

Activate Azure with DevOps

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

Student Lab Manual

Conditions and Terms of Use

Microsoft Confidential - For Internal Use Only

This training package is proprietary and confidential and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

© 2020 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at <https://www.microsoft.com/en-us/legal/intellectualproperty/permissions/default>

Microsoft®, Internet Explorer®, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

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

Contents

Introduction

Prerequisites

Exercise 1: Split the Deploy stage into multiple stages

Exercise 2: Configure the Dev and Staging deployment stage

Exercise 3: Configure the Production deployment stage

Exercise 4: Review environments and setup Approval Exercise 5: Deploy the Web App to Multiple

Environments Exercise 6: Set the Approval on the Service Connection

Lab 6.3.2: End-to-End DevOps: Deploy using Multi-Stage YAML with Approvals

Introduction

In this lab, you will continue working with previous lab and modify the YAML pipeline to create environments and approvals. You will create approvals for both the stages in your pipeline as well as for the service connection.

You'll learn:

- Introducing environments in your YAML pipeline
- Implementing approvals in the deployment stages
- Implementing approval for the service connection

Prerequisites

- Microsoft Azure subscription <https://azure.microsoft.com/>
- Lab 6.3.1 - Previous Multistage YAML Pipelines lab

Estimated Time To Complete This Lab

90 minutes

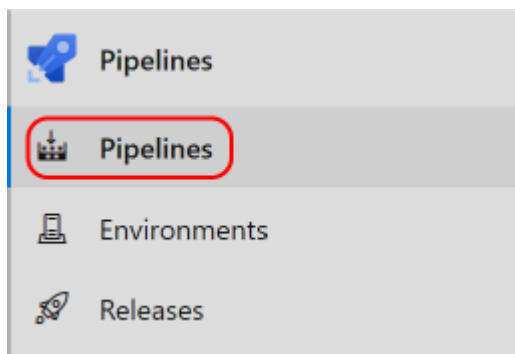
Exercise 1: Split the Deploy stage into multiple stages

1. In this exercise, we will divide the "Deploy" stage created in the previous lab into multiple stages. The idea is to create three separate stages:

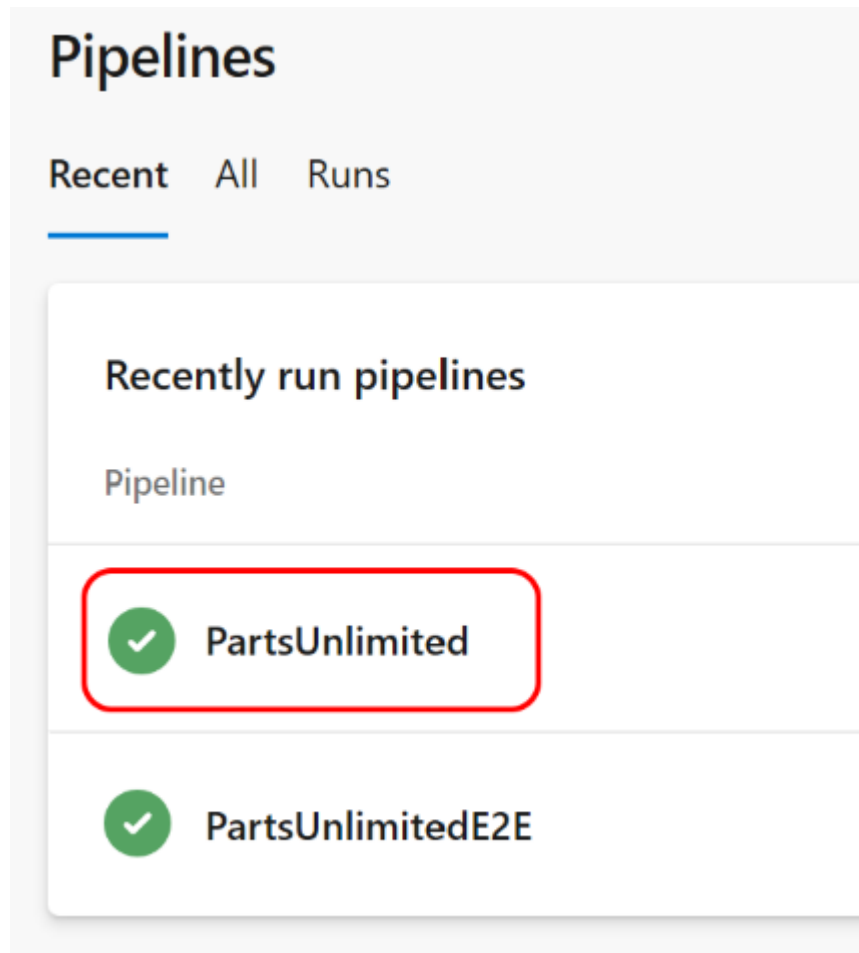
- **DeployAzureResources** - To deploy the ARM Template to create Azure resources in the Azure Portal
- **DeployToDevAndStaging** - To deploy the web application into Dev and Staging environments
- **DeployToProd** - To deploy the web app into the Production environment

The above is what we will use in this lab. This is just one example. Another way to split the deploy stage could be: ARM deployment, Dev deployment, Staging deployment and Production deployment. Or combine all into a single Deploy stage. This is a design approach for YAML that comes with pros and cons for each approach. We get greater control at the stage level when the pipeline runs. For example, we can skip or cancel the deployment to one of the environments without impacting other environments if they are split into separate stages.

