Embracing Continuous Delivery with Azure Pipelines

Overview

In this lab, you will learn about the release management features available in Azure Pipelines that automate the deployment of applications. These features help development and operations teams integrate with Azure DevOps Server to configure and automate complex deployments of their automated builds to target stages more easily. Development teams can also model their release processes and track approvals, sign-offs, and visualize their release status.

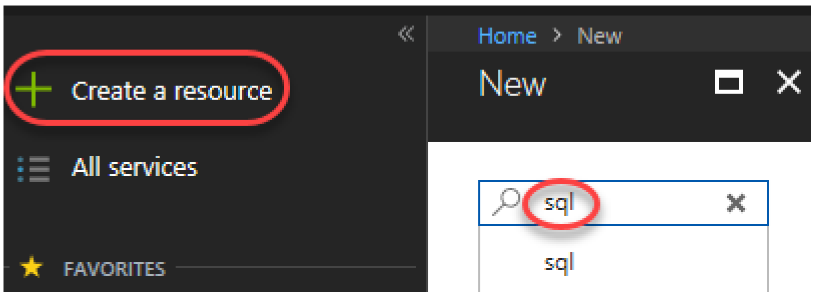
Prerequisites

- An Azure account (free from https://azure.com/free).
- Complete task 1 from the prerequisite instructions.

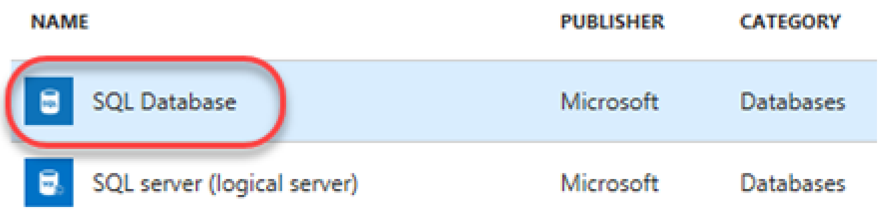
Exercise 1: Embracing Continuous Delivery with Azure DevOps

Task 1: Setting up Azure resources

1. Start off by creating the Azure resources needed for this lab. This includes a database and two app services: one for QA and one for production. Log into your account at https://portal.azure.com.
2. Click **Create a resource** and search for **"sql"**.

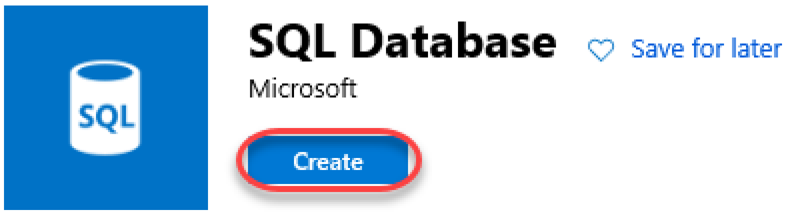


3. Select **SQL Database**.

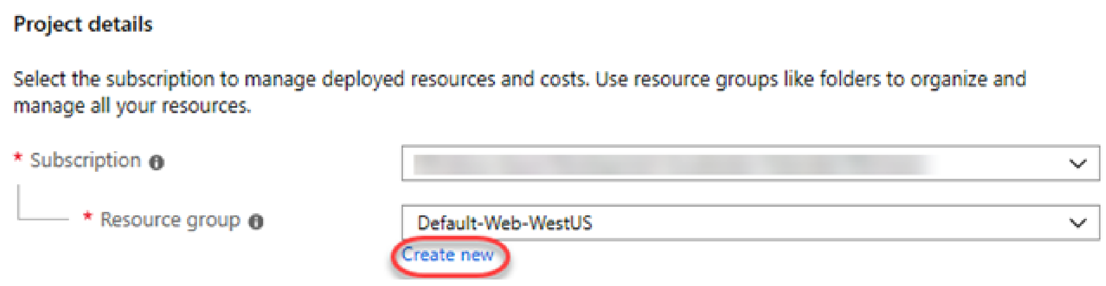


NAME	PUBLISHER	CATEGORY
SQL Database	Microsoft	Databases
SQL server (logical server)	Microsoft	Databases

4. Click **Create**.



5. Under **Resource group**, click **Create new**.

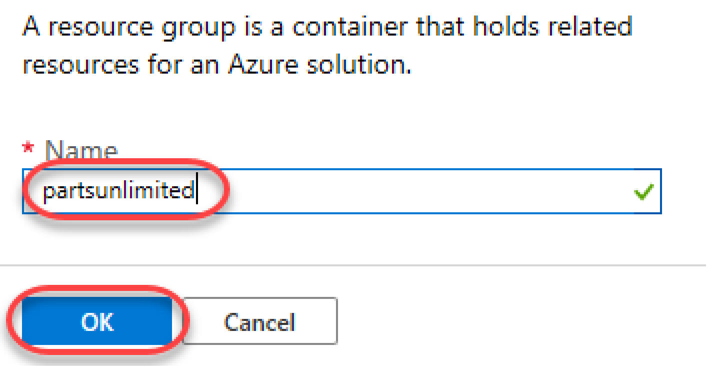


Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription	
* Resource group	Default-Web-WestUS

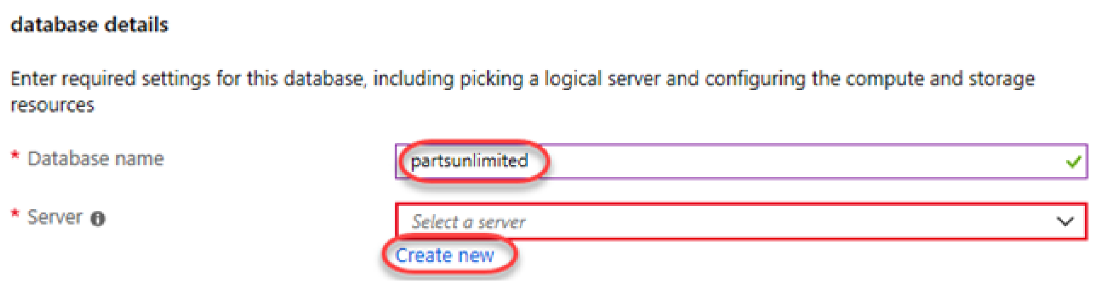
6. Enter a **Name** of **"partsunlimited"** and click **OK**.



A resource group is a container that holds related resources for an Azure solution.

* Name: partsunlimited

7. Enter a **Database name** of **"partsunlimited"** and click **Create new** to create a new server.

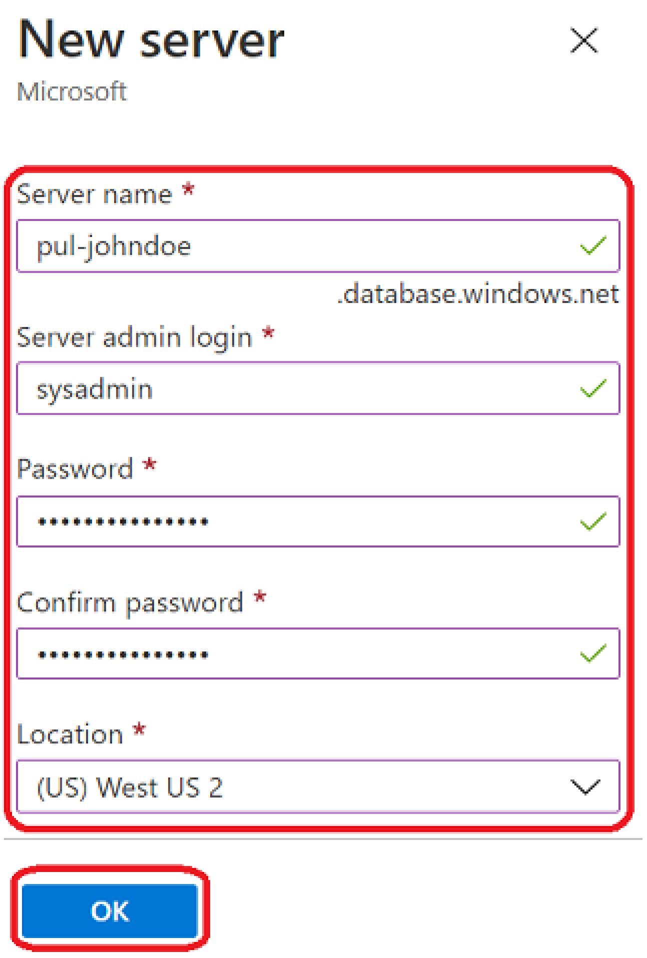


database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

* Database name	partsunlimited
* Server	Select a server

8. Enter a unique name for **Server name**, such as by including your name. Enter an admin username and password you can remember. Note that **"P2ssw0rd"** meets the password requirements. Click **OK** to confirm these options.


New server
Microsoft

Server name *
pul-johndoe
.database.windows.net

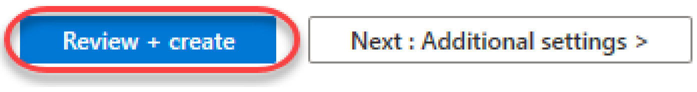
Server admin login *
sysadmin

Password *
•••••••••••••••

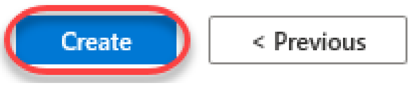
Confirm password *
•••••••••••••••

Location *
(US) West US 2

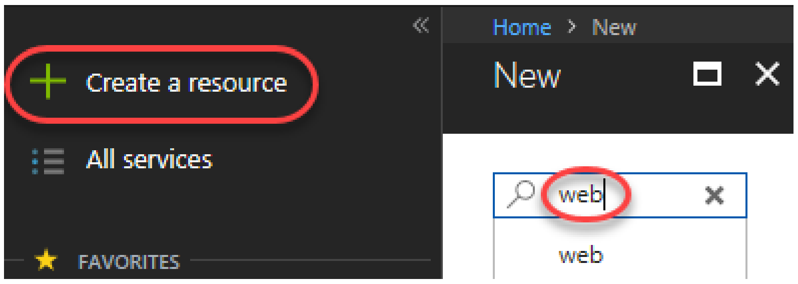
9. Click **Review + create**.



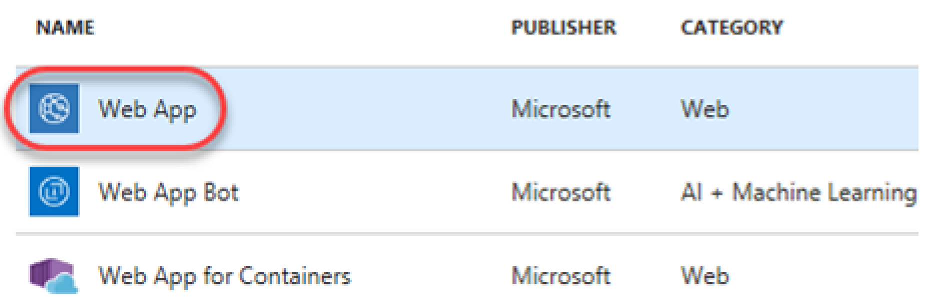
10. Click **Create**. It'll take some time to complete, but you can move on to the next step while it works in the background.



11. Click **Create a resource** and search for **"web"**.

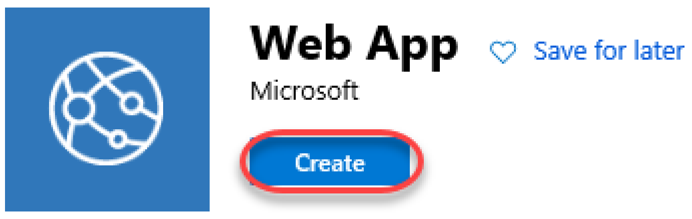


12. Select the **Web App** template.

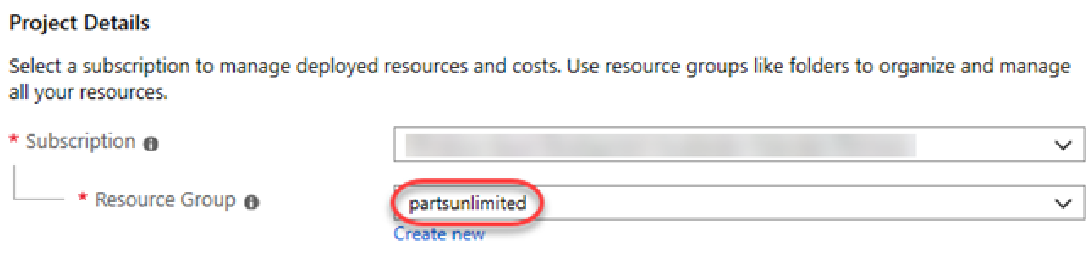


NAME	PUBLISHER	CATEGORY
Web App	Microsoft	Web
Web App Bot	Microsoft	AI + Machine Learning
Web App for Containers	Microsoft	Web

13. Click **Create**.



14. Under **Project Details**, select the same **Subscription** and **Resource Group** used for the database.

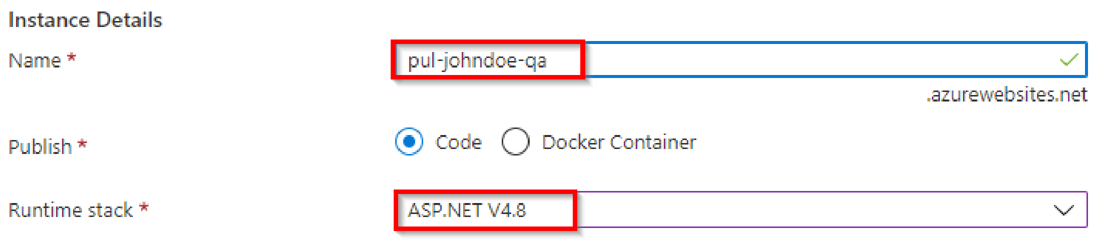


Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription: [redacted]
* Resource Group: partsunlimited

15. For **Name**, enter a unique name, such as by using your name as part. Since this will be for our QA deployment, append the name with **"-qa"**. Select the **ASP.NET 4.8 Runtime stack**.



