AZURE DEVOPS TUTORIAL: BUILDING CI/CD PIPELINE FOR JAVA APP WITH MYSQL

Hitesh Sharma (hiteshs@kth.se) and Avijit Roy (avijit@kth.se)

KTH DEVOPS COURSE (DD2482)

Table of Contents

Overview:

Innovation and ability to compete in the marketplace are some of the main factors to become successful in the industry. Continuous integration and continuous delivery help companies to deliver well and deliver often and allows companies to compete in the market. Azure DevOps makes this CI CD process setup easy and flexible. It can start with code available in git repository such as GitHub or by uploading the code in repo provided by Azure. You can build and deploy an application in different azure services such as virtual machine, app service, Azure Kubernetes Service or even in on-premises. In this tutorial, you will see how to set up a CI/CD pipeline starting from creating a project in azure to deploy an application on a live server.

Other than using Azure console to set up resources and configure CI CD pipeline, we are using Azure CLI (Command Line Interface) for all tasks. This makes the process faster and removes error that would cause by setting up configurations manually in different projects and environments. One can achieve this without any physical environment and using Azure CLI in azure portal or by installing Azure CLI on a physical system.

Things covered in this tutorial:

1. Setup Azure CLI, Devops extension and default values for the environment.
2. Creation of project with scrum settings in azure portal.
3. Uploading code in azure repo.
4. Deploy an Azure Container Registry repo.
5. Deploy an Azure MySQL DB server
6. Deploy an Azure web app for container
7. Setup build pipeline (continuous integration) for a java web application with MySQL database.
8. Run build pipeline.
9. Setup release pipeline (continuous delivery) for staging and production environment.
10. Run release pipeline.

Prerequisite:

1. Microsoft Azure account: To follow this tutorial you need to have an active Microsoft Azure Account. You can create a free trial account.
2. Azure DevOps Account: You can create a DevOps account in Azure using this link.
3. A java web application with a test project. You can use your code or the code used for this tutorial.
4. An Azure Container Registry service, an Azure web app service for container and an Azure MySql database server.

USP of this tutorial

This document is inspired by many online contents mentioned in references (including official Azure documentation that is too specific), but a significant part is written by the authors comparing between various technologies with tools provided by the Azure platform. The authors come up with the idea of expanding the scope of the article to this extent by including their personal experiences working in IT. This tutorial covers complete of end to end flow which is used in software projects across the globe. It is easy to follow as it in Azure which supports GUI based implementation that is quite simple. If needed, we could use Azure Resource Manager that is generally used for infrastructure creation like VMs, firewalls etc., but we like to focus on the DevOps pipelines that are recommended to be done in this way.

In this tutorial we will first set up accounts and environments, then set up all resources to deploy our shuttle app, such as container registry to push and store docker images, MySQL db server to store app user details and Azure web app to host the web application. Once all are setup, you will build the continuous integration and continuous deployment pipeline using simple yaml file and finally run the pipeline to build and deploy the application into azure. Anyone could easily work on Azure platform as Azure account provides 12 months of free services and USD200 in credit. Get one for you and prepare a full agile DevOps infrastructure in Azure for your startup by following this one guide.

Background and motivation for this tutorial

This document takes contains a complete end to end flow of Azure DevOps. Azure DevOps is a cloud-based, open, and integrated platform for enterprise-scale. Not only that, but it also demonstrates how a company utilizes such tools like Azure (Boards, repositories etc.) to set up their project. The authors of this document have worked professionally more than four years in DevOps and development teams respectively and considered framing one of the latest approach available in IT to deal with famous Agile way of working. This is possible under one umbrella technology of Microsoft Azure and Hitesh also worked as a Jira & bitbucket administrator in his company and could confidently say after working in it that it worked almost the same way on a cloud-based platform.

In agile methodology, we describe user stories and often take these from the backlog. These user stories are further divided into tasks and subtasks which are assigned to a team member. This all requires some tool to have presentable dashboards having graphical widgets, hours and tasks against different teams working together etc. This is facilitated my many tools Atlassian Jira is a widely used tool for this. A similar tool is Azure Boards which is there under the Azure ecosystem. Azure Boards is the specific Azure service focused on agile project management. This has been demonstrated in **part 1**.

Next step in any project flow involves an SCM (Source Code Management) tool Atlassian Bitbucket is quite famous in most of the IT companies. By comparing the similar technology in Azure, there comes Azure Repos which is short for Repository. After using it for this tutorial, we realized it provides all the necessary features of SCM tools- git repository, pull requests, semantic code search to point out the most general ones. Moreover, there is also support for any GIT client and API integration with web-hooks. This is clearly described in **part 2** with relevant graphics and implementation steps.

Now the project has been planned, it comes the Azure DevOps core that is pipeline. Before we start discussing pipelines let's have some pre-requisites prepared. For docker, we need a Docker registry server. In Azure we get an Azure container registry which is a private repo to store docker images. We are using Azure Container registry that we used here instead of public/ private registry of Dockerhub. That is needed in any company when you don't want to expose your product on a public registry and don't have expertise in setting up all-new registry server. Not only that, Azure never forces you to use its product, one is free to use other registry servers if you like. Integration with such services could be comfortably enabled here. Few features like Geo-replication feature of Azure registry is interesting, in this user just need to interact with one registry server for push/ pull of images but it can be automatically set-up to replicate in another region for example between West US and East Asia. This feature provides HA for docker images. Another important feature here we can relate is the pipeline within Azure registry. What this means is whenever there is any change in the base image the dependent applications that are utilizing the containers based out of those images could be automated in a pipeline to be replaced. Therefore, we thought to include this Azure container registry feature in this document as described in **part 3**.

In CI/CD pipelines we have Continuous Integration this is taken care of by the SCM tool in part 2. Now this part is focused on the pipelines feature provided under Azure pipeline menu. Apart from providing the services to build and deploy any program based on any language, it offers a wide variety of community-built build, test, and deployment tasks, along with hundreds of extensions like Slack and SonarCloud. One might think that AWS and Azure are rivals so there will not be any integration between them at least. But to their surprise, one of the important features that I wish to highlight here is that it offers the facility of deploying to any cloud that means implementing continuous delivery (CD) to any cloud provider it may be AWS or GCP. Visualizations for deployments are also clear and graphically presentable as shown in the **rest of the parts**.

We can find different option available to configure pipeline. These options are called tasks. Tasks are nothing but the building block to define the automation of build and release process. Task contains packaged scripts that are used to run a job, such as build or release. You can choose a predefined task from the list or create a custom task using yaml file. There are three types of tasks available that are defined in **part 4**.

User can make use of multi-phased builds and can put triggers and conditions for example- when it gets success in one stage then load certain parameters and perform forthcoming stages order. It offers support for YAML, test integration, release gates, reporting etc. This is described in **part 5** with the support of screenshots.

Next is a release pipeline that will take the artifacts created in build pipeline and deploy. Our java application is a web application and we have created a docker image for the application. In **part 6,** we have deployed an Azure web app for containers and Azure MySQL database server. Here, I would like to introduce a bit about how we manage artifacts created in the build step in IT companies. Some tools manage the binaries and version them. They offer offline access to artefacts and metadata, security and access control, ensuring compliance with a variety of license requirements. Therefore, the industry uses it, but it is underrated in general development or IT start-up companies. We strongly suggest using this. In Azure they have their Azure Artifactory library for serving this purpose.

