Amadis

WHITE PAPER

Building a CI/CD pipeline with Azure Devops:

Introduction to Devops:

Integrating development and operations environment to be similar to each other for seamless flow of the overall development procedure is called DevOps. It is a methodology; there was **Waterfall Methodology** (Figure 1) before where the various stages involved in a software development lifecycle followed a linear flow. Complete development of the code first followed by code review, validation, unit testing, user acceptance testing, staging, quality assurance and finally deployment to production. Here the biggest problem was if there is a problem in any of one stage particularly in the final ones we have to start the process all over again, which was very time consuming and resource intensive.

Later **Agile Methodology** (Figure 1) was developed where rather than following a linear workflow the next the step will not be started until there is a final output from the previous step, the whole process was divided in to smaller chunks each capable of carrying out its own functionality as and when required. The overall workflow was broadly classified in to four stages:

- **Version control**
- **Continuous integration**
- **Continuous delivery**
- **Continuous deployment**

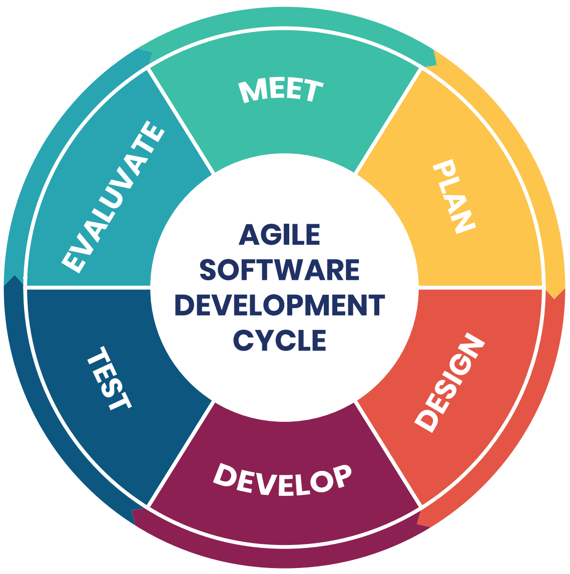


Figure 1

The first and foremost step in any software development lifecycle is of course writing the source code. Most of the time this source code requires contribution of multiple members of a team who might also be in different geographic locations. To efficiently coordinate the inputs from various teams involved in the development and to keep track of the changes being made to the code version control systems like GitHub, Subversion were developed. For example, if you use GitHub as your version control system it stores the source code and acts as a central repository from where all the developers can pull the latest version of the code or make a commit to it. It keeps track of your code and changes made to it, so in case you see errors after a particular commit you can revert to the previous version.

The second step of the DevOps process is whenever a commit is made to the source code and it is altered, new code is automatically pulled and built. That is unit testing, code review, code validation are done on the code. This process is called Continuous Integration (CI).

Once the code is built, it is then sent to the test servers to perform user acceptance testing through the process of Continuous Delivery (CD).

Finally, after the code is fully validated it is released on to the production servers through the process of Continuous Deployment.

