Microsoft

DevOps for Big Data Platform on Azure

Azure Data Services – DevOps for Big Data

Prepared by

Data SQL Ninja Engineering Team (askdmjfordmtools@microsoft.com)

Disclaimer

The High-Level Architecture, Migration Dispositions and guidelines in this document is developed in consultation and collaboration with Microsoft Corporation technical architects. Because Microsoft must respond to changing market conditions, this document should not be interpreted as an invitation to contract or a commitment on the part of Microsoft.

Microsoft has provided generic high-level guidance in this document with the understanding that MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION CONTAINED HEREIN.

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

Note: The detail provided in this document has been harvested as part of a customer engagement sponsored through the Azure Data Services Jumpstart Program.

Table of Contents

1 Overview

DevOps (an abbreviation of Development and Operations) is helping to decrease IT operational costs while improving software quality and accelerating time to market. In order to eliminate the operational boundaries that currently generate inefficiencies in terms of software quality and time to market for new features DevOps reorganizes whole software development process and its operational units. It includes the implementation of Continuous Delivery, Continuous Integration, Automated testing, Application monitoring and other recommended practices in Software Development and Operations.

In general, DevOps principals and benefits apply to all software, regardless of its nature. However, the relative maturity and impact of the DevOps recommended practices, and certainly the required tooling are different depending on the technology stack, so by adopting CI/CD process early in the development process organization will be well compensated by the gains in development cycle time and reduction in bug fixes while maintaining and maturing the CI/CD procedure.

In this whitepaper, we will be focusing on CI/CD pipeline for Big data. As Big data requires service that can orchestrate and operationalize processes to refine enormous stores of raw data into actionable business insights. Fundamentally, a CI/CD pipeline for a Big data project should include:

- Deployment of an entire cloud environment
- Management of related service identities and credentials
- Management of consistent set of interdependent pipelines including notebooks
- Perform integration and related tests

2 Continuous Integration (CI)

CI stands for *Continuous integration, that* is a set of coding practices that assist development teams to check in code to version control repositories frequently by implementing small code updates.

The technical object of CI is to build a consistent and automated way to maintain a life cycle that includes build, package, and test applications. With consistency in the integration process in place, teams are more likely to commit code changes more frequently, which leads to better collaboration and software quality.

3 Continuous Development (CD)

CD stands for *Continuous delivery,* that picks up where continuous integration concludes. CD ensures and automates the delivery of application code to selected environments. Ideally, this code has been built and tested successfully by the CI server as well.

CI/CD pipeline requires *continuous testing* as the objective is to deliver quality application to users. Continuous testing is often implemented as a set of automated regression, performance, and other tests that are executed inside the CI/CD pipeline.

To better illustrate a CI/CD pipeline, a fictional example involving collection of petabytes of game logs will be made up. One can think of CI/CD processes as similar to a traditional software development lifecycle.

4 Sample Project Overview

Imagine a gaming company that collects petabytes of game logs that are produced by games in the cloud. The company wants to analyze these logs to gain insights into customer preferences, demographics, and usage behavior. It also wants to identify up-sell and cross-sell opportunities, develop compelling new features, drive business growth, and provide a better experience to its customers.

To analyze these logs, the company needs to use reference data such as customer information, game information, and marketing campaign information that is in an on-premises data store. The company wants to utilize this data from the on-premises data store, combining it with additional log data that it has in a cloud data store.