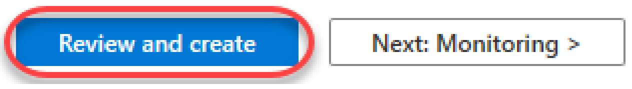
Instance Details

Name *: pul-johndoe-qa .azurewebsites.net

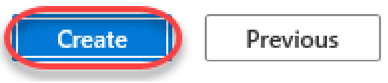
Publish *: [✓ Code] Docker Container

Runtime stack *: ASP.NET V4.8

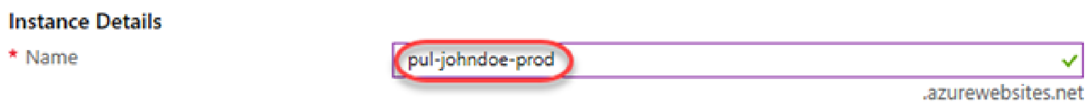
16. Click **Review and create**.



17. Click **Create**.



18. Repeat the process above to create a second app service for the production stage. This time, append the **Name** with **"-prod"** instead.

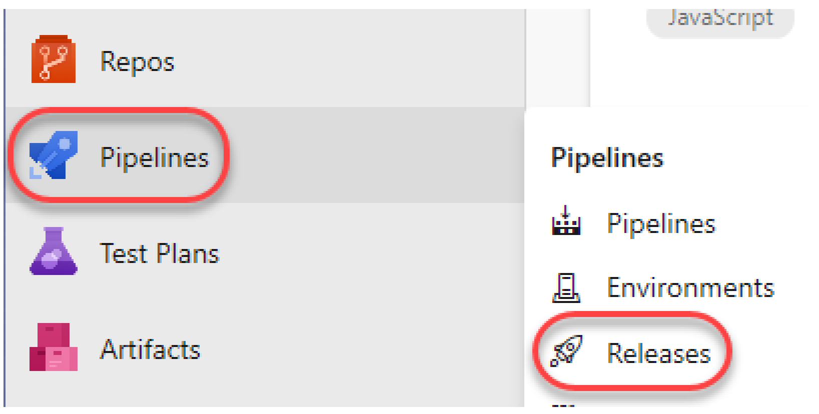

Instance Details

* Name: pul-johndoe-prod .azurewebsites.net

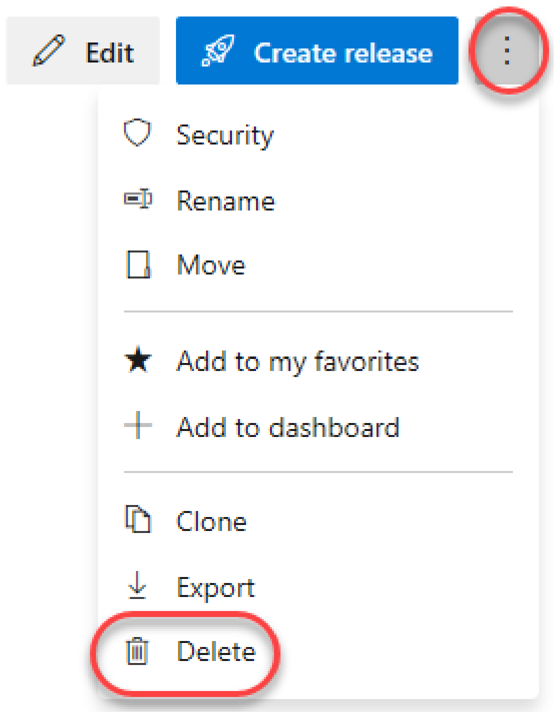
19. While the Azure resources are created, go ahead and configure the SQL server.
20. Click the **Resource groups** tab from the left menu. Locate and click the **partsunlimited** group.
21. Click the SQL server resource created earlier.
22. Select the **Firewalls and virtual networks** tab from the **Security** section.
23. Set **"Allow Azure services and resources to access this server"** to **Yes** to allow the App Services to access this SQL server.
24. Click **Save** on the top of the tab.
25. Continue on to the next task. Leave this browser tab open for later.

Task 2: Creating a continuous release to the QA stage

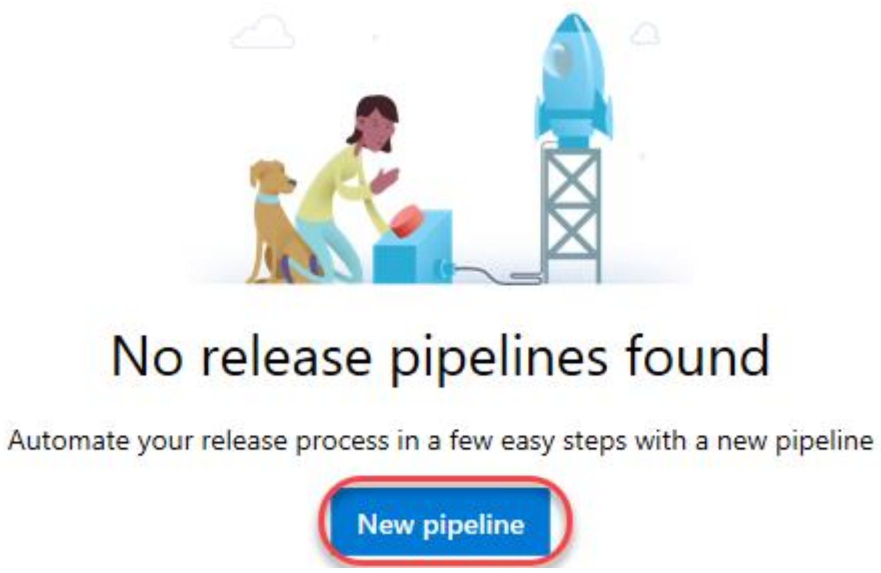
1. Navigate to your team project on Azure DevOps in a new browser tab. It should be at https://dev.azure.com/YOURACCOUNT/Parts%20Unlimited.
2. Navigate to **Pipelines | Releases**.



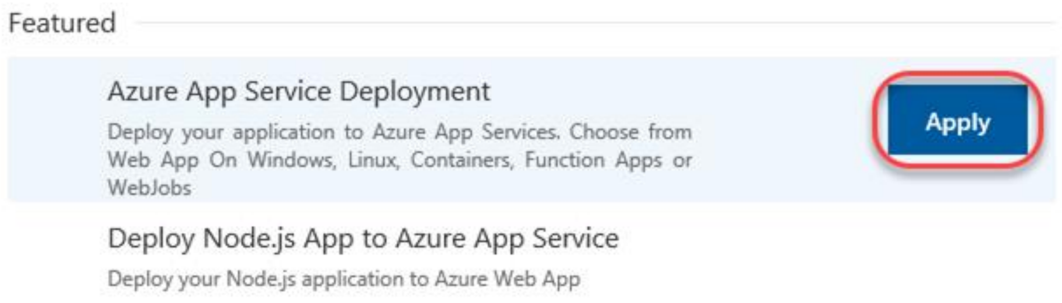
3. **Delete** the existing **PartsUnlimitedE2E** release pipeline. We'll start fresh here.



4. Click **New pipeline**.


No release pipelines found
Automate your release process in a few easy steps with a new pipeline
New pipeline

5. There are many starting templates to choose from, or you can even begin with an empty process template. In this case, select the **Azure App Service Deployment** and click **Apply**.



Featured

Azure App Service Deployment

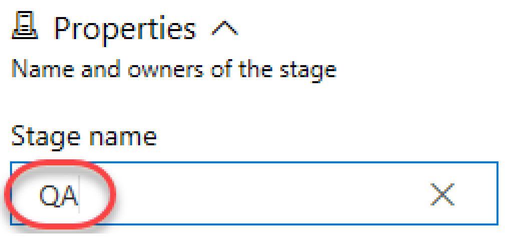
Deploy your application to Azure App Services. Choose from Web App On Windows, Linux, Containers, Function Apps or WebJobs

Apply

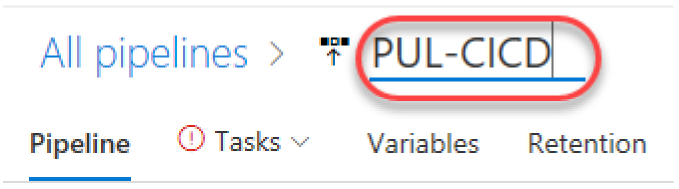
Deploy Node.js App to Azure App Service

Deploy your Node.js application to Azure Web App

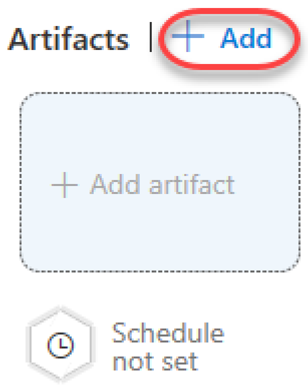
6. Rename the default stage to **“QA”**. This template will deploy to QA, and then to a production stage. We’ll set up this one first.



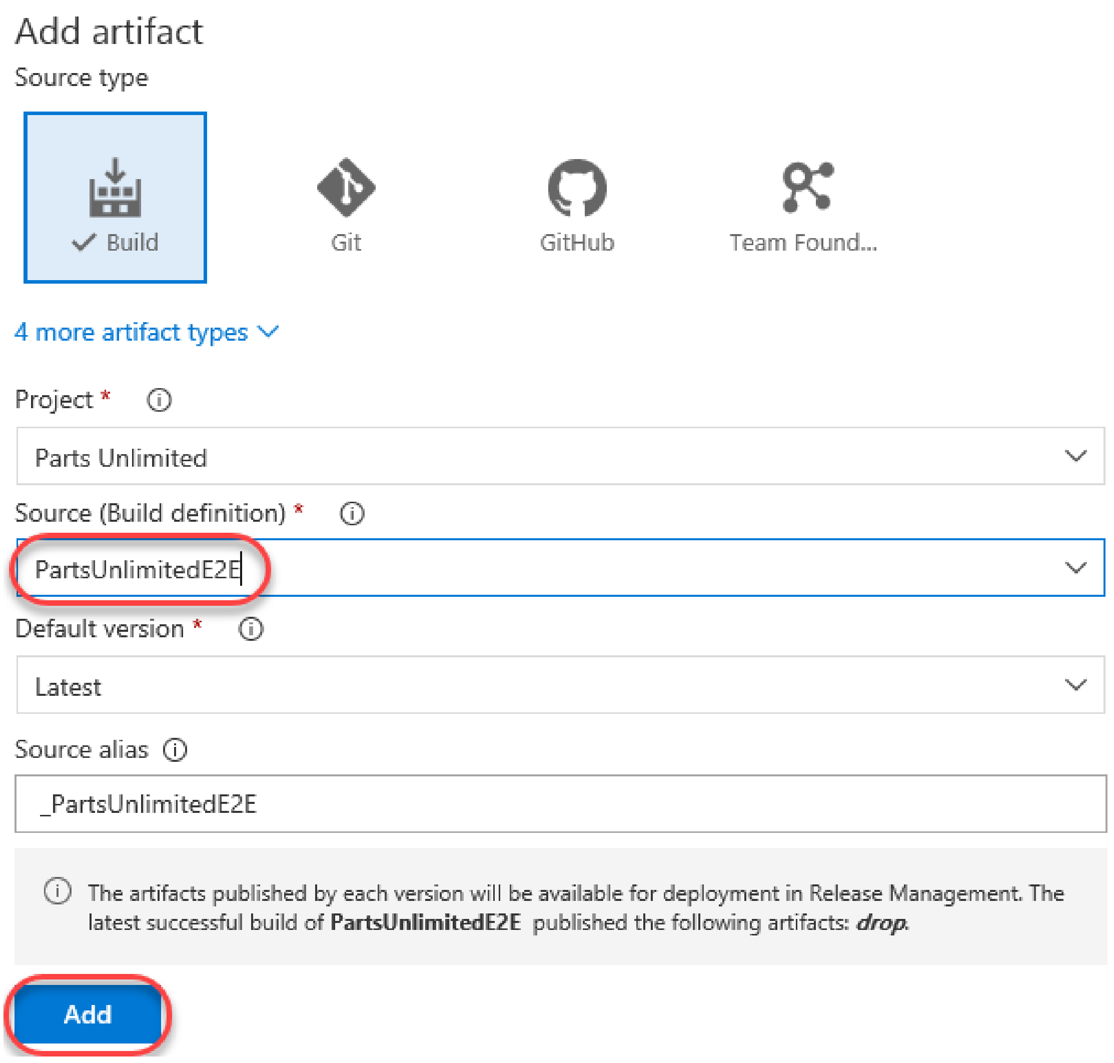
7. Rename the release pipeline to **“PUL-CICD”**.



