and Allow Push to Git Repository if you want to check-in and check-out.

Figure 4: Source Control

• If you want the users to use the OAuth feature over a generated token, make sure to select the Allow OAuth Access to Git option.

• Then, select your platform, copy the desired Git repository URL (navigate to your DevOps project and select the repository and use that URL), and the branch name.

Figure 5: Git

- Save the settings and then navigate to your user profile dropdown menu and select Settings.
- If you want to use the OAuth Access, select that option, and click the OAuth Authentication, and follow the popup to completion.
- If you are using the Personal Access Token, navigate back to the DevOps Project, and click the Clone option.

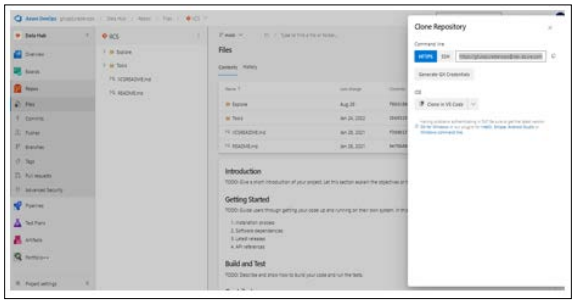

Figure 6: Clone

• Then, after getting the above menu, click Generate Git Credentials where you have a username and password listed. Copy the password given into IICS as your Personal Access Token.

Branching Strategies

Given the limitations of the Informatica CLI, we've decided to diverge from conventional branching strategies. Our approach involves managing two separate repositories:

1. IICS and 2. IICS Artifacts. Within the IICS repository, we maintain a sole Master branch, integrated into the Informatica Cloud development environment for streamlined check-in and check-out operations.

Note: The limitations, we can't store both source code and artifacts in the same repository. For that we introduced separate IICS Artifacts Repository, which houses Delta versions of our export code.

Interactions with IICS DevOps

After getting the IICS setup to utilize the DevOps repository, you would be able to check-in and check-out like you would with a GitHub source control setup. You would also see the usual ability to Pull from Git. From an IICS standpoint, it would work similarly to how it would within GitHub.

On the DevOps side, it would use the Informatica username for the commits and recreate the IICS folder structure. In the following image, you see the folder structure setup and a list of objects that make up the metadata of the objects in the IICS Repository.

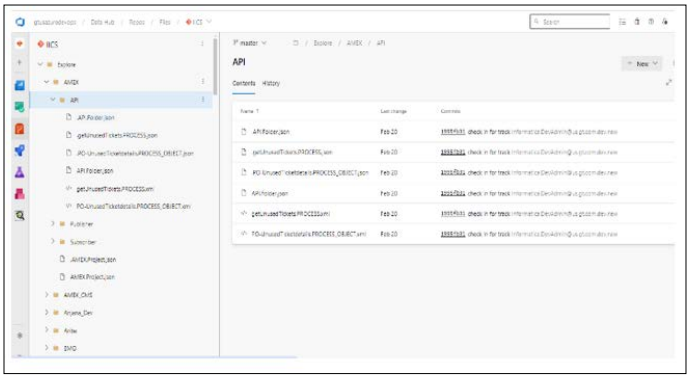

Figure 7: Metadata

CI/CD Overview

CI/CD, or continuous integration and continuous delivery, is a practice that automates the integration and delivery operations in a CI/CD pipeline. You can automate each integration and delivery operation using the Data Integration Service REST API or the infacmd command line programs.

A CI/CD pipeline includes the integration operations that developers use to design objects and the delivery operations that deliver the objects to the production environment. You can use the REST API or infacmd to automate integration and delivery in the following ways:

Deploy and test every change that a developer makes to an object. Developers receive instant feedback about whether objects pass or fail testing and the types of changes that objects require.

Deliver objects that pass testing to the production environment. Based on organizational requirements, you can deliver objects to

additional requirements, such as QA and UAT, before delivering the objects to the production environment.

Continuous integration and continuous delivery are fully logged and visible to the entire team so that team members can allocate time away from manual tasks.

