Create CI/CD pipelines for AI apps using Azure Pipelines, Docker, and Kubernetes

Azure Pipelines Machine Learning Kubernetes Service

An Artificial Intelligence (AI) application is application code embedded with a pretrained machine learning (ML) model. There are always two streams of work for an AI application: Data scientists build the ML model, and app developers build the app and expose it to end users to consume. This article describes how to implement a continuous integration and continuous delivery (CI/CD) pipeline for an AI application that embeds the ML model into the app source code. The sample code and tutorial use a Python Flask web application, and fetch a pretrained model from a private Azure blob storage account. You could also use an AWS S3 storage account.

> **Note**
>
> The following process is one of several ways to do CI/CD. There are alternatives to this tooling and the prerequisites.

Source code, tutorial, and prerequisites

You can download source code and a detailed tutorial from GitHub. Follow the tutorial steps to implement a CI/CD pipeline for your own application.

To use the downloaded source code and tutorial, you need the following prerequisites:

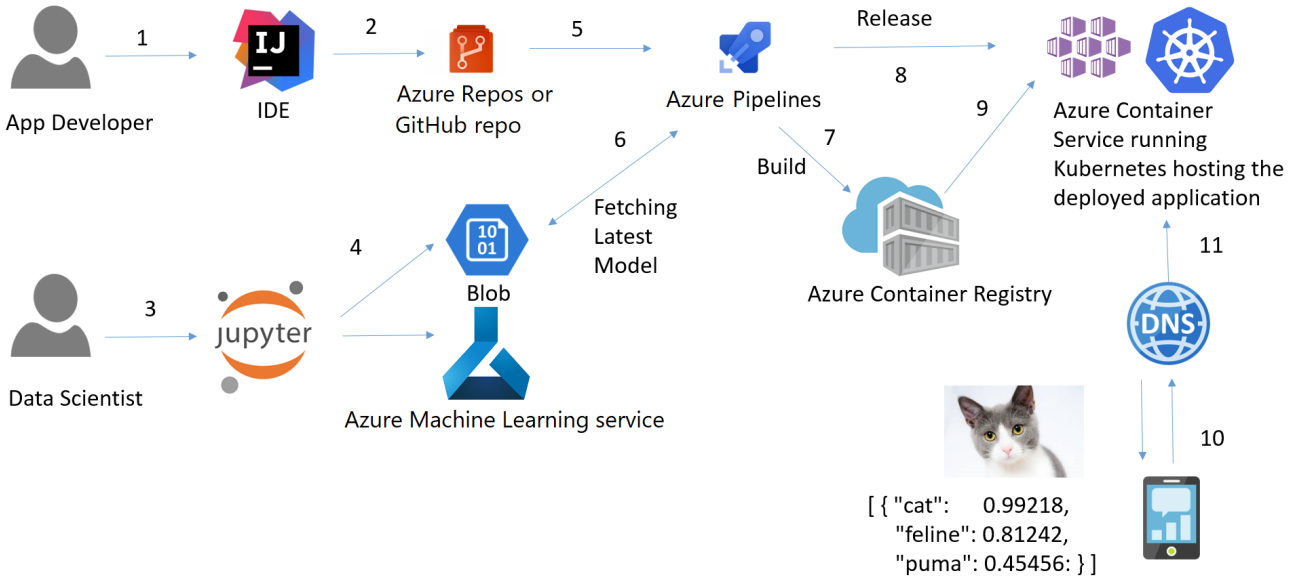
- The source code repository forked to your GitHub account
- An Azure DevOps Organization
- Azure CLI
- An Azure Container Service for Kubernetes (AKS) cluster
- Kubectl to run commands and fetch configuration from the AKS cluster
- An Azure Container Registry (ACR) account

CI/CD pipeline summary

Each new Git commit kicks off the Build pipeline. The build securely pulls the latest ML model from a blob storage account, and packages it with the app code in a single container. This decoupling of the application development and data science workstreams ensures that the production app is always running the latest code with the latest ML model. If the app passes testing, the pipeline securely stores the build image in a Docker container in ACR. The release pipeline then deploys the container using AKS.

CI/CD pipeline steps

The following diagram and steps describe the CI/CD pipeline architecture:



1. Developers work on the application code in the IDE of their choice.
2. The developers commit the code to Azure Repos, GitHub, or other Git source control provider.
3. Separately, data scientists work on developing their ML model.
4. The data scientists publish the finished model to a model repository, in this case a blob storage account.
5. Azure Pipelines kicks off a build based on the Git commit.
6. The Build pipeline pulls the latest ML model from blob storage and creates a container.
7. The pipeline pushes the build image to the private image repository in ACR.
8. The Release pipeline kicks off based on the successful build.

9. The pipeline pulls the latest image from ACR and deploys it across the Kubernetes cluster on AKS.
10. User requests for the app go through the DNS server.
11. The DNS server passes the requests to a load balancer, and sends responses back to the users.

See also

- Team Data Science Process (TDSP)
- Azure Machine Learning (AML)
- Azure DevOps
- Azure Kubernetes Services (AKS)