Accelerators.wiki

Filter pages by title

- Accelerators Wiki
  - Design Document
  - Backlog/Notes
  - CustomerTool & Core...
    - Creating Service C...
    - DeploymentEngine
    - Logic/App & Powe...
    - ProvisioningEngine

New page

DeploymentEngine

Last Author Bobby T. · Apr 21

Unfollow 2 Edit

Contents

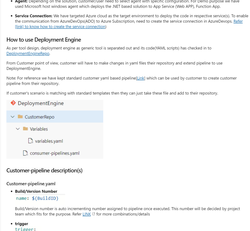
- Introduction
- High/Level Design
- How Deployment Engine differs from Provisioning Engine
- Technical Overview
- ADO Repository Reference
- How to use Deployment Engine
- Customer-pipeline description(s)
  - customer-pipeline.yaml
- Variable-file Description
  - variables.yaml
- Common Code Description
  - common-code-deploy.yaml
- build-dotnet.yaml Description
  - build-dotnet.yaml
- dotnet.yaml Description
  - deploy-dotnet.yaml file

Introduction

The Deployment Engine is a WIAL based tool mainly used for the deployment of various solutions based on .NET, Java etc. This tool target to deploying solutions to Azure environments so called stages (e.g. Dev,Test,ST,UR,UAT,Preprod and Production). Deployment engine will deploy the solutions(s) to the respective service(s) either at a time or based on the var's input. Deployment engine not only reduce the amount of code needed to deploy the solutions but also ensuring the resources to the Azure Services but also provides the error free deployment strategy across various subscriptions/Environments.

High/Level Design

How Deployment Engine differs from Provisioning Engine



Technical Overview

ADO Repository Reference

- **Source Code**: Source Code mainly YAML files has been managed in AzureDevOpsADO.
- **Agent**: Depending on the solution, customer/user need to select agent with specific configuration. For Demo purpose we have used Microsoft hosted windows agent which supports the .NET based solution in App Service (Web APP), Function App.
- **Service Connection**: We have targeted Azure cloud as the target environment to deploy the code in respective service(s). To enable the communication from AzureDevOpsADO to Azure Subscription, need to create the service connection in AzureDevops. Refer link to know how to create the service connection

How to use Deployment Engine

As per best design, deployment engine as generic tool is separated out and its code(YAML scripts) has checked-in in DeploymentEngineRepo.

From Customer point of view, customer will have to make changes in yaml files their repository and extend pipeline to use DeploymentEngine.

Note: For reference we have kept standard customer yaml based pipeline(yaml) which can be used by customer to create customer pipeline from their repository.

If customer's scenario is matching with standard templates then they can just take these file and add to their repository.

Customer-pipeline description(s)

Customer-pipeline.yaml

- **Build/Version Number**
  - name: $(BuildID)

  Build/Version number is auto incrementing number assigned to pipeline once executed. This number will be decided by project team which fits for the purpose. Refer UDA ⧉ for more combinations/details

- **trigger**
  - trigger:
    - none

The trigger is used to run the pipeline automatically. Azure Pipelines supports various types of actions. Depending on your pipeline type/trigger none will disable the CI trigger if nothing is set, the pipeline will be triggered whenever there is any change within the branch. Link for more reference ⧉

- **pool/vmImage**
  - pool:
    - vmImage: "windows-latest"

Azure Pipelines provides a predefined agent pools named as Microsoft-hosted agents and Self-hosted agents. This agent will depends on solutions.Link for more reference ⧉

- **resources**

```
resources:
  repositories:
    - repository: CodeDeployment
      name: Accelerators/DeploymentEngine
      type: git
      ref: master
```

The Resources is defined list of repositories resources referenced by the pipeline. The repositories represents list of repository referenced by the pipeline.

properties	Description
name	[string]The name of the triggering resources ⧉
type	[string]The type of the triggering resources ⧉ , Git - Git repository hosted on an external server Type of repository: git, github, github enterprise, and bitbucket.
reference	[string]Optional. The reference of dependents to: refs/heads/master. The branch checked out by default whenever the resource trigger fires. Template expressions are supported.
repository	[String]Required.e first property. Alias for the specified repository. Acceptable values [,A-Za-z0-9]*.

- **Type**

  Pipeline support the following values for the repository type: git, GitHub, and bitbucket. The git type refers to Azure Repos Git repos.

  If you specify type: git, the name value refers to the name of an Azure Repos Git repository.

  Note: If your pipeline is in the same Azure DevOps project as the repository, for example a repository named tools, you reference it by just the name, for example, tools.

