

# AZURE DEVOPS FOR .NET APPLICATION

## AUTOMATING ALL THE STAGES OF SDLC LIFECYCLE USING DEVOPS FOR SAAS .NET CLIENT

---



### PROBLEM STATEMENT

Since there is an on going, escalating demand for better software and unbroken service. So the issue with traditional software development that it takes so long for adoption by the time new features are delivered, end users needs have entirely changed. It is challenging to offer services as a service provider, and it appears to harm business growth. Hence to minimize the human involvement and to achieve the automated process, we are seeking for a solution that could be utilized repeatedly. As we work with a team it is important to manage, track work, issues and code defects associated with project.

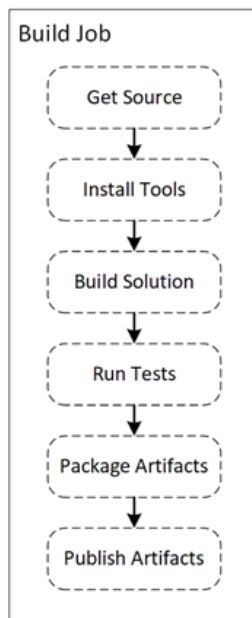
## SOLUTION

As per the above problem statement, the best solution we thought of was Azure DevOps. Basically, Azure DevOps includes many services provided under one umbrella as it is offered by Azure. It exceeds the expectations and addresses the majority of our requirements to connect with SaaS clients. Using Azure DevOps automation pipelines, the wait time for software development was reduced significantly. SaaS clients were able to shorten the time by using automation of typical constraints in older applications. As the requirements coming for us regarding .NET applications, it is very convenient for us to use Azure DevOps umbrella service because it integrates very well.

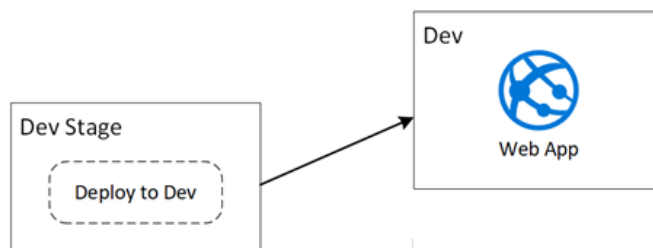
We are implementing Azure Pipeline to automate certain task-related compilation and deployment so that users can build, compile, and deploy codes on other computing platforms. This pipeline's objective is to eliminate manual software development processes; all modifications are carried out automatically in the project.

When a process is handled by humans, there are always some chances for human error as they perform the same tedious, repetitive duties, but with automation, it works without a hitch once it is configured. We have the two-stage option to separate pipelines: build and release. Both stages are distinct from one another, and the build stage covers the Design, Build, Document, Test, while deployment is covered in Release Stages.

### BUILD PIPELINE



## RELEASE PIPELINE



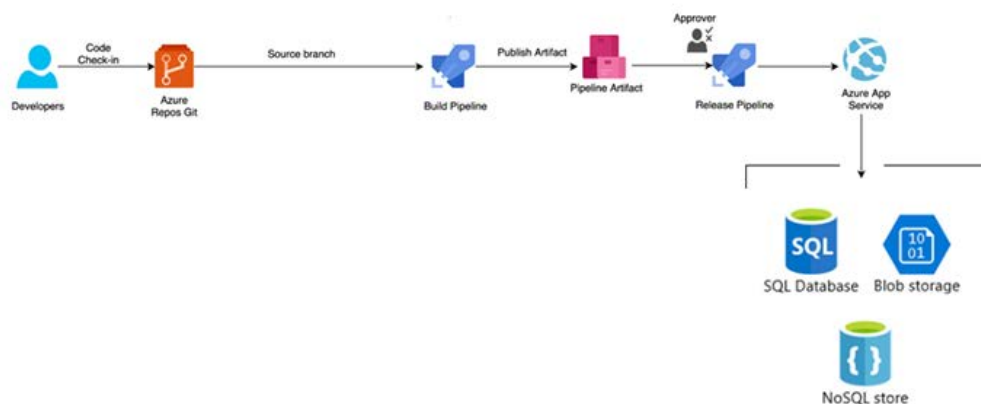
## MANAGING CODE

However managing code was also a challenge for our client, so we used Azure Repository's collection of version control tools which allowed them to keep track of the changes made to their code over time on the same portal. In addition, azure boards was also used to track plans and manage team discussions.

## DEPLOYING CODE

We used the Azure App Service as a platform to host our code in order to put our application into a production environment. Due to Azure's management of all infrastructure-related to app

services, end users were able to easily maintain their applications without worrying about it. However while connecting with app service we used Azure VM Scale Set which let us create and manage a group of load-balanced virtual machines (VMs). Increase or decrease the number of VMs automatically in response to demand or based on a schedule you define.