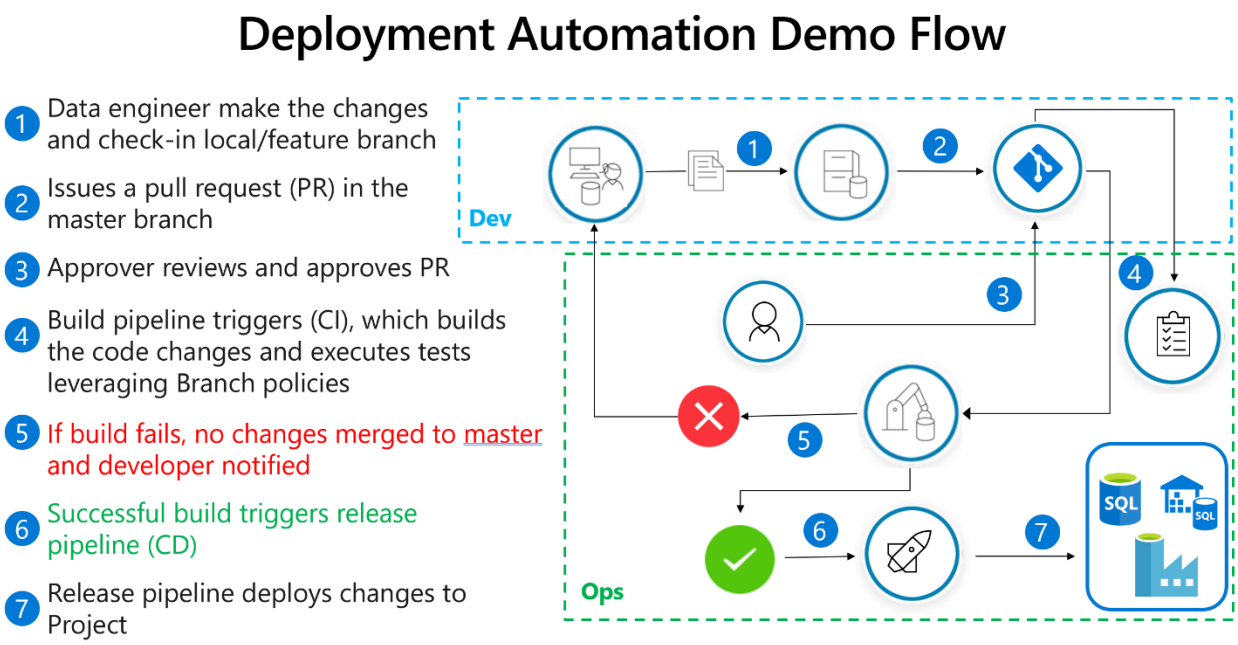
To extract insights, it hopes to process the joined data by using a Azure Databricks cluster in the cloud, and publish the transformed data into a cloud data warehouse such as Azure SQL Data Warehouse to easily build a report on top of it. They want to automate this workflow and monitor and manage it on a daily schedule. They also want to execute it when files land in a blob store container.

We will demonstrate the whole CI/CD pipeline process by architecting, developing and deploying this sample project in Azure cloud.

5 CI/CD Pipeline Toolchain

The pipeline is the fundamental concept of the DevOps toolchain, connecting tools, teams and processes with a visible and flexible collection of automated phases that allow consistent, secure and compliant promotion of software releases across various stages of development life cycle.

Below please find an illustration of the automatic deployment pipeline:

