AZ-400: Implement a secure continuous deployment using Azure Pipelines

Introduction to deployment patterns

Introduction

Completed

- 3 minutes

This module introduces deployment patterns and explains microservices architecture to help improve the deployment cycle and examine classical and modern deployment patterns.

Continuous Delivery is an extension of Continuous Integration. It's all about getting changes to customers quickly and using sustainable methods.

Continuous Delivery goes further, and changes that pass through production pipelines are released to customers.

Continuous Delivery is more than release management.

Continuous Delivery is all about the process, the people, and the tools that you need to make sure that you can deliver your software on demand.

Deployment is only one step within the Continuous Delivery process. To deploy on-demand or multiple times a day, all the prerequisites need to be in place.

For example:

Testing strategy

Your testing strategy should be in place. If you need to run many manual tests to validate your software, it is a bottleneck to delivering on-demand.

Coding practices

If your software isn't written in a safe and maintainable manner, the chances are that you can't maintain a high release cadence.

When your software is complex because of a large amount of technical Debt, it's hard to change the code quickly and reliably.

Writing high-quality software and high-quality tests are an essential part of Continuous Delivery.

Architecture

The architecture of your application is always significant. But when implementing Continuous Delivery, it's maybe even more so.

If your software is a monolith with many tight coupling between the various components, it's challenging to deliver your software continuously.

Every part that is changed might impact other parts that didn't change. Automated tests can track many these unexpected dependencies, but it's still hard.

There's also the time aspect when working with different teams. When Team A relies on the service of Team B, Team A can't deliver until Team B is done. It introduces another constraint on delivery.

Continuous Delivery for large software products is complex.

For smaller parts, it's easier. So, breaking up your software into smaller, independent pieces is a good solution in many cases.

One approach to solving these issues is to implement microservices.

Continuous Integration is one of the key pillars of DevOps.

Once you have your code in a version control system, you need an automated way of integrating the code on an ongoing basis.

Azure Pipelines can be used to create a fully featured cross-platform CI and CD service.

It works with your preferred Git provider and can deploy to most major cloud services, including Azure.

This module details continuous integration practice and the pillars for implementing it in the development lifecycle, its benefits, and properties.

Learning objectives

After completing this module, students and professionals can:

- Describe deployment patterns.
- Explain microservices architecture.
- Understand classical and modern deployment patterns.
- Plan and design your architecture.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore microservices architecture

Completed

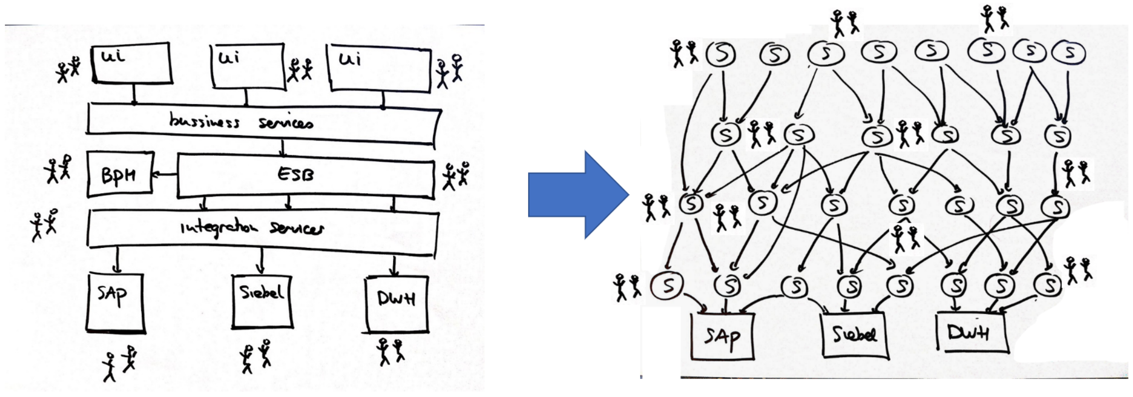
- 2 minutes

Today, you'll frequently hear the term microservices. A microservice is an autonomous, independently deployable, and scalable software component.

They're small, focused on doing one thing well, and can run autonomously. If one microservice changes, it shouldn't impact any other microservices within your landscape.

By choosing a microservices architecture, you'll create a landscape of services that can be developed, tested, and deployed separately. It implies other risks and complexity.

It would be best if you created it to keep track of interfaces and how they interact. And you need to maintain multiple application lifecycles instead of one.



In a traditional application, we can often see a multi-layer architecture.

One layer with the UI, a layer with the business logic and services, and a layer with the data services.

Sometimes there are dedicated teams for the UI and the backend. When something needs to change, it needs to change in all the layers.

When moving towards a microservices architecture, all these layers are part of the same microservice.

Only the microservice contains one specific function.

The interaction between the microservices is done asynchronously.

They don't call each other directly but use asynchronous mechanisms like queues or events.

Each microservice has its lifecycle and Continuous Delivery pipeline. If you built them correctly, you could deploy new microservice versions without impacting other system parts.

Microservice architecture is undoubtedly not a prerequisite for Continuous Delivery, but smaller software components help implement a fully automated pipeline.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

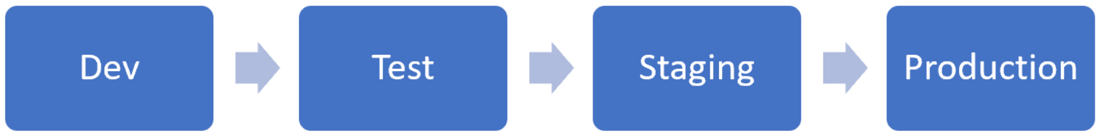
Examine classical deployment patterns

Completed

- 1 minute

When we have our prerequisites to deliver our software continuously, we need to start thinking about a deployment pattern.

Traditionally a deployment pattern was straightforward.



The software was built, and when all features had been implemented, the software was deployed to an environment where a group of people could start using it.