8. The first thing to define is exactly what should be deployed. Click **Add** in the **Artifacts** section to specify the artifact to deploy.



9. There are many types of artifacts, but this one will be pretty simple: a project built from the **PartsUnlimitedE2E** build pipeline that already exists in this team project. Click **Add**.


Add artifact

Source type

Project *

Parts Unlimited

Source (Build definition) *

PartsUnlimitedE2E

Default version *

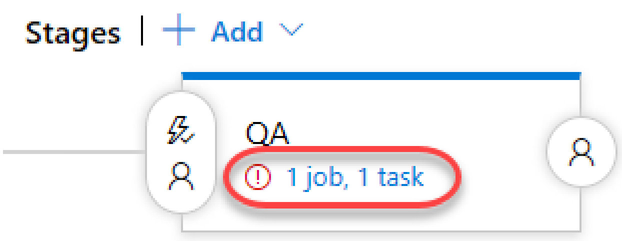
Latest

Source alias

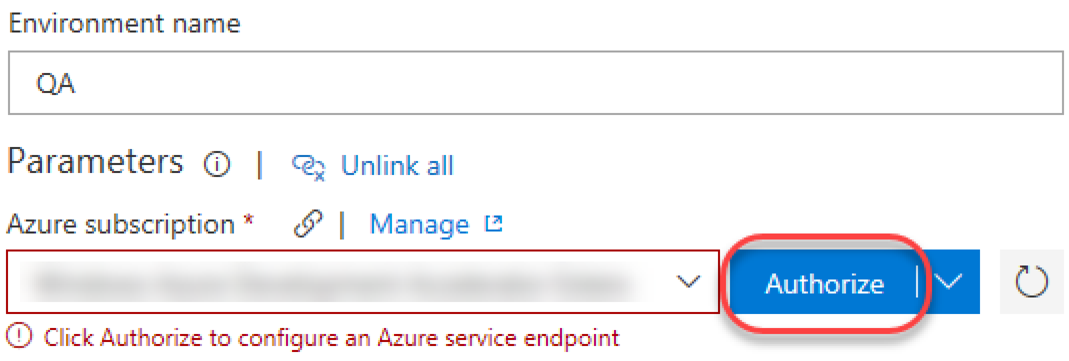
_PartsUnlimitedE2E

The artifacts published by each version will be available for deployment in Release Management. The latest successful build of **PartsUnlimitedE2E** published the following artifacts: ***drop***.

10. Now that the artifact has been defined, it's time to configure the deployment to QA. Click **1 job, 1 task** in the **QA** stage.



11. Select the **Azure subscription** you used earlier to create the resources and click **Authorize**. If you need to create a connection to an Azure account associated with a different Microsoft account, click **New** and follow that workflow before continuing.

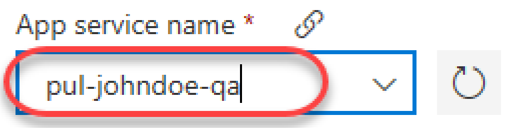


Environment name

QA

Click Authorize to configure an Azure service endpoint

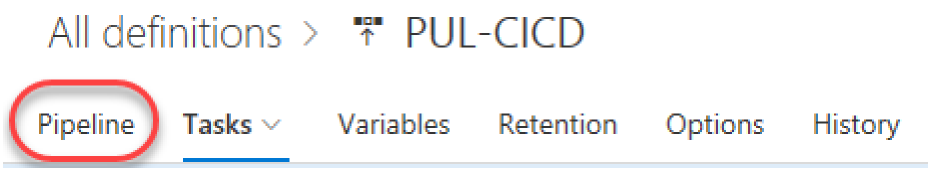
12. Follow the workflow to authorize access to your Azure account.
13. Enter the **App service name** used earlier when creating the QA app service.



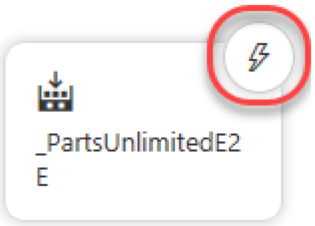
App service name *

pul-johndoe-qa

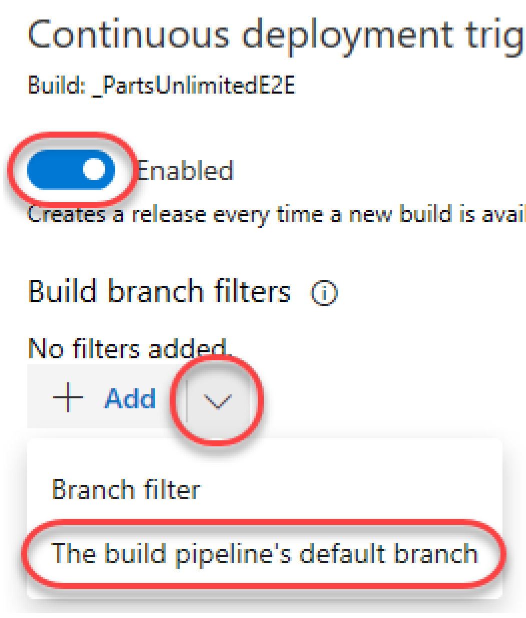
14. Return to the **Pipeline** tab.



15. Click the **Triggers** button to define what triggers will invoke this deployment.



16. **Enable** the **Continuous deployment trigger**. Add a **Build branch filter** that points at the **The build pipeline’s default branch**. This will kick off the deployment when the build completes.


Continuous deployment trigger

Build: _PartsUnlimitedE2E

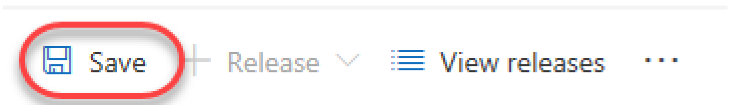
Enabled

Creates a release every time a new build is available.

Build branch filters

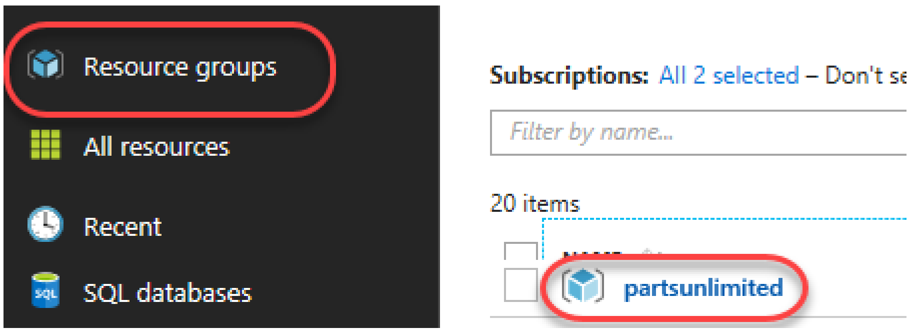
No filters added.

17. **Save** the release pipeline.

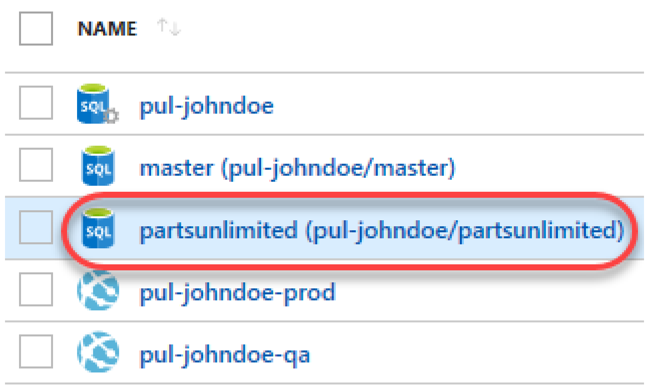


Task 3: Configuring the Azure app services

1. Return to the browser tab open to the Azure portal.
2. Click the **Resource groups** tab from the left menu. Locate and click the **partsunlimited** group created earlier.

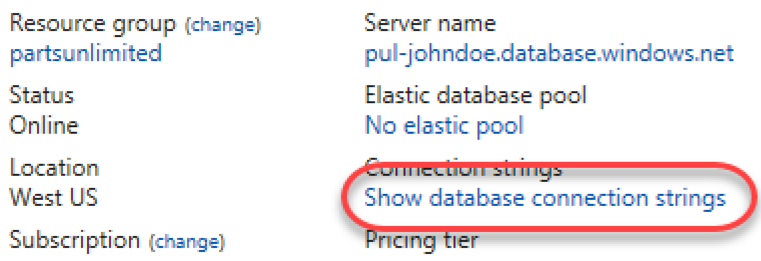


3. Click your SQL database (something like **pul-johndoe/partsunlimited**). Make sure you click the database you created and not the server. Note that it may take a few minutes for the database and server to become available, so click the **Refresh** button every once in a while to check in.



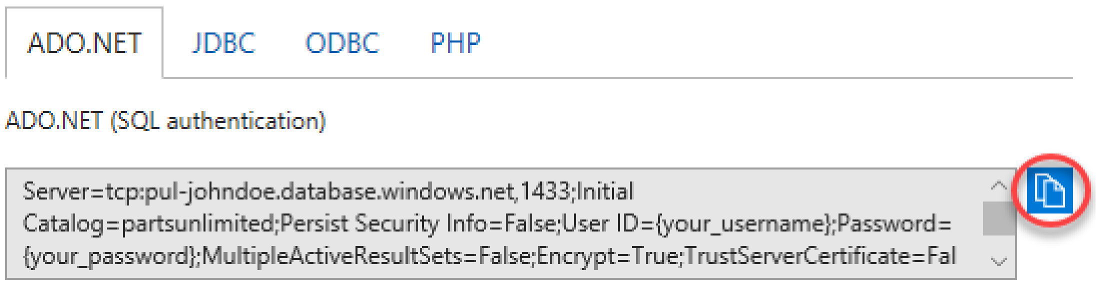
NAME
pul-johndoe
master (pul-johndoe/master)
partsunlimited (pul-johndoe/partsunlimited)
pul-johndoe-prod
pul-johndoe-qa

4. In the new blade, click **Show database connection strings**.



Resource group (change) partsunlimited	Server name pul-johndoe.database.windows.net
Status Online	Elastic database pool No elastic pool
Location West US	Connection strings Show database connection strings
Subscription (change)	Pricing tier

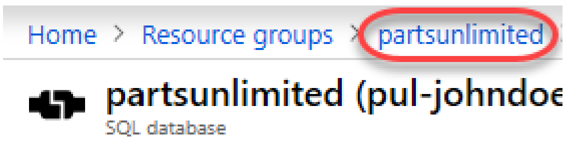
5. This will provide you with a list of connection strings based on platform. Copy the **ADO.NET** string to your clipboard so you can configure your new web site to use it. Close this blade.



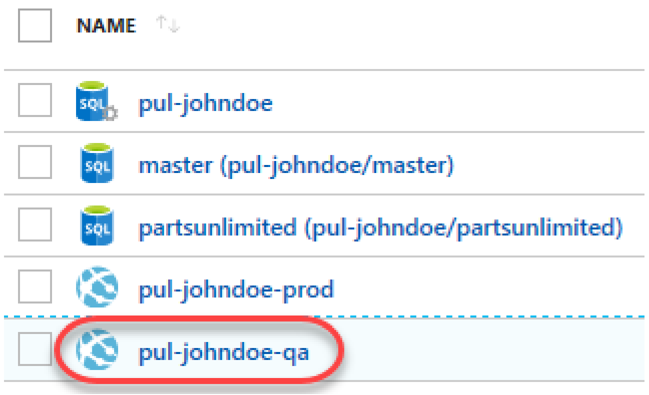
ADO.NET (SQL authentication)

```
Server=tcp:pul-johndoe.database.windows.net,1433;Initial
Catalog=partsunlimited;Persist Security Info=False;User ID={your_username};Password=
{your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=Fal
```

6. Open a new instance of **Notepad** and paste the connection string into it. This will make it easier to edit and retrieve later on in case anything happens to the clipboard copy.
7. Use the breadcrumb navigation to return to the **partsunlimited** resource group.