  If your pipeline is in the same Azure DevOps organization as the repository, but in a different Azure DevOps project, for example a project named Tools Project, you must qualify the repository name with the project name. Name: Tools Project/tools.

Variable-file Description

```
variables:
  - template: /Consumer-project/Variables/variables.yaml@self
```

The Variables file parsing from Consumer Project in the Same Repo.

variable.yml

The Variables is Stored in key value pair as below shown

```
variables:
  repositoryName: TobeUpdatedLater
  solutionType: TobeUpdatedLater
  solutionFolderPath: TobeUpdatedLater
  zipFileName: TobeUpdatedLater
  productName: TobeUpdatedLater
  project: TobeUpdatedLater
  buildConfiguration: TobeUpdatedLater
  azureSubscription_DEV: TobeUpdatedLater
  azureSubscription_TST: TobeUpdatedLater
  azureSubscription_ST: TobeUpdatedLater
  azureSubscription_UR: TobeUpdatedLater
  azureSubscription_UAT: TobeUpdatedLater
  azureSubscription_PRE: TobeUpdatedLater
  azureSubscription_PRD: TobeUpdatedLater
  resourceGroupName: TobeUpdatedLater
  appServiceName: TobeUpdatedLater
```

Above Devops variables are Pipeline variables and have limitations.
Variables are different from runtime parameters official docs on define variables ⧉
This is the Scenario of the pipeline Flow:

Common Code Description

Extend from a template with resources

```
extends:
  template: /Pipelines/Templates/common-code-deploy.yaml@CodeDeployment
```

You can a use extends to extend from a template in your Azure pipeline that contains resources.

common-code-deploy.yaml

The Extend template has different stages and jobs as shown below :

```
stages:
  - stage: Build
    displayName: Build
    jobs:
    - template: /Pipelines/Templates/job-templates/build-dotnet.yaml
```

properties	Description
stage	[string] Required as first property. It is ID of the stage.for more reference ⧉
display Name	[string] Human-Readable name for the stage
jobs	which makeup the stage.for more reference ⧉
template	Stage related information passed from pipeline when path of the template is given

build-dotnet.yaml Description

build-dotnet.yaml

We have to Validate the according to your requirement. In this Solution Our team has been done as below shown:

1. We have restore the DotNetCoreCLI with project folder
2. We have publish the project in the System directory.
3. We have archive to bundle this built output to a zip file
4. We have build and publish the Zip file to the Artifact.

```
parameters:
  - name: deploymentParameters
    type: object

jobs:
  - job: Validate
    steps:
    - checkout: self
    - ${{ each deploymentParameter in parameters.deploymentParameters }}:
      - task: DotNetCoreCLI@2
        displayName: 'restore'
        inputs:
          command: "restore"
          projects: "**/*.csproj"
          feedsToUse: "select"

      - task: DotNetCoreCLI@2
        displayName: "publish Integration solution"
        inputs:
          command: publish
          publishWebProjects: false
          projects: ${{ deploymentParameter.buildConfiguration }} --output publish_output"
          publishWebProjects: false
          zipAfterPublish: false
          includeNuGetOrProject: false

      - task: ArchiveFiles@2
        displayName: 'Archive files'
        inputs:
          rootFolderOrFile: '$(System.DefaultWorkingDirectory)/publish_output'
          includeRootFolder: false
          archiveFile: '$(System.DefaultWorkingDirectory)/${{ deploymentParameter.zipFileName }}'

      - task: PublishBuildArtifacts@1
        inputs:
          PathtoPublish: '$(System.DefaultWorkingDirectory)/${{ deploymentParameter.zipFileName }}'
          ArtifactName: '${{ deploymentParameter.productName }}'
```

Properties

1. The **jobs** property defines number of jobs. Each job represents a specific stage in the pipeline, and is responsible for performing a particular set of tasks.Link for more reference ⧉
2. The **steps** property define the individual task that we need perform the stage of the pipeline. Each step in the pipeline performs a specific function.Link for more reference ⧉
3. The **task** property is used simply packaged script that can be abstracted set of inputs. when you run a job all the tasks are run in sequence, one after the other.
4. The **inputs** is collect relevant information to the project's scope of the task.for more reference ⧉

input Values	Description
Command	[string] Allowed values: restore, pack, push, custom.
projects	[string] Path of the Projects file.matching pattern reference ⧉
feedToUse	[string] input alias: selectOrConfig
arguments	[string] when command = restore, we have add additionally arguments to be passed to the restore command.
publishWebProjects	[boolean]when we have Use command = publish. Default value: true.for more reference ⧉
modifyOutputPath	[boolean]Optional.
zipAfterPublish	[boolean]Optional.
nonfailOnFmtOut	[string]
includeRootFolder	[boolean]
archiveFiles	[string]we have give the path of the file ,where we have archive the file .
pathtoPublish	[string]we have give path of the file , where we are publishing.
artifactName	[string] we have give name of the artifact.