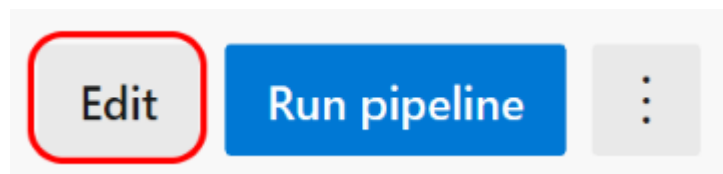
1. Navigate to [Azure DevOps](#). Navigate to the project we used in the last lab.
2. Navigate to **Pipelines**, then **Pipelines**.



3. Select and click on the YAML pipeline that we worked on in the previous lab.



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



5. Navigate down to the Deploy stage of the YAML file. For convenience, you can **collapse** the Build stage so that only Deploy stage will be visible. This is the stage we will be working on in this lab.

```

9    stages:
10 > - stage: Build ...
56
57   - stage: DeployAzureResources
58     jobs:
59     - job: DeployARMTemplate
60
61     pool:
62     - name: Hosted VS2017
63

```

6. Replace the stage name from Deploy to **DeployAzureResources** and job name to **DeployARMTemplate**

```

- stage: DeployAzureResources
  jobs:
  - job: DeployARMTemplate

```

```

8
9    stages:
10 > - stage: Build ...
55
56   - stage: DeployAzureResources
57     jobs:
58     - job: DeployARMTemplate
59     pool:
60     - name: Hosted VS2017
61     steps:
62       Settings
63       - task: DownloadPipelineArtifact@2
64       inputs:

```

7. We will also add **displayName** lines for both the stage and the job to add friendly name to this stage and job. Similar *displayName* can be used for *stages*, *jobs*, *deployment jobs*, and *tasks* to make them

more readable when the pipeline runs.

```
- stage: DeployAzureResources
  displayName: Deploy Azure Resources
  jobs:
  - job: DeployARMTemplate
    displayName: Deploy ARM Template
```

```
57 ✓ - stage: DeployAzureResources
58   · displayName: Deploy Azure Resources ·
59   · jobs:
60 ✓   · - job: DeployARMTemplate
61     ·   displayName: Deploy ARM Template
62
63 ✓   · pool:
64     ·   name: Hosted VS2017
```

8. We have successfully created the first stage of our deployment. Now, we will add **DeployToDevAndStaging** stage just before the Web App deployment task.

9. Place the cursor on the next line after the *AzureResourceManagerTemplateDeployment@3* is finished and before the *AzureRmWebAppDeployment@4* task begins.

```
84   ······ overrideParameters: '-WebsiteName $(WebsiteName)'
85   ······ deploymentMode: 'Incremental'
86
87
88
      Settings
89 ✓   ··· - task: AzureRmWebAppDeployment@4
90 ✓   ··· inputs:
91   ······ ConnectionType: 'AzureRM'
```

10. Here we will add the second stage **DeployToDevAndStaging** like below:

```
- stage: DeployToDevAndStaging
  displayName: Deploy to Dev and Staging Environments
  jobs:
  - deployment: DeployToDev
```

```

      displayName: Deploy To Dev Environment

    pool:
      name: Hosted VS2017

85 | | | | | deploymentMode: 'Incremental'
86 |
87 | - stage: DeployToDevAndStaging
88 |   displayName: Deploy to Dev and Staging Environments
89 |   jobs:
90 |   - deployment: DeployToDev
91 |     displayName: Deploy To Dev Environment
92 |
93 |   pool:
94 |     name: Hosted VS2017
95 |
96 |   - task: AzureRmWebAppDeployment@4
97 |     inputs:

```

Note: At this point the YAML file might show syntax error. Ignore this for now. We will fix it in the next exercise.

Note: Unlike the previous stage, instead of using *job* here, we are using *deployment* under Jobs. A [deployment job](#) is a special type of job tailored to an environment.