Home > Resource groups > partsunlimited
partsunlimited (pul-johndoe
SQL database

8. Click the **QA** app service created earlier.


NAME

pul-johndoe

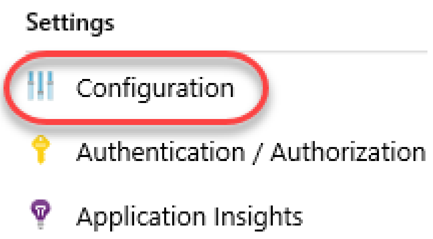
master (pul-johndoe/master)

partsunlimited (pul-johndoe/partsunlimited)

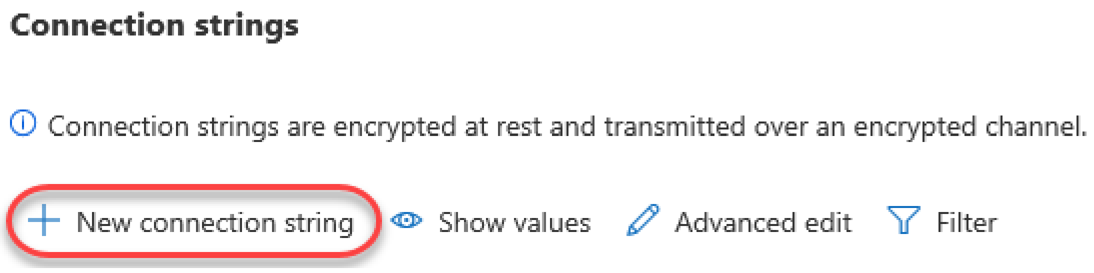
pul-johndoe-prod

pul-johndoe-qa

9. Select the **Configuration** tab from the **Settings** section.

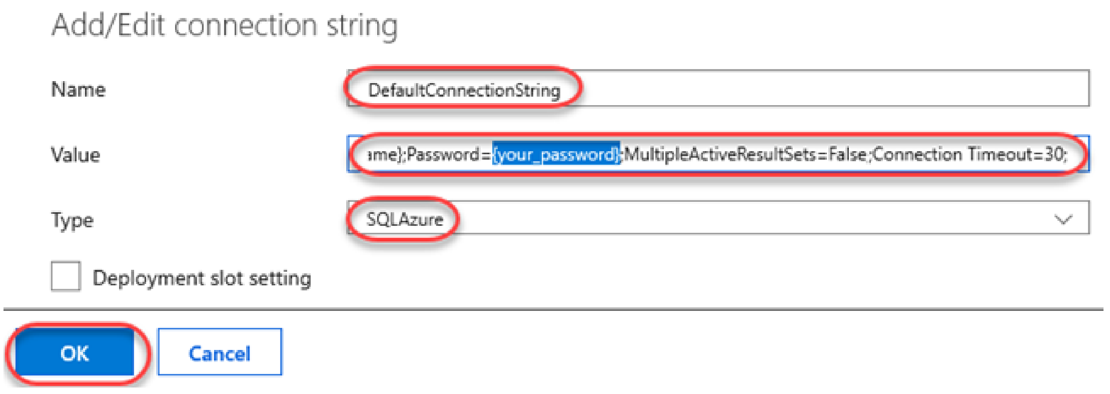


10. On this blade you can configure settings for your app, such as connection strings. Click **New connection string**.


Connection strings

Connection strings are encrypted at rest and transmitted over an encrypted channel.

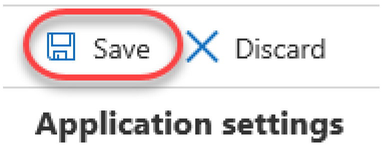
11. Locate the **Connection strings** section and add a new entry with the key **"DefaultConnectionString"** and the value pasted from the clipboard. You'll need to locate the "{your_username}" and "{your_password}" sections and replace them (including braces) with the actual SQL credentials entered earlier. Be sure the **Type** is set to **SQLAzure** and click **OK**.


Add/Edit connection string

Name	DefaultConnectionString
Value	ıme};Password={your_password};MultipleActiveResultSets=False;Connection Timeout=30;
Type	SQLAzure

☐ Deployment slot setting

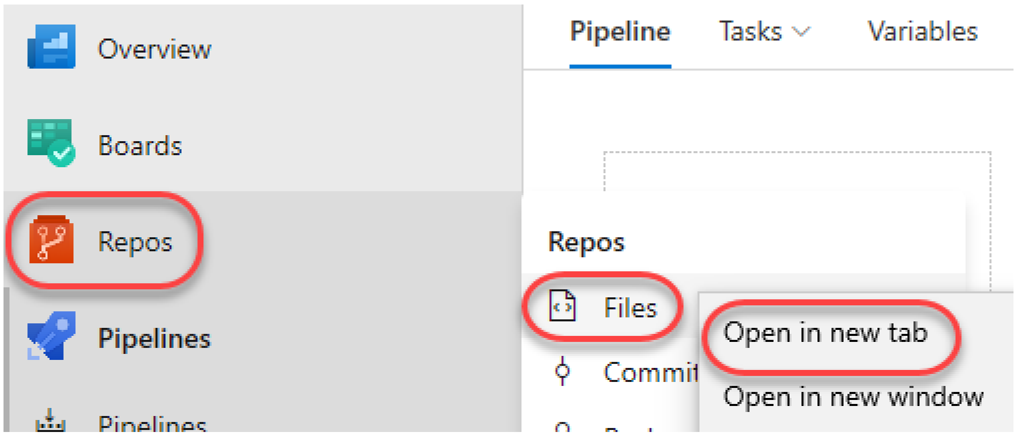
12. Click **Save** to commit.



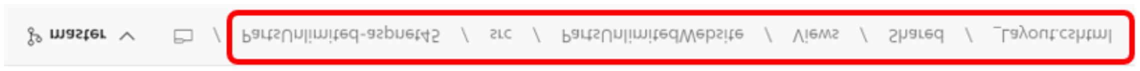
13. Repeat the process above to add the same connection string to the production app service.

Task 4: Invoking a continuous delivery release to QA

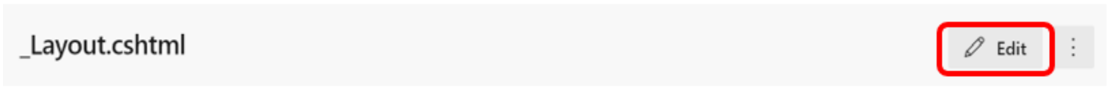
1. Return to the browser tab open to your Azure DevOps project.
2. Now that the release pipeline is in place, it's time to commit a change in order to invoke a build and release. You'll need to make a few changes like this over the course of this lab, so it's recommended that you use a separate tab for **Code | Files** to keep that part of the process separate.



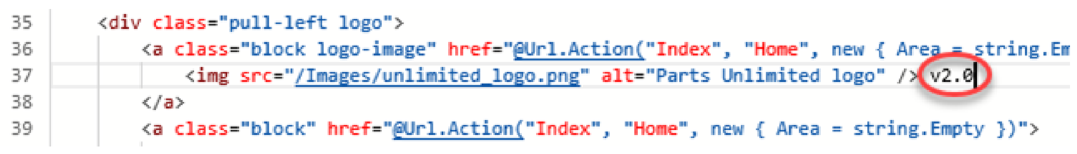
3. Navigate to **PartsUnlimited-aspnet45/src/PartsUnlimitedWebsite/Views/Shared/_Layout.cshtml**. This is a file that defines the general layout of the site and is a good place to make a change that will be easily visible after deployment.


master
PartsUnlimited-aspnet45 \ src \ PartsUnlimitedWebsite \ Views \ Shared \ _Layout.cshtml

4. Click **Edit** to edit the file inline.

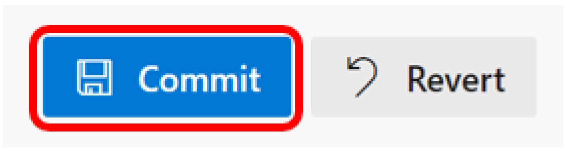


5. Locate the **Parts Unlimited Logo** and add the text **"v2.0"** after it. This will be an easy thing to check for after deployment.

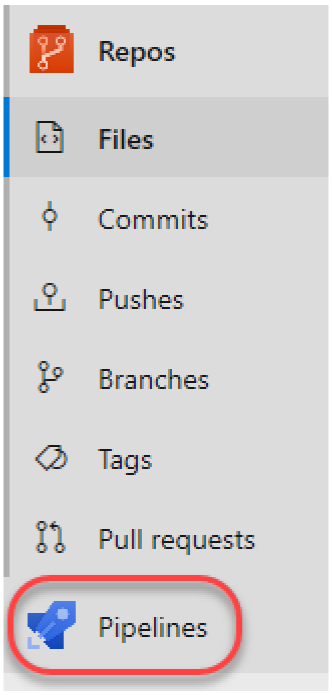


```
<div class="pull-left logo">
    <a class="block logo-image" href="@Url.Action("Index", "Home", new { Area = string.Em
        <img src="/Images/unlimited_logo.png" alt="Parts Unlimited logo" /> v2.0
    </a>
    <a class="block" href="@Url.Action("Index", "Home", new { Area = string.Empty })">
```

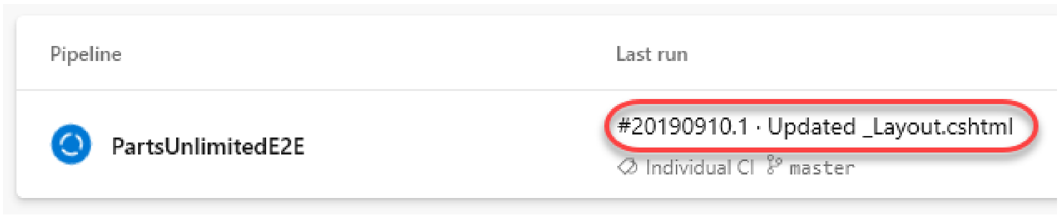
6. **Commit** the changes back to the repo. This will kick off a build based on the preconfigured build pipeline.



7. Navigate to the **Pipelines** hub.

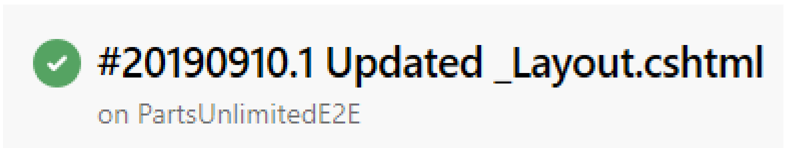


8. Open the most recent build for **PartsUnlimitedE2E**. It may be queued, in progress, or already completed.



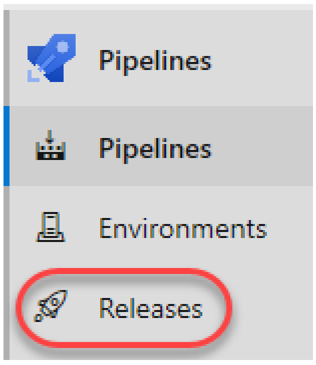
Pipeline	Last run
PartsUnlimitedE2E	#20190910.1 · Updated _Layout.cshtml Individual CI master