The traditional or classical deployment pattern was moving your software to a development stage, a testing stage, maybe an acceptance or staging stage, and finally a production stage.

The software moved as one piece through the stages.

The production release was, in most cases, a Big Bang release, where users were confronted with many changes at the same time.

Despite the different stages to test and validate, this approach still involves many risks.

By running all your tests and validation on non-production environments, it's hard to predict what happens when your production users start using it.

You can run load tests and availability tests, but in the end, there's no place like production.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Understand modern deployment patterns

Completed

- 1 minute

End-users always use your application differently. Unexpected events will happen in a data center, multiple events from multiple users will cooccur, triggering some code that hasn't been tested in that way.

To overcome, we need to embrace that some features can only be tested in production.

Testing in production sounds a bit scary, but that shouldn't be the case.

When we talked about separating our functional and technical releases, we already saw that it's possible to deploy features without exposing them to all users.

When we take this concept of feature toggling and use it with our deployment patterns, we can test our software in production.

For example:

- Blue-green deployments.
- Canary releases.
- Dark launching.
- A/B testing.
- Progressive exposure or ring-based deployment.
- Feature toggles.

Take a critical look at your architecture

Are your architecture and the current state of your software ready for Continuous Delivery?

Topics you might want to consider are:

- Is your software built as one giant monolith, or is it divided into multiple components?
- Can you deliver parts of your application separately?
- Can you guarantee the quality of your software when deploying multiple times a week?
- How do you test your software?

- Do you run one or multiple versions of your software?
- Can you run multiple versions of your software side by side?
- What do you need to improve to implement Continuous Delivery?

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices describe microservice?

○

A microservice is a multi-function software to deliver more than one thing very well.

○

A microservice is software built as one monolith.

○

A microservice is an autonomous, independently deployable, and scalable software component.

2.

Which of the following choices is a microservice characteristic?

○

Each microservice can be tested based on other microservice results.

○

If one microservice changes, it shouldn't impact any other microservices within your landscape.

○

If one microservice changes, it affects other services, and you need to wait for other services to go online.

3.

Which of the following choices represent a classical deployment pattern?

○

Dev, Test, Staging, and Production deployment.

○

Blue-green deployments.

○

A/B testing.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module-introduced deployment patterns and explained microservices architecture to help improve the deployment cycle and examine classical and modern deployment patterns.

You learned how to describe the benefits and usage of:

- Describe deployment patterns.
- Explain microservices architecture.
- Understand classical and modern deployment patterns.
- Plan and design your architecture.

Learn more

- Deployment jobs - Azure Pipelines | Microsoft Docs .

- What are Microservices? - Azure DevOps | Microsoft Docs .
- Design a CI/CD pipeline-using Azure DevOps - Azure Example Scenarios | Microsoft Docs .

Continue

Implement blue-green deployment and feature toggles

Introduction

Completed

- 1 minute

This module describes the blue-green deployment process and introduces feature toggle techniques to implement in the development process.

Learning objectives

After completing this module, students and professionals can:

- Explain deployment strategies.
- Implement blue-green deployment.
- Understand deployment slots.
- Implement and manage feature toggles.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.
- You need to create an Azure DevOps Organization and a Team Project for some exercises. If you don't have it yet, see: Create an organization - Azure DevOps .
    - If you already have your organization created, use the Azure DevOps Demo Generator and create a new Team Project called "Parts Unlimited" using the template "PartsUnlimited." Or feel free to create a blank project. See Create a project - Azure DevOps .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

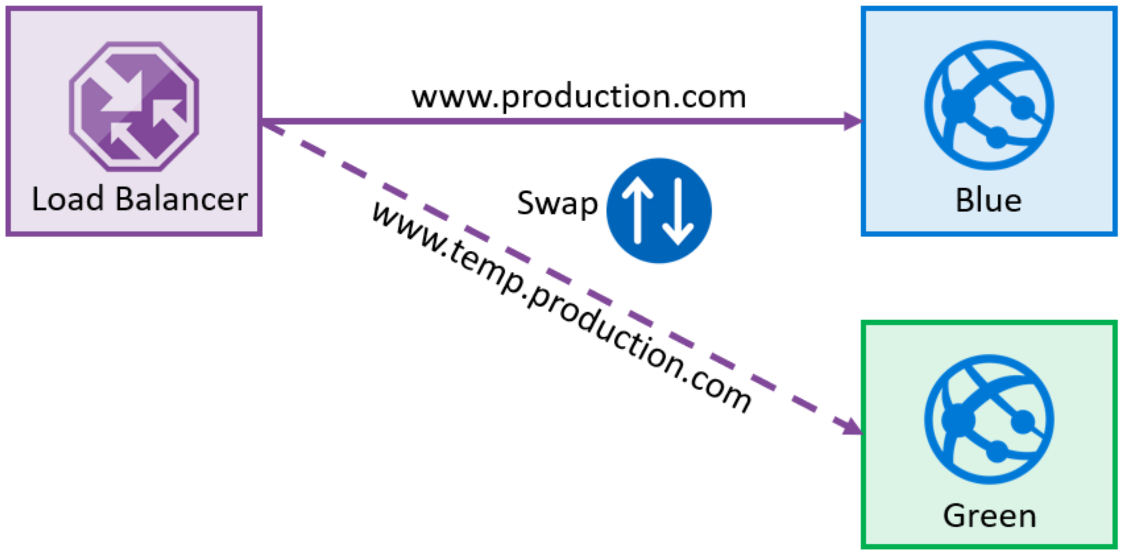
What is blue-green deployment?

Completed

- 3 minutes

Blue-green deployment is a technique that reduces risk and downtime by running two identical environments. These environments are called blue and green.

Only one of the environments is live, with the live environment serving all production traffic.



For this example, blue is currently live, and green is idle.

As you prepare a new version of your software, the deployment and final testing stage occur in an environment that isn't live: in this example, green. Once you've deployed and thoroughly tested the software in green, switch the router or load balancer so all incoming requests go to green instead of blue.