11. Similarly, we will create the last stage **DeployToProd** like below. Place this at the end of the YAML file after the web app deploy task.

```

- stage: DeployToProd
  displayName: Deploy to Production
  jobs:
    - deployment: DeployToProd
      displayName: Deploy To Prod Environment

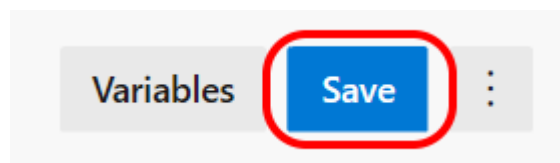
    pool:
      name: Hosted VS2017

```

```
102 |.....packageForLinux: '$(System.ArtifactsDirectory)/**/*.zip'
103 |
104 | - stage: DeployToProd
105 |   displayName: Deploy to Production
106 |   jobs:
107 |     - deployment: DeployToProd
108 |       displayName: Deploy To Prod Environment
109 |
110 |   pool:
111 |     name: Hosted VS2017
```

12. Click **Save** at the top-right to save the changes made so far.

Note: The YAML file, at this point, is incomplete and the pipeline cannot run. However, we have set **trigger: none** at the beginning of the YAML file in the previous lab. This will ensure that saving the modifications in the YAML file does not trigger Continuous Integration.



Exercise 2: Configure the Dev and Staging deployment stage

1. In the previous exercise, we fully configured the first deploy stage (DeployAzureResources). However, the second deploy stage (DeployToDevAndStaging) is incomplete and we need to modify it in this exercise.
2. In this stage, we will create Dev and Staging [environments](#). We will also add tasks to deploy the Web App in the Dev and Staging slots of the Azure App Service.
3. Within the DeployToDevAndStaging Stage, place the cursor as shown below between **name: Hosted VS2017** and **- task: AzureRmWebAppDeployment@4** lines. Indent it inline with *Pool*:

```

87 - stage: DeployToDevAndStaging
88   displayName: Deploy to Dev and Staging Environments
89   jobs:
90     - deployment: DeployToDev
91       displayName: Deploy To Dev Environment
92     pool:
93       name: Hosted VS2017
94     |
95     - task: AzureRmWebAppDeployment@4
96
97
98
```

4. Add following snippet and indent it as shown below. This will create the **PUL-Dev** environment.

Note: We are using the simplest deployment strategy of runOnce. You can find additional details about the deployment strategies [here](#).

```

environment: 'PUL-Dev'
strategy:
  runOnce:
    deploy:
      steps:

```

```

91     ... displayName: Deploy To Dev Environment
92     .
93     ... pool:
94     ... | name: Hosted VS2017
95
96     ... environment: 'PUL-Dev'
97     ... strategy:
98     ... | runOnce:
99     ... | | deploy:
100    ... | | | steps:
101
102    ... - task: AzureRmWebAppDeployment@4
103        ... | inputs:

```

5. Indent the **AzureRmWebAppDeployment@4** task appropriately by pressing Tab until it aligns with **steps**. The stage should look like this:

```

87 - stage: DeployToDevAndStaging
88   displayName: Deploy to Dev and Staging Environments
89   jobs:
90   - deployment: DeployToDev
91     displayName: Deploy To Dev Environment
92     .
93     pool:
94     | name: Hosted VS2017
95
96     environment: 'PUL-Dev'
97     strategy:
98     | runOnce:
99     | | deploy:
100    | | | steps:
101
102    | Settings
103    | - task: AzureRmWebAppDeployment@4
104    |   inputs:
105    |     | ConnectionType: 'AzureRM'
106    |     | azureSubscription: 'Microsoft Azure Internal Consumption'
107    |     | appType: 'webApp'
108    |     | WebAppName: '$(WebsiteName)'
109    |     | packageForLinux: '$(System.ArtifactsDirectory)/**/*.zip'

```

6. Since this is the Dev stage, we want the Web App to be deployed to the Dev slot of the Azure App Service. Modify the **AzureRmWebAppDeployment@4** task to add the following lines after

WebAppName and before **packageForLinux**:

```

deployToSlotOrASE: true
ResourceGroupName: 'PartsUnlimitedRG'
SlotName: 'Dev'

```

```

96     ...environment: 'PUL-Dev'
97     ...strategy:
98     ...    runOnce:
99     ...    deploy:
100    ...    steps:
101
102    Settings
103    - task: AzureRmWebAppDeployment@4
104      inputs:
105        ConnectionType: 'AzureRM'
106        azureSubscription: 'Microsoft Azure Internal Consumption'
107        appType: 'webApp'
108        WebAppName: '$(WebsiteName)'
109        deployToSlotOrASE: true
110        ResourceGroupName: 'PartsUnlimitedRG'
111        SlotName: 'Dev'
112        packageForLinux: '$(System.ArtifactsDirectory)/**/*.zip'

```

7. We will also replace **\$(System.ArtifactsDirectory)** with **\$(Pipeline.Workspace)** in *packageForLinux*: value. This is to avoid having to add Download Pipeline Artifact task in every job under every stage. When we use the *deployment* job, it automatically downloads the published artifacts to **\$(Pipeline.Workspace)/{artifact}** location.

You can find more about Publish and Download Artifacts [here](#).