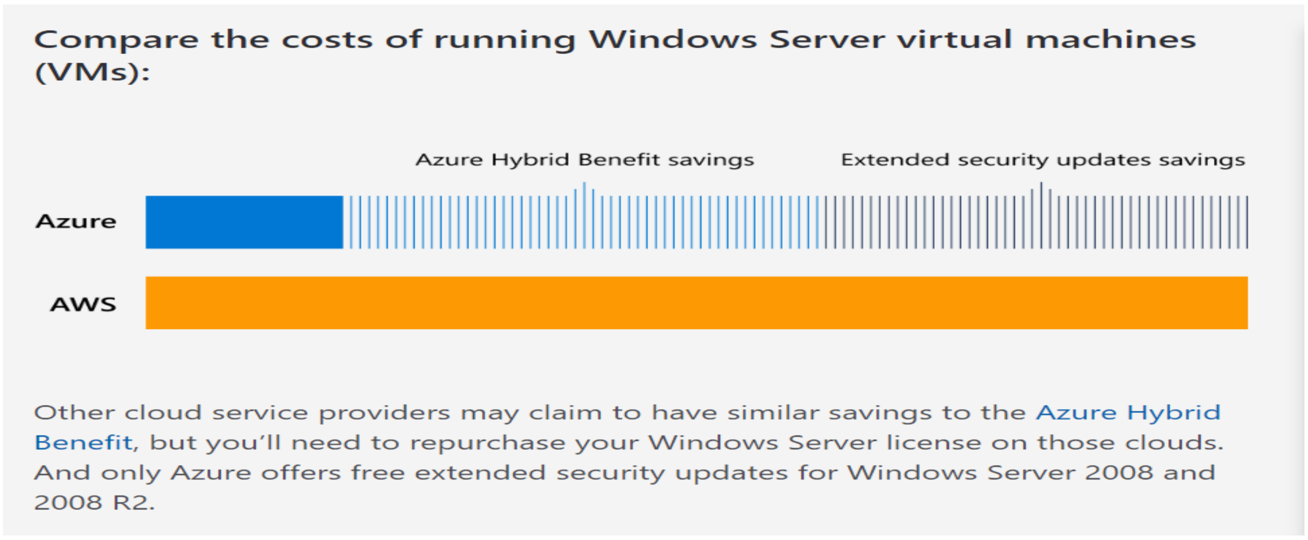
Last **parts 6 and 7**, demonstrates the implementation of a simple web application taken from the internet as the purpose is to demonstrate the configuration, build and deployment which DevOps deals with. IT includes Azure MySQL DB server as a database. This provides a complete view of how the pipeline finally looks when triggered and what are all the components doing behind it.

<u>An Important fact to consider:</u>

The author of this tutorial Hitesh is an AWS SysOps administrator and Avijit is AWS Architect. We have chosen Azure DevOps over AWS documentation because while working for the companies they have seen many projects being migrated from AWS setup to Azure platform.

Hitesh leading a DevOps team knows the reason behind it. Both cloud providers are the leader in the market and offer benefits to IT companies for bulk membership, but Microsoft provided significant discounts to take over the market which was earlier dominated by AWS. Many of the companies are in a stage of migrating or have already migrated to Azure and it is climbing the position of number 1 cloud service provider.

The industry is going towards Azure so why should we not upskill us. It has become an important skill to master the top cloud provider which is soon going to be Microsoft Azure.



Other cloud service providers may claim to have similar savings to the Azure Hybrid Benefit, but you'll need to repurchase your Windows Server license on those clouds. And only Azure offers free extended security updates for Windows Server 2008 and 2008 R2.

Java app: Overview

For this tutorial, we will use Java maven project with MySQL to store and retrieve data. We will use this web app called **MyShuttle** where employees can login used login page and see their trip expenses. We need a web server with tomcat 9 to deploy the web app and a MySQL database server. For this tutorial, we will create docker image to deploy application and use Azure web app service to host the app. Dockerfile, Docker-compose file and database script file, all are included in code repo. You can download code from:

https://aviroy1988@dev.azure.com/aviroy1988/My_Shuttel_Web_CI_CD/_git/MyShuttelWeb

Do you know?

Why is Azure DevOps most popular over other cloud vendors?

It allows users to prepare tasks or utilize the existing once i.e. facilitates the Power of open source Community.

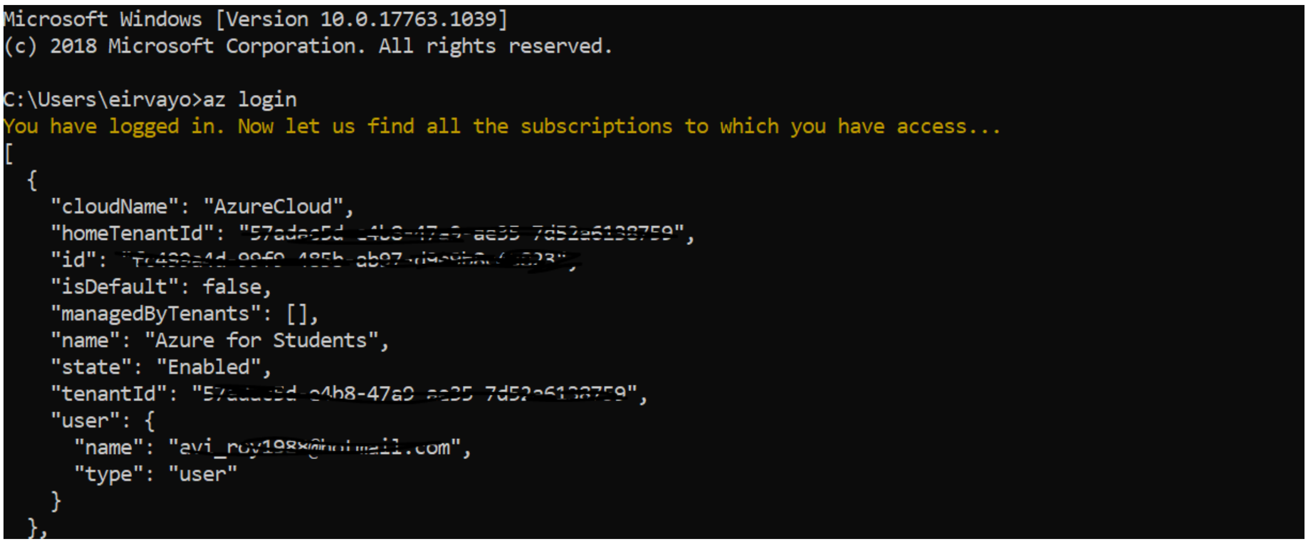
Azure DevOps gives unlimited private agile planning, Git hosting, continuous integration and release management for continuous delivery to the cloud and on-premises as opposed to its rival AWS.

Let's get started with Azure CLI installation and default value setup

You can install latest version CLI following the Microsoft Docs. At a minimum, you should install version 2.0.49 to use azure CLI with DevOps extension. You can check the installation by running **az -version** command which will give you installed version.

Next, add azure devops extension by running **az extension add --name azure-devops.**

You should sign in to azure portal to start using your azure account. You can do this using **az login** command which will redirect you to your default browser, login using your azure account credentials. Once login is successful you will be able to see your available subscriptions in the command window.



```
Microsoft Windows [Version 10.0.17763.1039]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\eirvayo>az login
You have logged in. Now let us find all the subscriptions to which you have access...
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "[redacted]",
    "id": "[redacted]",
    "isDefault": false,
    "managedByTenants": [],
    "name": "Azure for Students",
    "state": "Enabled",
    "tenantId": "[redacted]",
    "user": {
      "name": "[redacted]",
      "type": "user"
    }
  },
```

Next step is to set up default values for your devops project and other azure resources, such as Resource Group, Subscription, location (cloud location, where your resources will be deployed), DevOps organization (created during azure devops signup).

First create a Resource Group (a logical group, under which you will upload other resources for this tutorial) using Azure CLI.

az group create --name "MyShuttelApp_ResourceGroup" --location northeurope

Add, default values for all azure resources

az configure --defaults group=MyShuttelApp_ResourceGroup location=northeurope subscription=<azure subscription ID or Name>

Now, set up default organization for azure devops. You can find the organization name by logging into dev.azure.com. For this tutorial, the default organization is **aviroy1988**

az devops configure --defaults organization=<url of your default organization>

Part 1: Project creation in Azure

In this part of this tutorial, we will create a new project in Azure CLI, set it as default project and validate the same from the portal.