9. Follow the build until completion.

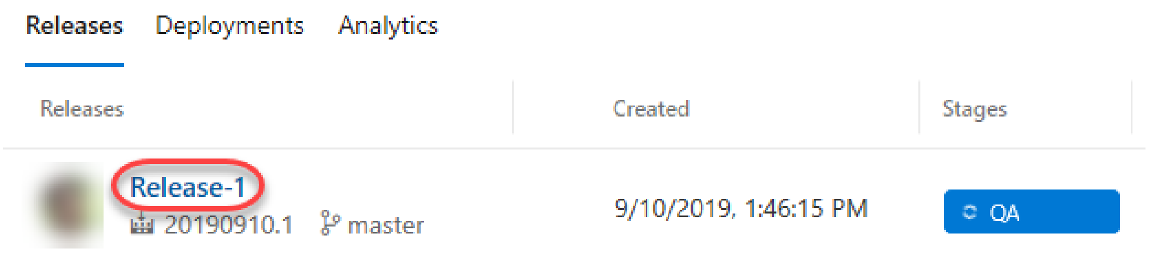


#20190910.1 Updated _Layout.cshtml
on PartsUnlimitedE2E

10. Click the **Releases** tab to see the new release get going.

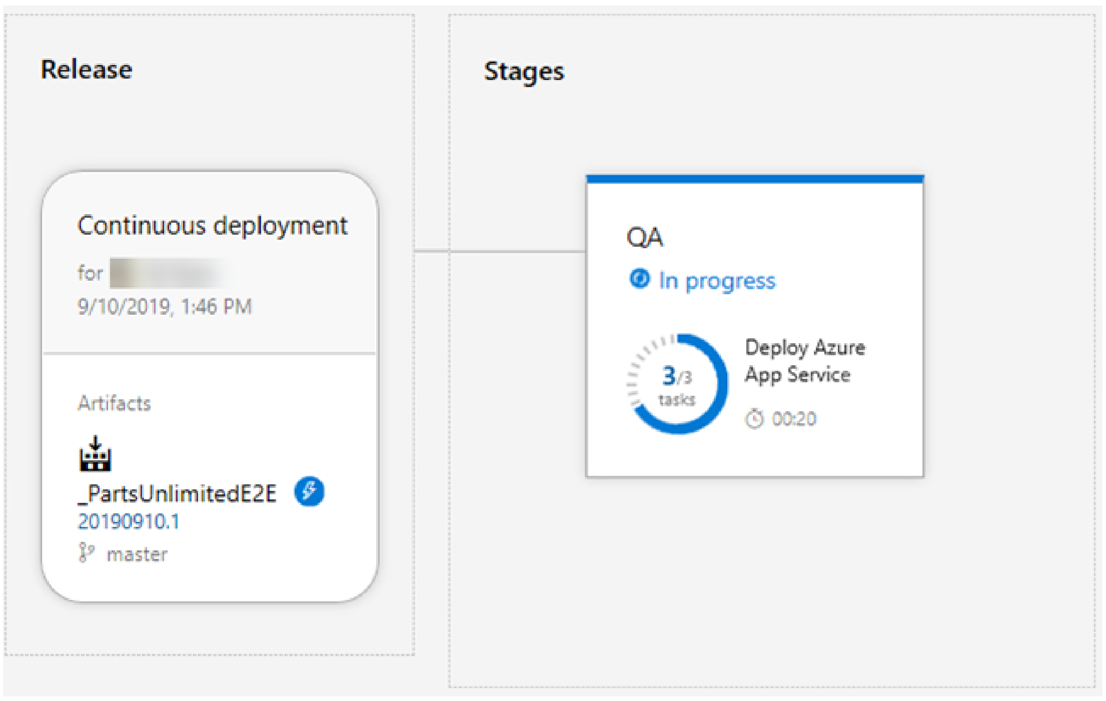


11. Click the release. As with the build, it may be queued, in progress, or already completed when you get here.

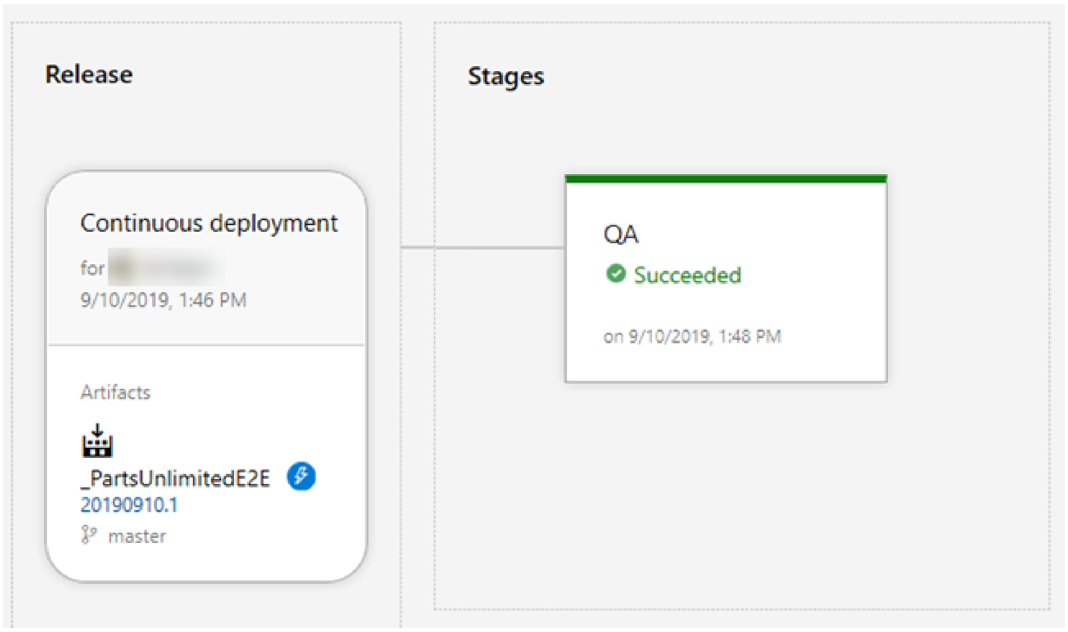


Releases	Created	Stages
Release-1 20190910.1 master	9/10/2019, 1:46:15 PM	QA

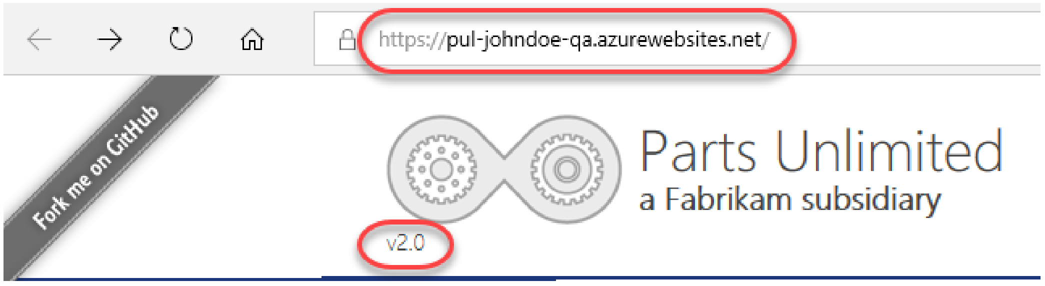
12. Follow the release until it succeeds.



13. You can now close this tab if you prefer.

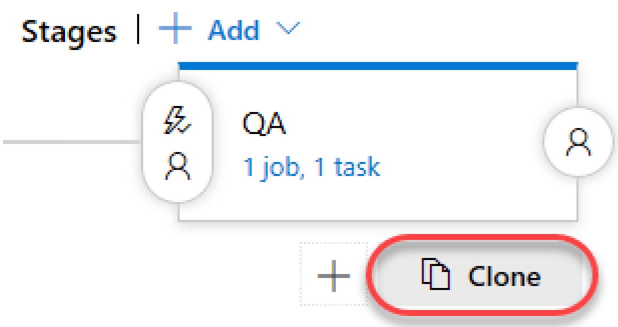


14. In a new browser tab, navigate to the QA site. It will be the name of your app service plus **".azurewebsites.net"**. It should show the **v2.0** added earlier.

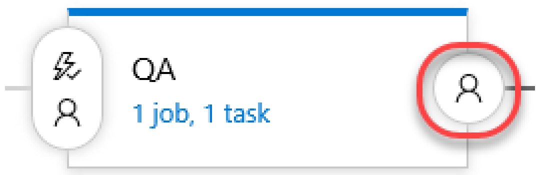


Task 5: Creating a gated release to the production stage

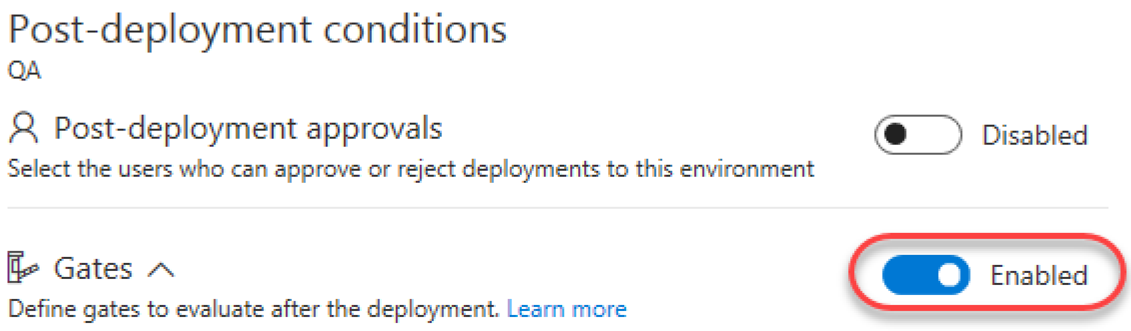
1. Return to the Azure DevOps browser tab.
2. As release pipelines get more sophisticated, it becomes important to define gates to ensure quality throughout the release pipeline. Since the next stage we're deploying to is production, we'll need to be sure to include both automated quality gates as well as a manual approver gate. Return to the release pipeline browser tab and click **Clone** in the **QA** stage. Since the production stage is virtually the same, we can reuse almost all of the existing configuration.



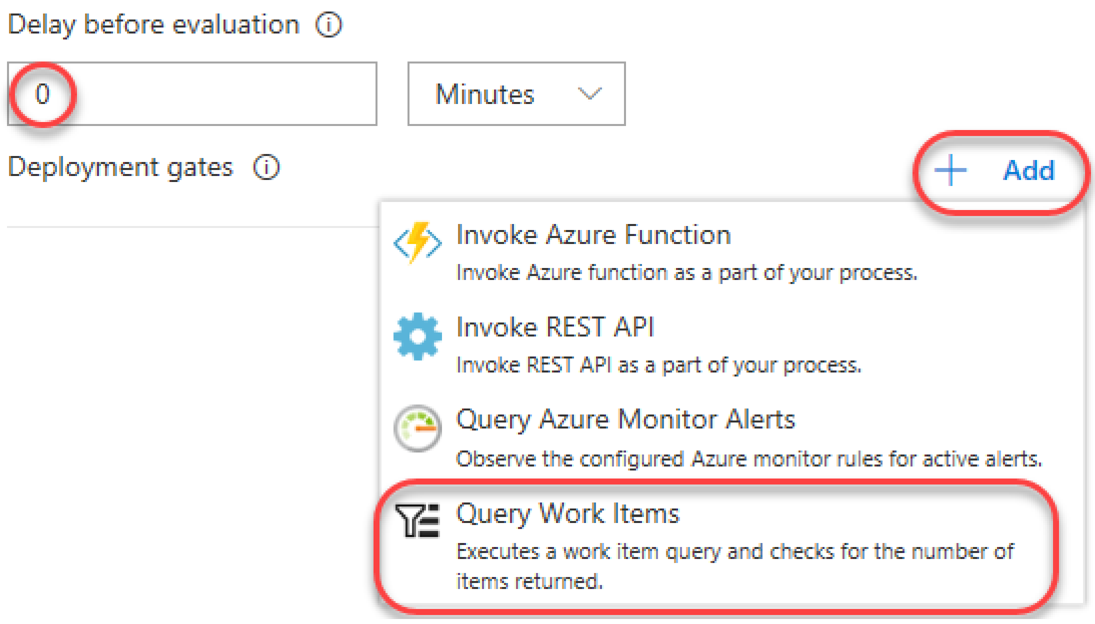
3. The new stage is added after the current one, which is what we want. However, before we can consider the QA deployment successful, we'll need to define a post-deployment condition. Click the **Post-deployment conditions** button on the **QA** stage.



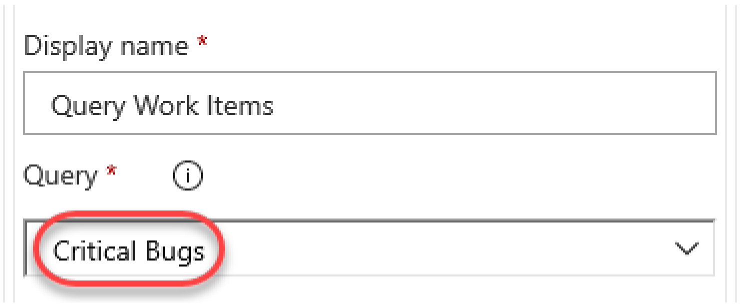
4. **Enable** the **Gates** option.


Post-deployment conditions
QA
Post-deployment approvals
Select the users who can approve or reject deployments to this environment
Disabled
Gates
Define gates to evaluate after the deployment. Learn more
Enabled

5. There are several kinds of gates available that can automatically test virtually anything you need to make sure a deployment is in good shape. These could be the return values of Azure functions or REST APIs, queries to Azure for alerts, or work item queries in Azure DevOps. You can also configure how long the platform should delay before evaluating the gates for the first time. In this case, change that to **0** so it will test them immediately after deployment. Then click **Add | Query Work Items**.


Delay before evaluation
0
Minutes
Deployment gates
Add
Invoke Azure Function
Invoke Azure function as a part of your process.
Invoke REST API
Invoke REST API as a part of your process.
Query Azure Monitor Alerts
Observe the configured Azure monitor rules for active alerts.
Query Work Items
Executes a work item query and checks for the number of items returned.

6. Select the **Query** for **Shared Queries | Critical Bugs**. We will make it our policy that the QA deployment cannot be considered a success until all critical bugs have been resolved.

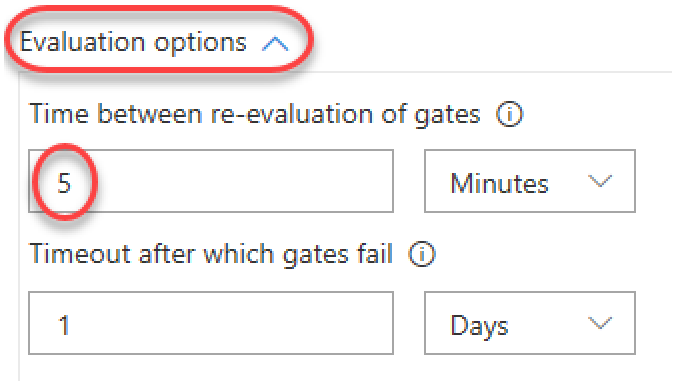

Display name *

Query Work Items

Query *

Critical Bugs

7. Expand **Evaluation options** and update the **Time between re-evaluation of gates** to **5**. If this gate fails, we want it to reevaluate the query every 5 minutes until it clears because engineers will need some time to confirm those critical bugs are fixed in the current version. However, if those bugs aren't cleared and the release isn't manually failed, this configuration will automatically fail the gate after 1 day.