Green is now live, and blue is idle.

This technique can eliminate downtime because of app deployment. Besides, blue-green deployment reduces risk: if something unexpected happens with your new version on the green, you can immediately roll back to the last version by switching back to blue.

When it involves database schema changes, this process isn't straightforward. You probably can't swap your application. In that case, your application and architecture should be built to handle both the old and the new database schema.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore deployment slots

Completed

- 3 minutes

When using a cloud platform like Azure, doing blue-green deployments is relatively easy. You don't need to write your code or set up infrastructure. You can use an out-of-the-box feature called deployment slots when using web apps.

Deployment slots are a feature of Azure App Service. They're live apps with their hostnames. You can create different slots for your application (for example, Dev, Test, or Stage). The production slot is the slot where your live app stays. You can validate app changes in staging with deployment slots before swapping them with your production slot.

You can use a deployment slot to set up a new version of your application, and when ready, swap the production environment with the new staging environment. It's done by an internal swapping of the IP addresses of both slots.

Swap

The swap eliminates downtime when you deploy your app with seamless traffic redirection, and no requests are dropped because of swap operations.

To learn more about Deployment slots and swap, see also:

- Set up Staging Environments in Azure App Service .
- Considerations on using Deployment Slots in your DevOps Pipeline .
- What happens during a swap.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Exercise - set up a blue-green deployment

Completed

- 60 minutes

In this demonstration, you'll investigate Blue-Green Deployment.

Steps

Let's now look at how a release pipeline can be used to implement blue-green deployments.

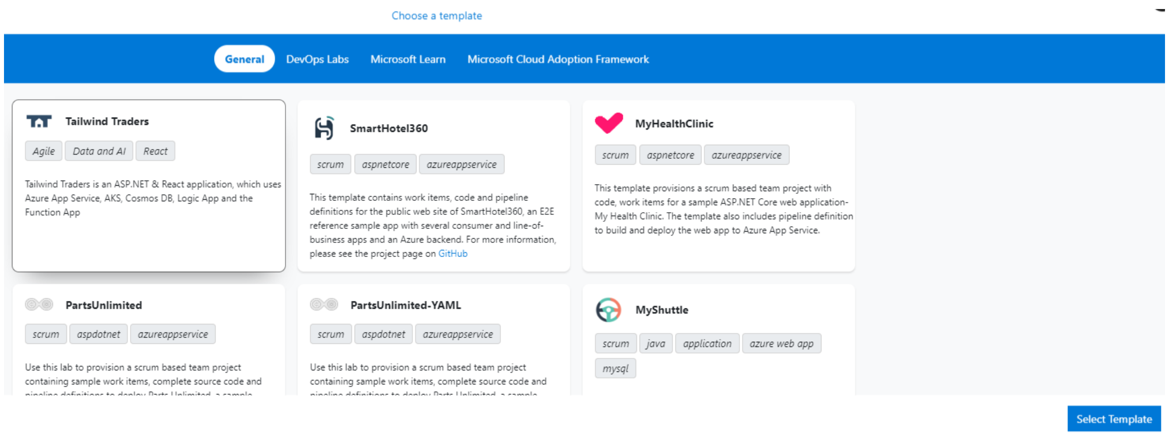
We'll start by creating a new project with a release pipeline that can deploy the **Parts Unlimited** template again.

An initial app deployment

1. Navigate to Azure DevOps Demo Generator in a browser: https://azuredevopsdemogenerator.azurewebsites.net and click **Sign in** .

   You'll be prompted to sign in if necessary.

2. In the **Create New Project** window, select your existing Organization, set the **Project Name** to **PU Hosted,** and click **Choose template** .



Tailwind Traders
Agile | Data and AI | React
Tailwind Traders is an ASP.NET & React application, which uses Azure App Service, AKS, Cosmos DB, Logic App and the Function App

SmartHotel360
scrum | aspnetcore | azureappservice
This template contains work items, code and pipeline definitions for the public web site of SmartHotel360, an E2E reference sample app with several consumer and line-of-business apps and an Azure backend. For more information, please see the project page on GitHub

MyHealthClinic
scrum | aspnetcore | azureappservice
This template provisions a scrum based team project with code, work items for a sample ASP.NET Core web application-My Health Clinic. The template also includes pipeline definition to build and deploy the web app to Azure App Service.

PartsUnlimited
scrum | aspdotnet | azureappservice
Use this lab to provision a scrum based team project containing sample work items, complete source code and

PartsUnlimited-YAML
scrum | aspdotnet | azureappservice
Use this lab to provision a scrum based team project containing sample work items, complete source code and

