Activate Azure with DevOps

Module 06: End-to-End DevOps - Lab 03 - Multistage YAML Pipelines

Student Lab Manual

Parts of this lab has been taken from https://azuredevopslabs.com/labs/azuredevops/yaml/. View additional publicly available labs at https://azuredevopslabs.com/.

Contents

Lab 6.3.1: End-to-End DevOps: Deploy using Multi-Stage YAML

Introduction

In this lab, you will create a pipeline using YAML to build the code, run the unit tests, create the infrastructure in Azure and deploy the app.

You'll learn:

- The basic features of multistage YAML pipelines
- Deploying infrastructure as code
- Understand the value of pipelines as code

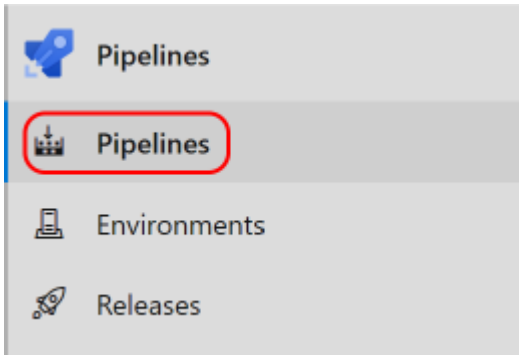
Prerequisites

- Microsoft Azure subscription https://azure.microsoft.com/
- Lab 6.2 YAML Pipelines
- **Note**: You may want to disable other CI pipelines before starting the lab to prevent triggering multiple builds at once and causing agent delays in the queue
- If you've set up any branch policies on the master branch from the earlier labs, you may also need to temporarily disable those

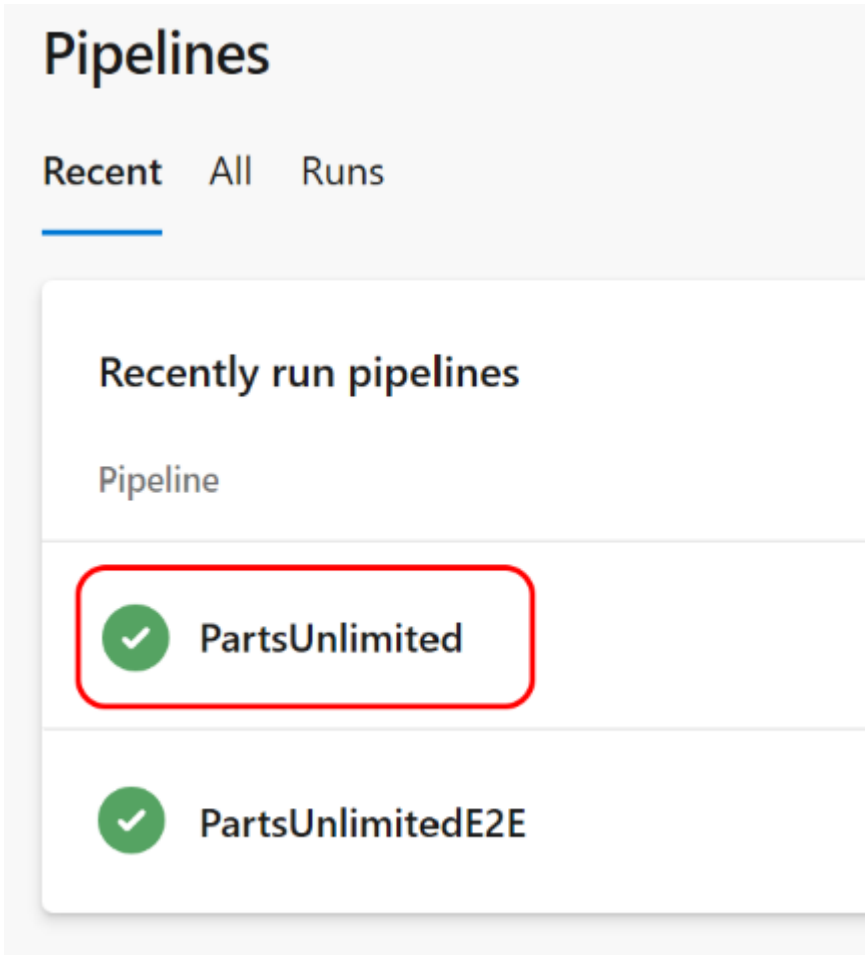
Estimated Time To Complete This Lab
60 minutes

Exercise 1: Add stages to YAML Pipelines

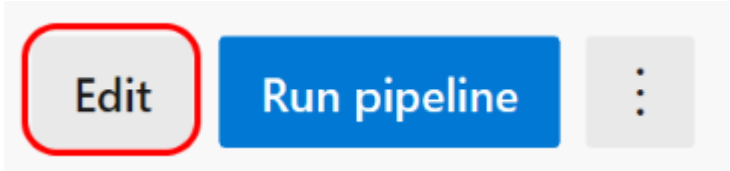
1. Navigate to the project used in the previous (YAML Pipeline) lab in Azure DevOps.
2. Navigate to **Pipelines**, then **Pipelines**.



3. Select and click on the YAML pipeline that we created in the build lab.

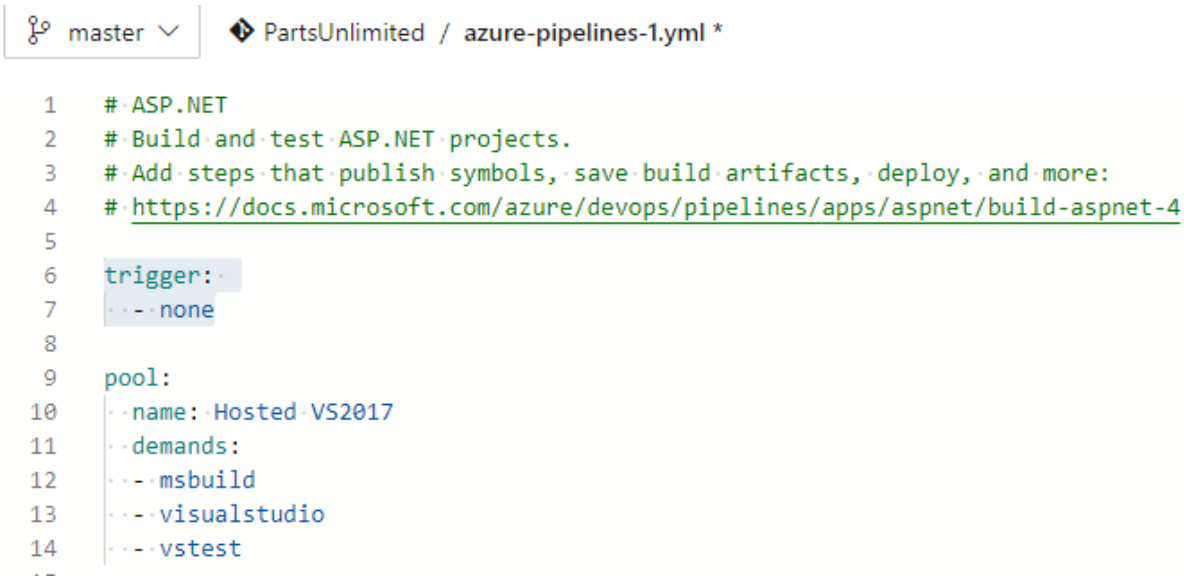


4. From the menu on the right, select Edit.



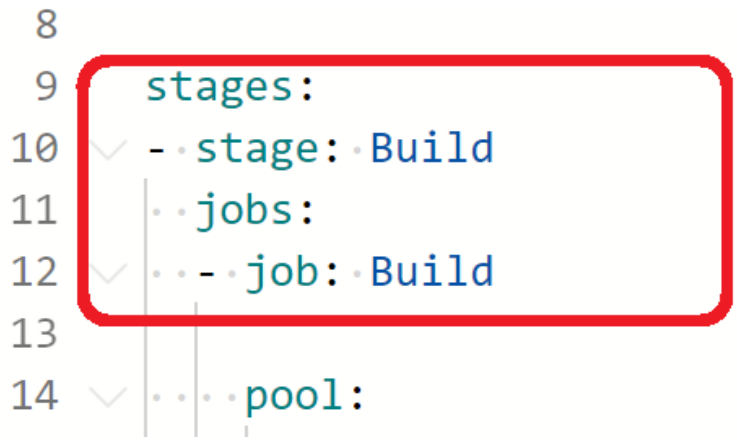
5. Now, first we will disable the Continuous Integration trigger so that saving the YAML file every time in this lab will not automatically trigger running of the pipeline. Change the **trigger** value to **none** like below:

```
trigger:
- none
```



6. We will add the stage configuration lines below after the **trigger** section to define a **Build** stage in the YAML pipeline. You can define any stages you need to better organize and track pipeline progress. It's important that everything lines up as shown in the screenshot below. Remove extra spaces if neccessary.

```
stages:
- stage: Build
  jobs:
  - job: Build
```



7. Highlight the remainder of the YAML file and indent it four spaces (press Tab button twice). This will simply take the existing build definition and relocate it as a child of the **jobs** node.

```
 9  stages:
10  - stage: Build
11    jobs:
12    - job: Build
13
14      pool:
15        name: Hosted VS2017
16        demands:
17        - msbuild
18        - visualstudio
19        - vstest
20
21      variables:
22        solution: '**/*.sln'
```

8. At the bottom of the file, add the configuration below to define a second stage.

```
- stage: Deploy
  jobs:
  - job: Deploy
    pool:
      name: Hosted VS2017
    steps:
```