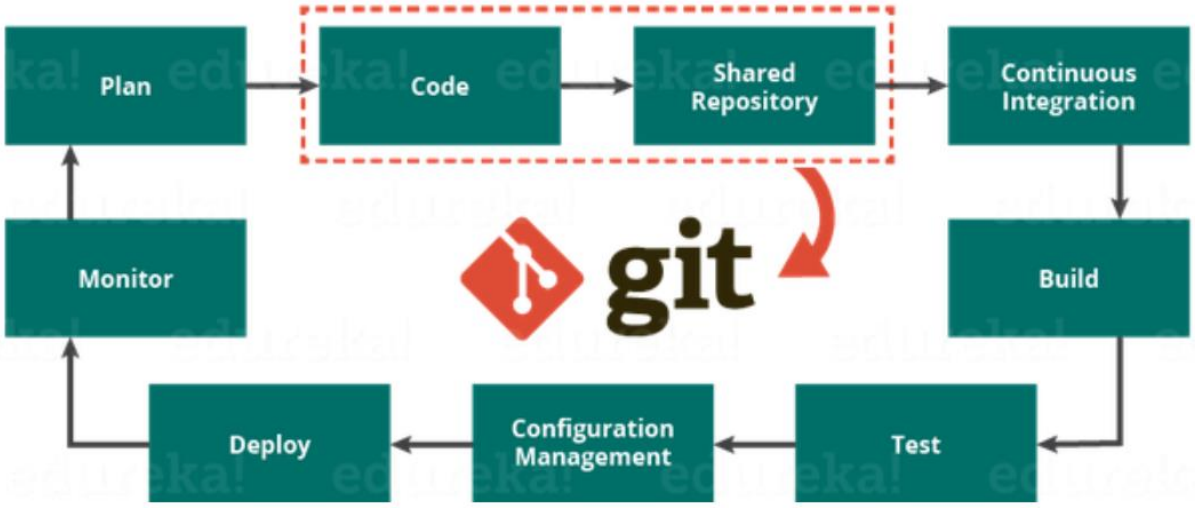


To start from scratch, and keep things easy the following choices would be considered to fully support our sample project:

6 Source Control

One of the fundamental requirements of DevOps is to have a common repository that holds latest code so that the development team can work all together and this truly helps to achieve quality product and it could be anything as per company standards. In our sample project, we will leverage GitHub as source control/repository for our CI/CD process. It will be used by developers to manage and maintain versions of their codebase for various projects and related set of files as they change over the time.

In DevOps, to implement CI/CD pipeline its must to have latest project changes available in hand. As DevOps gathers latest code and builds definition that performs multiple task as per user requirements and same is used by the release definitions that helps to deploy the latest codes on the production environment. It could be any client machine or any environment where final product gets ready for use. The downstream CI/CD processes will trigger based on pull requests and changes to the master branch.



The above diagram shows the entire life cycle of DevOps starting from planning the project to its deployment and monitoring. Git plays a vital role in succeeding at DevOps.

7 Azure Data Factory

Azure Data Factory is one of integral part of our solution. It is a *cloud-based data integration service that allows you to create data-driven workflows in the cloud for orchestrating and automating data movement and data transformation*.

In Azure Data Factory, continuous integration & delivery means moving Data Factory pipelines from one environment (development, test, production) to another. To do continuous integration & delivery, you can use Data Factory UX integration with Azure Resource Manager templates. The Data Factory UX can generate a Resource Manager template from the **ARM template** dropdown. When you select **Export ARM template**, the portal generates the Resource Manager template for the data factory and a configuration file that includes all your connections strings and other parameters. Then you've to create one configuration file for each environment (development, test, production). The main Resource Manager template file remains the same for all the environments. Read more

The pipelines (data-driven workflows) in Azure Data Factory typically perform the following four steps:

8 ARM Templates

When you are using Azure, you are consuming resources. For example, when you want to deploy one virtual machine, in fact you also need one storage account to store the virtual hard disk on it, one network interface to have an internal IP, etc.... It's the concept of Azure Resource Manager (ARM).

Azure Resource Manager is like a Lego box, and you can build your environments on it.

One of the concepts about DevOps is automation. When you're using Azure and want to automate your resources deployment, the best practices will be to use an ARM Template with all your Azure resources already configured: VM Size, DNS name, IP, etc.... To do that you need to create a JSON file.

Read more

9 Azure Databricks

Azure DevOps is a collection of services that provide an end-to-end solution for the five core practices of DevOps: planning and tracking, development, build and test, delivery, and monitoring and operations. Azure DevOps provides CI/CD automation for Azure Databricks.

Read more

9.1 Libraries and Notebooks

In our sample project, we have written a very simple Scala code that moves game log files from azure blob storage to data lake store. The library contains a unit test. As we follow DevOps practices, our first job before refactoring any code is to create a CI/CD pipeline so that we can control and measure any changes we make. There is also an integration test notebook that runs the other notebooks in sequence and verifies that they can be called in a pipeline.

The plan is to use notebooks for all the Databricks code, it is highly recommended for mature projects to manage code into libraries that follow object-oriented design and are fully unit tested. Frequently, developers will start by prototyping code in a notebook, then factor out the code into a library.