MyShuttle
scrum | java | application | azure web app | mysql

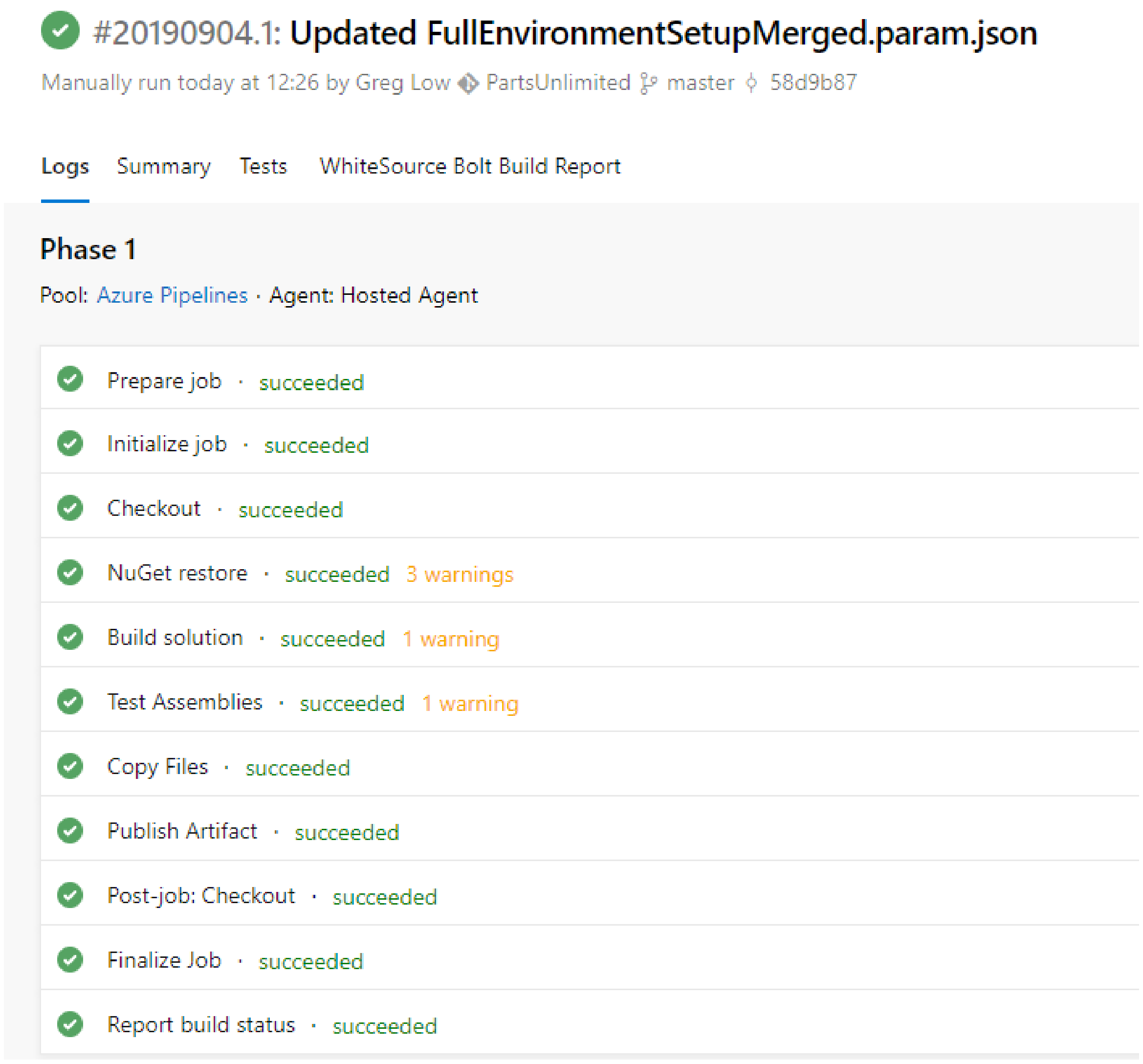
3. Click on the **PartsUnlimited** project (not the PartsUnlimited-YAML project), click **Select Template** , and click **Create Project** . When the deployment completes, click **Navigate to the project** .

4. In the main menu for **PU Hosted** , click **Pipelines** , then click **Builds** , then **Queue,** and finally **Run** to start a build.

   The build should succeed.

   Note

   Warnings might appear but can be ignored for this walkthrough.



#20190904.1: Updated FullEnvironmentSetupMerged.param.json

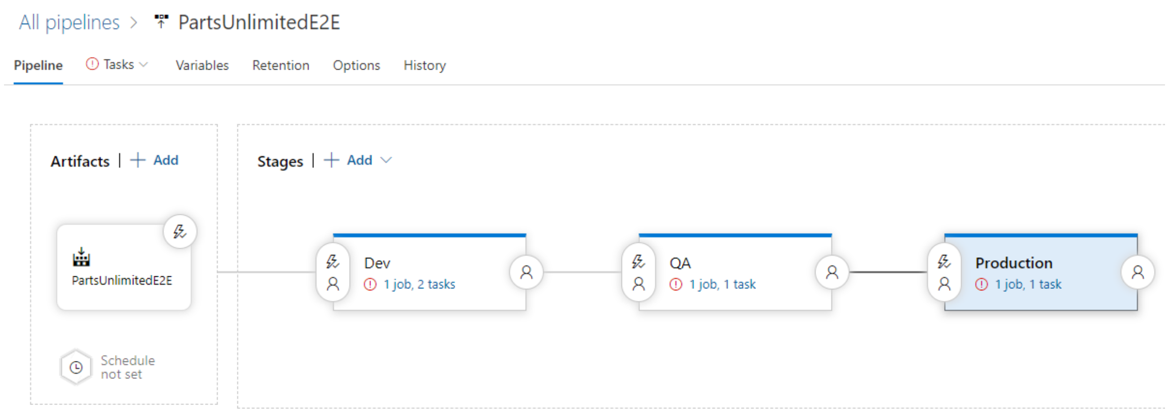
Manually run today at 12:26 by Greg Low PartsUnlimited master 58d9b87

Phase 1

Pool: Azure Pipelines · Agent: Hosted Agent

```
Prepare job · succeeded
Initialize job · succeeded
Checkout · succeeded
NuGet restore · succeeded 3 warnings
Build solution · succeeded 1 warning
Test Assemblies · succeeded 1 warning
Copy Files · succeeded
Publish Artifact · succeeded
Post-job: Checkout · succeeded
Finalize Job · succeeded
Report build status · succeeded
```

5. In the main menu, click **Releases** . Because a continuous integration trigger was in place, a release was attempted. However, we haven't yet configured the release so it will have failed. Click **Edit** to enter edit mode for the release.