Evaluation options

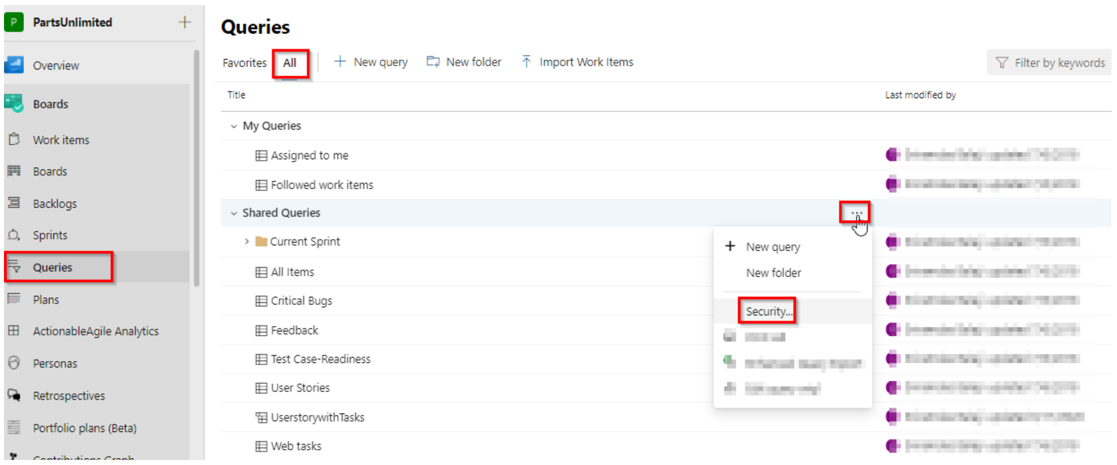
Time between re-evaluation of gates

5 Minutes

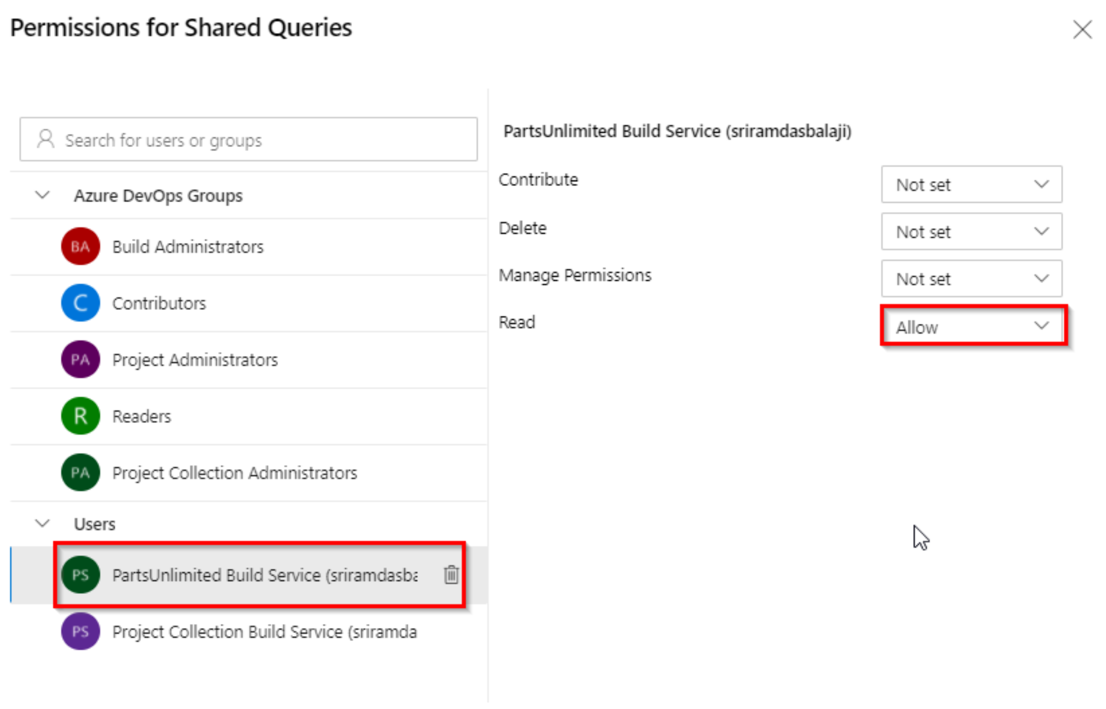
Timeout after which gates fail

1 Days

8. For the Query Gate to work, Project Build Service would require Read permission to queries. Go to **Azure Boards** > **Queries** > **All** > **Shared Queries** > **"..."** > **Security**.



9. Search for the **"(YOUR PROJECT NAME) Build Service "**, if not included by default (not Project Collection Build Service!) and give it **Read** > **Allow** permission.


Permissions for Shared Queries

Search for users or groups

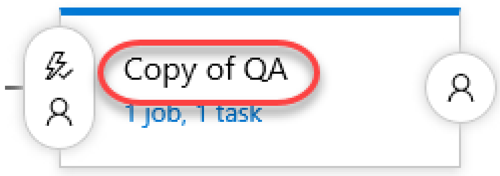
Azure DevOps Groups
- Build Administrators
- Contributors
- Project Administrators
- Readers
- Project Collection Administrators

Users
- PartsUnlimited Build Service (sriramdasba
- Project Collection Build Service (sriramda

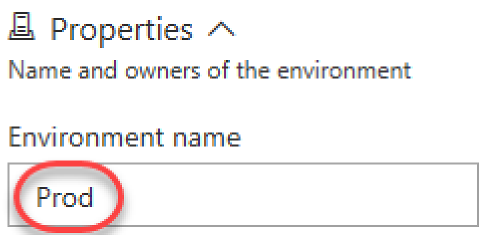
PartsUnlimited Build Service (sriramdasbalaji)

Contribute	Not set
Delete	Not set
Manage Permissions	Not set
Read	Allow

10. We can now turn our focus to the production stage. Select the **Copy of QA** stage.



11. Rename it to **"Prod"**.

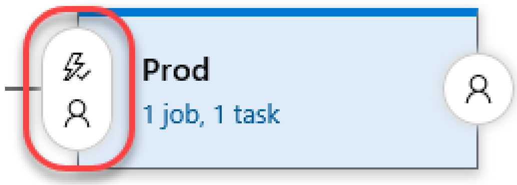

Properties

Name and owners of the environment

Environment name

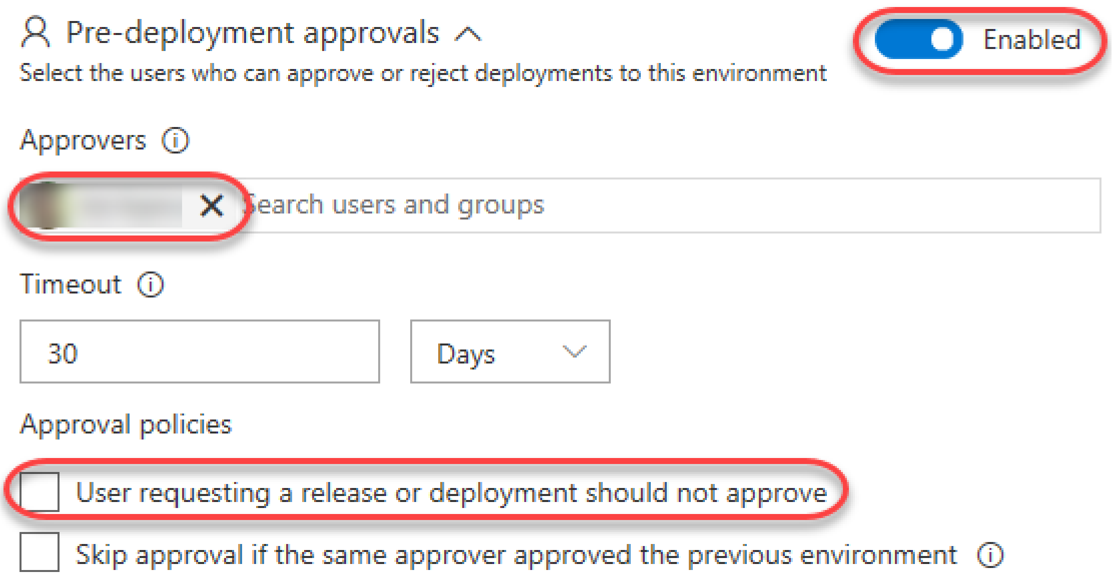
Prod

12. Click its **Pre-deployment conditions** button.

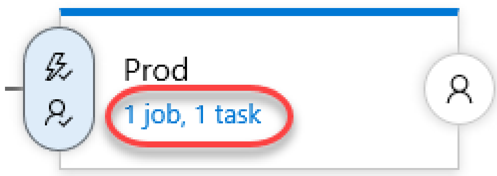


13. **Enable** the **Pre-deployment approvals** and add yourself as an **Approver**. The idea here is that you won't be asked to approve the production deployment until after the QA deployment has succeeded. At that point, someone on this list will need to approve the

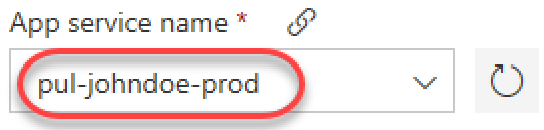
deployment to production. Also clear the box for **User requesting a release of deployment should not approve** if it's checked. For the purposes of this lab, you can approve releases you have requested.


Pre-deployment approvals
Select the users who can approve or reject deployments to this environment
Enabled
Approvers
Search users and groups
Timeout
30
Days
Approval policies
User requesting a release or deployment should not approve
Skip approval if the same approver approved the previous environment

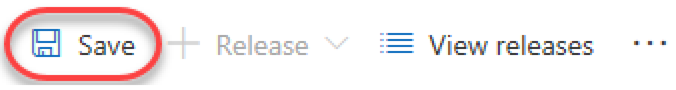
14. Click the **Prod** stage's **1 job, 1 task**.



15. Update the **App service name** to reflect the "**-prod**" (instead of **"-qa"**).

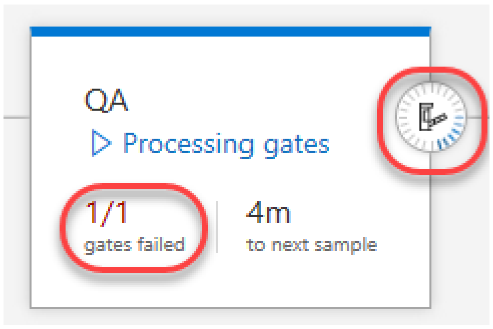

App service name *
pul-johndoe-prod

16. **Save** the release pipeline.

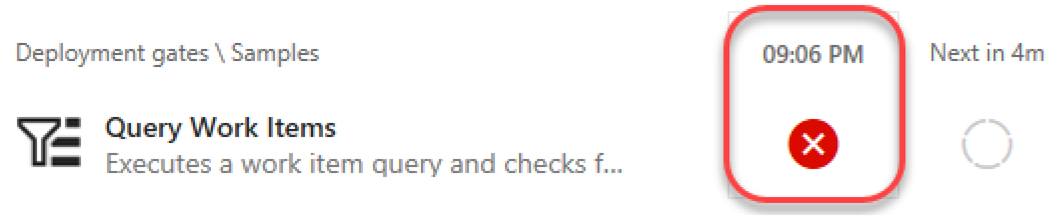


17. Repeat the process for changing the codebase at **"PartsUnlimited-aspnet45/src/PartsUnlimitedWebsite/Views/Shared/_Layout.cshtml"** followed earlier in a new tab. This time, update the version number from **"2.0"** to **"3.0"**. This will invoke the release pipeline.
18. As before, follow the release until it is deploying to QA. Click to follow the release itself once available.

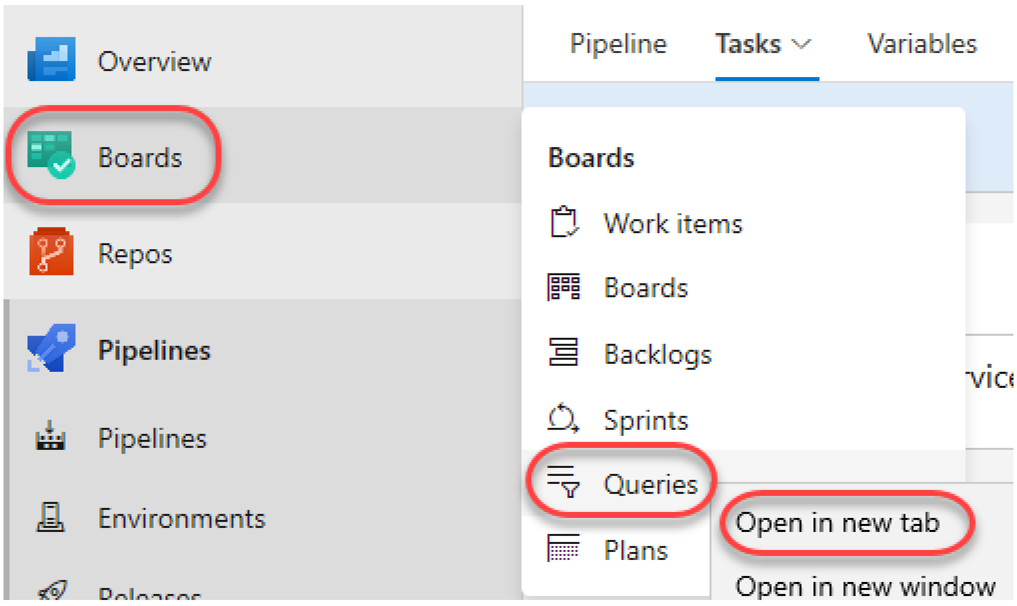
19. Eventually there will be an issue with the QA deployment. While it deployed out successfully, one of the quality gates failed. This will need to be resolved for that stage to be approved. Click the **View post-deployment gates** button.