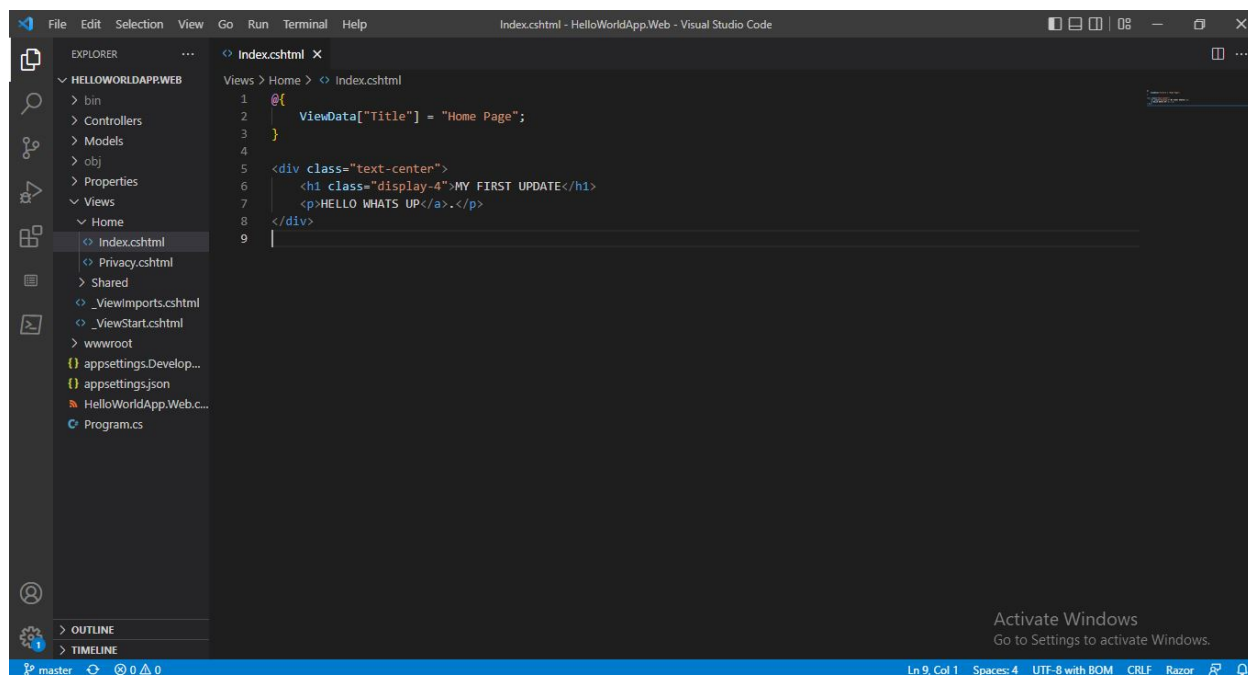
## STEPS

1. Code written should connect to git
2. Connect azure repos with git
3. Create a Project in devops
4. Create a Build Pipeline
5. Connect azure repos to build pipeline
6. Choose build template
7. Queue a build
8. Create a Release Pipeline
9. Create Release Pipeline
10. Choose Release Template
11. Add tasks to Release Definition
12. Check Release Progress

13. Schedule Release
14. Connect release pipeline to app services
15. Attach app service with scale set
16. Start app service

## SCREENSHOT (sample application and pipeline)

### SAMPLE HELLO APP APPLICATION ON VSCODE CONNECTED TO GIT AND AZURE REPO



```
Index.cshtml - HelloWorldApp.Web - Visual Studio Code
1  @{
2  |   ViewData["Title"] = "Home Page";
3  | }
4
5  <div class="text-center">
6  |   <h1 class="display-4">MY FIRST UPDATE</h1>
7  |   <p>HELLO WHATS UP</a>.</p>
8  | </div>
9  |
```

### AZURE REPOS

Files

Name ↑	Last change	Commits
HelloWorld.Web	Sep 15	<a href="#">69a7cb96</a> update8 asadnedaria
.gitignore	Sep 15	<a href="#">acb9819a</a> Initial Status asadnedaria
HelloWorld.sln	Sep 15	<a href="#">acb9819a</a> Initial Status asadnedaria

## AZURE BUILD PIPELINE

Tasks Variables Triggers Options History Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources HelloWebApp master

Agent job 1 Run on agent

- Restore .NET Core
- Build .NET Core
- Test .NET Core
- Publish .NET Core
- Publish Artifact Publish build artifacts