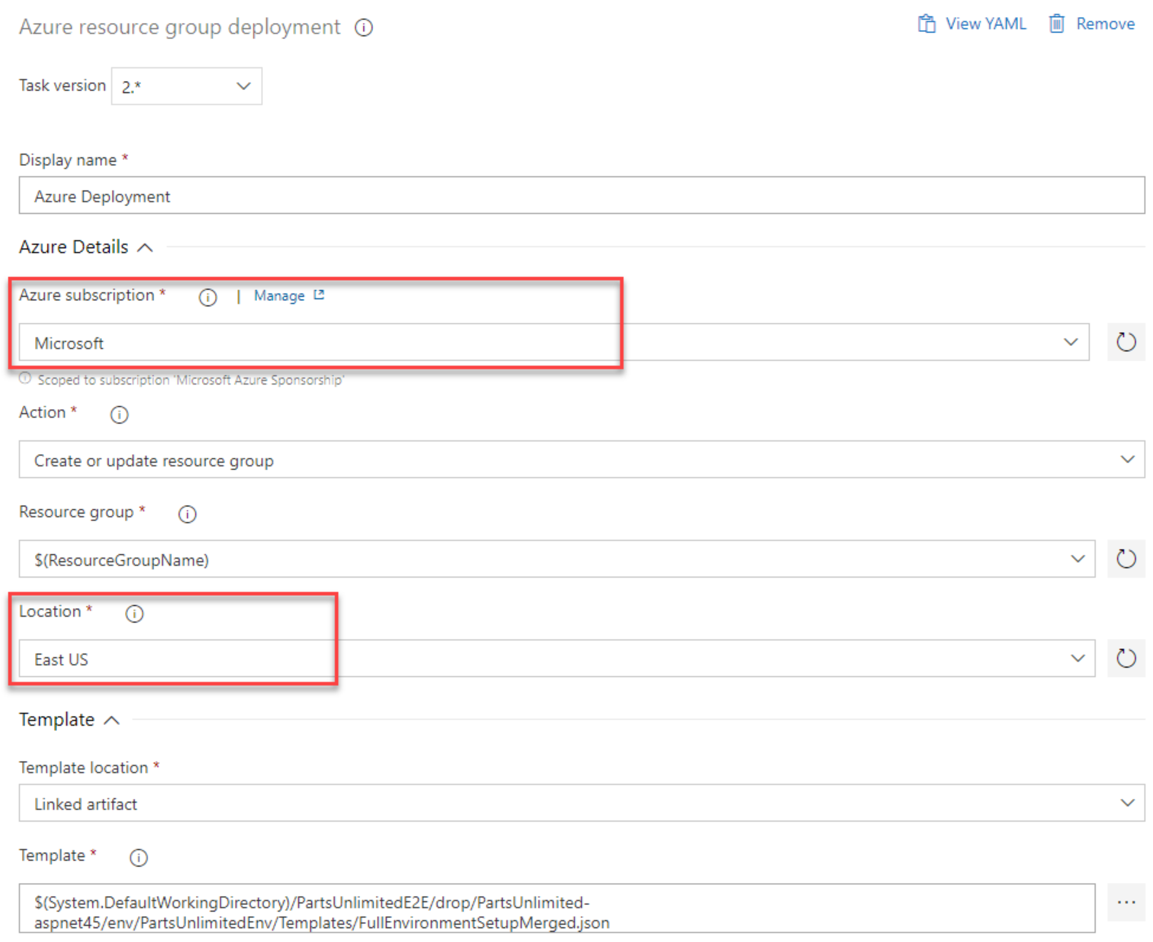


6. Select the Dev stage from the drop-down list beside **Tasks** , then click to select the **Azure Deployment** task.

7. In the **Azure resource group deployment** pane, select your Azure subscription, then click **Authorize** when prompted. When authorization completes, choose a **Location** for the web app.

   Note

   You might be prompted to sign in to Azure at this point.



Azure resource group deployment

Task version 2.*

Display name * Azure Deployment

Azure Details

Azure subscription * | Manage

Microsoft

Scoped to subscription 'Microsoft Azure Sponsorship'

Action * Create or update resource group

Resource group * $(ResourceGroupName)