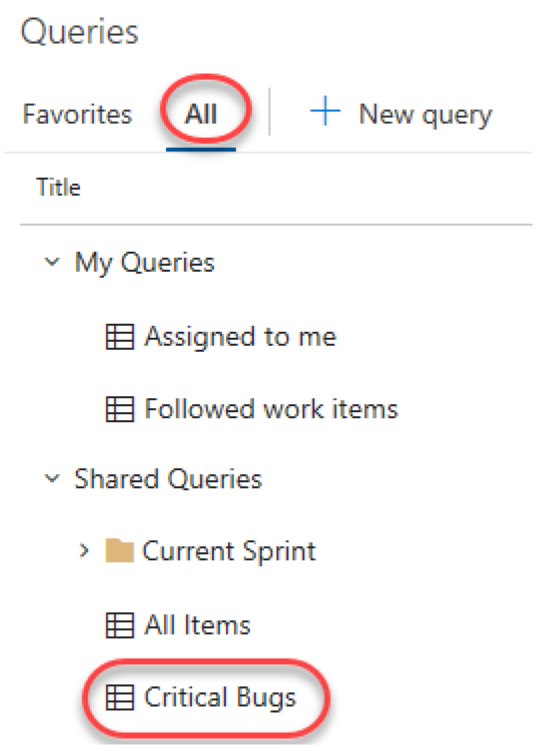
20. It looks like the **Query Work Items** gate is failing. This means that there is a critical bug that must be cleared.


Deployment gates \ Samples
Query Work Items
Executes a work item query and checks f...
09:06 PM
Next in 4m

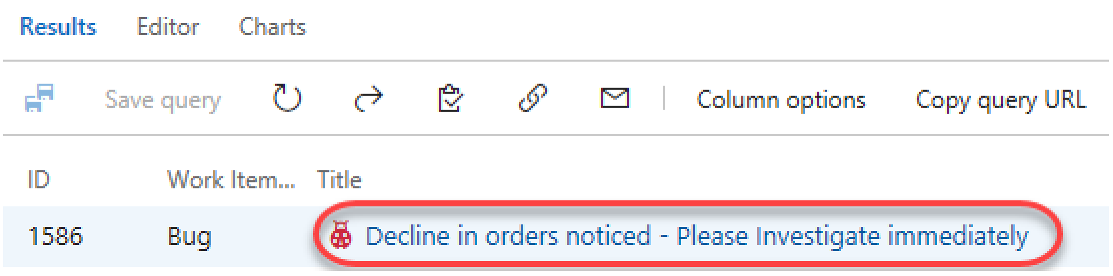
21. Open a new tab to **Boards | Queries** to locate the bug.



22. Use the query list on the **All** tab to open the **Shared Queries | Critical Bugs** query.

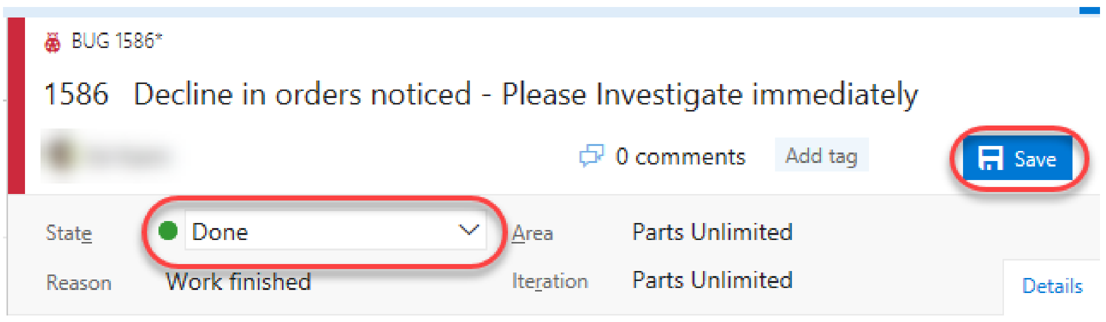


23. Open the one bug by clicking it.



ID	Work Item...	Title
1586	Bug	Decline in orders noticed - Please Investigate immediately

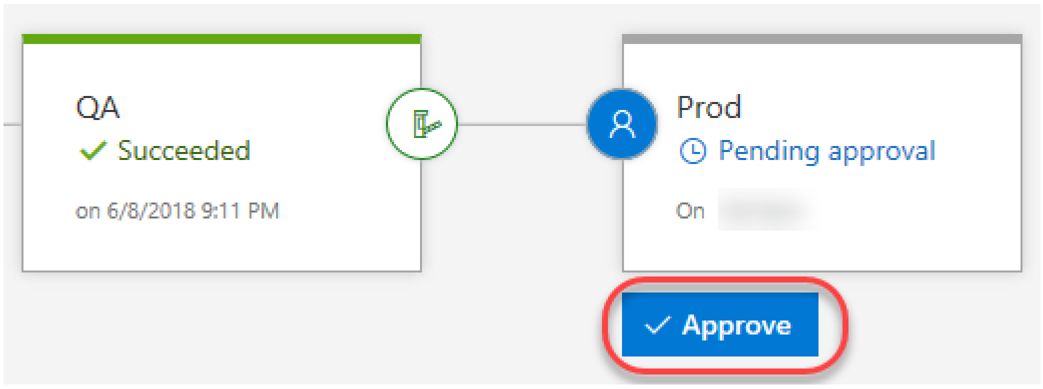
24. Ordinarily you'd check the site to confirm the bug was fixed, but we'll skip ahead here and mark it **Done**. Click **Save** and close the tab.



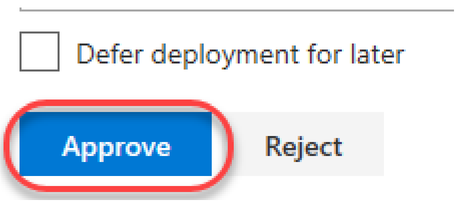
1586 Decline in orders noticed - Please Investigate immediately

State: Done
Reason: Work finished
Area: Parts Unlimited
Iteration: Parts Unlimited

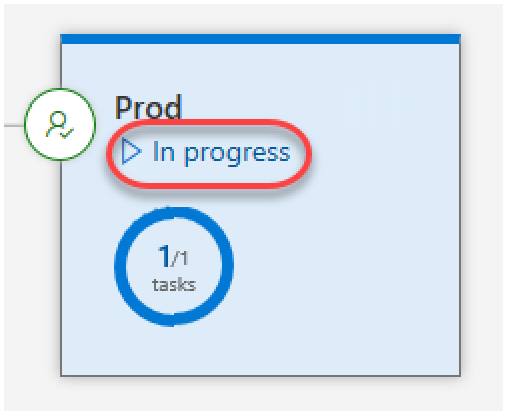
25. Return to the release pipeline tab. Depending on timing, it may take up to five minutes for Azure DevOps to check the query again. Once it does, the bugs will be cleared and the release will be approved. You can then click **Approve** to approve deployment to production.



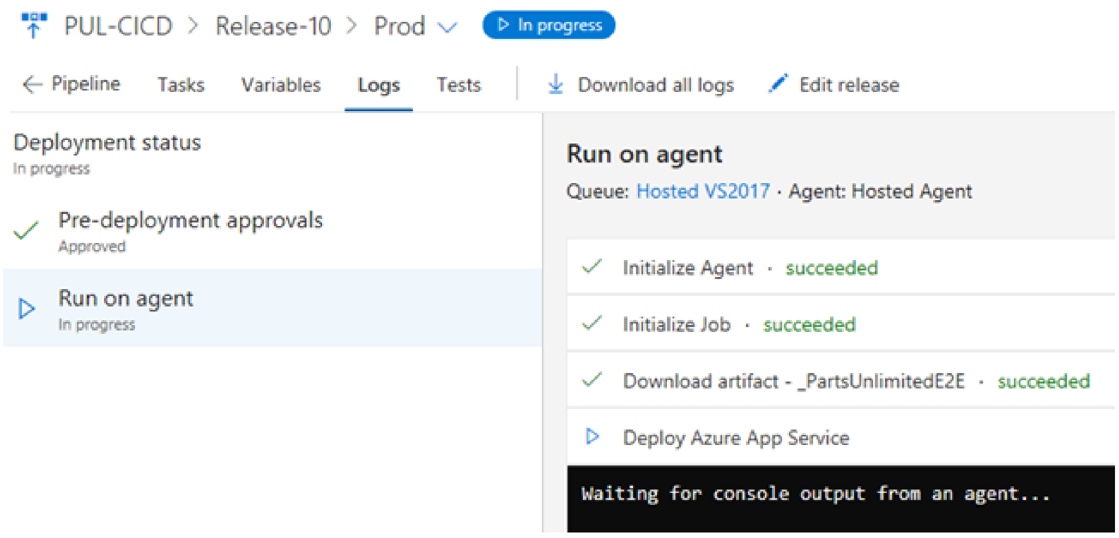
26. Confirm the deployment.



27. Click the **In progress** link to follow the release workflow.



28. It may take a moment for the release to kick off and complete.



Deployment status
In progress

✓ Pre-deployment approvals
Approved

Run on agent
In progress

Run on agent
Queue: Hosted VS2017 · Agent: Hosted Agent

✓ Initialize Agent · succeeded

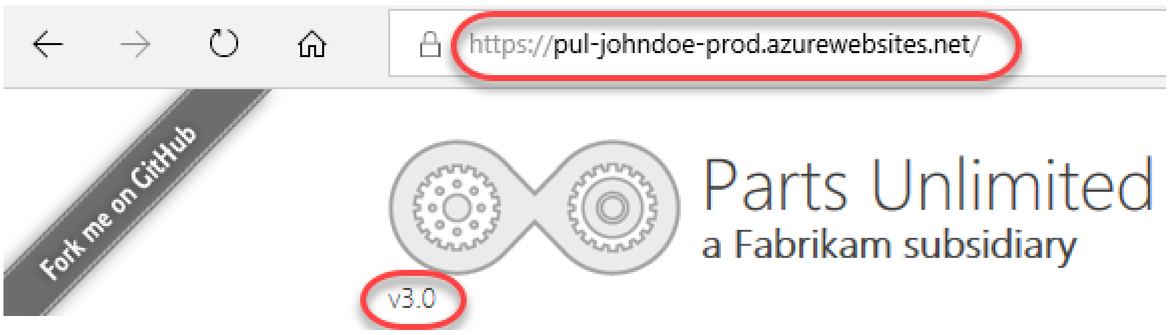
✓ Initialize Job · succeeded

✓ Download artifact - _PartsUnlimitedE2E · succeeded

Deploy Azure App Service

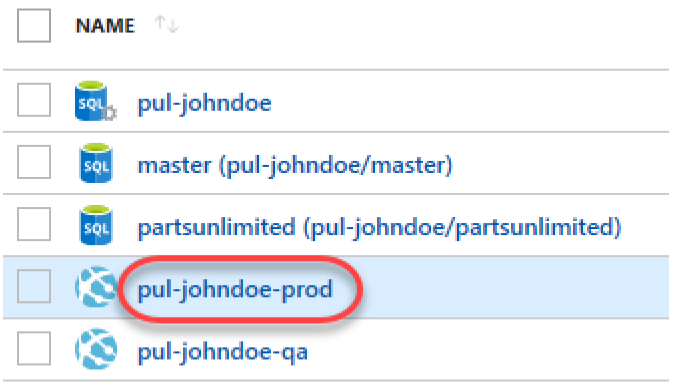
```
Waiting for console output from an agent...
```

29. Once the release has completed, open a new tab to the production app service URL. It should be the same as your QA URL, but with **"-prod"** instead of **"-qa"**. Note the **v3.0**.



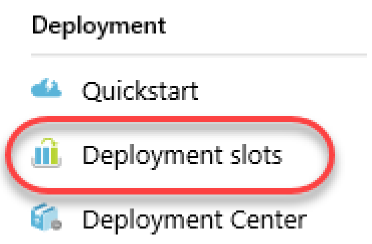
Task 6: Working with deployment slots

1. Return to the browser window open to the Azure portal.
2. Azure App Services offer **deployment slots**, which are parallel targets for application deployment. The most common scenario for using a deployment slot is to have a staging stage for your application to run against productions services, but without replacing the current production application. If the staging deployment passes review, it can immediately be "swapped" in as the production slot with the click of a button. As an additional benefit, the swap can be quickly reversed in the event an issue is uncovered with the new build. Locate the resource group created earlier and click the **prod** app service.

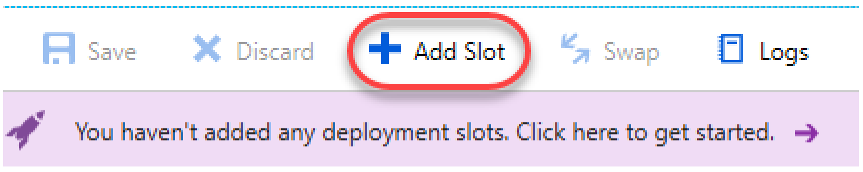


NAME
pul-johndoe
master (pul-johndoe/master)
partsunlimited (pul-johndoe/partsunlimited)
pul-johndoe-prod
pul-johndoe-qa

3. Select the **Deployment slots** tab. If you see a message indicating that your current pricing tier does not support slots, follow the workflow to upgrade this service to a **Production** tier or higher and then return here. You may need to refresh the browser for the **Add Slot** option to become enabled.