Your AzureRmWebAppDeployment@4 task should now look like below:

```

102  Settings
103  - task: AzureRmWebAppDeployment@4
104    inputs:
105      ConnectionType: 'AzureRM'
106      azureSubscription: 'Microsoft Azure Internal Consumption'
107      appType: 'webApp'
108      WebAppName: '$(WebsiteName)'
109      deployToSlotOrASE: true
110      ResourceGroupName: 'PartsUnlimitedRG'
111      SlotName: 'Dev'
112      packageForLinux: '$(Pipeline.Workspace)/**/*.zip'

```

8. Next, we will add a snippet for the Staging environment. Copy the following *deployment* section.

```

90     - deployment: DeployToDev
91       displayName: Deploy To Dev Environment
92     pool:
93       name: Hosted VS2017
94     environment: 'PUL-Dev'
95     strategy:
96       runOnce:
97         deploy:
98           steps:
99             - task: AzureRmWebAppDeployment@4
100               inputs:
101                 ConnectionType: 'AzureRM'
102                 azureSubscription: 'Microsoft Azure Internal Consumption'
103                 appType: 'webApp'
104                 WebAppName: '$(WebsiteName)'
105                 deployToSlotOrASE: true
106                 ResourceGroupName: 'PartsUnlimitedRG'
107                 SlotName: 'Dev'
108                 packageForLinux: '$(Pipeline.Workspace)/**/*.zip'

```

9. Paste this section on the next line after **AzureRmWebAppDeployment@4** task ends and before **DeployToProd** stage starts. Correctly indent the pasted section to align with the previous *DeployToDev* deployment job. **Note:** The first **DeployToDev** deployment job is collapsed for better readability.

```

87     - stage: DeployToDevAndStaging
88       displayName: Deploy to Dev and Staging Environments
89       jobs:
90     > - deployment: DeployToDev ...
112
113     |
114
115     - stage: DeployToProd
116       displayName: Deploy to Production

```

```

90 > .. - deployment: DeployToDev ...
112
113 .. - deployment: DeployToDev
114 ..   displayName: Deploy To Dev Environment
115 ..
116 ..   pool:
117 ..     name: Hosted VS2017
118 ..
119 ..   environment: 'PUL-Dev'
120 ..   strategy:
121 ..     runOnce:
122 ..       deploy:
123 ..         steps:
124
125         Settings
126         .. - task: AzureRmWebAppDeployment@4
127         ..   inputs:
128         ..     ConnectionType: 'AzureRM'
129         ..     azureSubscription: 'Microsoft Azure Internal Consumption'
130         ..     appType: 'webApp'
131         ..     WebAppName: '$(WebsiteName)'
132         ..     deployToSlotOrASE: true
133         ..     ResourceGroupName: 'PartsUnlimitedRG'
134         ..     SlotName: 'Dev'
135         ..     packageForLinux: '$(Pipeline.Workspace)/**/*.zip'
136 .. - stage: DeployToProd
137 ..   displayName: Deploy to Production

```

10. Now, we will rename the second *DeployToDev* deployment job to **DeployToStaging**, rename *displayName* to **Deploy to Staging Environment**, and change the *environment* to **PUL-Staging**.

```

> .. - deployment: DeployToDev ...
.. - deployment: DeployToStaging
..   displayName: Deploy To Staging Environment
..
..   pool:
..     name: Hosted VS2017
..
..   environment: 'PUL-Staging'
..   strategy:

```

11. We also want to ensure that *DeployToDev* deployment runs first and only after it is successfully finished, *DeployToStaging* starts. For enforcing this dependency, we will use **dependsOn**.

```
deployment: DeployToStaging
displayName: Deploy To Staging Environment
dependsOn: DeployToDev
```

```
.. - deployment: DeployToStaging
  .. displayName: Deploy To Staging Environment
  .. dependsOn: DeployToDev
  ..
  .. pool:
```

12. Lastly, we will change the *SlotName* to **Staging** in the **AzureRmWebAppDeployment@4** task. This deploys the web app to the Staging slot in this deployment job.

```
120 .. environment: 'PUL-Staging'
121 .. strategy:
122   .. runOnce:
123     .. deploy:
124       .. steps:
125
126   Settings
127   .. - task: AzureRmWebAppDeployment@4
128     .. inputs:
129       .. ConnectionType: 'AzureRM'
130       .. azureSubscription: 'Microsoft Azure Internal Consumption'
131       .. appType: 'webApp'
132       .. WebAppName: '$(WebsiteName)'
133       .. deployToSlotOrASE: true
134       .. ResourceGroupName: 'PartsUnlimitedRG'
135       .. SlotName: 'Staging'
136       .. packageForLinux: '$(Pipeline.Workspace)/**/*.zip'
137   .. stage: DeployToProd
```