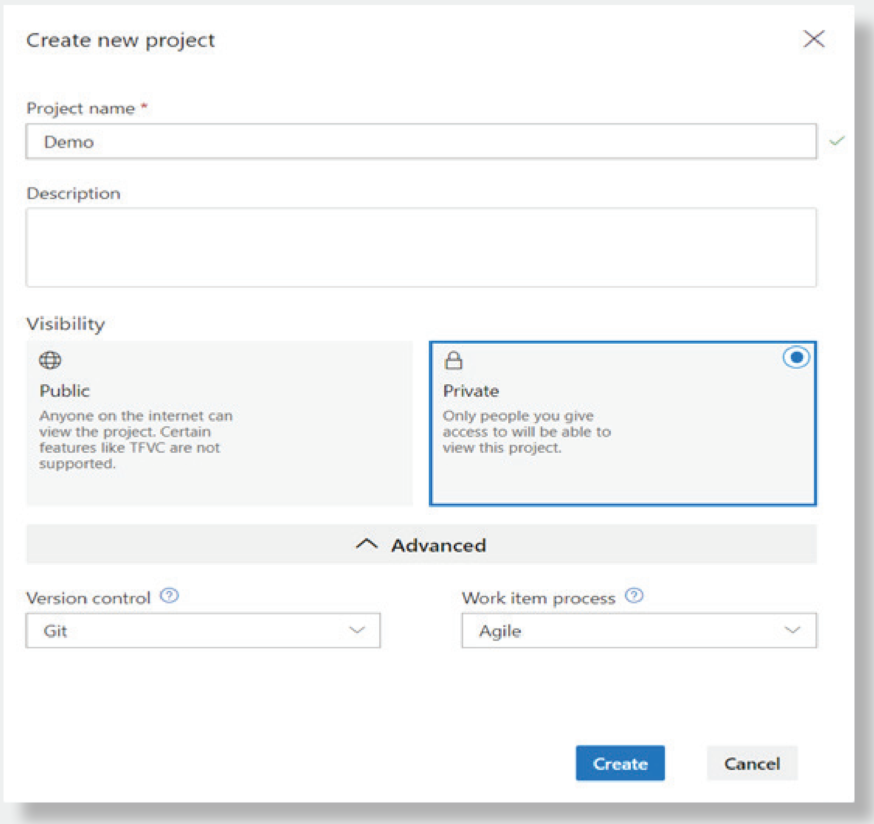
As you can see, from the above explanation that, continuous integration and continuous delivery (CI/CD) form the backbone of DevOps. Things start to get interesting when you combine these practices with programmable infrastructure and a suite of services that allow you to automate the entire lifecycle of an application. In this white paper, we try to give you a practical example of what that all looks like when you are building, testing, and deploying applications with Azure DevOps Services. This will give you an overall idea of the end-to-end process of building a fully automated build and release pipeline for a Node and Express application. We will use Azure DevOps Services to create the CI/CD pipeline and Azure App Service for deploying to development/staging and production.

Create Your Azure DevOps Organization

The first step is to navigate to dev.azure.com and sign in to Azure DevOps. If you have never done this before, you will need to create a new organization. You need to have at least one organization, which is used to store your projects, structure your repositories, set up your teams, and manage access to data. You can start by creating a single organization, for more advanced scenarios, you can refer to plan your organization structure in Microsoft's documentation.

Creating a Build Pipeline:

Once you have an organization created, the next step is to create a project. When you click on the Create Project button in the Azure DevOps portal you will be taken to a page (Figure 2) where you can select the level of visibility you want to have in your project (Public or Private). When you click on the advanced settings, you can also select the version control system that you want to use on your project like GitHub or Team foundation server, etc.



Create new project

Project name * Demo

Description

Visibility

Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

[✓ Private
Only people you give access to will be able to view this project.]

