Continuous Integration and Continuous Deployment

Introduction:

Continuous Integration and Continuous Deployment (CI/CD) is an approach that allows developers to automate the software delivery process. This approach involves integrating the code changes frequently and testing the application to ensure that it's functioning correctly. Continuous Deployment is an extension of CI, which automates the deployment process of code changes to production environments.

Azure is a cloud computing service that offers various tools and services for building, testing, deploying, and managing applications. In this document, we'll cover the steps for implementing CI/CD with Azure.

Steps for Implementing CI/CD with Azure:

1. Create a new Azure DevOps project:

Azure DevOps is a platform that provides tools for planning, developing, testing, and deploying applications. To get started, create a new Azure DevOps project. Follow these steps:

- Go to the Azure DevOps portal.
- Click on "Create Project."
- Enter a name for your project and select the appropriate organization.
- Choose a version control system (Git, TFVC).
- Click on "Create."

2. Set up Continuous Integration:

Continuous Integration (CI) involves automating the process of building and testing code changes. In Azure DevOps, you can use Azure Pipelines for CI. Follow these steps:

- Go to your Azure DevOps project.
- Click on "Pipelines" and select "Create Pipeline."
- Select your code repository and choose the appropriate pipeline type.
- Select your build agent and click on "Continue."
- Configure the pipeline settings, such as trigger conditions, build stages, and tasks.
- Save the pipeline configuration.

3. Set up Continuous Deployment:

Continuous Deployment (CD) involves automating the process of deploying code changes to production environments. In Azure DevOps, you can use Azure Release Pipelines for CD. Follow these steps:

- Go to your Azure DevOps project.
- Click on "Releases" and select "Create Release Pipeline."

- Select your build pipeline as the artifact source.
- Configure the release pipeline settings, such as stages, deployment triggers, and tasks.
- Save the release pipeline configuration.

4. Configure Deployment Environments:

To deploy code changes to production environments, you need to configure the deployment environments. In Azure DevOps, you can use Azure Environments for this purpose. Follow these steps:

- Go to your Azure DevOps project.
- Click on "Environments" and select "New Environment."
- Enter a name for your environment and select the appropriate resource group.
- Choose the appropriate deployment target (Azure Kubernetes Service, Virtual Machines, etc.).
- Configure the environment settings, such as variables, security, and approvals.
- Save the environment configuration.

5. Test and Deploy:

After configuring the CI/CD pipeline and deployment environment, you can test and deploy your application. Follow these steps:

- Make changes to your code and push them to the code repository.
- The Azure Pipelines will automatically trigger the build process and create a new build artifact.
- The Azure Release Pipelines will automatically trigger the deployment process and deploy the code changes to the production environment.
- Monitor the deployment process and verify that the application is functioning correctly.