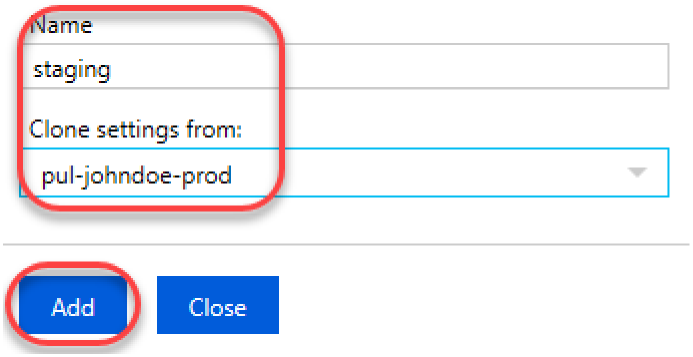


4. Click **Add Slot**. Note that the **production** slot is considered a "default" slot and is not shown as a separate slot in the user experience.



You haven't added any deployment slots. Click here to get started.

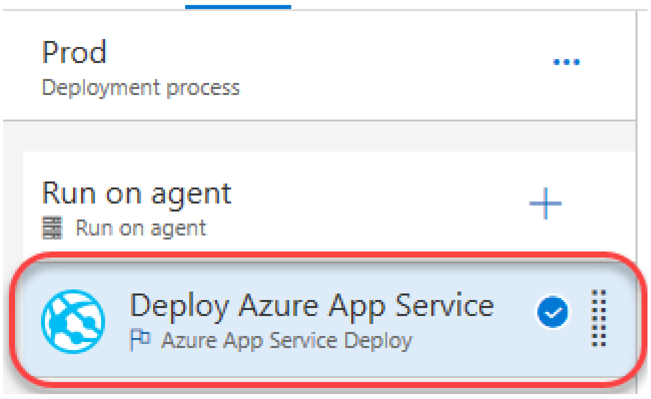
5. Enter a **Name** of **"staging"** and select the **Configuration Source** that matched your existing deployment (there should be only one). Click **Add** to create the slot.



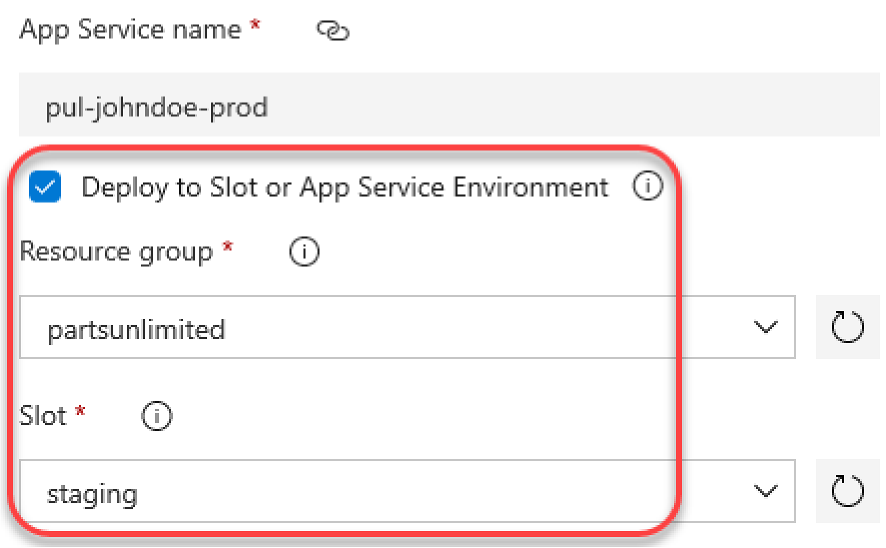
Name: staging

Clone settings from: pul-johndoe-prod

6. Return to the Azure DevOps tab with the **Prod** stage pipeline editor.
7. Select the **Deploy Azure App Service** task.



8. Check **Deploy to Slot…** and set the **Resource group** and **Slot** to those created earlier.


App Service name *

pul-johndoe-prod

Deploy to Slot or App Service Environment

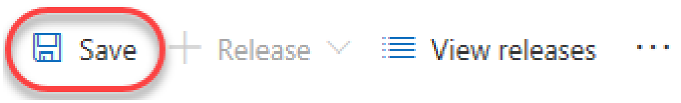
Resource group *

partsunlimited

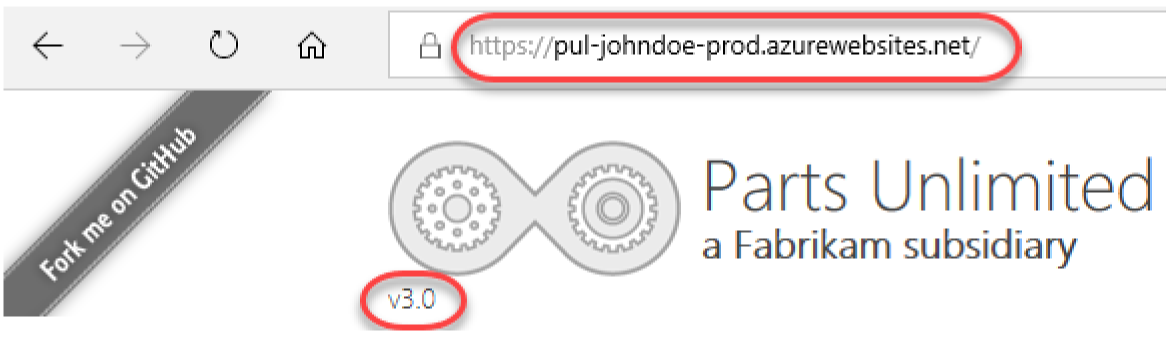
Slot *

staging

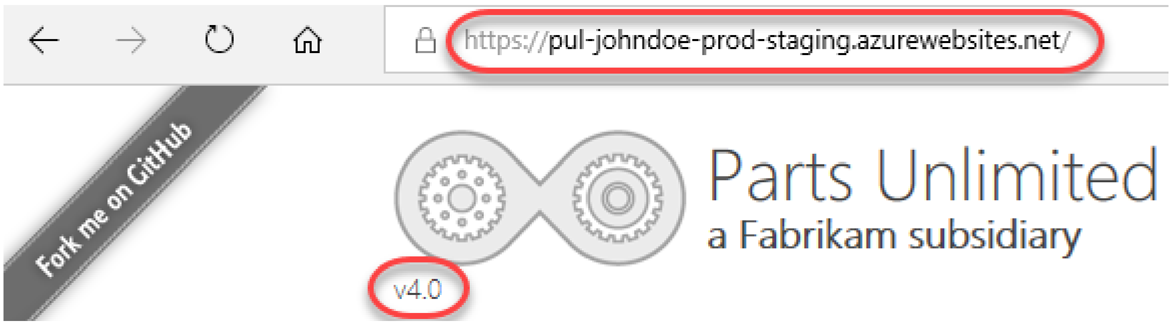
9. **Save** the release pipeline.



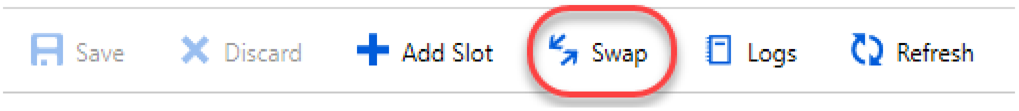
10. Follow the workflow from earlier to commit a change to the codebase at **"PartsUnlimited-aspnet45/src/PartsUnlimitedWebsite/Views/Shared/_Layout.cshtml"** by updating the layout template from **"3.0"** to **"4.0"**.
11. Follow the release pipeline through deployment and approve the release to production when requested.
12. When the production deployment has completed, refresh that browser tab. Note that there shouldn't be any change since the deployment was pushed to a different slot.



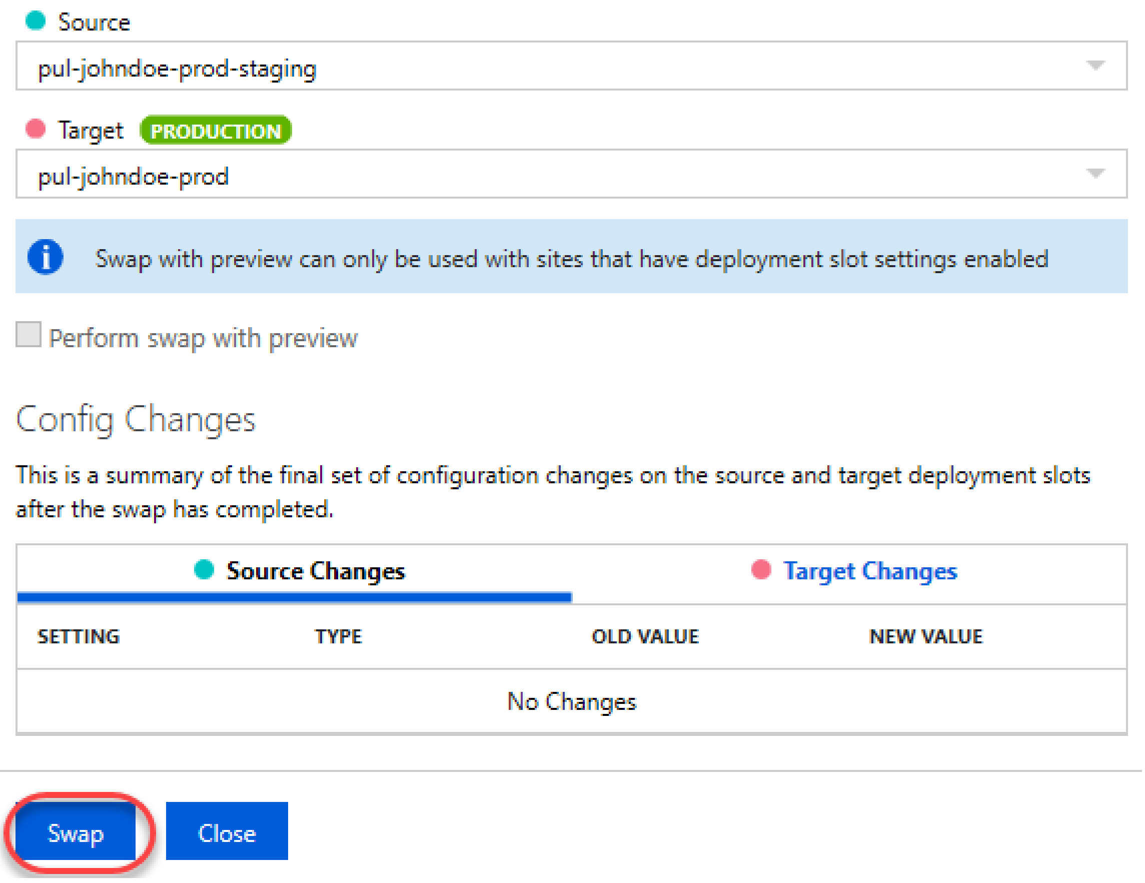
13. Open a new tab to the **staging** slot. This will be the same as your production URL, but with **"-staging"** appended to the app service name within the domain. This should reflect the new **v4.0**.



14. Return to the browser window open to the **Azure portal**. Click **Swap** in the deployment slots blade.



15. The default options here are exactly what we want: to swap the production and staging slots. Click **Swap**. Note that if your apps rely on **slot-level configuration settings** (such as connection strings or app settings marked "slot"), then the worker processes will be restarted. If you're working under those circumstances and would like to warm up the app before the swap completes, you can select the **Swap with preview** swap type.



Source: pul-johndoe-prod-staging

Target PRODUCTION: pul-johndoe-prod

Swap with preview can only be used with sites that have deployment slot settings enabled

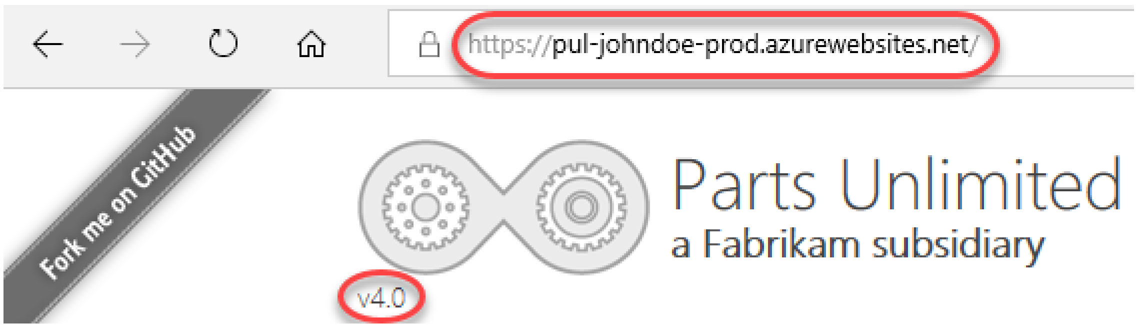
☐ Perform swap with preview

Config Changes

This is a summary of the final set of configuration changes on the source and target deployment slots after the swap has completed.

SETTING	TYPE	OLD VALUE	NEW VALUE
No Changes			

16. Return to the **prod** browser window (not the staging slot) and refresh. It will now be the 4.0 version.


https://pul-johndoe-prod.azurewebsites.net/
Fork me on GitHub
Parts Unlimited
a Fabrikam subsidiary
v4.0