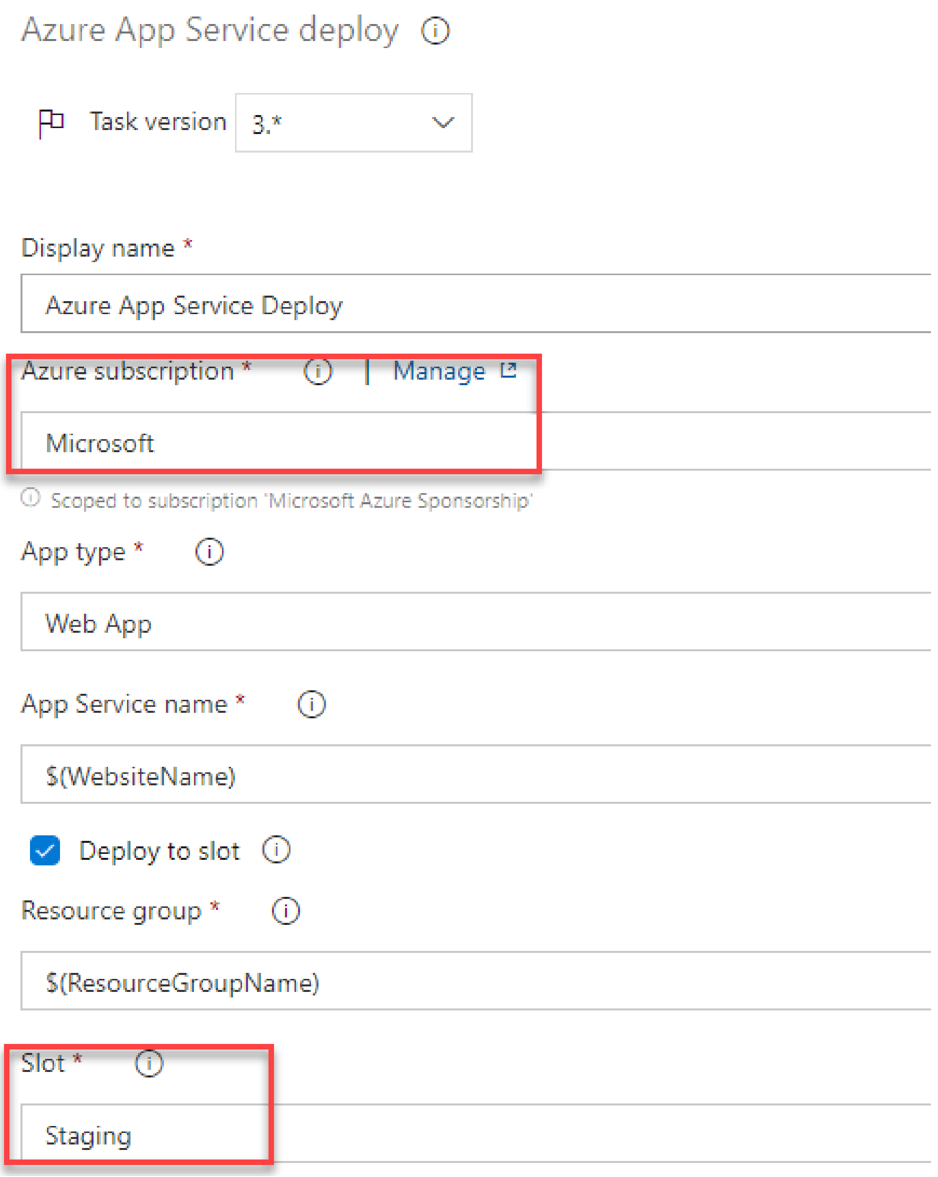
Location * East US

Template

Template location * Linked artifact

Template * $(System.DefaultWorkingDirectory)/PartsUnlimitedE2E/drop/PartsUnlimited-aspnet45/env/PartsUnlimitedEnv/Templates/FullEnvironmentSetupMerged.json

8. Click **Azure App Service Deploy** in the task list to open its settings. Again, select your Azure subscription. Set the **Deployment slot** to **Staging** .


Azure App Service deploy

Task version: 3.*

Display name *: Azure App Service Deploy

Azure subscription *: Microsoft

Scoped to subscription 'Microsoft Azure Sponsorship'

App type *: Web App

App Service name *: $(WebsiteName)

☑ Deploy to slot

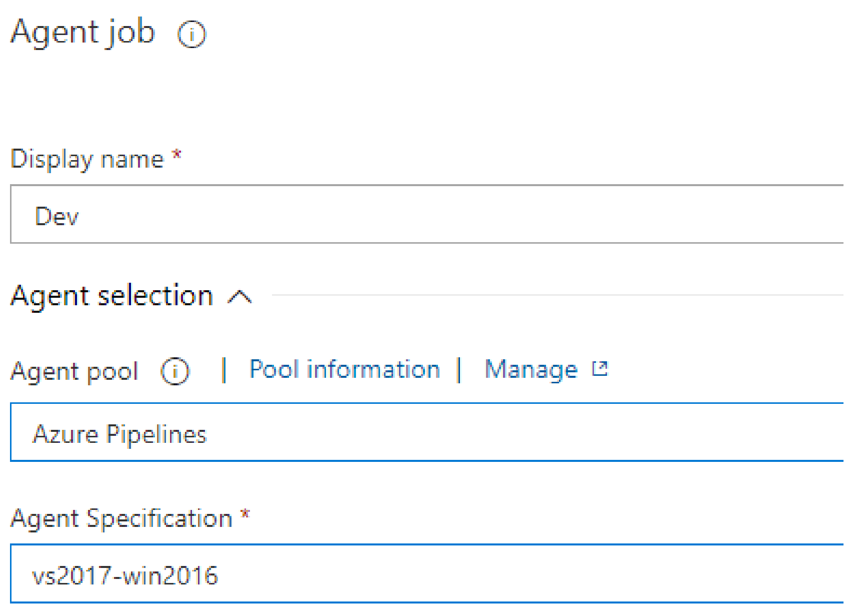
Resource group *: $(ResourceGroupName)

Slot *: Staging

Note

The template creates a production site and two deployment slots: Dev and Staging. We'll use Staging for our Green site.

9. In the task list, click **Dev,** and in the **Agent job** pane, select **Azure Pipelines** for the **Agent pool** and **windows-latest** for the **Agent Specification** .