13. Finally, before we start working on the last stage in the next exercise, **Save** the changes made so far.

Variables

Save

⋮

Exercise 3: Configure the Production deployment stage

1. Add an *environment* section to the last line of the YAML file. Set the environment name to **PUL-Production**

```
environment: 'PUL-Production'
strategy:
  runOnce:
    deploy:
      steps:
```

```
137 - stage: DeployToProd
138   displayName: Deploy to Production
139   jobs:
140   - deployment: DeployToProd
141     displayName: Deploy To Prod Environment
142
143     pool:
144     | name: Hosted VS2017
145
146     environment: 'PUL-Production'
147     strategy:
148     | runOnce:
149     | | deploy:
150     | | | steps:
```

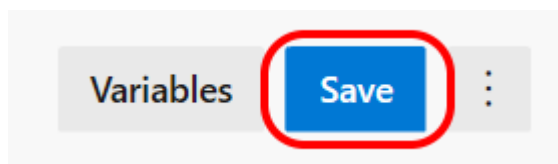
2. Ensure the environment section is correctly indented to align with the *pool* section.
3. Set the cursor on a new line at the end of the YAML definition. This will be the location where our next task will be added for Azure App Service Deployment.
4. Copy and paste the **AzureRmWebAppDeployment@4** task from the previous stage and remove the following three lines:

```
deployToSlotOrASE: true
ResourceGroupName: 'PartsUnlimitedRG'
SlotName: 'Staging'
```

```
146 |     environment: 'PUL-Production'
147 |     strategy:
148 |       runOnce:
149 |         deploy:
150 |           steps:
151 |
152 |           - task: AzureRmWebAppDeployment@4
153 |             inputs:
154 |               ConnectionType: 'AzureRM'
155 |               azureSubscription: 'Microsoft Azure Internal Consumption'
156 |               appType: 'webApp'
157 |               WebAppName: '$(WebsiteName)'
158 |               packageForLinux: '$(Pipeline.Workspace)/**/*.zip'
```

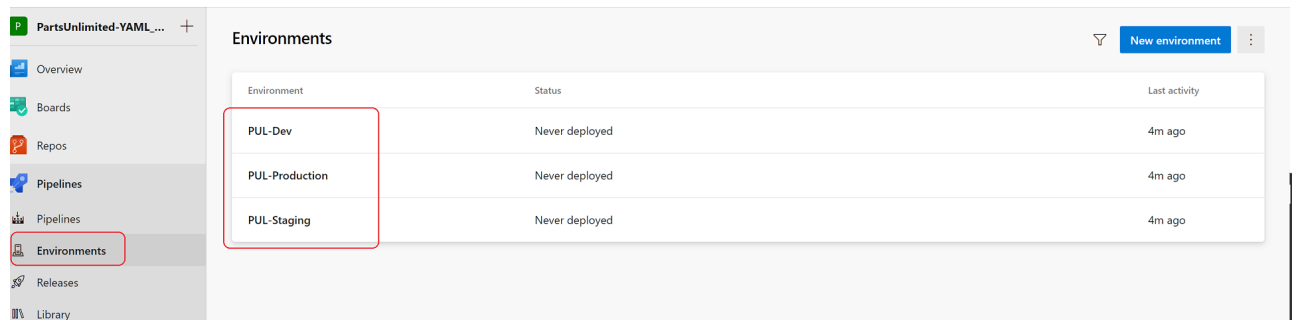
Note: Since this is the Production environment, we are deploying the app in the production slot or in the actual Azure App Service instead of any slot of this App Service.

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



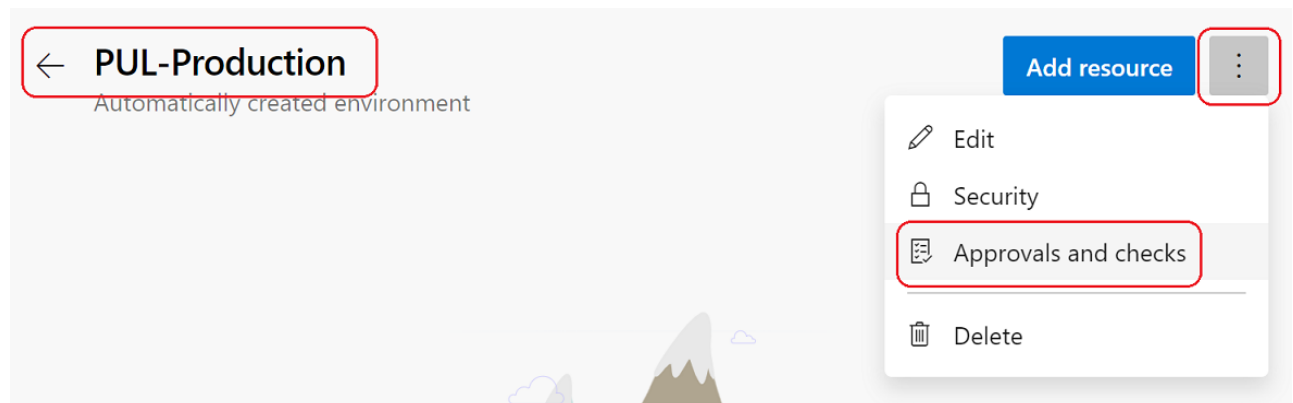
Exercise 4: Review environments and setup Approval

1. Before we trigger the run of this pipeline, let's review the environments that got created after we saved the YAML file. We will also setup Approvals on the Production environment.
2. Click on **Environments** under the Pipelines section. Notice that three environments are automatically created: PUL-Dev, PUL-Staging and PUL-Production.