Settings

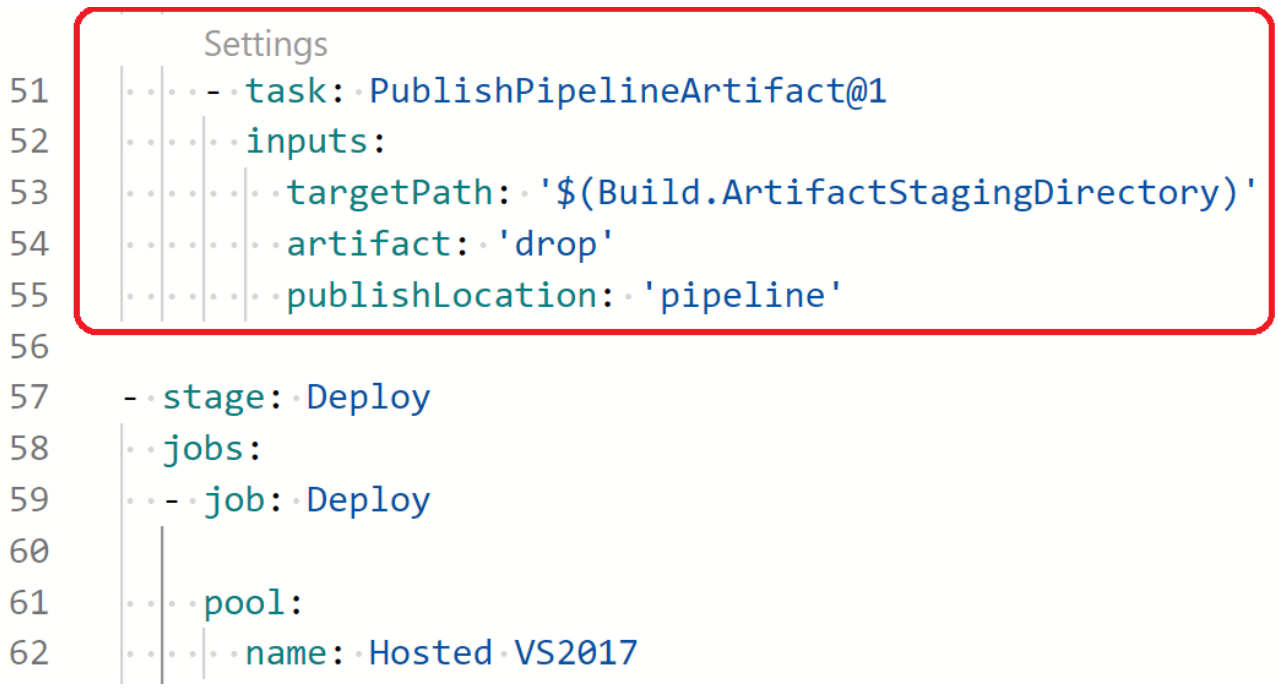
```
51      - task: PublishPipelineArtifact@1
52        inputs:
53          targetPath: '$(Build.ArtifactStagingDirectory)'
54          artifact: 'drop'
55          publishLocation: 'pipeline'
56
57  - stage: Deploy
58    jobs:
59    - job: Deploy
60      pool:
61        name: Hosted VS2017
62      steps:
```

9. In this exercise we have modified the existing YAML pipeline created in the previous lab to divide it into two stages: **Build** and **Deploy**. In the next exercise, we will start adding tasks and steps under the newly defined Deploy stage.

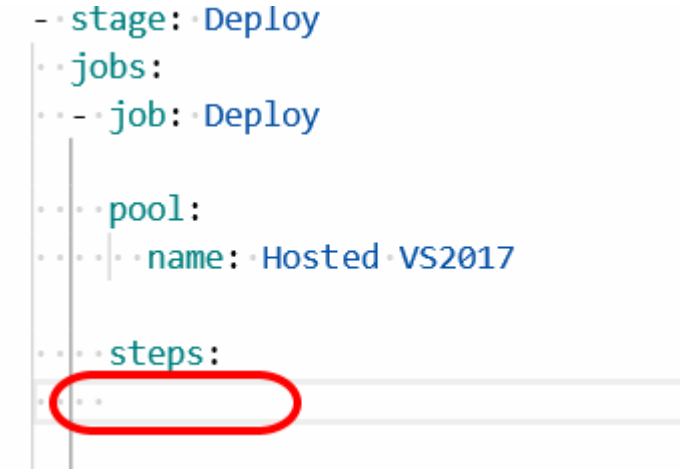
Exercise 2: Download pipeline artifacts for the Deploy stage

1. It's important to note that the two stages created in the previous exercise will run independently. As a result, the output from the first stage (Build) will not be available to the second stage (Deploy) without special consideration.

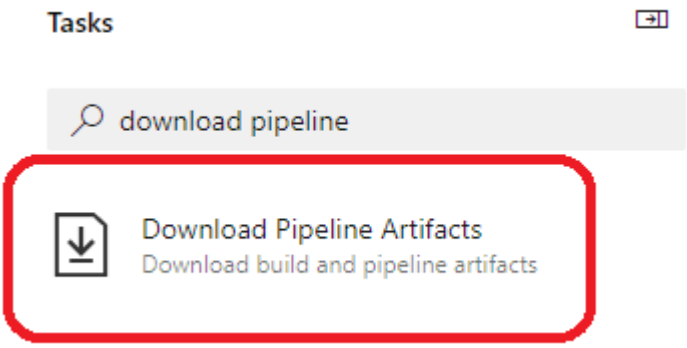
   For this, we should already have a task that publishes the pipeline artifacts at the end of the build stage from the previous lab. We need to add a another task to download these artifacts at the beginning of the deploy stage. Check your build stage to ensure you have a **Publish Pipeline Artifact** task at the end of the build stage.



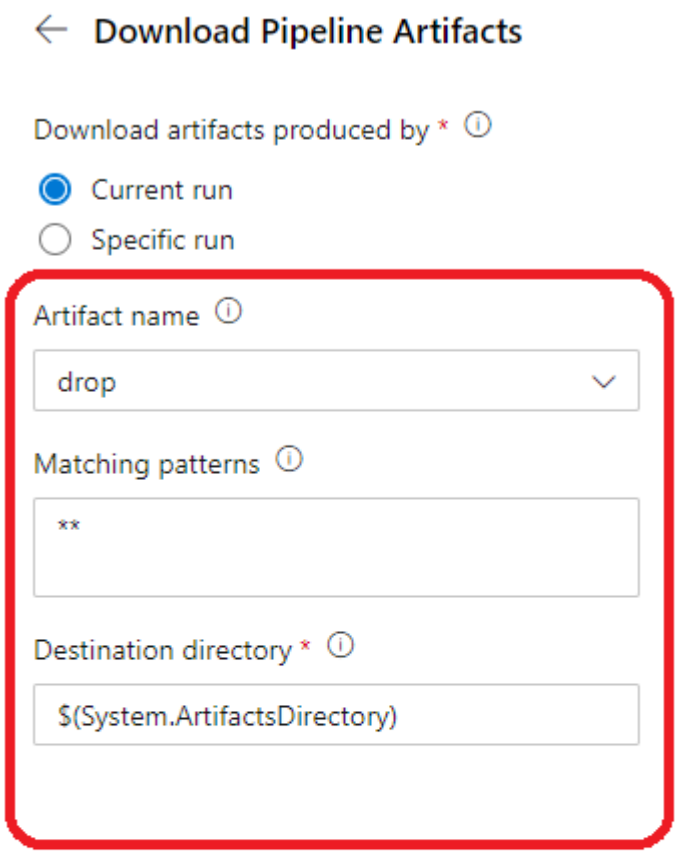
2. Place the cursor on the first line under the **steps** node of the deployment stage.



3. Search the tasks for **download pipeline** and select the **Download Pipeline Artifacts** task.



4. Set following values to the fields and then and click **Add**:
   Download artifacts produced by: Current run
   Artifact name: drop
   Matching patterns: **
   Destination directroy: $(System.ArtifactsDirectory)



5. Indent the download task four spaces (two tabs) if it isn't already. You may also want to add an empty line before and after to make it easier to read.

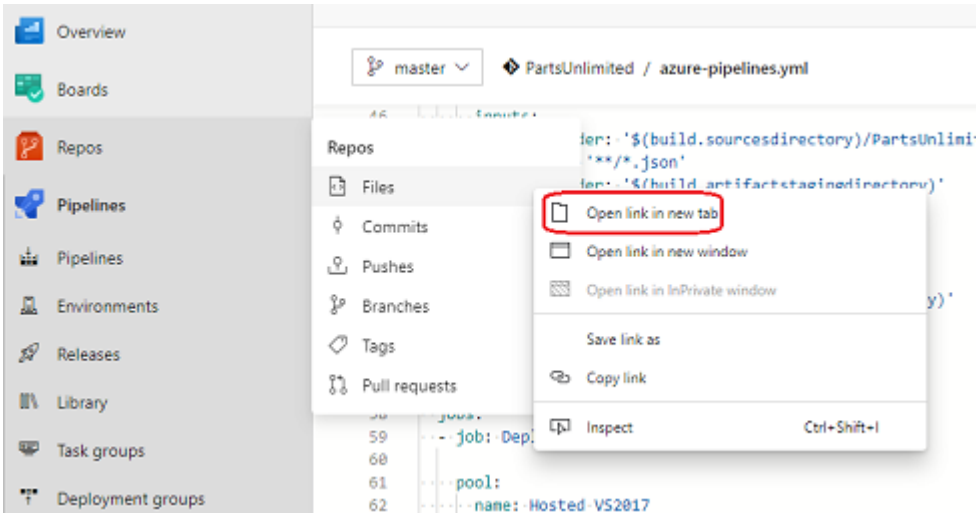
```
57    - stage: Deploy
58      jobs:
59      - job: Deploy
60        pool:
61          name: Hosted VS2017
62        steps:
          Settings
63        - task: DownloadPipelineArtifact@2
64          inputs:
65            buildType: 'current'
66            artifactName: 'drop'
67            targetPath: '$(System.ArtifactsDirectory)'
```