- **Common code.yml**

  parameters have more control over what values can be passed to a pipeline.Parameters are only available at template parsing time. Parameters are expanded just before the pipeline runs so that values surrounded by ${{ }} are replaced with parameter values.

Here using the Each condition and loop Conditions to deploy the Environment into solutions(Like DEV,TEST,ST,UR,UAT,PRE,PRD) in the Azure Cloud. The Deployment Parameters are passed from the Consumer pipeline as Shown Below :
The Deployment deploying in the sequential manner by one one.
The subscriptions listed in the code are as follows:

1. azureSubscription_DEV
2. azureSubscription_TST
3. azureSubscription_ST
4. azureSubscription_UR
5. azureSubscription_UAT
6. azureSubscription_PRE
7. azureSubscription_PRD

```
  parameters:
    deploymentParameters: ${{ parameters.deployments }}

  - ${{ each deploymentEnvironment in split(parameters.deploymentEnvironments, ',') }}:
    - ${{ if eq(deploymentEnvironment, 'DEV') }}:
      displayName: Deploy_${{ deploymentEnvironment }}
      - ${{ if eq(deploymentEnvironment, 'DEV') }}: eq( deploymentEnvironment, '') }}:
        dependsOn: CodeDeployment_DEV
      - ${{ if eq(deploymentEnvironment, 'TST') }}: eq( deploymentEnvironment, '') }}:
        dependsOn: CodeDeployment_TST
      - ${{ if eq(deploymentEnvironment, 'ST') }}: eq( deploymentEnvironment, '') }}:
        dependsOn: CodeDeployment_ST
      - ${{ if eq(deploymentEnvironment, 'UR') }}: eq( deploymentEnvironment, '') }}:
        dependsOn: CodeDeployment_UR
      - ${{ if eq(deploymentEnvironment, 'UAT') }}: eq( deploymentEnvironment, '') }}:
        dependsOn: CodeDeployment_UAT
      - ${{ if eq(deploymentEnvironment, 'PRE') }}: eq( deploymentEnvironment, '') }}:
        dependsOn: CodeDeployment_PRE
      - ${{ if eq(deploymentEnvironment, 'PRD') }}: eq( deploymentEnvironment, '') }}:
        dependsOn: CodeDeployment_PRD

      variables:
      - ${{ each azureSubscription in parameters.azureSubscriptions }}:
        - name: azureSubscriptionPackageConnection
          ${{ if eq(azureSubscription.azureSubscription_DEV, '') }}: eq( deploymentEnvironment, '') }}:
            value: ${{ azureSubscription.azureSubscription_DEV }}
          ${{ if eq(azureSubscription.azureSubscription_TST, '') }}: eq( deploymentEnvironment, '') }}:
            value: ${{ azureSubscription.azureSubscription_TST }}
          ${{ if eq(azureSubscription.azureSubscription_ST, '') }}: eq( deploymentEnvironment, '') }}:
            value: ${{ azureSubscription.azureSubscription_ST }}
          ${{ if eq(azureSubscription.azureSubscription_UR, '') }}: eq( deploymentEnvironment, '') }}:
            value: ${{ azureSubscription.azureSubscription_UR }}
          ${{ if eq(azureSubscription.azureSubscription_UAT, '') }}: eq( deploymentEnvironment, '') }}:
            value: ${{ azureSubscription.azureSubscription_UAT }}
          ${{ if eq(azureSubscription.azureSubscription_PRE, '') }}: eq( deploymentEnvironment, '') }}:
            value: ${{ azureSubscription.azureSubscription_PRE }}
          ${{ if eq(azureSubscription.azureSubscription_PRD, '') }}: eq( deploymentEnvironment, '') }}:
            value: ${{ azureSubscription.azureSubscription_PRD }}
```

A jobs Creates the other Specific stage in the pipeline name as **Deploy**Stage

```
jobs:
  - deployment: Deploy
    displayName: Deploy
    environment: ${{ deploymentEnvironment }}
    strategy:
      runOnce:
        deploy:
```