Advanced

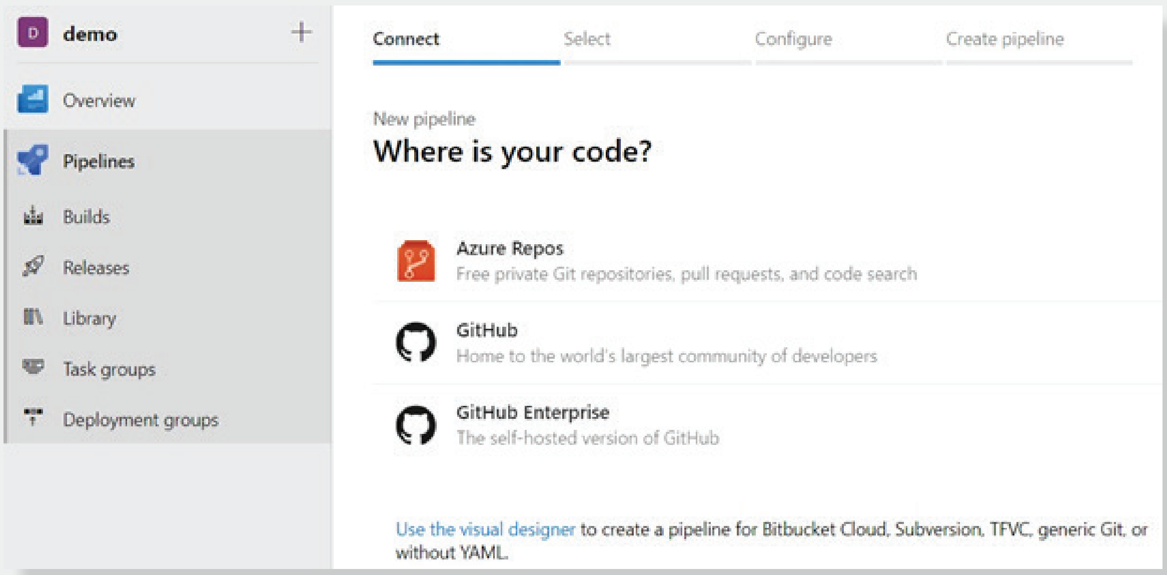
Version control: Git

Work item process: Agile

Create Cancel

Figure 2

Once you have created a projected, you will have to create a build pipeline. Once you click on the pipelines button in your project dashboard you will be given option to choose the pipeline you want to create and you can choose the build pipeline. You will be taken to a similar window (Figure 3) upon selecting the option:



demo

Overview
Pipelines
Builds
Releases
Library
Task groups
Deployment groups

Connect Select Configure Create pipeline

New pipeline

Where is your code?

Azure Repos
Free private Git repositories, pull requests, and code search

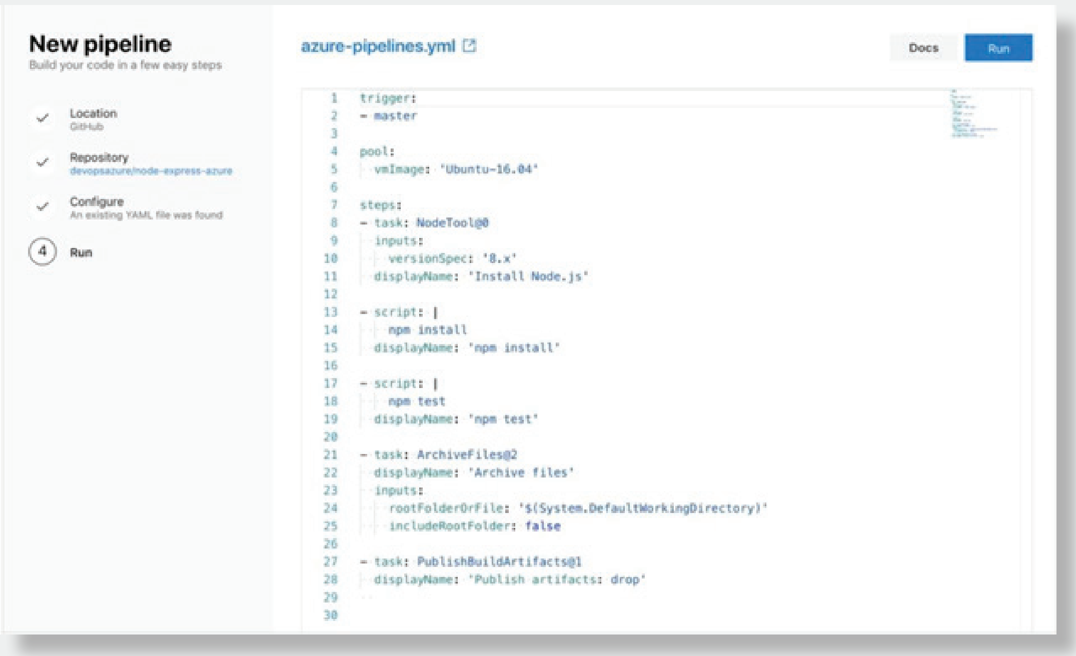
GitHub
Home to the world's largest community of developers

GitHub Enterprise
The self-hosted version of GitHub

Use the visual designer to create a pipeline for Bitbucket Cloud, Subversion, TFVC, generic Git, or without YAML.