6. In the next exercise we will now create the Azure resources using the JSON files downloaded in this task.

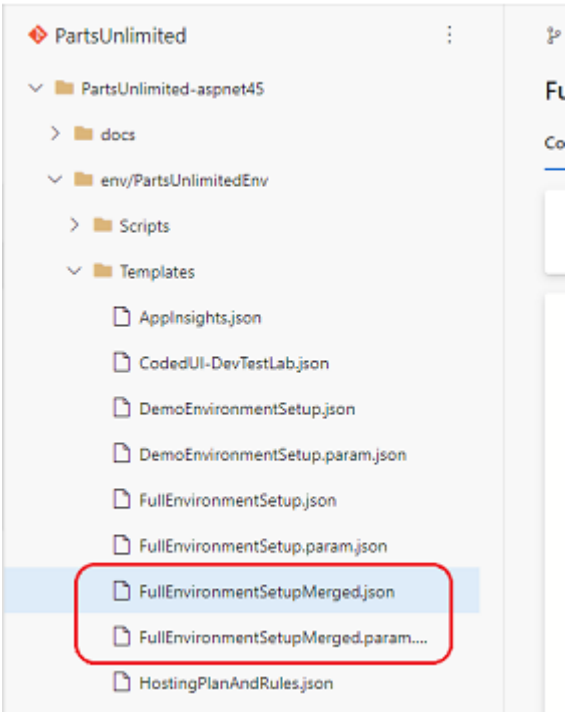
Exercise 3: Create Azure resources using YAML pipeline

1. In this exercise, we will create required Azure resources (Azure Web App, Azure SQL Database, etc.) by using the ARM Template available in the repository and published and downloaded in the previous exercises.

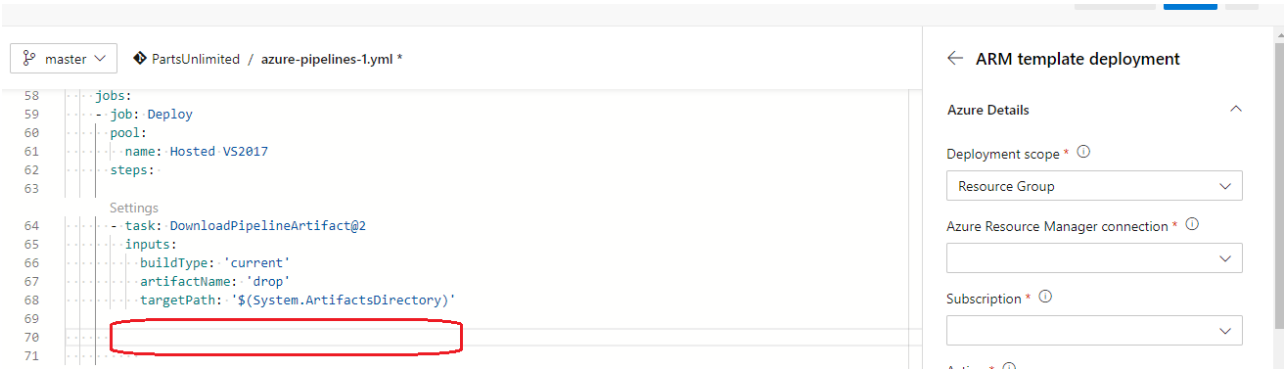
2. Move the mouse pointer on **Repos** and right-click on **Files**. Click on **Open link in new tab** to open repository in a new browser tab.



3. Navigate to **PartsUnlimited-aspnet45/env/PartsUnlimitedEnv/Templates** and locate **FullEnvironmentSetupMerged.json** and **FullEnvironmentSetupMerged.param.json**. These are the two files we will use to build our Azure resources. Review these files.

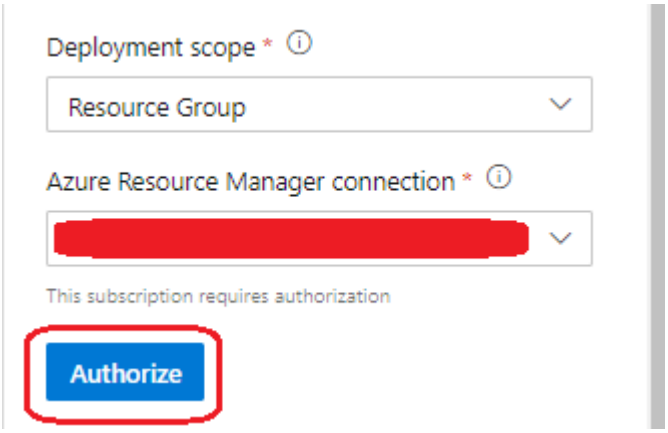


4. Switch back to the browser tab for editing Pipelines. Place the cursor on the next empty line in the YAML file. In the **Tasks**, search for **ARM template deployment** task and click on it.



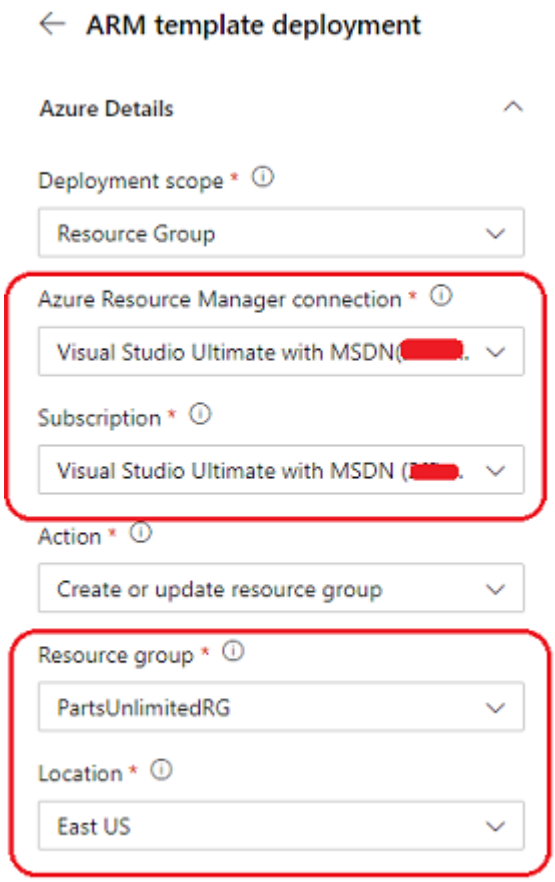
5. This is the task that will take the ARM templates mentioned above and deploy Azure resources using those templates. Enter following information in this task:

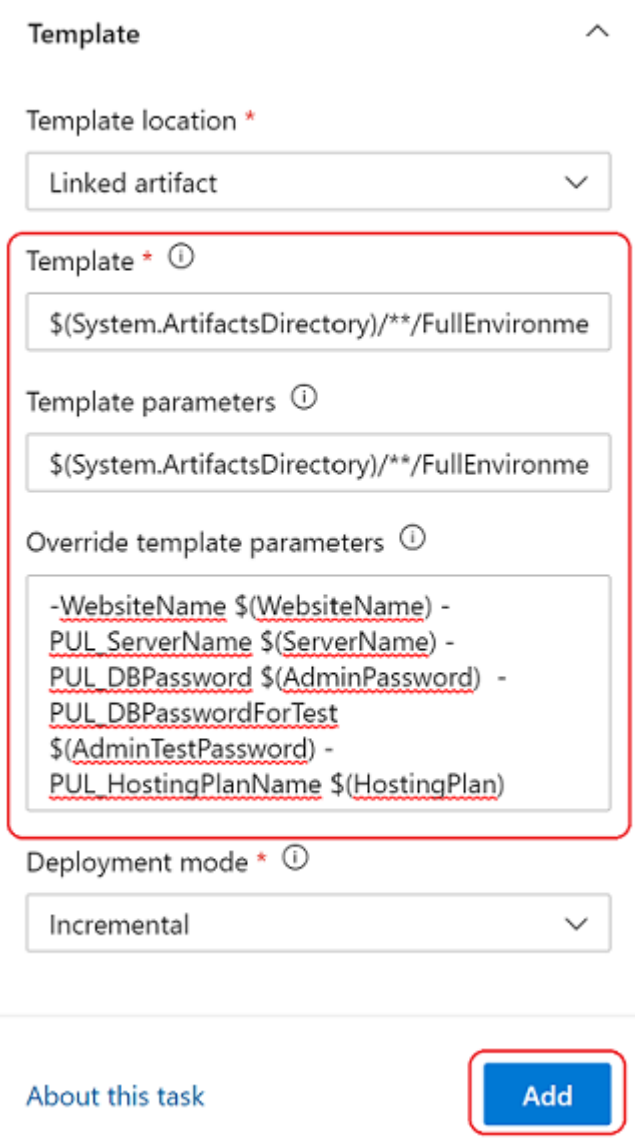
    - **Deployment scope:** Resource Group
    - **Azure Resource Manager Connection:** From the drop down, select the right subscription
      Note: If Authorization is requested, click **Authorize** and follow the path to complete authorization.