An efficient development environment for Databricks is therefore a combination of notebooks that are developed interactively, and libraries that are developed in a 'traditional' manner. Libraries might contain core business logic and algorithms, and some of that code may also run in other environments. Notebooks might contain integration logic, such as connectivity to upstream data lakes and downstream data marts.

10 Managed Identities

Azure Data Factory can conveniently store secrets into Azure Key Vault. In the sample project, Key Vault is used to store the Personal Access Token for Azure Databricks. As Azure Data Factory supports managed identities, granting access merely means creating an access policy in the ARM template.

Azure Databricks supports two different mechanisms for storing secrets. In the sample project, we create a Databricks-backed secret scope and grant full permission to all users. You can easily modify the provisioning script to restrict permissions, if you are using the Azure Databricks Premium tier). You can also use an Azure Key Vault-backed secret scope, but you will have to create it manually as the Databricks REST API does not currently allow automating it.

In the sample project, we use the Databricks secret scope to store the Storage Access Key for the ADLS Gen2 storage account where we will output the result of our transformed data to extract insights.

Read more

11 Test and Monitoring

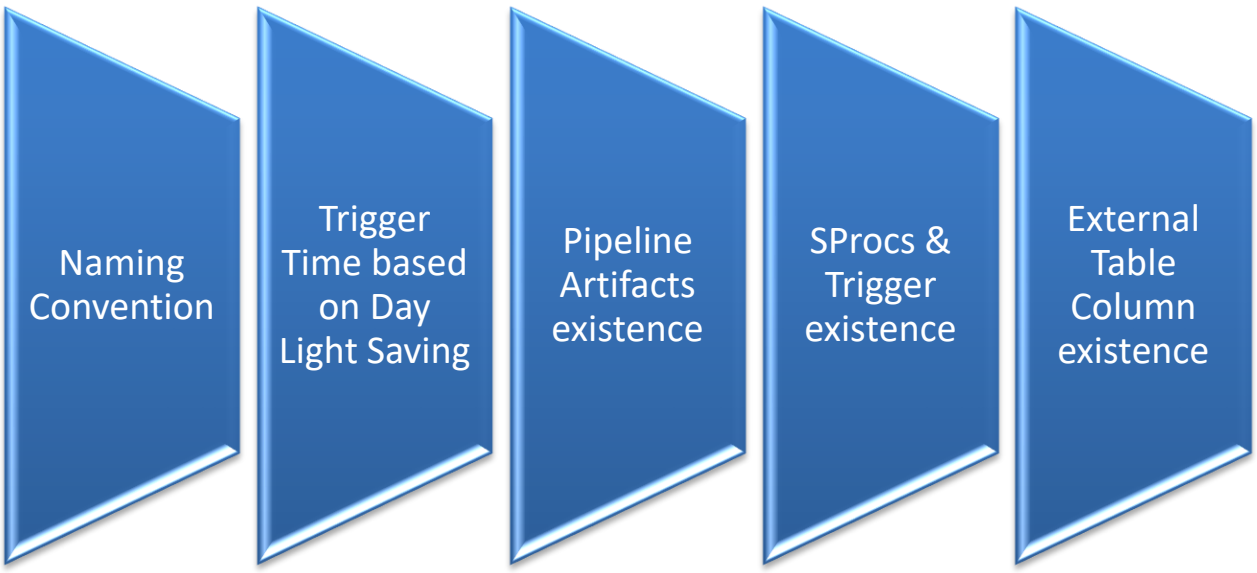
Testing is very much required to align their efforts in the DevOps cycle. Testing team needs to make sure that all their test cases are automated and achieve near 100% code coverage. They also need to make sure that their environments are standardized and the deployment on their testing boxes are automated. All pre-testing tasks, cleanups, post-testing tasks, etc. are automated and aligned with the CI cycle.

Testing team should also be able to identify problems in testing phase and report them proactively. To achieve this, they need to set up monitoring on the designated environment to be able to uncover bugs before they cause a failure. Setting up specialized counters like response times, memory & CPU utilization, etc. can provide a lot of insight into the end-user experience.