Figure 3

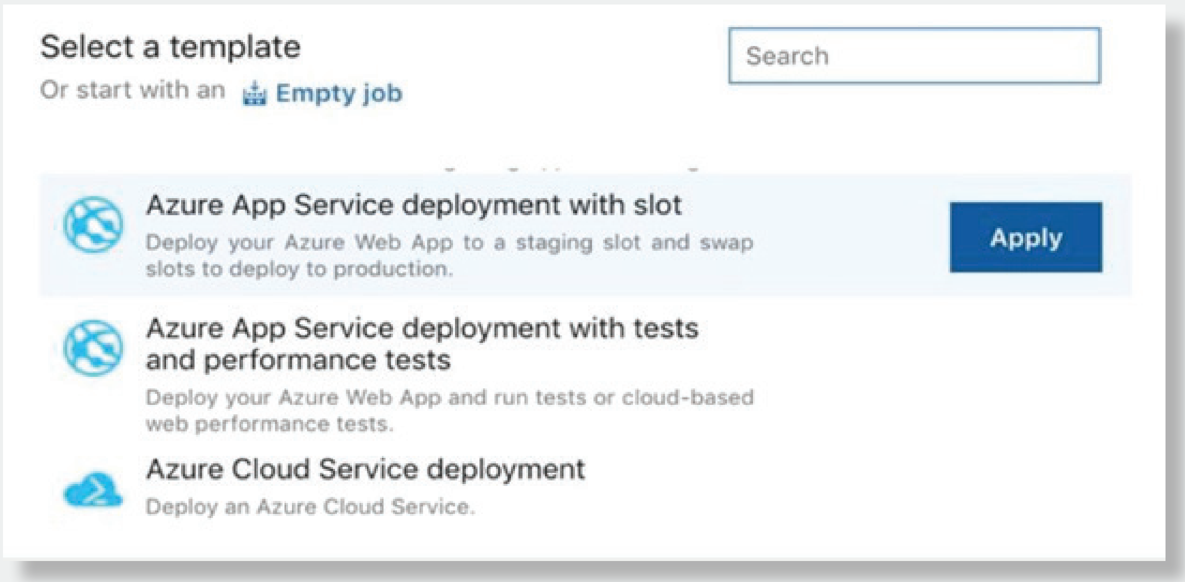
Here you can choose the source repository that you want to use for your project. This can be from external sources like GitHub or your own local repository. For example, if you are choosing GitHub you will need to authorize the Azure DevOps service to connect to your GitHub account on your behalf. Once you have associated a source repo with your project and connected to it you will be shown a screen like the one in Figure 4. Based on the template you choose for creating the pipeline (This depends on the type of application you are building) a YAML file is created which has all the instructions to pull the source code from your repository, testing and validating it. Once you click on Save and Run, your build pipeline would be created.


New pipeline
Build your code in a few easy steps
Location
GitHub
Repository
devopsazure/node-express-azure
Configure
An existing YAML file was found
4 Run
azure-pipelines.yml
Docs
Run

```yaml
trigger:
- master

pool:
  vmImage: 'Ubuntu-16.04'

steps:
- task: NodeTool@0
  inputs:
    versionSpec: '8.x'
  displayName: 'Install Node.js'

- script: |
    npm install
  displayName: 'npm install'

- script: |
    npm test
  displayName: 'npm test'

- task: ArchiveFiles@2
  displayName: 'Archive files'
  inputs:
    rootFolderOrFile: '$(System.DefaultWorkingDirectory)'
    includeRootFolder: false

- task: PublishBuildArtifacts@1
  displayName: 'Publish artifacts: drop'
```

Figure 4

Creating a Release pipeline:



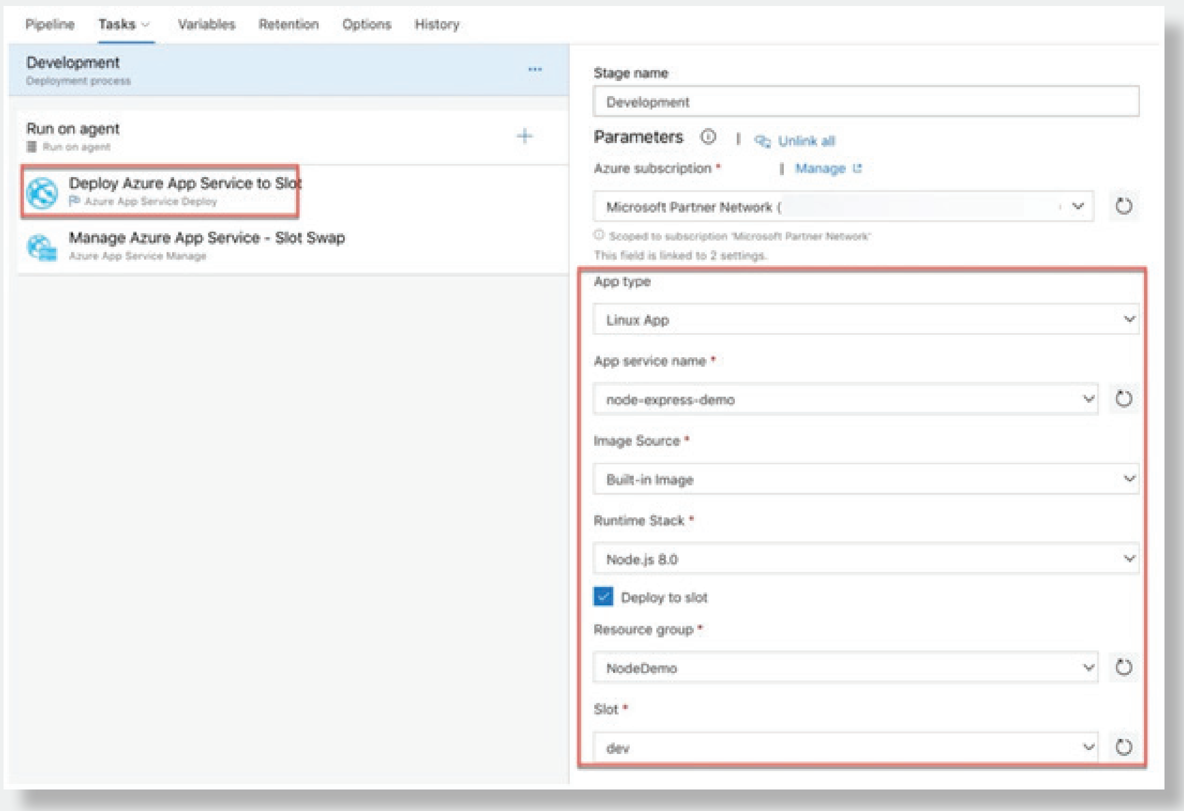
Azure App Service deployment with slot
Deploy your Azure Web App to a staging slot and swap slots to deploy to production.

Azure App Service deployment with tests and performance tests
Deploy your Azure Web App and run tests or cloud-based web performance tests.

Azure Cloud Service deployment
Deploy an Azure Cloud Service.

Figure 5

Before creating a release pipeline, we first need to have an empty shell or a resource to deploy the application. Let us consider we are developing a web application; we can make use of the Azure App services to power our web application. We can set things up so our CI/CD pipeline can build and deploy the app into a development/ staging slot. Then we can also choose if we want to have a manual approval before moving the app into the production slot. We can create these resources directly from either the Azure portal or using ARM template and power shell. Once the required dependencies are created in Azure environment, go to the Azure DevOps portal again and click on the release pipeline. You will be asked to choose a template, for our example we need a web application deployed with separate slots for development and production so we are going to choose Azure App service deployment with slot (see Figure 6). Once you click on Apply, a new deployment stage would be created with in the release pipeline. You can name this stage as development in accordance with our example and can configure the tasks that need to be performed in this deployment stage. You can also choose which slot the tasks would be deployed to (We would be choosing dev for this example, Figure 6)



Stage name: Development

Azure subscription *: Microsoft Partner Network (

Scoped to subscription 'Microsoft Partner Network'

This field is linked to 2 settings.