6. Once authorized successfully continue with following information:

    - **Subscription:** Select the correct subscription
    - **Action:** Create or update resource group
    - **Resource group:** PartsUnlimitedRG (This resource group will be created when the task runs)
    - **Location:** East US
    - **Template location:** Linked artifact
    - **Template:** $(System.ArtifactsDirectory)/**/FullEnvironmentSetupMerged.json
    - **Template parameters:** $(System.ArtifactsDirectory)/**/FullEnvironmentSetupMerged.param.json
    - **Override template parameters:** -WebsiteName $(WebsiteName) -PUL_ServerName $(ServerName) -PUL_DBPassword $(AdminPassword) -PUL_DBPasswordForTest $(AdminTestPassword) -PUL_HostingPlanName $(HostingPlan)
    - **Deployment mode:** Incremental
    - Click **Add**


ARM template deployment
Azure Details
Deployment scope *
Resource Group
Azure Resource Manager connection *
Visual Studio Ultimate with MSDN
Subscription *
Visual Studio Ultimate with MSDN
Action *
Create or update resource group
Resource group *
PartsUnlimitedRG
Location *
East US



7. Your YAML file should now looks like this:

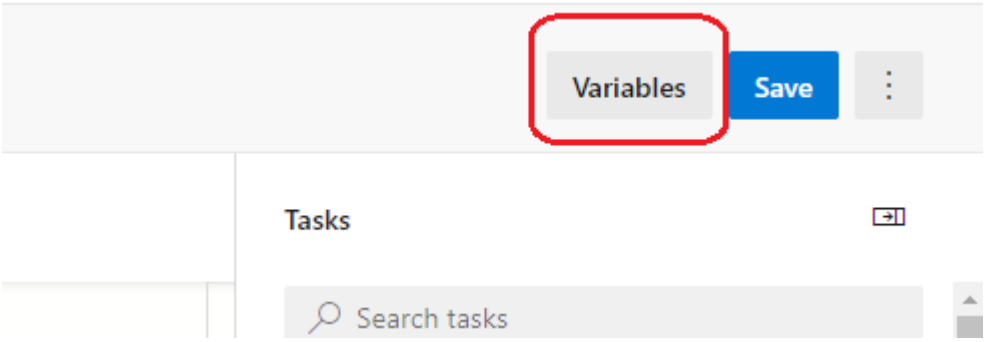
```yaml
- job: Deploy
  pool:
    name: Hosted VS2017
  steps:

  Settings
  - task: DownloadPipelineArtifact@2
    inputs:
      buildType: 'current'
      artifactName: 'drop'
      targetPath: '$(System.ArtifactsDirectory)'

  Settings
  - task: AzureResourceManagerTemplateDeployment@3
    inputs:
      deploymentScope: 'Resource Group'
      azureResourceManagerConnection: 'Visual Studio Ultimate with MSDN([redacted])'
      subscriptionId: '[redacted]'
      action: 'Create Or Update Resource Group'
      resourceGroupName: 'PartsUnlimitedRG'
      location: 'East US'
      templateLocation: 'Linked artifact'
      csmFile: '$(System.ArtifactsDirectory)/**/FullEnvironmentSetupMerged.json'
      csmParametersFile: '$(System.ArtifactsDirectory)/**/FullEnvironmentSetupMerged.param.json'
      overrideParameters: '-WebsiteName $(WebsiteName) -PUL_ServerName $(ServerName) -PUL_DBPassword $(AdminPa
      deploymentMode: 'Incremental'
```

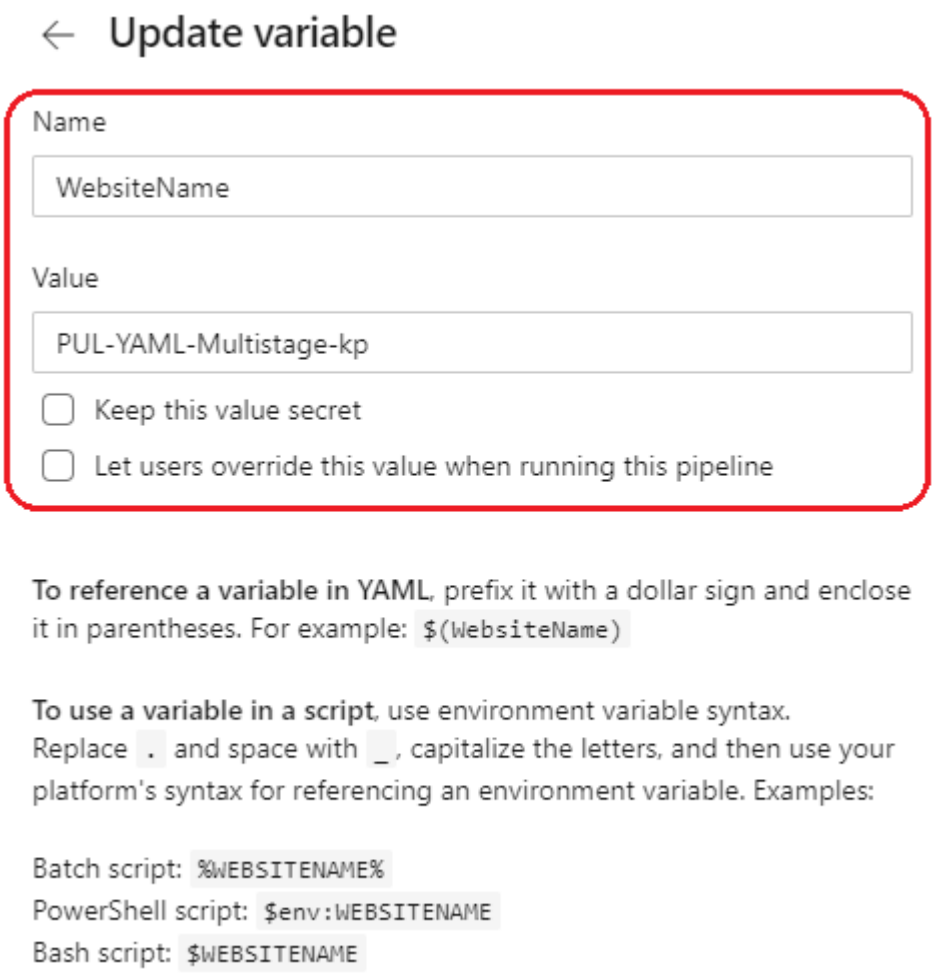
Note: In case your sections are 'overindented' then you can select that entire section and press **Shift + Tab** twice to unindent the selected section.

8. We need to define the variables that we used in the previous step for overriding the template parameters. Click on **Variables** at the top-right of the screen.



9. Click on **New Variable** to add New Variable. Add **Name** as WebsiteName and **Value** as the name you intend to use for your website.

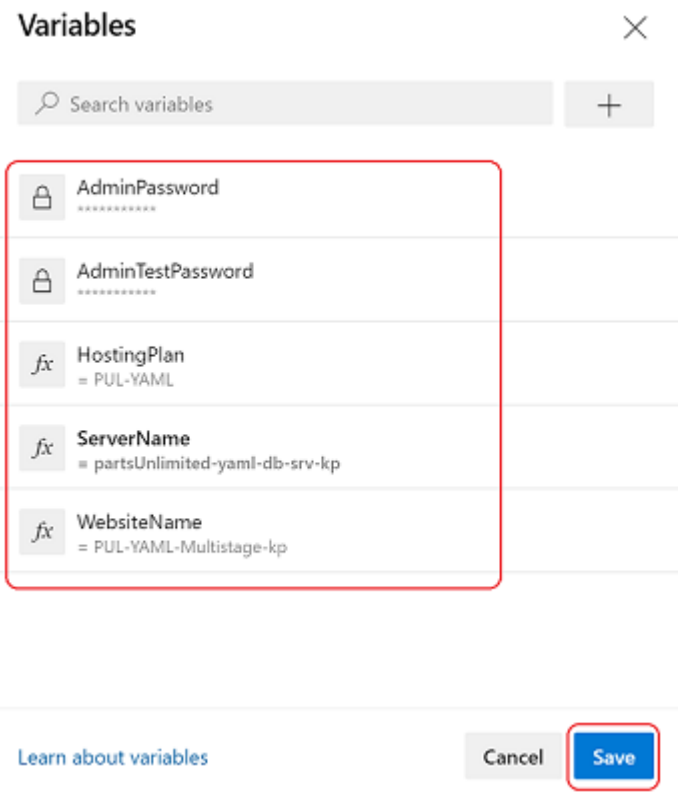
Note: Remember, the website name needs to be unique across all of Azure. You can achieve this by adding your initials at the end of the name.



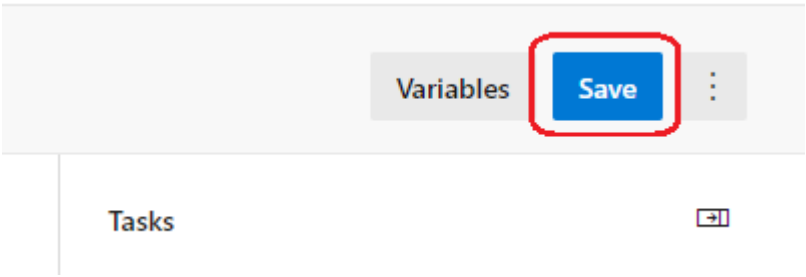
10. Now we will define all the remaining variables mentioned in **Override template parameters** section above:

- **ServerName:** partsUnlimited-yaml-db-srv-[your initials] (Server name has to be unique across all of Azure. Add your initials in the server name to make it unique)
- **AdminPassword:** Pa$$w0rd (You can use any password you want and also check the box for **Keep this value secret**)
- **AdminTestPassword:** Pa$$w0rd (You can use any password you want and also check the box for **Keep this value secret**)
- **HostingPlan:** PUL-YAML
- Click **Save**