11.1 Unit Test

Unit testing is a testing process in which the lowest assessable parts of any software application, called units, are individually and independently tested for proper functionality.

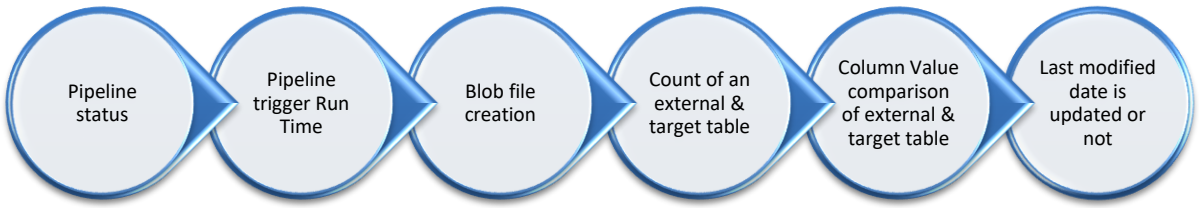
Below find an illustration of series of steps need to be tested in our project as part of unit testing.



11.2 Functional Test

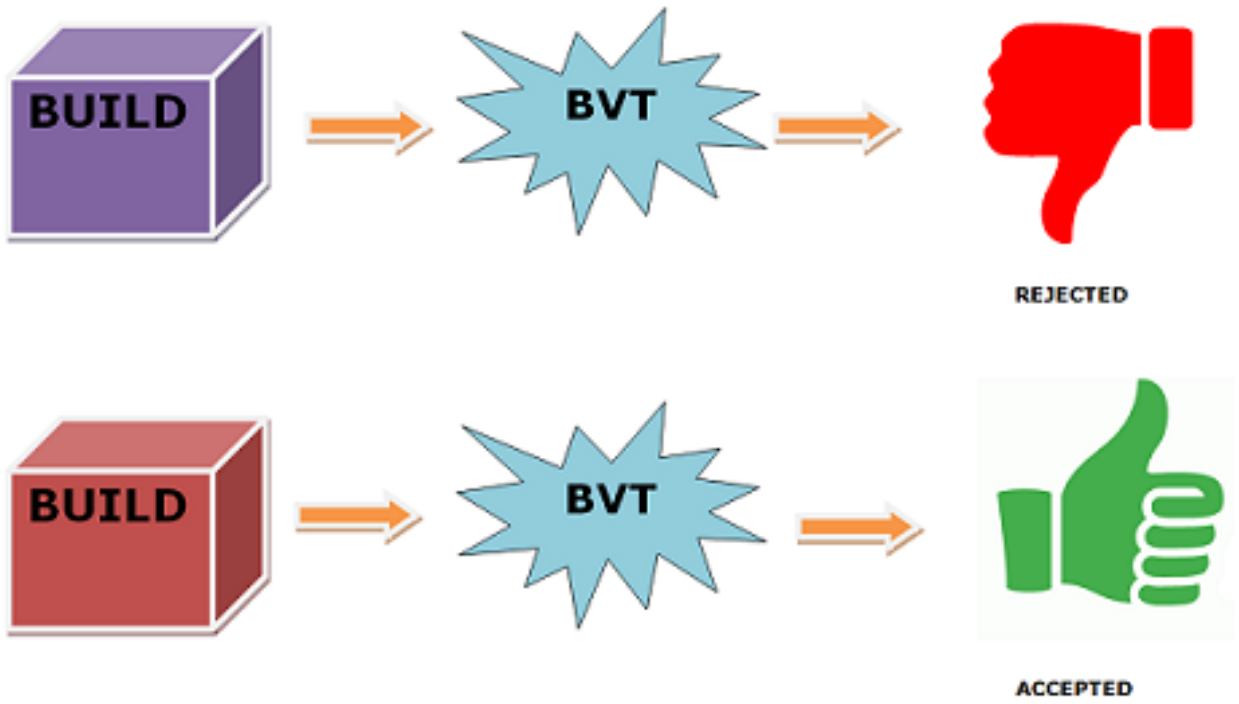
Functional Testing is another type of testing that validates each function of the software application operates in accordance with the requirement specification.