.NET Core Link settings View YAML Remove

Task version 2.x

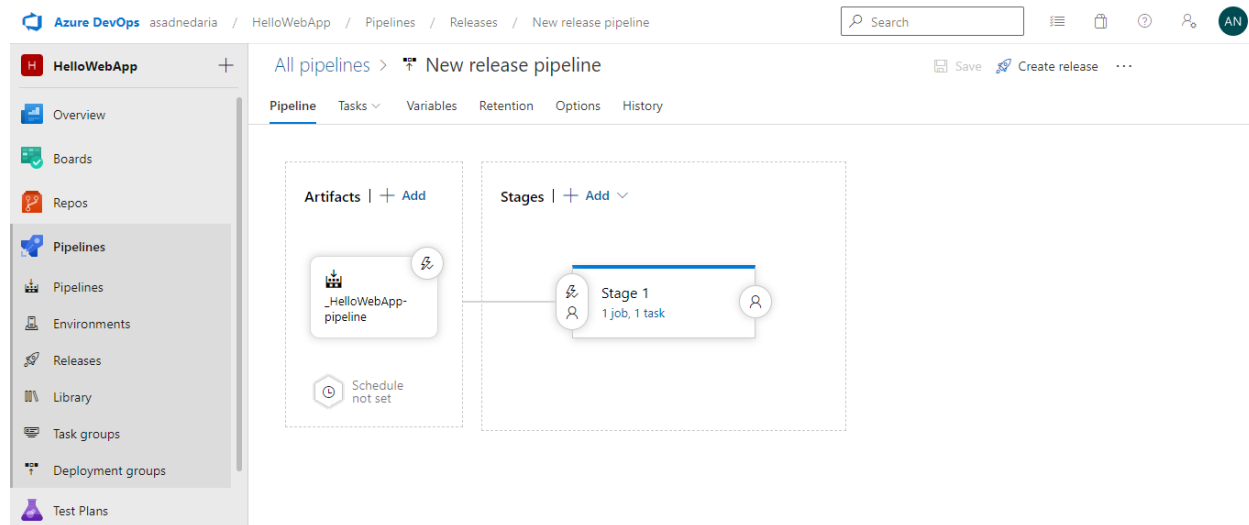
Display name \* Restore

Command \* restore

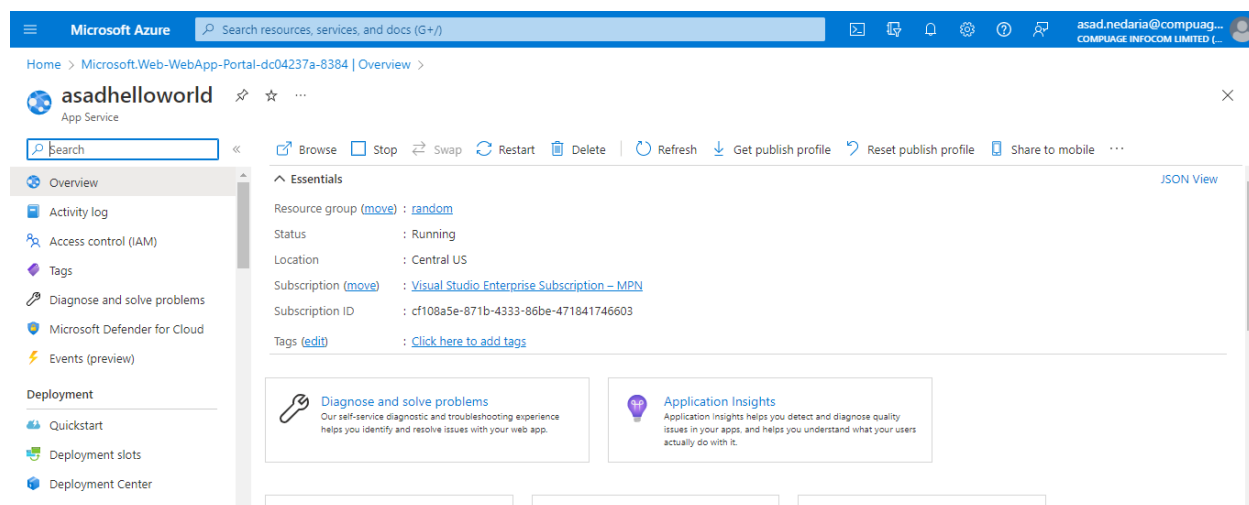
Path to project(s) \*\*/\*.csproj

Arguments

## AZURE RELEASE PIPELINE



## SAMPLE HELLOWORLD APPSERVICE



## CHALLENGE FACED

Some of the common challenges faced by our SaaS clients were:

1. Team Communication- Miscommunication or a lack of communication is one of the main reasons projects fail.
2. Version Control- Every change to the code is recorded by version control software through which we can access previous versions.

3. Security Vulnerabilities- We use HTTPS and SSL encryption to make sure that no third parties can intercept or change the data as it is being sent between you and Azure DevOps.
4. Scaling Up The Infrastructure- In a traditional infrastructure you have a fixed static infrastructure which needs upfront cost.
5. Complex Reports- Most of the reports provided by SaaS service providers are very complex and hard to understand.
6. Adopting Automated Testing- Automated tests are excellent at spotting mistakes with an application's fundamental features and functionalities without manual error.
7. Reaching time constraints- As most of our client have specific time available to publish their application in production environment.

## **BUSINESS BENEFIT**

In our case the deadline for the SaaS client's launch was always met. The offshore team was organized to participate while maintaining compliance norms. The team kept using CI/CD pipelines to create and deploy infrastructure which made their task easier. Using this implementation we were able to boost our deployment speed, meet compliance requirements, and cut operating costs to a great extent. The benefit with Azure CI/CD pipelines is you can deploy to multiple target environments like Virtual Machines, Containers, or any On-prem or Cloud Platform.

**By Mr.ASAD NEDARIA**