Note: Notice that the environments indicate that they were created just a few minutes ago (when we saved the YAML file) and have never been deployed to. We could have manually created these environments as well. However, it's easier to create those directly from the YAML definition.




3. Now, we want to ensure that deployments to the PUL-Production environment don't happen without the approval of an individual/group that is responsible for the Production releases. For this we will use **Approvals**.
4. Select the **PUL-Production** environment. Select the vertical "..." section at the top-right. Select **Approvals and Checks**



5. Select **Approvals** and add yourself as the **Approver**. Click **Create**.

Add your first check

Checks allow you to manage how this resource is used.
Changes made to checks are effective immediately, applicable to all existing and new pipelines.

-  **Approvals**
Approvers should grant approval for deployment
-  **Evaluate artifact (preview)**
Ensure artifacts adhere to custom policies (container i...
-  **Invoke Azure Function**
Invoke an Azure Function

[See all](#)

Approvals

Approvers

  × +

Instructions to approvers (optional)

Advanced

- ☒ Allow approvers to approve their own runs

Control options




Timeout

30

Days

Add your first

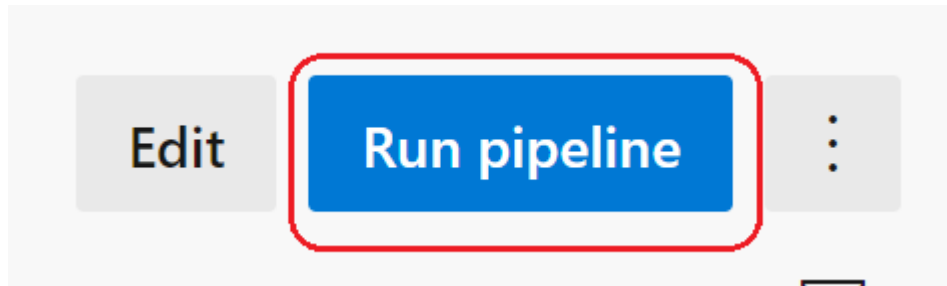
Checks allow you to manage
Changes made to checks
new pipelines.

-  **Approvals**
Approvers should grant approval for deployment
-  **Evaluate artifact (p**
Ensure artifacts adhere
-  **Invoke Azure Func**
Invoke an Azure Func

[See all](#)

Exercise 5: Deploy the Web App to Multiple Environments

1. Let's run the pipeline to check if everything is working the way we are expecting.
2. Navigate back to Pipelines, select your YAML pipeline and click **Run pipeline**.



3. While we will not make any modifications at this time, select **Advanced Options**, expand **Stages to run**. You will notice all the stages we have configured in the YAML definitions are showing up here with the option to uncheck any of the stage.


Since we have combined the deployment of Dev and Staging environments into a single stage, we cannot unselect Dev or Staging independently. Click **Cancel**.

Run pipeline

×

Select parameters below and manually run the pipeline

Branch/tag

 master ▼

Select the branch, commit, or tag

Advanced options

Variables

This pipeline has no defined variables

>

Stages to run


Run as configured

>


Resources

Use latest version of all resources

>



Stages to run



Deselect stages you want to skip for this run

☒ **Build**
No dependencies

☒ **Deploy Azure Resources**
Depends on: Build


☒ **Deploy to Dev and Staging Environments**
Depends on: Deploy Azure Resources

☒ **Deploy to Production**
Depends on: Deploy to Dev and Staging Environments


4. Click **Run** to start the pipeline. Review the summary.


Note: Here you can see multiple stages under the **Stages** section. Also, notice the friendly names of those stages based on the *displayName* value in the YAML definitions.



Summary


Manually run by  r

[View 12 changes](#)



Repository and version
 PartsUnlimited
master 2e98e8f

Time started and elapsed
 Just now
24s

Related
 0 work items
 0 artifacts

Tests and coverage
 [Get started](#)

Stages Jobs

 **Build**
0/1 completed 22s
 Build 22s
[Cancel](#)






☐ **Deploy Azure Resour...**
Not started

☐ **Deploy to Dev and St...**
Not started

☐ **Deploy to Production**
Not started

5. Click **Jobs** to review the jobs defined in the YAML definitions.

20 / 25


Stages	Jobs
Name	
	Build
	Deploy ARM Template
	Deploy To Dev Environment
	Deploy To Staging Environment
	Deploy To Prod Environment

6. At any time during the running of the pipeline, click on any stage or job to view more details.

← Jobs in run #20200430.3


PartsUnlimited (1)

Build


>  Build

1m 36s


Deploy Azure Resources

▼  Deploy ARM Template


2m 24s

 Initialize job


2s

 Checkout PartsUnlimite...