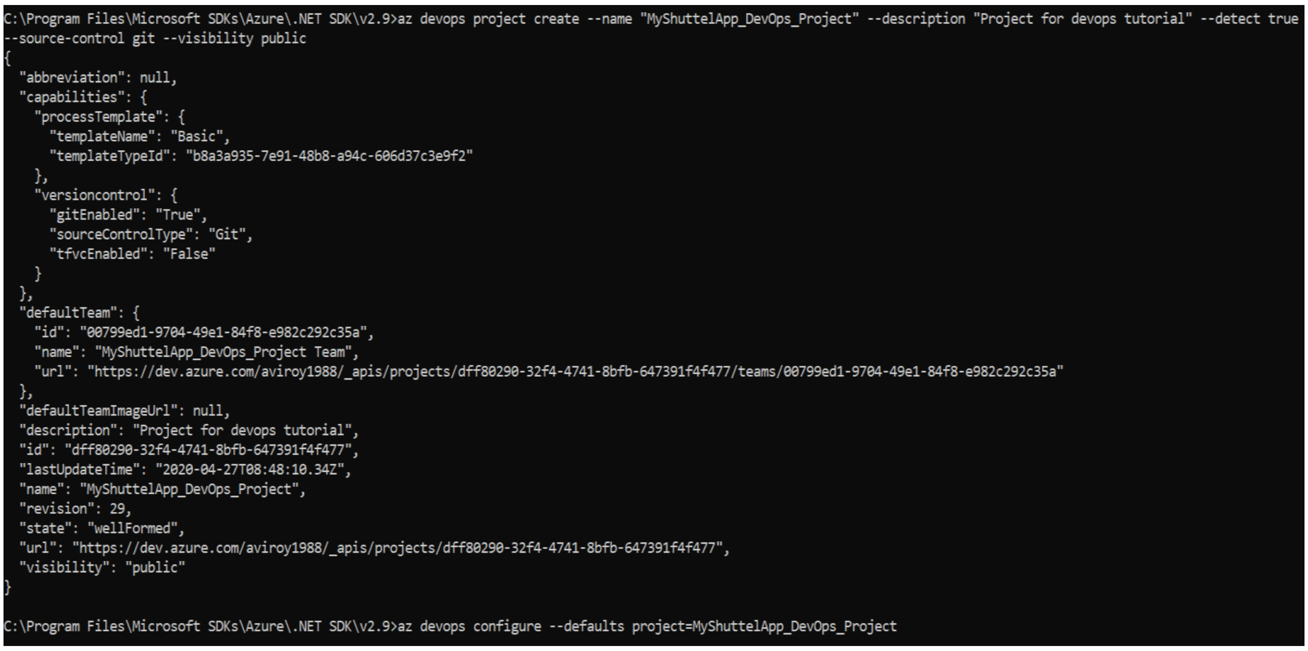
Run create-project command to create new devops project

az devops project create --name "MyShuttelApp_DevOps_Project" --description "Project for devops tutorial" --detect true --process scrum --source-control git --visibility public

This command will create a project with name as **MyShuttelApp_DevOps_Project**, project process as **Scrum,** git will be used as source control and the project is public.

Now, set this project as default project for this tutorial.

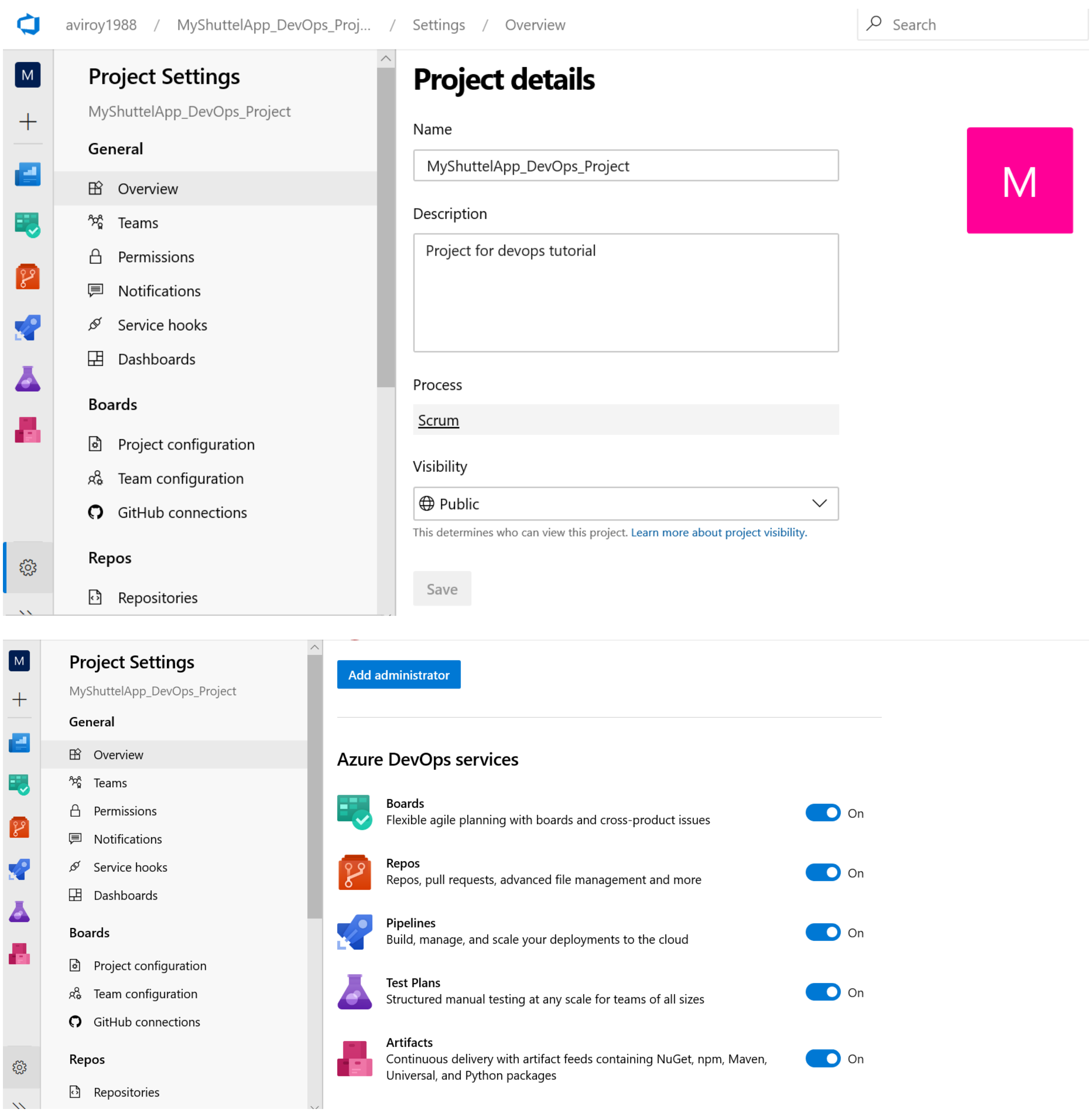
az devops configure --defaults project=MyShuttelApp_DevOps_Project



```
C:\Program Files\Microsoft SDKs\Azure\.NET SDK\v2.9>az devops project create --name "MyShuttelApp_DevOps_Project" --description "Project for devops tutorial" --detect true
--source-control git --visibility public
{
  "abbreviation": null,
  "capabilities": {
    "processTemplate": {
      "templateName": "Basic",
      "templateTypeId": "b8a3a935-7e91-48b8-a94c-606d37c3e9f2"
    },
    "versioncontrol": {
      "gitEnabled": "True",
      "sourceControlType": "Git",
      "tfvcEnabled": "False"
    }
  },
  "defaultTeam": {
    "id": "00799ed1-9704-49e1-84f8-e982c292c35a",
    "name": "MyShuttelApp_DevOps_Project Team",
    "url": "https://dev.azure.com/aviroy1988/_apis/projects/dff80290-32f4-4741-8bfb-647391f4f477/teams/00799ed1-9704-49e1-84f8-e982c292c35a"
  },
  "defaultTeamImageUrl": null,
  "description": "Project for devops tutorial",
  "id": "dff80290-32f4-4741-8bfb-647391f4f477",
  "lastUpdateTime": "2020-04-27T08:48:10.34Z",
  "name": "MyShuttelApp_DevOps_Project",
  "revision": 29,
  "state": "wellFormed",
  "url": "https://dev.azure.com/aviroy1988/_apis/projects/dff80290-32f4-4741-8bfb-647391f4f477",
  "visibility": "public"
}

C:\Program Files\Microsoft SDKs\Azure\.NET SDK\v2.9>az devops configure --defaults project=MyShuttelApp_DevOps_Project
```

You can validate the project by logging into azure portal


aviroy1988 / MyShuttelApp_DevOps_Proj... / Settings / Overview

Search

Project Settings

MyShuttelApp_DevOps_Project

General

- Overview
- Teams
- Permissions
- Notifications
- Service hooks
- Dashboards

Boards

- Project configuration
- Team configuration
- GitHub connections

Repos

- Repositories

Project details

Name: MyShuttelApp_DevOps_Project

Description: Project for devops tutorial

Process: Scrum

Visibility: Public

This determines who can view this project. Learn more about project visibility.

Save

Add administrator

Azure DevOps services

Boards	Flexible agile planning with boards and cross-product issues	On
Repos	Repos, pull requests, advanced file management and more	On
Pipelines	Build, manage, and scale your deployments to the cloud	On
Test Plans	Structured manual testing at any scale for teams of all sizes	On
Artifacts	Continuous delivery with artifact feeds containing NuGet, npm, Maven, Universal, and Python packages	On

Figure 1: Project creation

This project demonstrates the complete structure of a project that we do in a project with an actual industrial setup. Azure DevOps provides us with all those necessary tools under one umbrella called Azure DevOps. It is interesting and easy for any DevOps team to manage. Figure 1 shows a summary of the newly created project.