Below find an illustration of series of steps need to be tested in our project as part of functional testing.



11.3 Build Verification Test

Below find an illustration of build verification test.

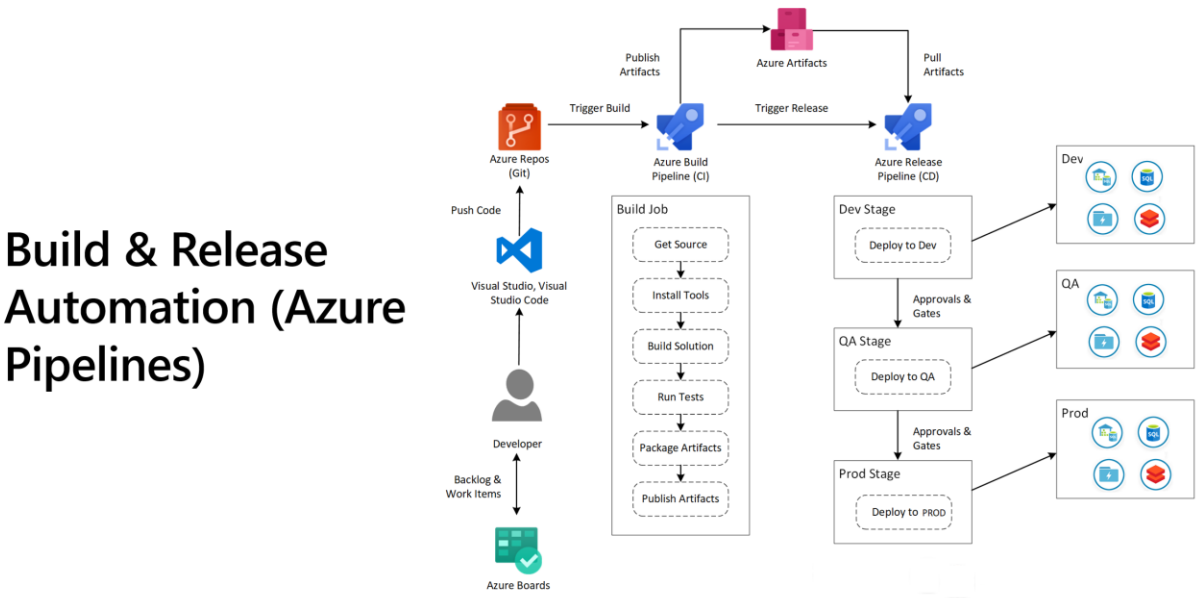


12 CI/CD Pipeline

Azure DevOps Projects presents a simplified experience where you can bring your existing code and Git repo or choose a sample application to create a continuous integration (CI) and continuous delivery (CD) pipeline to Azure.

Read more

Below find an illustration of build and release pipeline automation steps in Azure.



12.1 Build Pipeline

https://docs.microsoft.com/en-in/azure/devops/pipelines/index?view=azure-devops

12.2 Release Pipeline

https://docs.microsoft.com/en-in/azure/devops/pipelines/index?view=azure-devops

13 Conclusion

Azure DevOps helps to host and manage code centrally which is a key to optimization goal of organizations. It enables code versioning, code management and provides templates to model work around Agile Framework or the Capability Maturity Model Integration which makes it easier to manage. Azure DevOps provides a robust platform to create and manage pipelines for continuous integration and deployment.

14 References | Useful Links

https://docs.microsoft.com/en-in/azure/devops/pipelines/index?view=azure-devops

https://docs.microsoft.com/en-us/azure/devops/test/run-automated-tests-from-test-hub?view=azure-devops

https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-azure-devops/

https://docs.microsoft.com/en-us/azure/azure-resource-manager/vs-resource-groups-project-devops-pipelines

15 Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data SQL Ninja Team (askdmjfordmtools@microsoft.com). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the Azure Database Migration Guide.