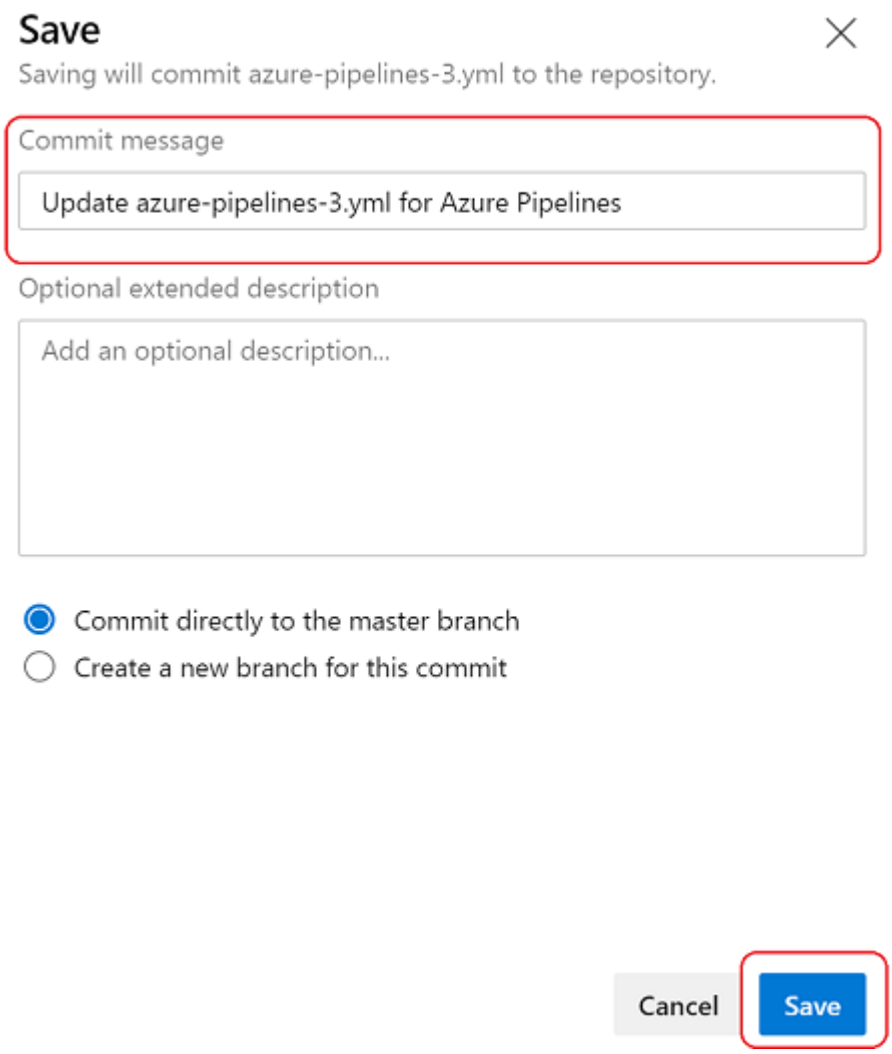
```
Note: If you decide to use some other password of your choice please make
sure if meets following requirements:
    * Minimum of 8 characters
    * Requires 3 out of 4 of the following:
        ** Lowercase characters
        ** Uppercase characters
        ** Numbers (0-9)
        ** Symbols
    * Cannot contain the login name
```



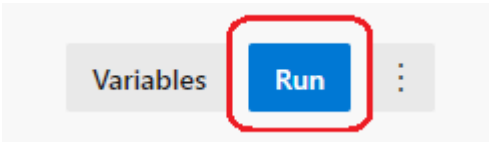
11. At this point it'd be best to check if the above task creates the environment in Azure as we expect. We will save and run the pipeline before continuing to the next exercise. Click on **Save** at the top-right.

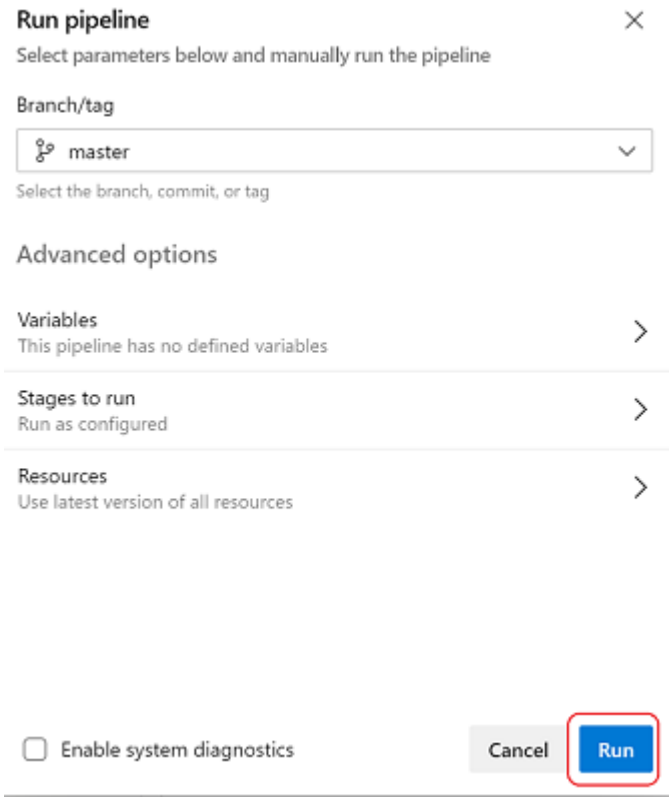


12. Add a commit message and click **Save**.



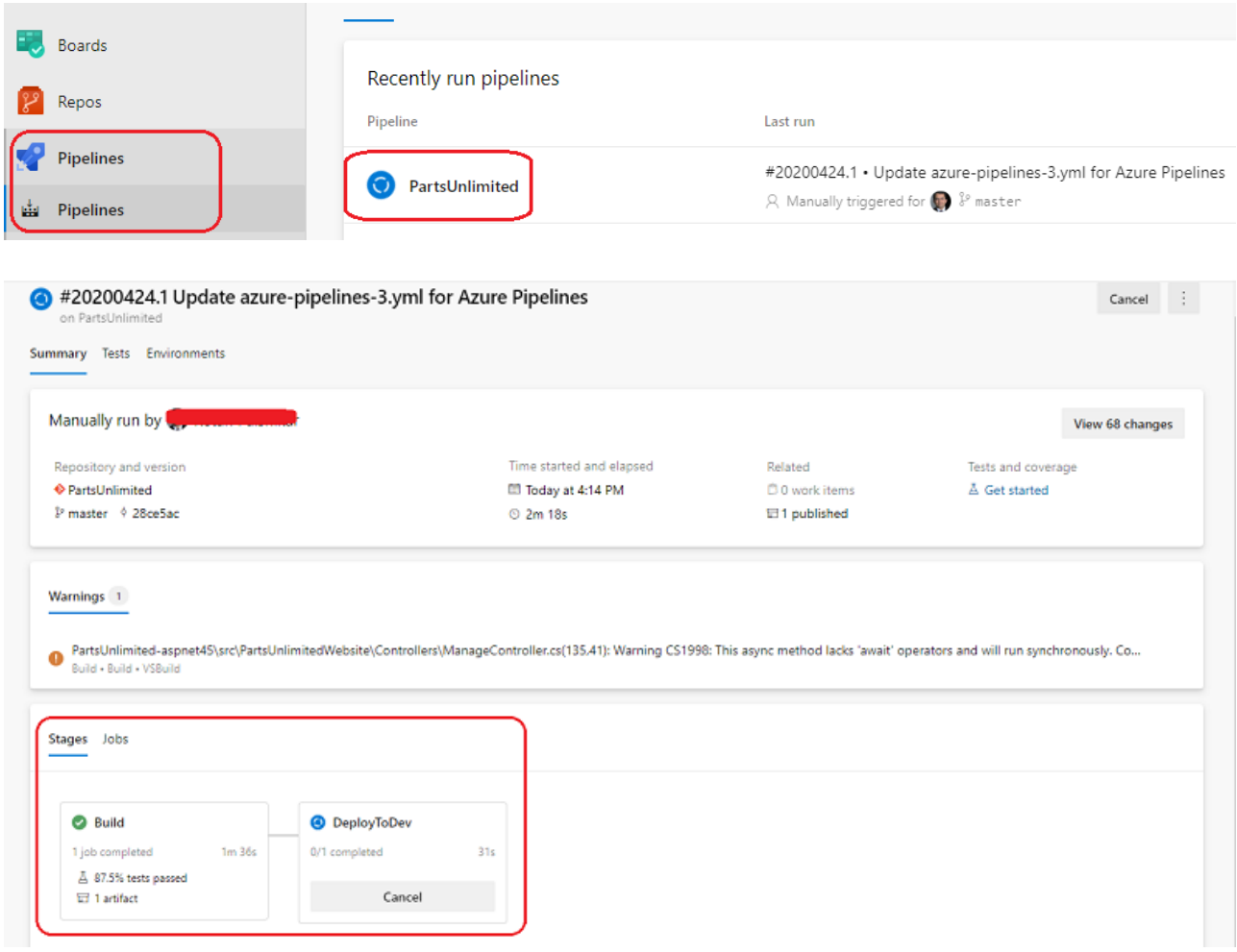
13. Click on **Run** at the top-right corner and click **Run** again without making any changes.