App type: Linux App

App service name *: node-express-demo

Image Source *: Built-in Image

Runtime Stack *: Node.js 8.0

[✓] Deploy to slot

Resource group *: NodeDemo

Slot *: dev

Figure 6

Once you have configured the task for this deployment stage, you can click Save. If you go to the release pipeline again and check the deployment triggers, you will see that continuous deployment is enabled by default. This means that going forward, each new build will trigger a deployment to our development slot in Azure App Service. Click the "Release" button on the top right of the release pipeline screen and create a new release. Click on the "Create" button to deploy the application to the development deployment slot. You should see a successful status in the properties of the release. Now that you have successfully deployed your web app to the development slot and verified it, you can consider moving it to the production slot. For doing that Head back over to the Azure DevOps portal and go to **Pipelines** > **Releases**. Click on the "Edit" button to modify the pipeline. Highlight the **Development** stage and click the dropdown to clone the stage (Figure 7).

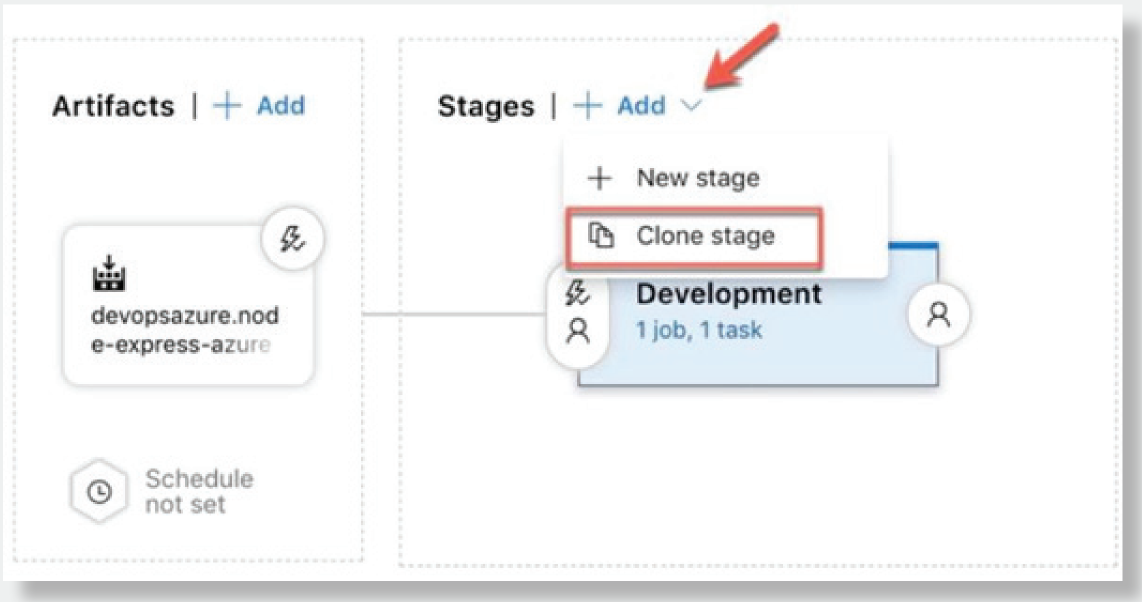


Figure 7

Rename the stage to production. Next, click the pre-deployment conditions button for the **Production** stage. Enable pre-deployment approvals and add yourself as an approver. We are doing this because we do not want automated deployments going straight into production. We are not building a continuous deployment pipeline for production. We are building continuous delivery pipeline.

Continuous delivery is a process that ensures our application is production ready. When we are doing a scheduled deployment, we can do so with confidence since we know the application has been through a pipeline of tests beforehand.

Next, click on the "task" link on the **Production** stage. We need to modify this task so that it does not deploy our code into the development slot. Simply uncheck "slot" and this will infer that the production slot of the web app should be used during the deployment. Click Save when complete. Now you have successfully created CI/CD pipeline for your application using Azure DevOps services.

Amadis

Guhan Palanisamy
Cloud Solutions Engineer, Amadis Technologies
Phone: 862-300-9516
Email: guhanp@amadisglobal.com

100 Overlook Center, 2nd Floor, Princeton, NJ 08540
Phone: +1 609 375 2755
Email: info@amadisglobal.com