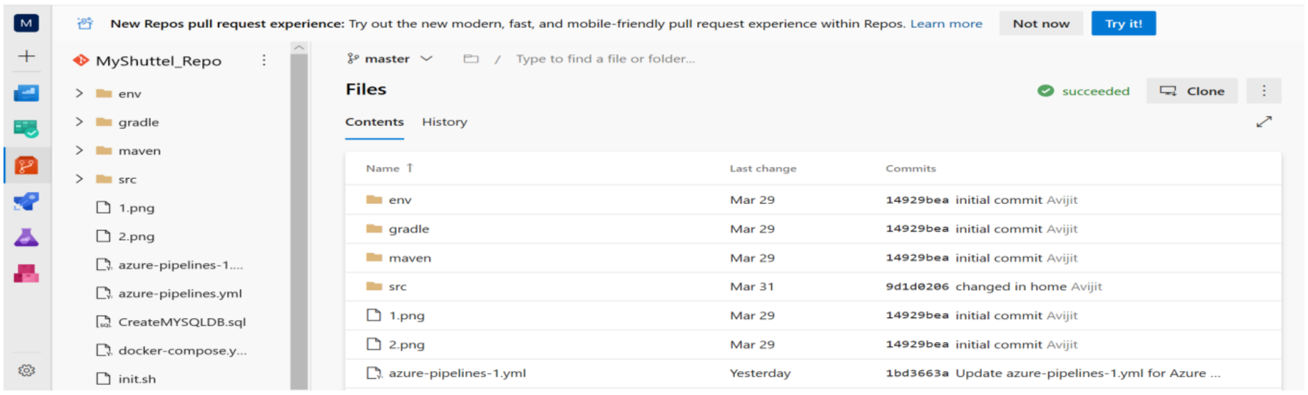
Part 2: Repo creation and code upload

In this tutorial, git is used as a code repository and a simple Java web application with MySQL database is used. You can upload code from GitHub, Team server, or another azure repo. Here we will upload code that is already present in another azure public repo.

You will use **repos create** command to create a new repository and **repos import create** to upload code from another git repo.

Create repo: **az repos create --name="MyShuttel_Repo" --detect=true**

Import code: **az repos import create --git-source-url https://aviroy1988@dev.azure.com/aviroy1988/My_Shuttel_Web_CI_CD/_git/MyShuttelWeb --detect=true --repository MyShuttel_Repo**



New Repos pull request experience: Try out the new modern, fast, and mobile-friendly pull request experience within Repos. Learn more   Not now   Try it!

MyShuttel_Repo

- env
- gradle
- maven
- src
- 1.png
- 2.png
- azure-pipelines-1....
- azure-pipelines.yml
- CreateMYSQLDB.sql
- docker-compose.y...
- init.sh

master   Type to find a file or folder...

Files   succeeded   Clone

Contents   History

Name	Last change	Commits
env	Mar 29	14929bea initial commit Avijit
gradle	Mar 29	14929bea initial commit Avijit
maven	Mar 29	14929bea initial commit Avijit
src	Mar 31	9d1d0206 changed in home Avijit
1.png	Mar 29	14929bea initial commit Avijit
2.png	Mar 29	14929bea initial commit Avijit
azure-pipelines-1.yml	Yesterday	1bd3663a Update azure-pipelines-1.yml for Azure ...

<u>Do you know?</u>

Containers packages up an application with all the dependencies, such as libraries, and deploy it as one single package. Docker is getting widely famous because it implements containers. So we have also included docker image in our tutorial.

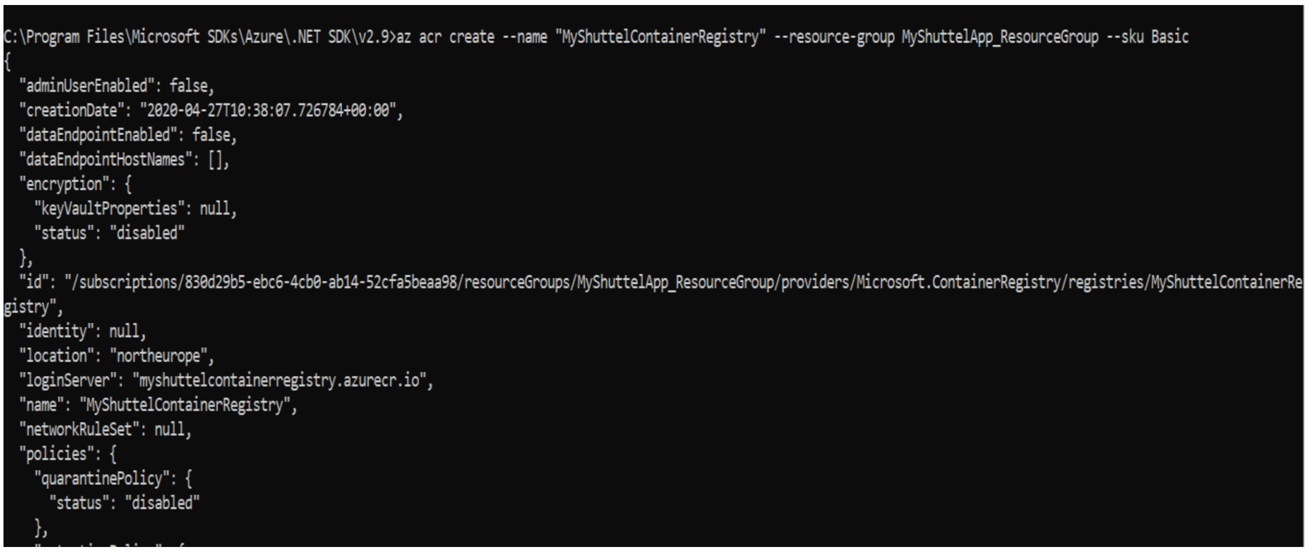
Part 3: Create Azure Container Registry

In this part, you will use an Azure container registry. This is a docker repository where you will push and store docker images of MyShuttelApp.

We can create the container registry by only one command which shows the power of using CLI over console interface.

az acr create --name "MyShuttelContainerRegistry" --resource-group MyShuttelApp_ResourceGroup --sku Basic --admin-enabled true

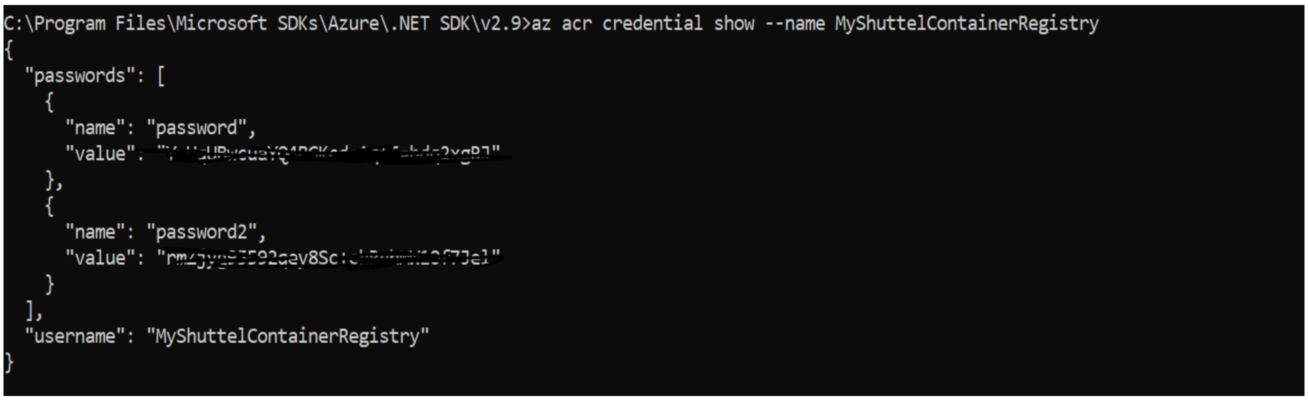
Using this command, we are creating a private container registry with name MyShuttelContainerRegistry and this will be placed in MyShuttelApp_ResourceGroup. Here, **sku** define the billing tier of this container registry and admin access is enabled for this.



```
C:\Program Files\Microsoft SDKs\Azure\.NET SDK\v2.9>az acr create --name "MyShuttelContainerRegistry" --resource-group MyShuttelApp_ResourceGroup --sku Basic
{
  "adminUserEnabled": false,
  "creationDate": "2020-04-27T10:38:07.726784+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/830d29b5-ebc6-4cb0-ab14-52cfa5beaa98/resourceGroups/MyShuttelApp_ResourceGroup/providers/Microsoft.ContainerRegistry/registries/MyShuttelContainerRe
gistry",
  "identity": null,
  "location": "northeurope",
  "loginServer": "myshuttelcontainerregistry.azurecr.io",
  "name": "MyShuttelContainerRegistry",
  "networkRuleSet": null,
  "policies": {
    "quarantinePolicy": {
      "status": "disabled"
    },
```

You would require the username and password to access container registry server during the creation of Azure web app service. Credentials are viewable using below command

az acr credential show --name MyShuttelContainerRegistry



```
C:\Program Files\Microsoft SDKs\Azure\.NET SDK\v2.9>az acr credential show --name MyShuttelContainerRegistry
{
  "passwords": [
    {
      "name": "password",
      "value": [redacted]
    },
    {
      "name": "password2",
      "value": [redacted]
    }
  ],
  "username": "MyShuttelContainerRegistry"
}
```

Save these values for the creation of Azure Web app service with a container.