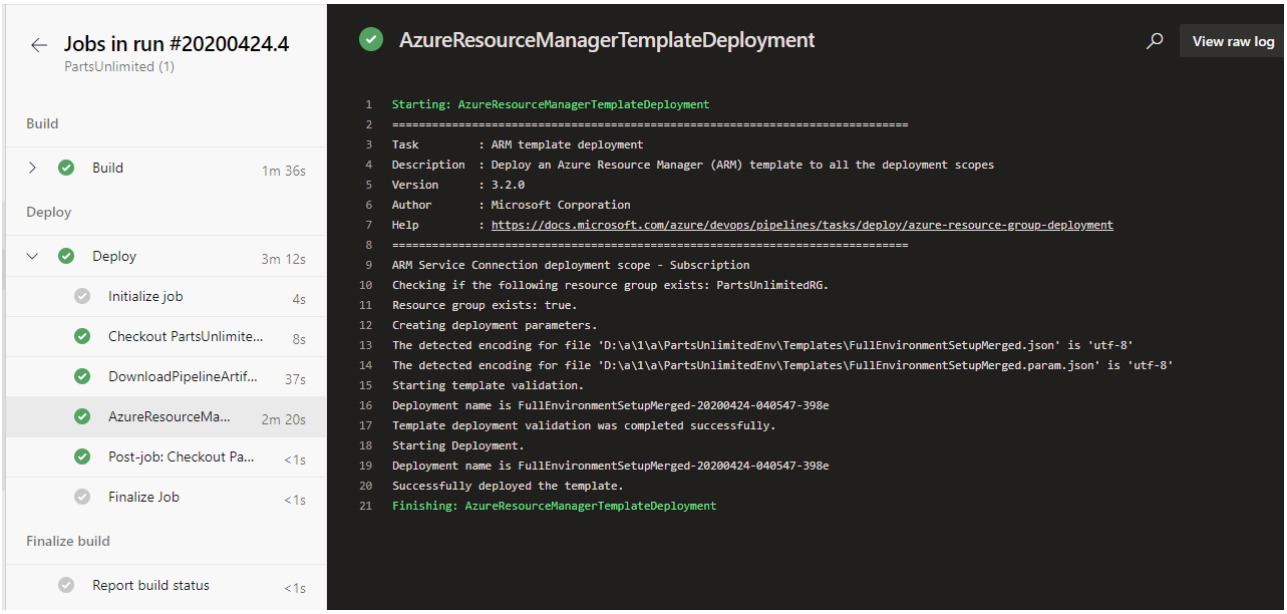
14. Click on **Pipelines** and view the recent run triggered. Click on the run to view more details.

    Note: As the Build stage finishes and Deploy stage starts, you maybe asked to **View** and provide permissions to Azure resources. Click on **Permit**.

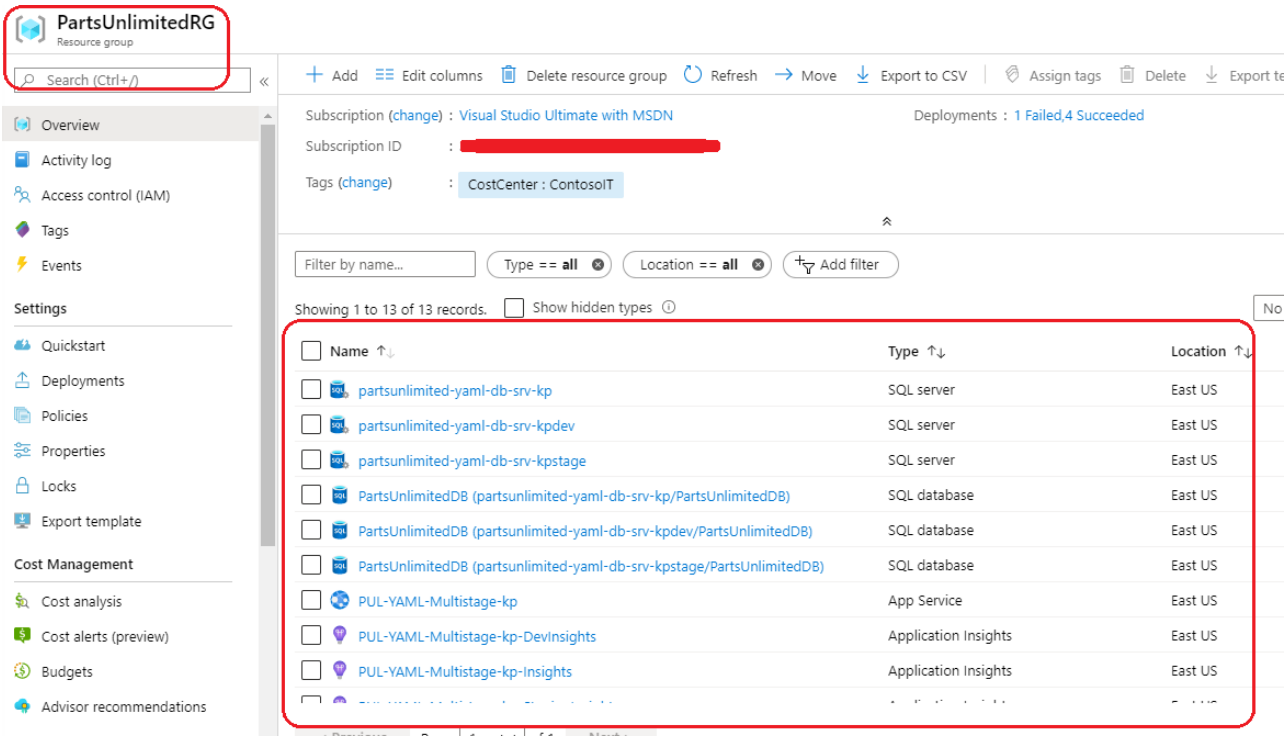


Note: You can also see the two stages (Build and Deploy) of the pipeline here.

15. Once the pipeline run is finished, you can see the summary like this:



16. Go to **Azure Portal** and navigate to the PartsUnlimitedRG that got created. It should list all the resources the deployment created:

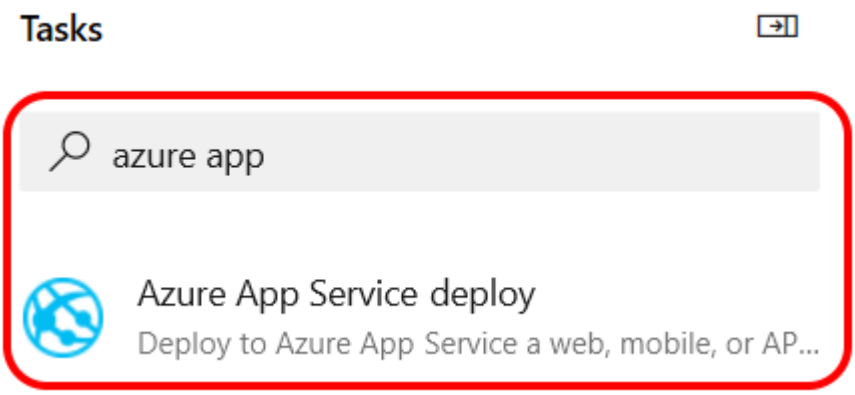


Exercise 4: Add task in the YAML pipeline to deploy the Web App

1. Now that the environment is created in Azure, we will continue with adding the last task in the YAML pipeline to deploy the web app. Navigate back to **Azure DevOps** and to the project. Click on **Pipelines**, select the pipeline you were working on in the previous exercise. Click on **Edit Pipeline**.

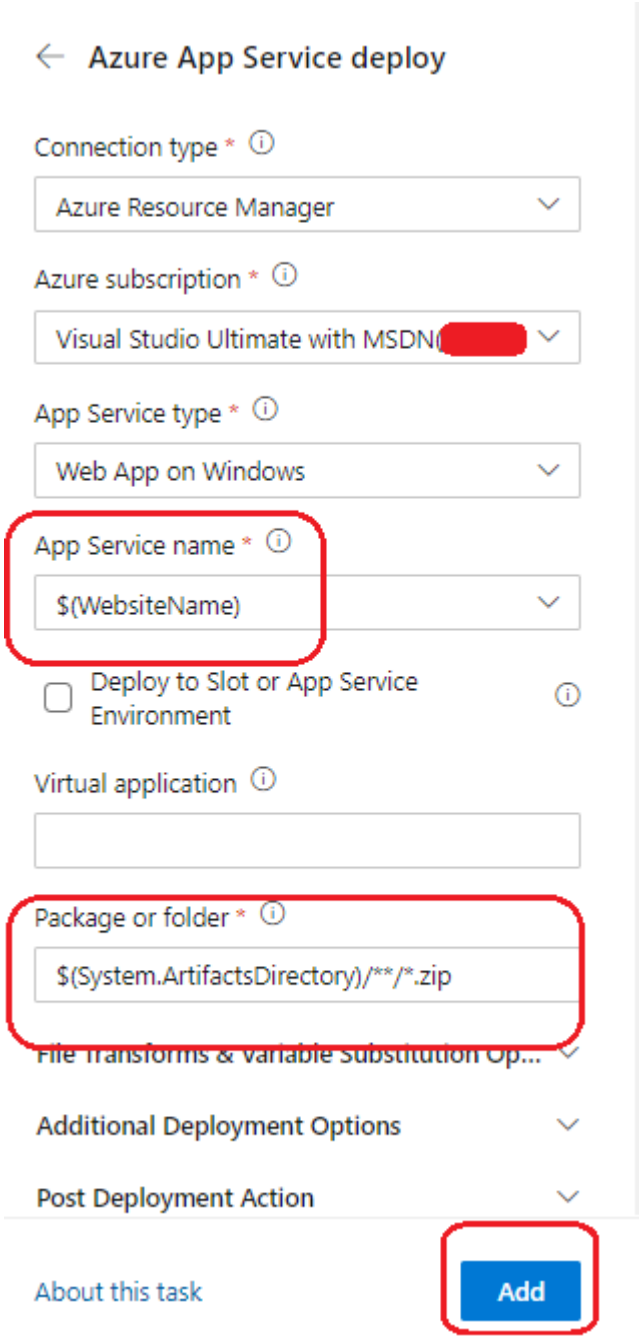
2. Set the cursor on a new line at the end of the YAML definition. This will be the location where new tasks are added.