Agent job

Display name *

Dev

Agent selection

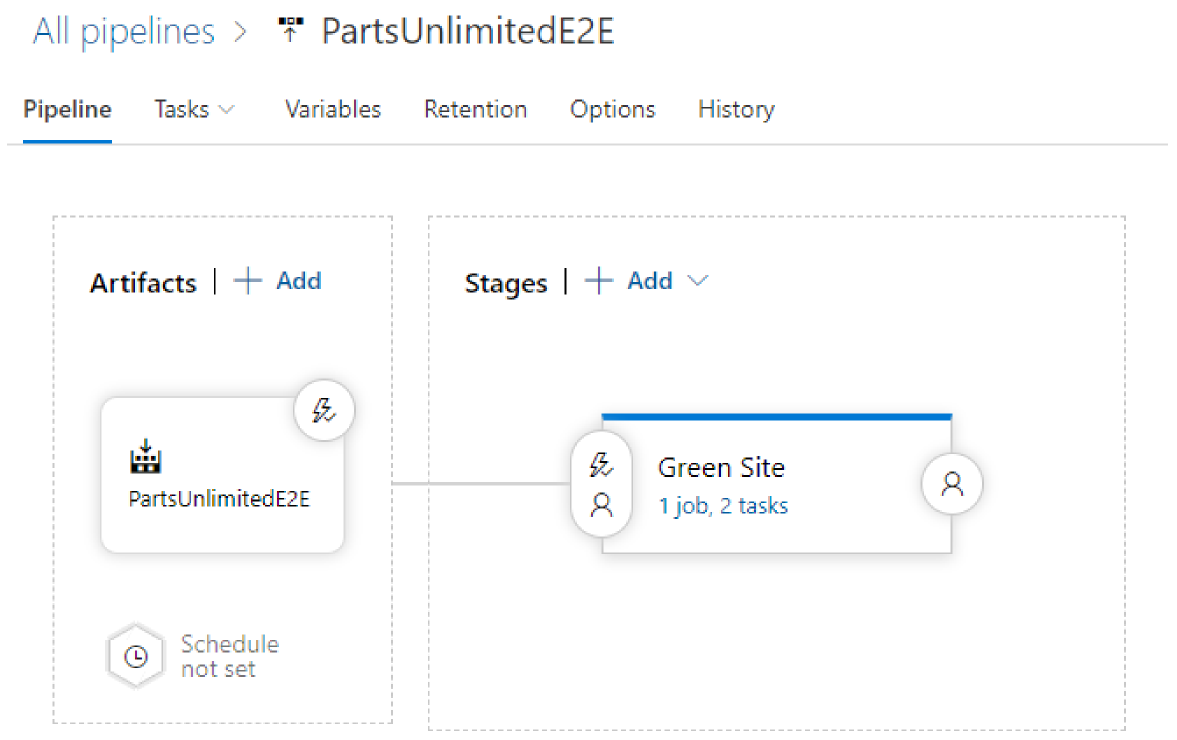
Agent pool | Pool information | Manage

Azure Pipelines

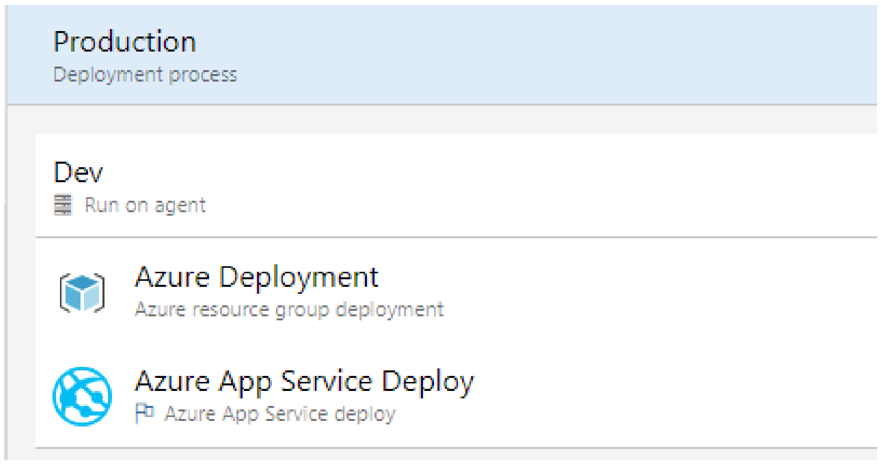
Agent Specification *

vs2017-win2016

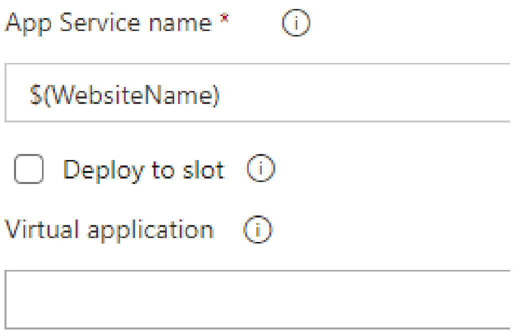
10. From the top menu, click **Pipelines** . Click the **Dev** stage, and in the properties window, rename it to **Green Site** . Click the **QA** stage and click **Delete** and **Confirm** . Click the **Production** stage and click **Delete** and **Confirm** . Click **Save,** then **OK** .



11. Hover over the **Green Site** stage and click the **Clone** icon when it appears. Change the **Stage name** to **Production** . From the **Tasks** drop-down list, select **Production** .

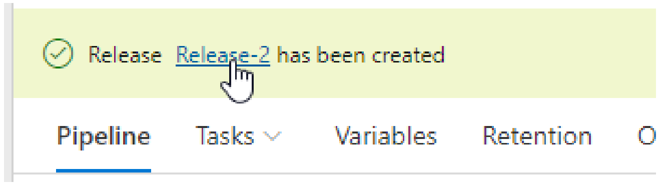

Production
Deployment process
Dev
Run on agent
Azure Deployment
Azure resource group deployment
Azure App Service Deploy
Azure App Service deploy

12. Click the **Azure App Service Deploy** task and uncheck the **Deploy to slot** option. Click **Save** and **OK** .

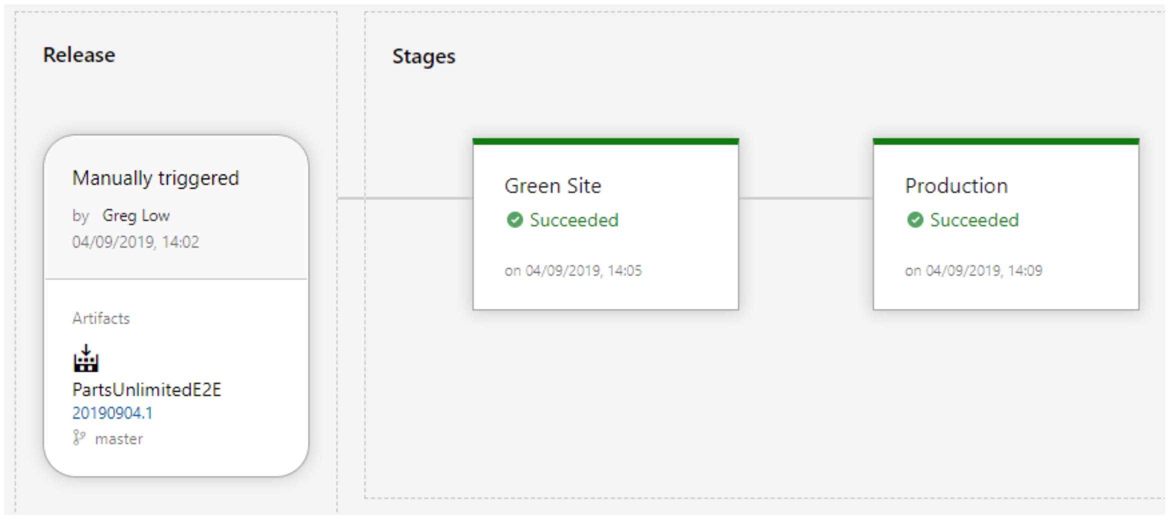

App Service name *
$(WebsiteName)
☐ Deploy to slot
Virtual application

The production site isn't deployed to a deployment slot. It's deployed to the main site.