Part 4: Create Azure MySQL DB server

In this section, we will deploy a MySQL DB server hosted and managed by azure cloud. This database server will be up and running only in minute (Isn't it awesome !!) and using only two commands.

Create DB:

az mysql server create --resource-group MyShuttelApp_ResourceGroup --name myshutteldbserver --admin-user mysqldbuser --admin-password Password@123 --sku-name GP_Gen5_2

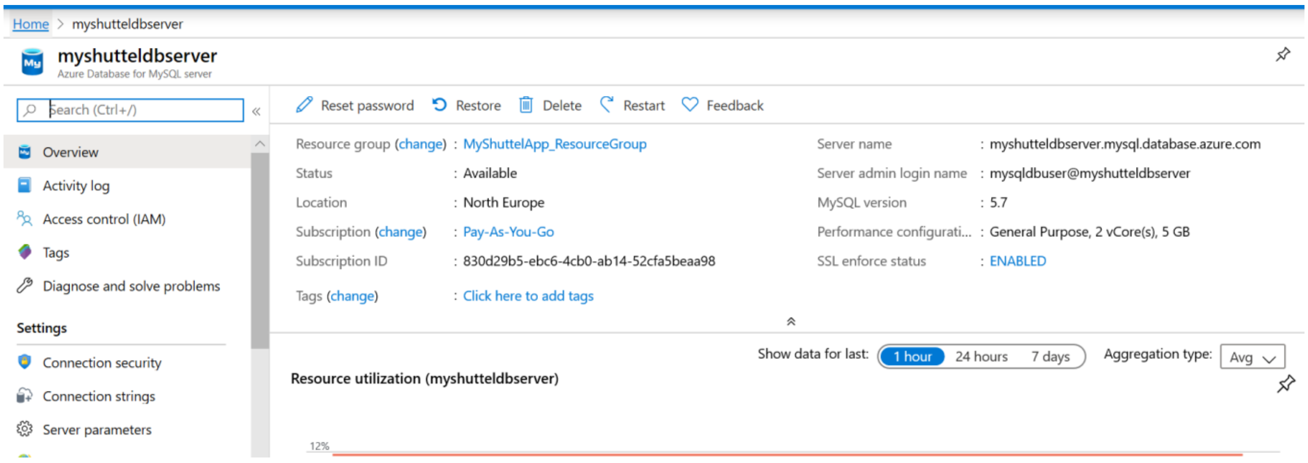
Add a firewall rule to access DB server from other Azure services

Create firewall:

az mysql server firewall-rule create --resource-group MyShuttelApp_ResourceGroup --server myshutteldbserver --name AllowAllAzureIps --start-ip-address 0.0.0.0 --end-ip-address 0.0.0.0

In the first command, we are creating a MySQL db server instance, with name as **myshutteldbserver**, admin-user as **mysqldbuser** and password. Here, **sku** again defines the pricing tier.

Once the command executed successfully, we can validate it from portal.azure.com.


Home > myshutteldbserver

myshutteldbserver
Azure Database for MySQL server

Resource group (change)	: MyShuttelApp_ResourceGroup		Server name	: myshutteldbserver.mysql.database.azure.com
Status	: Available		Server admin login name	: mysqldbuser@myshutteldbserver
Location	: North Europe		MySQL version	: 5.7
Subscription (change)	: Pay-As-You-Go		Performance configurati...	: General Purpose, 2 vCore(s), 5 GB
Subscription ID	: 830d29b5-ebc6-4cb0-ab14-52cfa5beaa98		SSL enforce status	: ENABLED
Tags (change)	: Click here to add tags			

Resource utilization (myshutteldbserver)

Part 5: Create Azure Web app service for container

In this section, we will create an Azure web app service to host our shuttle app in a fully-manged platform to perform OS patching, capacity provisioning, servers and load balancing. This can be configured easily with only a few commands in Azure CLI.

Create app service plan:

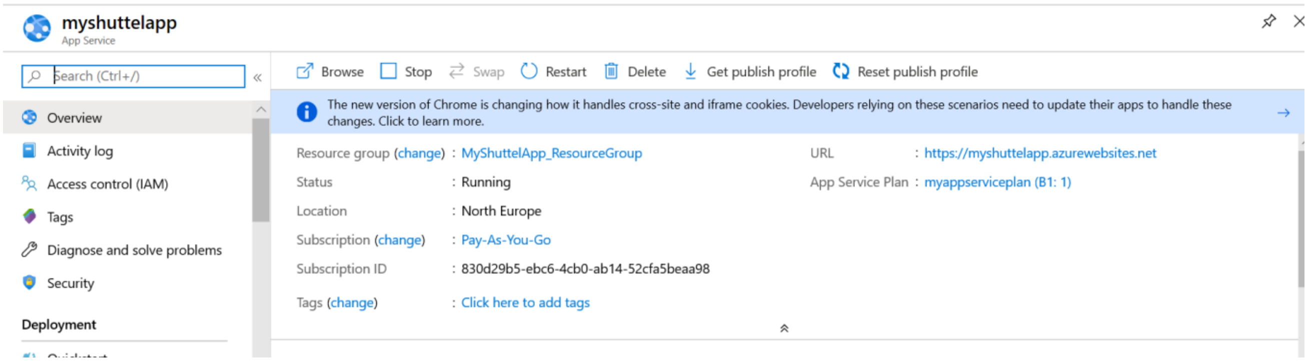
az appservice plan create -n myappserviceplan --is-linux

Here, we are defining the app service plan name and mentioned that it will be used for Linux environment

Create Web app service using app service plan:

az webapp create --name myshuttelapp --resource-group MyShuttelApp_ResourceGroup --plan myappserviceplan --deployment-container-image-name myshuttelcontainerregistry.azurecr.io/web:latest --docker-registry-server-user MyShuttelContainerRegistry --docker-registry-server-password YvUqUBwcuaYQ4BGKcdsAqf6=hdq2xgRJ

In this command, we are creating the web app with name as **myshuttelapp**, defined the source of the docker image that will be used to deploy the app. The username and password we saved from azure container registry creation step, are also provided to access the registry. With this command, our app is up and running. You can browse the app using link: https://myshuttelapp.azurewebsites.net/


myshuttelapp
App Service

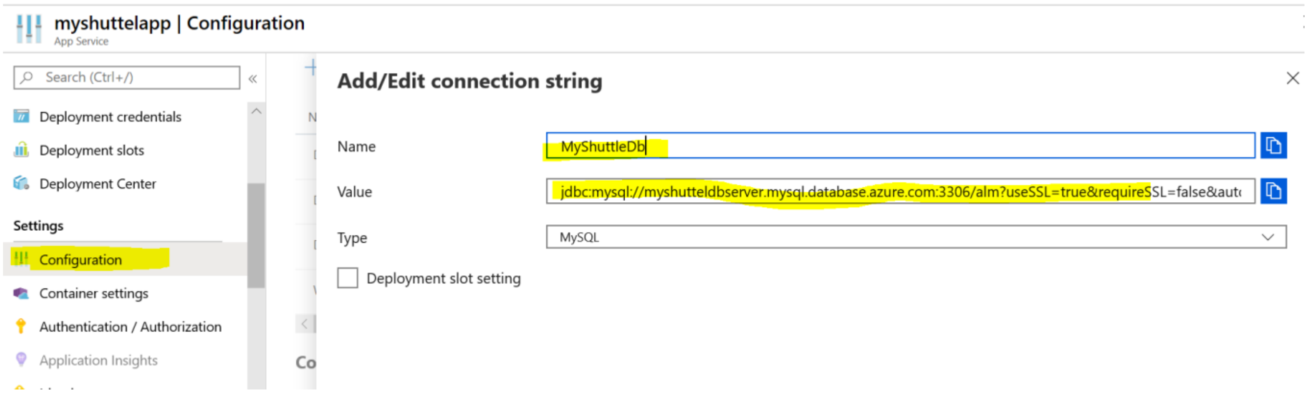
The new version of Chrome is changing how it handles cross-site and iframe cookies. Developers relying on these scenarios need to update their apps to handle these changes. Click to learn more.