3. Search the tasks for **"Azure App"** and select the **Azure App Service Deploy** task.



4. Add following details in the task:

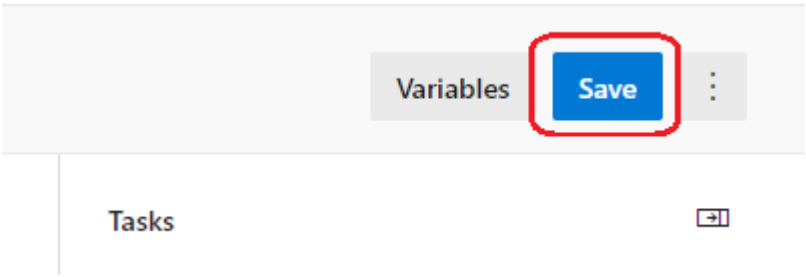
    - **Azure subscription:** Select the subscription used in the previous exercise
    - **App Service name:** $(WebsiteName)
    - **Package or folder:** $(System.ArtifactsDirectory)/**/*.zip
    - Click **Add**



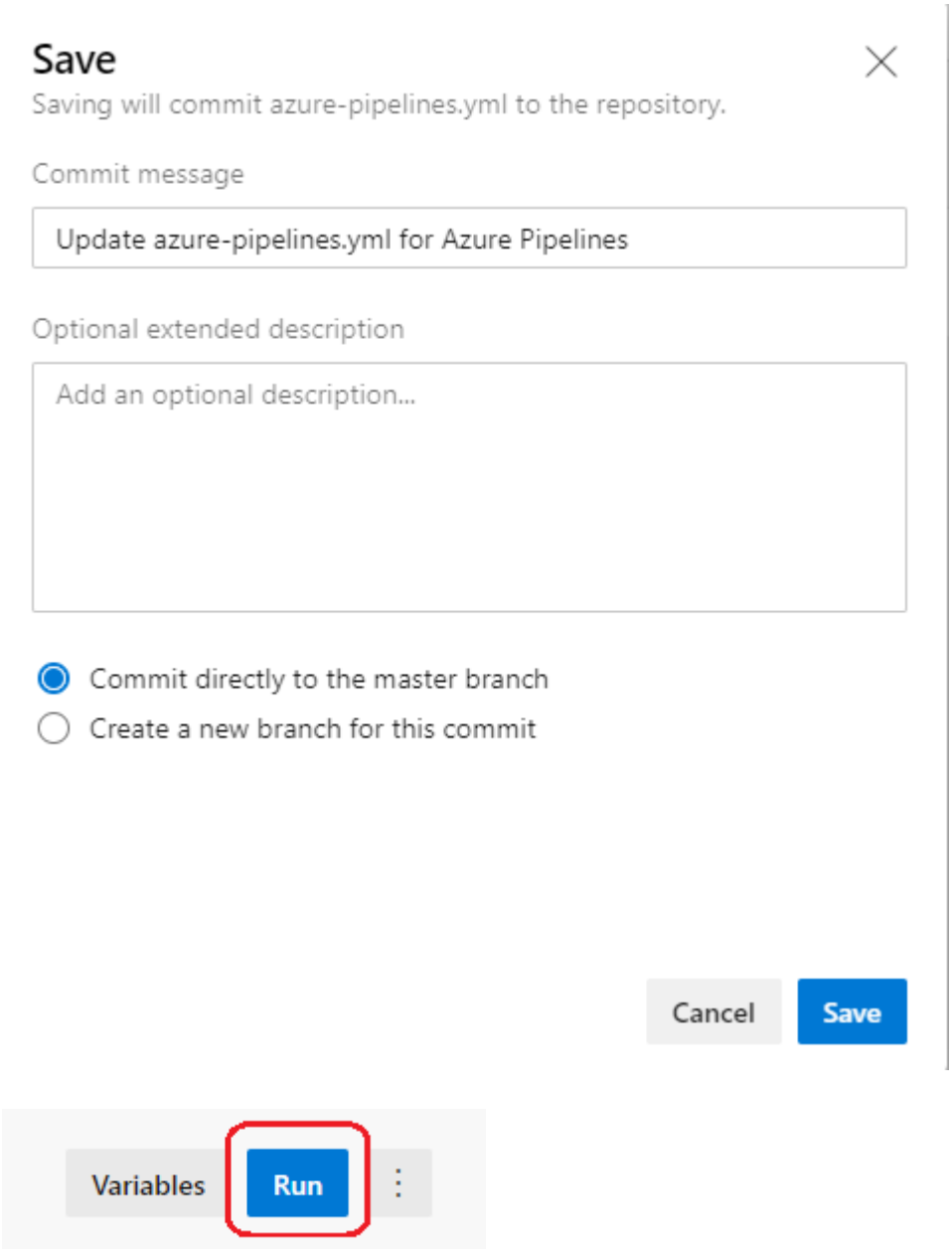
5. The YAML that defines the task will be added to the cursor location in the file. Ensure it is indented and thus a child of the **steps** task. If not, indent it four spaces (two tabs).

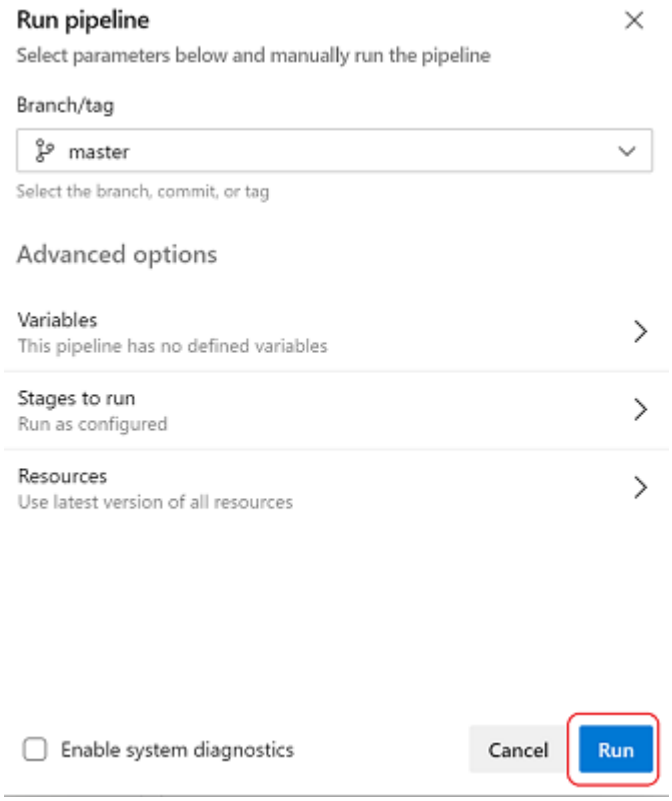
```
Settings
    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'Visual Studio Ultimate with MSDN([redacted])'
        appType: 'webApp'
        WebAppName: '$(WebsiteName)'
        packageForLinux: '$(System.ArtifactsDirectory)/**/*.zip'
```