- **deployment** is a stage name
- **displayName** is the property of the stage .
- **Environment** also holds information about the deployment strategy for running the steps defined inside the job.

dotnet.yaml Description

```
steps:
  - template: /Pipelines/Templates/step-templates/deploy-dotnet.yaml
```

- **steps** can insert a template to reuse one or more steps across several jobs. in addition to the steps from the template, each job can define more steps.

deploy-dotnet.yaml file

In this template we have use for Each loop to perform one by one task.
Depending on the requirement we have used the task and we can perform
In our team Solution we have used as shown below.
This tasks can perform the DownloadBuildArtifacts and deploy artifacts to Azure Webapp.

```
  parameters:
    - name: deploymentParameters
      type: object
    - name: serviceConnection
      type: string

  steps:
    - ${{ each deploymentParameter in parameters.deploymentParameters }}:
      - task: DownloadBuildArtifacts@0
        displayName: 'Download artifacts'
        inputs:
          buildType: 'current'
          downloadType: 'single'
          artifactName: ${{ deploymentParameter.productName }}
          downloadPath: '$(System.DefaultWorkingDirectory)/${{ deploymentParameter.zipFileName }}'

      - task: AzureWebAppDeploy@4
        displayName: 'Azure webapp deploy'
        inputs:
          ConnectionType: 'AzureRM'
          appType: webApp
          azureSubscription: ${{ parameters.serviceConnection }}
          resourceGroupName: ${{ deploymentParameter.resourceGroupName }}
          webAppName: ${{ deploymentParameter.appServiceName }}
          package: './${{ deploymentParameter.productName }}/${{ deploymentParameter.zipFileName }}'
```

Inputs	Description
buildType	[string]Download artifacts produced by Whether to download artifacts produced by the current build or from a specific build.
downloadType	Downloads a specific artifact or specific files from the build.for more reference ⧉
ConnectionType	The name you will use to refer to this service connection in task properties. This is not the name of your Azure subscription.
appType	[string] The name of the Azure Applications.
azureSubscription	The access to the service Connection.
resourceGroupName	The name of the resource where we have to deploy.

WebappName	The name of the Application.
package	The name of the Zip file.

- **Commmon-code.yaml**

while creating parameter we have declared parameter in same yaml file only.

If you organizing your pipeline into multiple stages, you have use the stage has a keyword.The **repository Name** parameter is a variable used in a software development project to specify the name of the repository that will be used for storing the project's source code. The value of this parameter is defined by the variables repository Name variable. The **solutionType** parameter is a variable used in a software development project to specify the type of solution being developed. The value of this parameter is defined by the variables solution Type variable file .

```
parameters:
  repositoryName: ${{ variables.repositoryName }}
  solutionType: ${{ variables.solutionType }}
  deploymentEnvironments: ${{variables.deploymentEnvironments}}
```

Azure Subscriptions is a feature of Microsoft Azure that allows users to create the multiple Azure environments within a single Azure account. Each subscription is essentially a logical container for Azure resources, and it provides a way to manage access, billing, and resource usage for those resources.Each of these subscriptions represents a different environment, such as development, testing, staging, user acceptance testing, pre-production, and production. By using Azure Subscriptions, users can deploy and manage their applications across multiple environments while maintaining control over access, security, and costs.
variables. These variables are used to specify the credentials required to access each subscription

```
azureSubscriptions:
  - azureSubscription_DEV: ${{ variables.azureSubscription_DEV }}
    azureSubscription_TST: ${{ variables.azureSubscription_TST }}
    azureSubscription_SIT: ${{ variables.azureSubscription_SIT }}
    azureSubscription_UR:  ${{ variables.azureSubscription_UR }}
    azureSubscription_UAT: ${{ variables.azureSubscription_UAT }}
    azureSubscription_PRE: ${{ variables.azureSubscription_PRE }}
    azureSubscription_PRD: ${{ variables.azureSubscription_PRD }}
```

The "deployments" section defines a list of deployment configurations that include the following properties: "zipFileName", "productName", "project", "buildConfiguration", "resourceGroupName", and "WebappName". Each configuration specifies the necessary details for deploying a project or application to a specific Azure resource group and web app instance.

```
deployments:
  - zipFileName: ${{ variables.zipFileName }}
    productName: ${{ variables.productName }}
    project: ${{ variables.project }}
    buildConfiguration: ${{ variables.buildConfiguration }}
    resourceGroupName: ${{ variables.resourceGroupName }}
    WebappName: ${{ variables.webAppName }}
```

2 visits in last 30 days

ST

Add a comment...