Resource group (change)	: MyShuttelApp_ResourceGroup		URL	: https://myshuttelapp.azurewebsites.net
Status	: Running		App Service Plan	: myappserviceplan (B1: 1)
Location	: North Europe			
Subscription (change)	: Pay-As-You-Go			
Subscription ID	: 830d29b5-ebc6-4cb0-ab14-52cfa5beaa98			
Tags (change)	: Click here to add tags			

We need to add the connection string to access the MySQL server, use **webapp config** command to add a connection string

az webapp config connection-string set -g MyShuttelApp_ResourceGroup -n myshuttelapp -t mysql --settings MyShuttleDb="jdbc:mysql://myshutteldbserver.mysql.database.azure.com:3306/alm?useSSL=true&requireSSL=false&autoReconnect=true&user=mysqldbuser@myshutteldbserver&password=Password@123"

You can verify the details for configuration section of this myshuttelapp



myshuttelapp | Configuration

App Service

Add/Edit connection string

Name: MyShuttleDb

Value: jdbc:mysql://myshutteldbserver.mysql.database.azure.com:3306/alm?useSSL=true&requireSSL=false&aut

Type: MySQL

Deployment slot setting

Do you know?

YAML is a human friendly format. It is the default configuration file format used by Ruby on Rails and used in a number of notable tools such as Ansible and Google App Engine. Here, we have used for Azure Tasks. It was released in 2001 but recently from 2015 it gained popularity.

Part 6: Create Azure CI CD pipeline

Now we are ready with all the resources that we need to deploy our app code. In this section, we will create the azure pipeline to build and deploy the code and database. In the azure pipeline, you can create pipelines and releases using classic editor where you have different predefined steps to add. Another option is to create the pipeline using YAML file. In this tutorial, we will create YAML file for the pipeline to run it using azure CLI.

Writing YAML file for CI CD pipeline:

A pipeline is defined as one or more stages which describes a CI CD process. In a pipeline, these stages are major divisions. Stages can be as 'Build the application', 'Test the app', 'Deploy the application'.

A stage consists of one or more jobs and a job is a unit of work that is assignable on the same machine. Both stages and jobs can be set up as dependency graph such that 'Run stage 2 after stage 1' or 'Run this job if the previous job is successful'.

A job is a combination of a linear series of steps. A step can be a task, scripts or reference to external templates.

To build the CI CD pipeline for this tutorial, we divide whole process in two stages. Build stage and Deploy deployment stage. We will discuss briefly about these stages and the jobs and steps inside that.

We have written build stage with only one job named as MavenPackageAndPublishArtifacts. The purpose of this job is to create package for deployment and publish them on pipeline. This job consists 4 steps, those are,

1. **Maven pom.xml**: In this step, a maven build is run using pom.xml present in project folder and a deployment package is created. This package consists of all the required library and packages to run the application. A code coverage tool, JoCoCo is added which will automatically recognise the test cases present in the project run them and publish test report and code coverage report.

2. **Copy files in Artifacs staging directory:** Packages created in the previous step and other files such as database scripts are copied into pipeline artifact directory to make it available for deployment.

3. **Build Docker Image:** In this step, a docker image is build depending on docker-compose file present in project directory.

4. **Push Docker Image:** Docker image created in the previous step is pushed to docker registry in this step. Please note you need to provide subscription information to access docker registry in this step which is defined in azureSubscription tag.

Once the build stage succeeds, deploy stage will start. Like the previous stage, we have one job to deploy the app on Linux and deploy database script to create the database schema on MySQL DB server. This job consists of two steps

1. **Deploy Database Script on MySQL server:** Database script copied on build pipeline is run on MySQL server in this step. Database server name, database name and subscription details are required for this step to run
2. **Deploy App on Azure web app container:** In this final step, the docker image is deployed on azure web app. Docker image is pulled for docker container registry therefore subscription details are required here too.

Other than stages we have other section such as trigger, variables. Trigger section is used to enable the pipeline for continuous integration and deployment. In this yaml file, the master branch of the code repository is defined as the trigger, which means any commit made on the master branch will trigger this pipeline to build and deploy the application with recent changes. In the variable section, common variables that are used in different steps of this yaml can be assigned here. It is always advisable not to include sensitive information inside yaml file, rather store them in pipeline variables and use reference in yaml file.

The yaml file, SQL script and docker-compose file are already present in our repository which is linked into final yaml file. Here is the yaml file that we are going to run to create azure CI CD pipeline.

```
# Maven package Java project Web App to Linux on Azure
# Build your Java project and deploy it to Azure as a Linux web app
# Add steps that analyze code, save build artifacts, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/java

trigger:
- master
variables:
  # Azure Resource Manager connection created during pipeline creation
  azureSubscription: 'adec6298-0ab4-424c-852d-499d0027c5c4'

  # Web app name
  webAppName: 'myshuttelapp'

  # Environment name
  environmentName: 'Prod'

  # Agent VM image name
  vmImageName: 'ubuntu-latest'

stages:
- stage: Build
  displayName: Build stage
  jobs:
  - job: MavenPackageAndPublishArtifacts
    displayName: Maven Package and Publish Artifacts
    pool:
      name: Azure Pipelines
      vmImage: 'ubuntu-latest'
      demands: maven

    steps:
```

```
  - task: Maven@3
    displayName: 'Maven pom.xml'
    inputs:
      mavenPomFile: 'pom.xml'
      options: '-DskipITs -settings ./maven/settings.xml'
      codeCoverageToolOption: JaCoCo
      codeCoverageSourceDirectories: src/main

  - task: CopyFiles@2
    displayName: 'Copy Files to: $(build.artifactstagingdirectory)'
    inputs:
      SourceFolder: '$(System.DefaultWorkingDirectory)'
      Contents: |
       **/target/*.war
       *.sql
      TargetFolder: $(Build.ArtifactStagingDirectory)

  - upload: $(Build.ArtifactStagingDirectory)
    artifact: drop



  - task: DockerCompose@0
    displayName: 'Build Docker Image'
    inputs:
      containerregistrytype: 'Azure Container Registry'
      azureSubscription: 'Pay-As-You-Go(830d29b5-ebc6-4cb0-ab14-52cfa5beaa98)'
      azureContainerRegistry: '{"loginServer":"myshuttelcontainerregistry.azurecr.io", "id" :
"/subscriptions/830d29b5-ebc6-4cb0-ab14-
52cfa5beaa98/resourceGroups/MyShuttelApp_ResourceGroup/providers/Microsoft.ContainerRegistry/regis
tries/MyShuttelContainerRegistry"}'
      dockerComposeFile: '**/docker-compose.yml'
      action: 'Build services'
      additionalImageTags: '$(Build.BuildNumber)'
      includeLatestTag: true


  - task: DockerCompose@0
    displayName: 'Push Docker Image'
    inputs:
      containerregistrytype: 'Azure Container Registry'
      azureSubscription: 'Pay-As-You-Go(830d29b5-ebc6-4cb0-ab14-52cfa5beaa98)'
      azureContainerRegistry: '{"loginServer":"myshuttelcontainerregistry.azurecr.io", "id" :
"/subscriptions/830d29b5-ebc6-4cb0-ab14-
52cfa5beaa98/resourceGroups/MyShuttelApp_ResourceGroup/providers/Microsoft.ContainerRegistry/regis
tries/MyShuttelContainerRegistry"}'
      dockerComposeFile: '**/docker-compose.yml'
      action: 'Push services'
      additionalImageTags: '$(Build.BuildNumber)'
      includeLatestTag: true


- stage: Deploy
  displayName: Deploy stage
  dependsOn: Build
  condition: succeeded()
  jobs:
  - deployment: DeployLinuxWebApp
  displayName: Deploy Linux Web App
  environment: $(environmentName)
  pool:
    vmImage: $(vmImageName)
  strategy:
    runOnce:
      deploy:
        steps:
        - task: AzureMysqlDeployment@1
          displayName: 'Execute Azure MySQL : SqlTaskFile'
          inputs:
```

```
        azureSubscription: 'Pay-As-You-Go (830d29b5-ebc6-4cb0-ab14-52cfa5beaa98)'

        ServerName: myshutteldbserver.mysql.database.azure.com

        DatabaseName: alm

        SqlUsername: 'mysqldbuser@myshutteldbserver'

        SqlPassword: 'Password@123'

        SqlFile: '$(Pipeline.Workspace)/drop/CreateMYSQLDB.sql'


    - task: AzureWebAppContainer@1

      displayName: 'Azure Web App on Container Deploy: MyShuttelApp'

      inputs:

        azureSubscription: 'Pay-As-You-Go (830d29b5-ebc6-4cb0-ab14-52cfa5beaa98)'

        appName: myshuttelapp

        containers: 'myshuttelcontainerregistry.azurecr.io/web:latest'
```