6. Click **Save** to commit the changes.

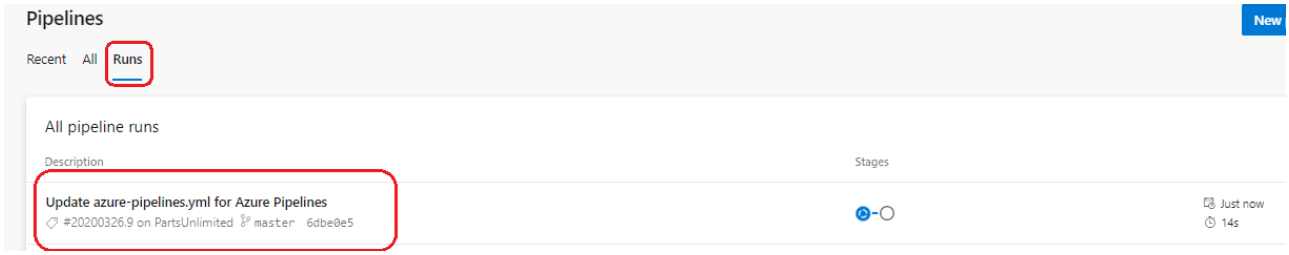


7. Confirm the **Save** and **Run** the pipeline again.

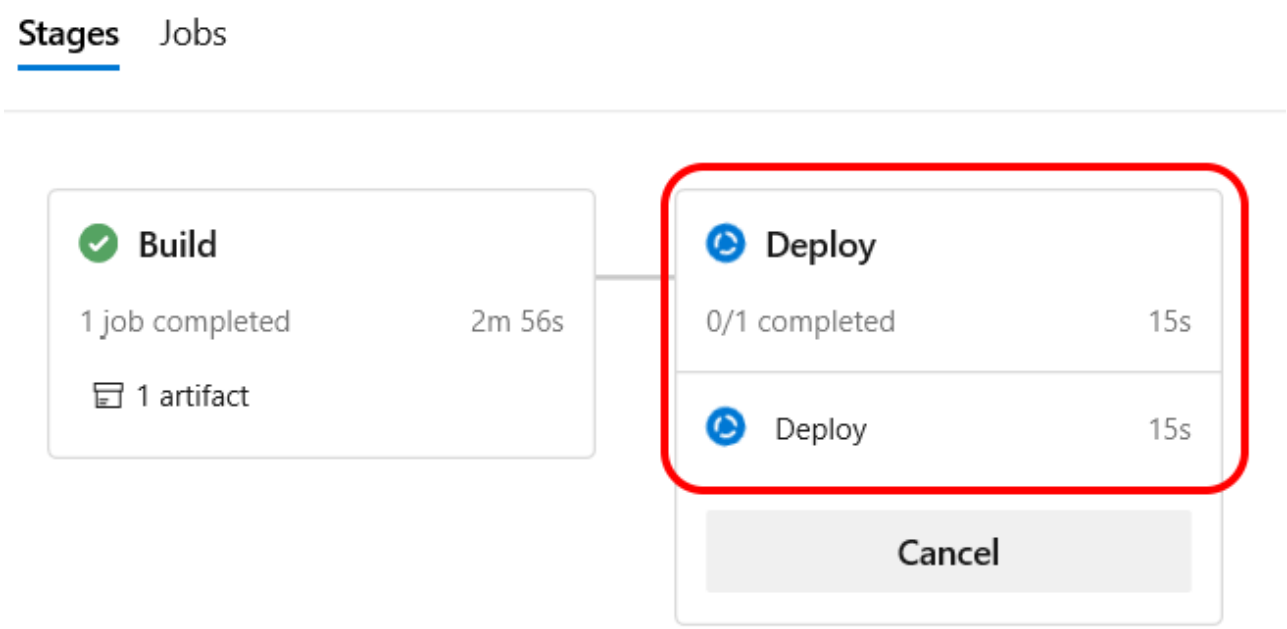




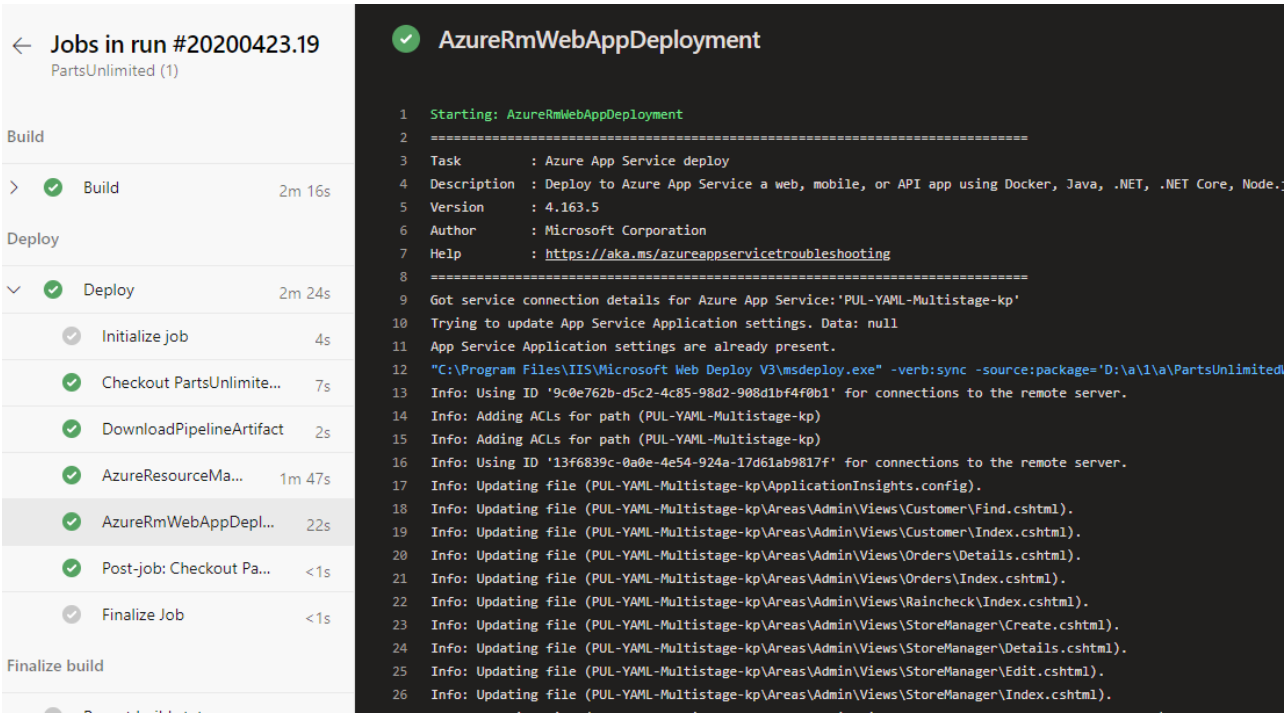
8. Return to the **Pipelines** view and from the **Runs** tab, click the new run to open it.



9. When the Build stage completes, click the **Deploy** stage to follow each task.



10. Expand the **AzureRmWebAppDeployment** task to review the steps performed during the Azure deployment. Once the task completes, your app will be live on Azure.

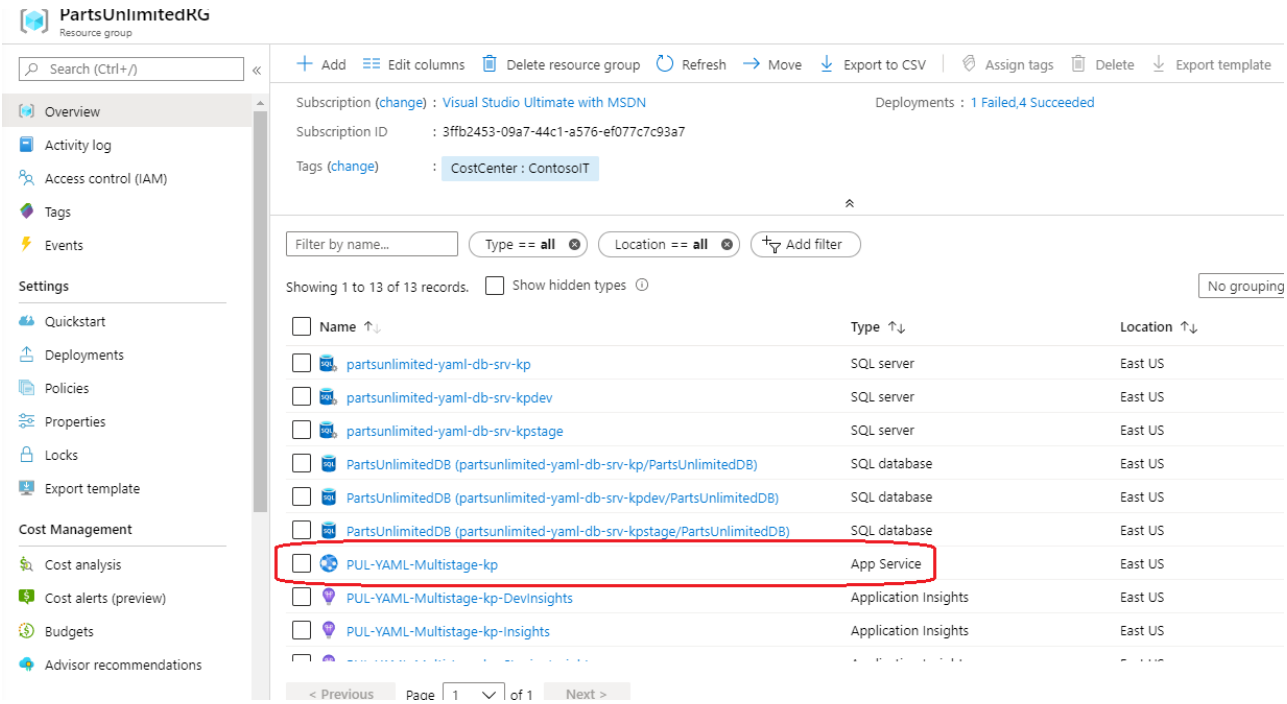


Exercise 5: Review the deployed site

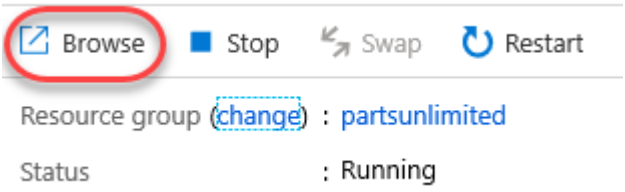
1. Return to the Azure portal browser tab.

2. Navigate to the **PartsUnlimitedRG** resource group.

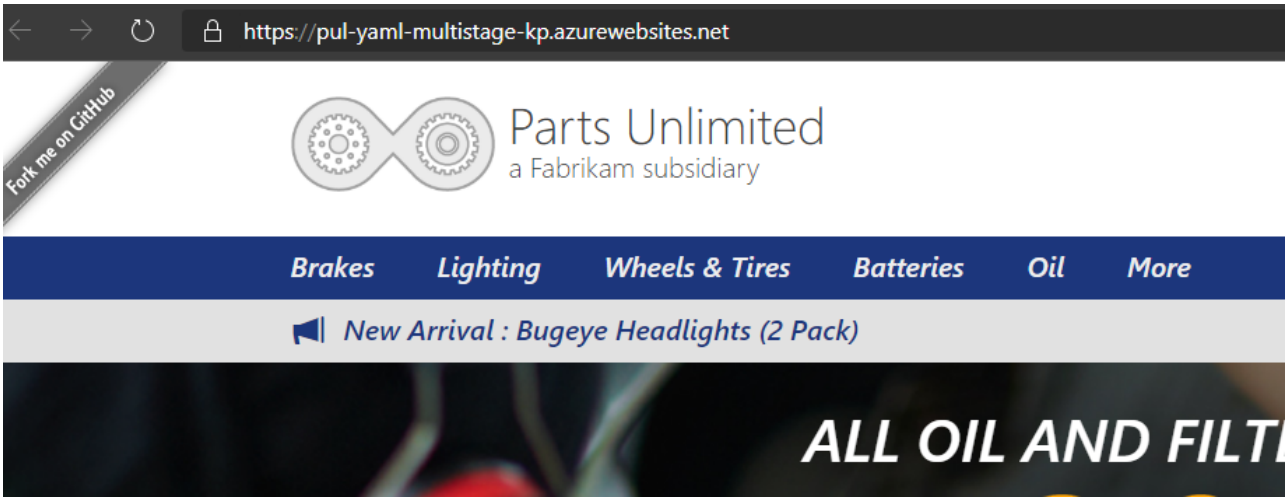
3. Click on **PUL-YAML-Multistage-kp** App Service



4. Click **Browse** to open your site in a new tab.



5. The deployed site should load as below.



Note: Back in the **PartsUnlimitedRG** resource group, you might have noticed other App Services (Slot) and database servers with "dev" and "stage" suffixes. These can be used to deploy the application in mutiple phases/rings where deployment first goes to Dev and then to Stage and then finally to the Production environment. We will explore this in the next lab.