6s

 DownloadPipelineArtifact


4s


 AzureResourceMa...

2m 11s


 Post-job: Checkout PartsU...


Deploy to Dev and Staging Environments

 Deploy To Dev Environment

 Deploy To Staging Environm...

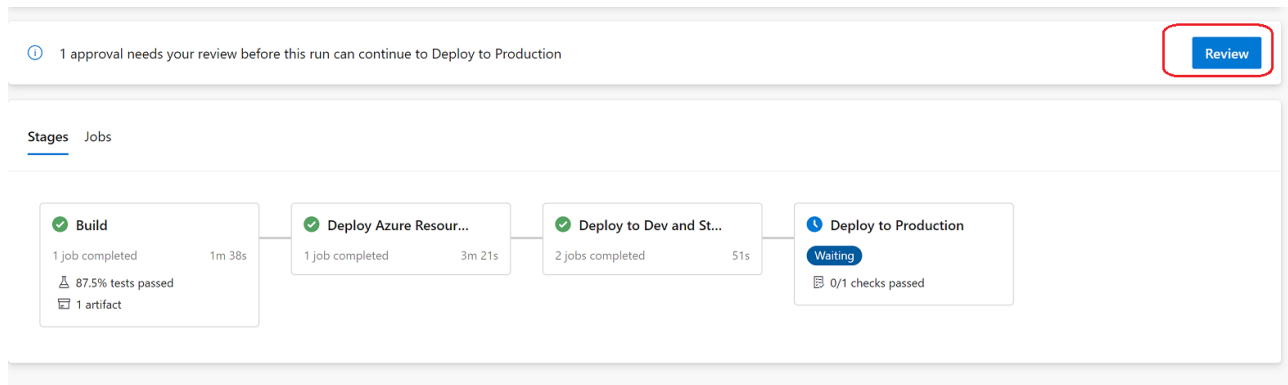
Deploy to Production

 Deploy To Prod Environment

 AzureResourceManagerTemplateDeployment

```
1 Starting: AzureResourceManagerTemplateDeployment
2 =====
3 Task      : ARM template deployment
4 Description : Deploy an Azure Resource Manager (ARM) template to all the deployment scopes
5 Version   : 3.2.0
6 Author    : Microsoft Corporation
7 Help      : https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/azure-resource-group-deployer
8 =====
9 ARM Service Connection deployment scope - Subscription
10 Checking if the following resource group exists: PartsUnlimitedRG.
11 Resource group exists: true.
12 Creating deployment parameters.
13 The detected encoding for file 'D:\a\1\PartsUnlimitedEnv\Templates\FullEnvironmentSetupMerged.json' is 'utf-8'
14 The detected encoding for file 'D:\a\1\PartsUnlimitedEnv\Templates\FullEnvironmentSetupMerged.param.json' is 'utf-8'
15 Starting template validation.
16 Deployment name is FullEnvironmentSetupMerged-20200430-022032-b2b5
17 Template deployment validation was completed successfully.
18 Starting Deployment.
19 Deployment name is FullEnvironmentSetupMerged-20200430-022032-b2b5
```

7. As the deployment reaches the Production environment, it will pause for the review and approval first. Click on **Review** and **Approve** the deployment.



Approval

Waiting • Timeout in 29d

Resource

PUL-Production

Requirement

All approvers must approve

Waiting on me

Reject Approve

Waiting

8. Also, back in the pipeline summary, click on **Environments** to review the deployment for all the environments.

✓

#20200430.3 Update azure-pipelines-3.yml for Azure Pipelines

on PartsUnlimited (1) ⚙ Retained

Summary

Tests

Environments

Code Coverage

PUL-Dev

Jobs	Resource	Duration
✓ Deploy To Dev Environment		🕒 19s

PUL-Staging

Jobs	Resource	Duration
✓ Deploy To Staging Environment		🕒 20s

PUL-Production

Jobs	Resource	Duration
✓ Deploy To Prod Environment		🕒 27s

9. As the deployment completes, we want to check our deployment in Azure. Navigate to the [Azure Portal](#). Select the PartsUnlimitedRG resource group and click on the App Service for the Dev slot. Click on **Browse** and verify that the site is running. Repeat this for the Staging slot and Production application.

PUL-YAML-Multistage-kp

App Service

Search (Ctrl+ /)

⏪

Overview

Activity log

Access control (IAM)

Browse

Stop

Swap

Restart

Delete

Get publish profile

Reset pu

App Service has installed a patch that changes cross-site and iframe cookie handling due to upcoming cha update their apps to handle these changes. Click to learn more.