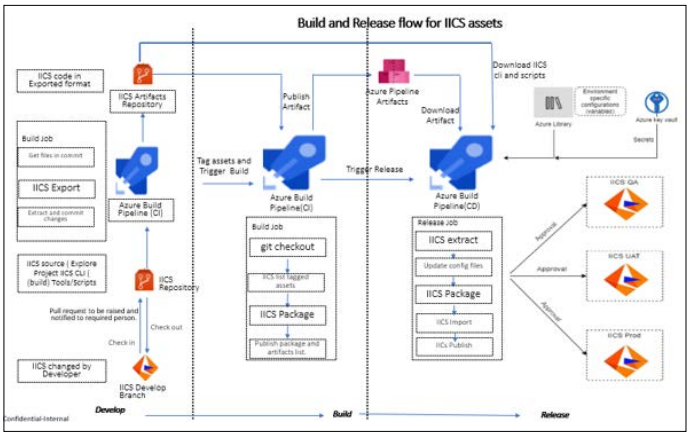

Figure 8: IICS Build Pipelines

Unlike any programming language, compilation of code is not required here.

Developer will commit the IICS code from Dev environment to Azure repo (Master branch) in IICS.
As we cannot use this committed code for packaging due to IICS CLI limitation, a build is setup to maintain the exported format of IICS code in another repo.

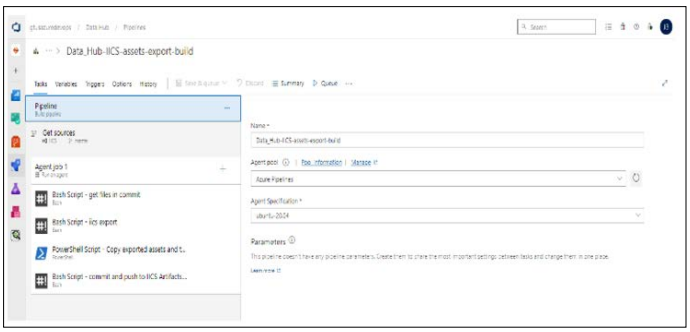

Figure 9: IICS Repo Structure

- This build will be triggered for each developer commit and it will perform below actions.
- Get the assets which are part of the commit.
- Export these assets from Dev org using IICS cli.
- Extract the exported zip file using IICS cli.
- Commit the extracted files into another repository (IICS Artifacts)
- Whenever a story or task is completed and the assets are ready to move to QA, couple of manual tasks needs to be done.
- Tag the assets as "promote_to_qa".

Release Pipilines

Once these dependent objects are in place in target environment, we can use release pipeline for deploying the assets. The release pipeline consists of the steps below.

- The build artifact is extracted using iics extract command of CLI.
- Bash script is used to update environment specific configurations for assets like service connectors and app connectors.
- The configuration variables are maintained in Azure Library which is linked to release stage.
- Secrets are downloaded from Azure key vault.
- Package the assets as zip using iics package command of CLI.
- Import the assets from zip created in target environment using iics import command of CLI
- Publish the assets in target environment using iics publish command of CLI in order i.e., the dependent assets like service connectors and app connections followed by processes and task flows
- Un-tag the assets in dev environment after successful deployment
- Also check-in the changes to 'qa' branch of IICS Artifacts repository.
- For any new configuration change, we must add the variables in Library and update the script in the release pipeline to include them during deployment. Need to escape special characters if they are present in variable values (like '/' with '\/') and also xml compatible characters for some special characters (like '$' to '$')
  - For example, variable value as [value="Admin&2&"] will correspond to [value=\"Admin\\&\\$2\\$\\&\"]
  - If the xml values are not proper, then import will fail

Conclusion

In conclusion, adopting CI/CD for Informatica Intelligent Cloud Services empowers organizations to accelerate the delivery of data integration solutions, improve collaboration among development teams, and enhance the overall efficiency and reliability of data pipelines. By following best practices and continuously refining CI/CD processes, organizations can maximize the value derived from their investment in IICS [1-5].

References

1. (2022) Progressive experimentation with feature flags - Azure DevOps. Microsoft Learn https://learn.microsoft.com/en-us/devops/operate/progressive-experimentation-feature-flags.
2. (2023) Set up staging environments - Azure App Service. Microsoft Learn https://learn.microsoft.com/en-us/azure/app-service/deploy-staging-slots?tabs=portal.
3. (2022) How Microsoft plans with DevOps - Azure DevOps. Microsoft Learn https://learn.microsoft.com/en-us/devops/plan/how-microsoft-plans-devops.
4. (2022) Automated Deployment of IICS Assets- CI/CD using Informatica API's. Informatica https://knowledge.informatica.com/s/article/Automated-Deployment-of-IICS-Assets-CI-CD-using-Informatica-API-s?language=en_US.
5. (2023) Developer Tool Guide. Informatica https://docs.informatica.com/data-quality-and-governance/informatica-data-quality/10-5-1/developer-tool-guide/continuous-integration-and-continuous-delivery--ci-cd-/continuous-integration/deploy-objects.html.