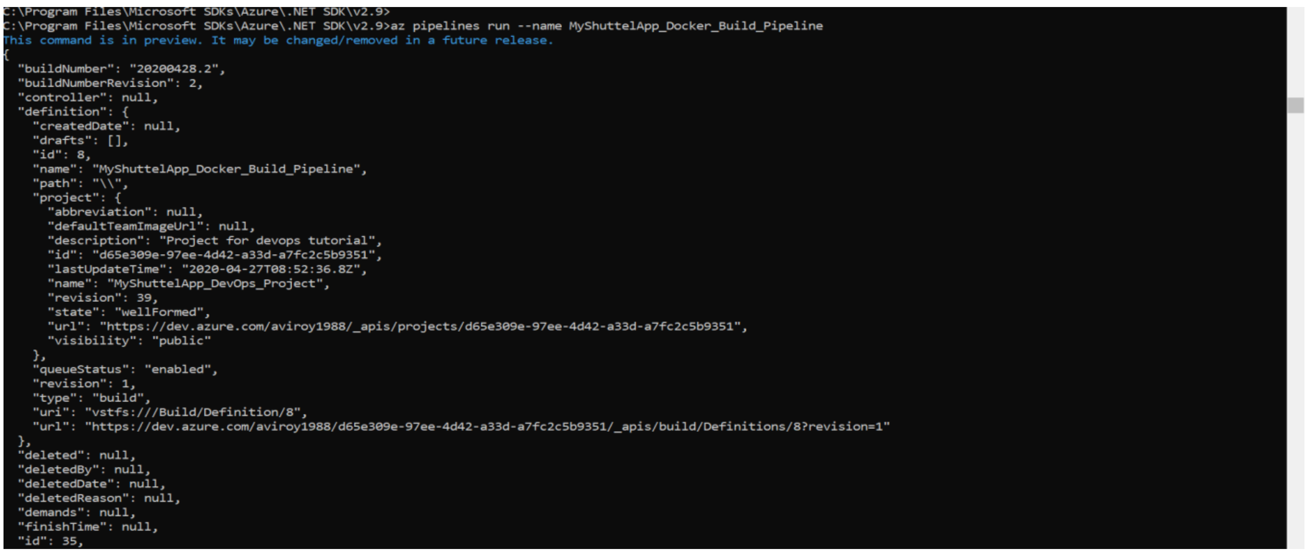
Run the command to create the pipeline using yaml file

az pipelines create --name "MyShuttelApp_Docker_Build_Pipeline" --description "pipeline for java maven project with MySql database" --repository MyShuttel_Repo --repository-type tfsgit --branch master --yml-path azure-pipelines.yml

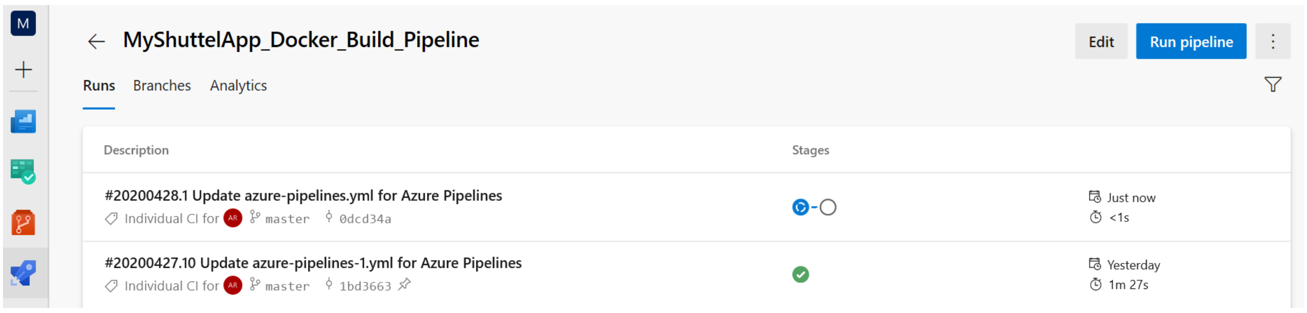
This command will create and run the pipeline also set the trigger for continuous integration and deployment. Please note the **–repository-type** in this command, as we are using repo within azure, therefore the value is **tfsgit**, if you use repo from GitHub or another git repo, change it to git only.

We can manually run the pipeline from CLI using pipeline run command

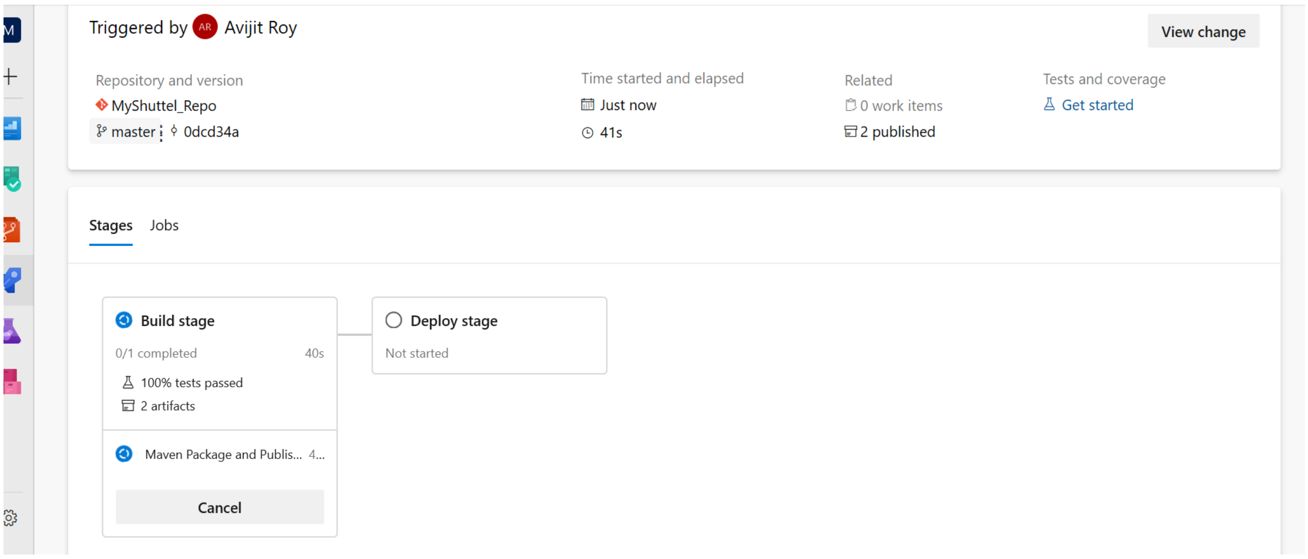
az pipelines run --name MyShuttelApp_Docker_Build_Pipeline



```
C:\Program Files\Microsoft SDKs\Azure\.NET SDK\v2.9>
C:\Program Files\Microsoft SDKs\Azure\.NET SDK\v2.9>az pipelines run --name MyShuttelApp_Docker_Build_Pipeline
This command is in preview. It may be changed/removed in a future release.
{
  "buildNumber": "20200428.2",
  "buildNumberRevision": 2,
  "controller": null,
  "definition": {
    "createdDate": null,
    "drafts": [],
    "id": 8,
    "name": "MyShuttelApp_Docker_Build_Pipeline",
    "path": "\\",
    "project": {
      "abbreviation": null,
      "defaultTeamImageUrl": null,
      "description": "Project for devops tutorial",
      "id": "d65e309e-97ee-4d42-a33d-a7fc2c5b9351",
      "lastUpdateTime": "2020-04-27T08:52:36.8Z",
      "name": "MyShuttelApp_DevOps_Project",
      "revision": 39,
      "state": "wellFormed",
      "url": "https://dev.azure.com/aviroy1988/_apis/projects/d65e309e-97ee-4d42-a33d-a7fc2c5b9351",
      "visibility": "public"
    },
    "queueStatus": "enabled",
    "revision": 1,
    "type": "build",
    "uri": "vstfs:///Build/Definition/8",
    "url": "https://dev.azure.com/aviroy1988/d65e309e-97ee-4d42-a33d-a7fc2c5b9351/_apis/build/Definitions/8?revision=1"
  },
  "deleted": null,
  "deletedBy": null,
  "deletedDate": null,
  "deletedReason": null,
  "demands": null,
  "finishTime": null,
  "id": 35,
```


MyShuttelApp_Docker_Build_Pipeline

Description	Stages	
#20200428.1 Update azure-pipelines.yml for Azure Pipelines Individual CI for AR master 0dcd34a		Just now <1s
#20200427.10 Update azure-pipelines-1.yml for Azure Pipelines Individual CI for AR master 1bd3663	✓	Yesterday 1m 27s


Triggered by AR Avijit Roy

Repository and version	Time started and elapsed	Related	Tests and coverage
MyShuttel_Repo	Just now	0 work items	Get started
master 0dcd34a	41s	2 published	

Build stage
0/1 completed 40s
100% tests passed
2 artifacts
Maven Package and Publis... 4...