Resource group (change) : PartsUnlimitedRG

Status : Running

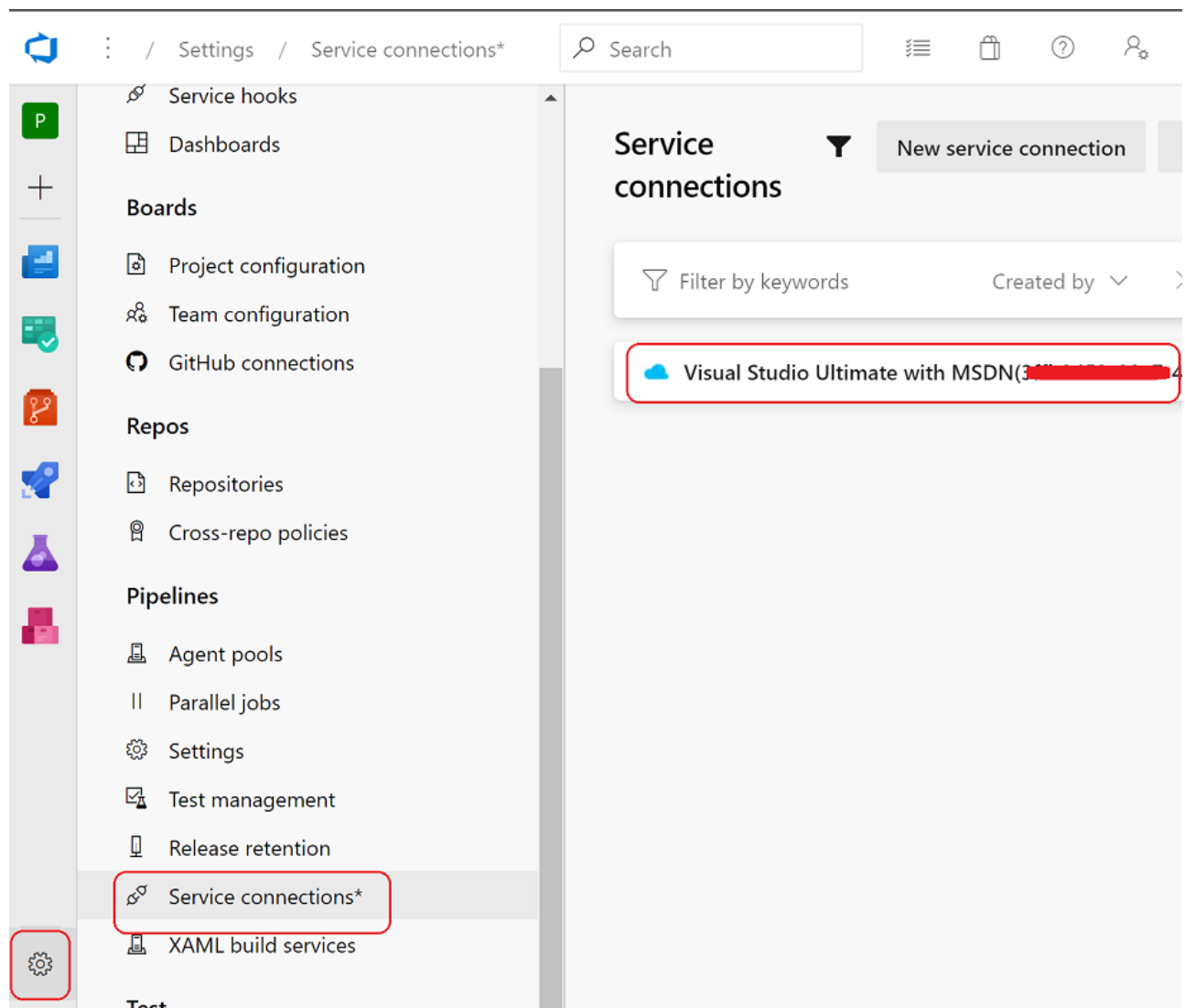
URL

App St

23 / 25

Exercise 6: Set the Approval on the Service Connection

1. In this last exercise, we will set up Approval on the Service Connection used in the pipeline.
2. At this point, everyone should be quite comfortable navigating through Azure DevOps. Also, setting up an Approval on the Service Connection is very similar to setting the Approvals in Environments. For these reasons, the following steps are deliberately provided with fewer screenshots.
3. Navigate to **Project Settings** at the bottom-left of the Project you are working on.
4. Under the **Pipelines** section in the Project Settings, select **Service connections**.
5. Notice a Service connection has already been created. This service connection was created the first time you authorized the connection from Azure DevOps to the Azure Portal in an earlier lab.



6. Select the Service connection and click on the vertical (...) to configure the settings of this connection.
7. Click on **Approvals and checks**. This UI should be similar to that of the Approvals and Checks for the Environments.
8. Select **Approvals** and add yourself as the approver. Click **Create**.

9. Run the pipeline again. In each stage that the service connections is invoked, the pipeline will now ask for an approval. This will protect unauthorized and unintended deployments from Azure DevOps into a particular Azure subscription!