Deploy stage
Not started

Do you know?

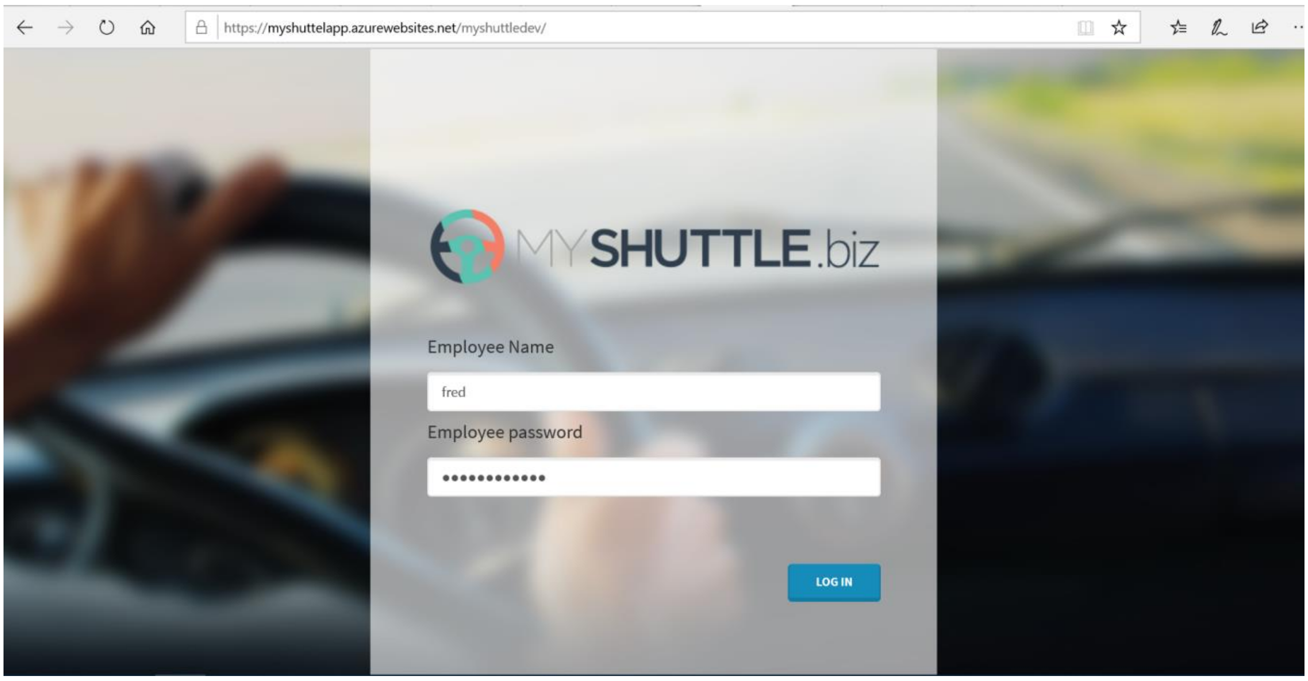
Why is pipeline named so?

Pipeline is called so because it flows in the same fashion as a tubular water pipe flowing water. We can also say it performs compile, build and deploy one after the other where human generally see the input and the result at the two ends only, rest is all automated.

Part 7: Accessing the deployed application

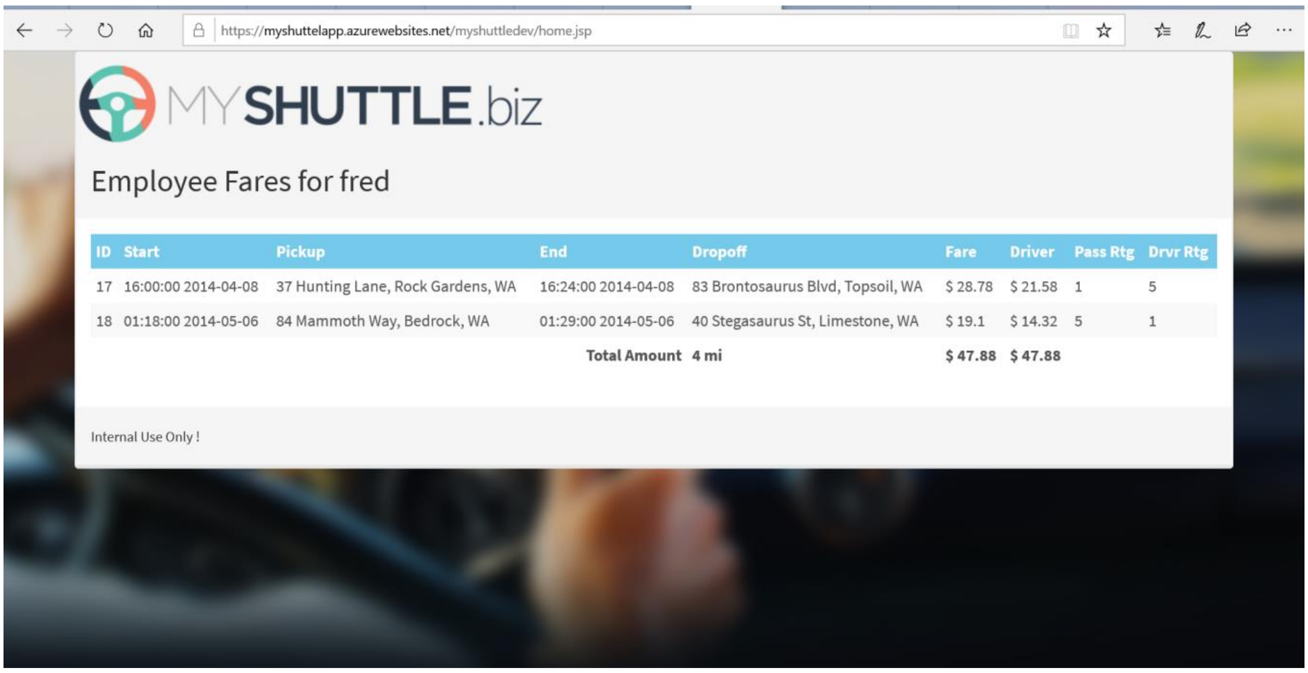
Finally, MyShuttel app is deployed and ready to browse. You can go to MyshutteApp, Azure web app resource and use browse button or use the following link to browse the application login page.

https://myshuttelapp.azurewebsites.net/myshuttledev/



Employee Name: fred

Employee password: ••••••••••••


https://myshuttelapp.azurewebsites.net/myshuttledev/home.jsp
MYSHUTTLE.biz

Employee Fares for fred

ID	Start	Pickup	End	Dropoff	Fare	Driver	Pass Rtg	Drvr Rtg
17	16:00:00 2014-04-08	37 Hunting Lane, Rock Gardens, WA	16:24:00 2014-04-08	83 Brontosaurus Blvd, Topsoil, WA	$ 28.78	$ 21.58	1	5
18	01:18:00 2014-05-06	84 Mammoth Way, Bedrock, WA	01:29:00 2014-05-06	40 Stegasaurus St, Limestone, WA	$ 19.1	$ 14.32	5	1
			Total Amount	**4 mi**	**$ 47.88**	**$ 47.88**		

Internal Use Only !

In this demo, we presented the overall flow of an implementation of a project with CI CD pipeline including the infrastructure set up to build, deploy and host the application. One can easily follow this tutorial to deploy and host their web application with continuous integration and delivery in place.

References:

https://docs.microsoft.com/en-us/azure/devops-project/azure-devops-project-java?toc=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fazure%2Fdevops-project%2Ftoc.json&bc=https%3A%2F%2Fdocs.microsoft.com%2Fen-us%2Fazure%2Fbread%2Ftoc.json

https://github.com/hsachinraj/GitHub-AzureDevOps

https://azure.microsoft.com/