13. Click **Create release,** then **Create** to create the new release. When created, click the release link to view its status.


Release Release-2 has been created

After a while, the deployment should succeed.



Release

Manually triggered
by Greg Low
04/09/2019, 14:02

Artifacts
PartsUnlimitedE2E
20190904.1
master

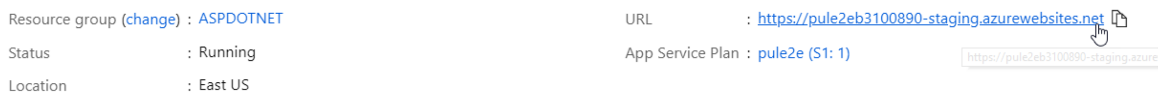
Stages

Green Site
Succeeded
on 04/09/2019, 14:05

Production
Succeeded
on 04/09/2019, 14:09

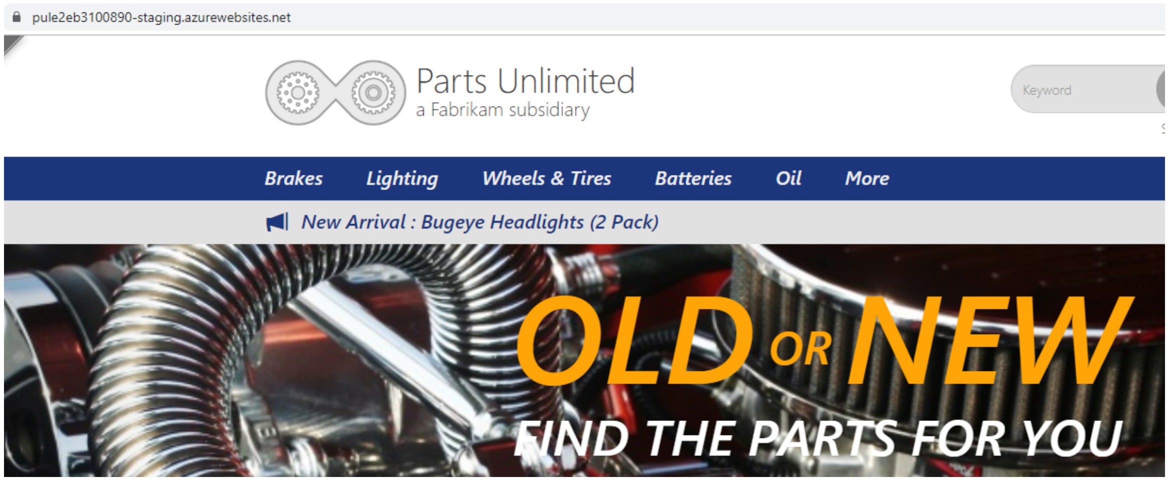
Test the green site and the production site

14. Open the blade for the **ASPDOTNET** resource group created by the project deployment in the Azure portal. Notice the names of the web apps that have been deployed. Click to open the *Staging* * web app's blade. Copy the URL from the top left-hand side.



Resource group (change)	: ASPDOTNET
Status	: Running
Location	: East US
URL	: https://pule2eb3100890-staging.azurewebsites.net
App Service Plan	: pule2e (S1: 1)

15. Open a new browser tab and navigate to the copied URL. It will take the application a short while to compile, but then the Green website (on the Staging slot) should appear.

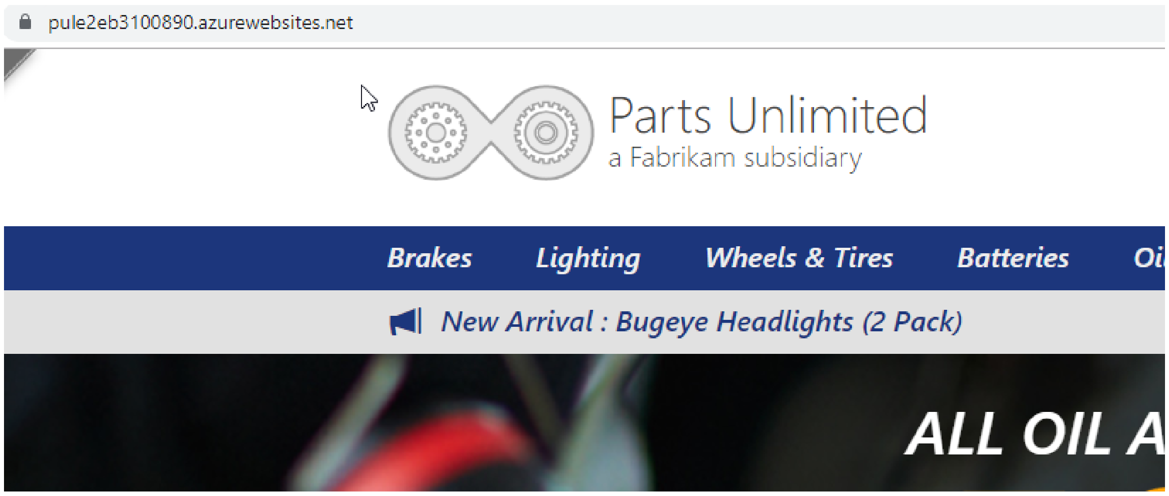


pule2eb3100890-staging.azurewebsites.net

Note

You can tell that the staging slot is being used because of the ***-staging*** *suffix in the website URL.*

16. Open another new browser tab and navigate to the same URL but without the **-staging** slot. The production site should also be working.



Note

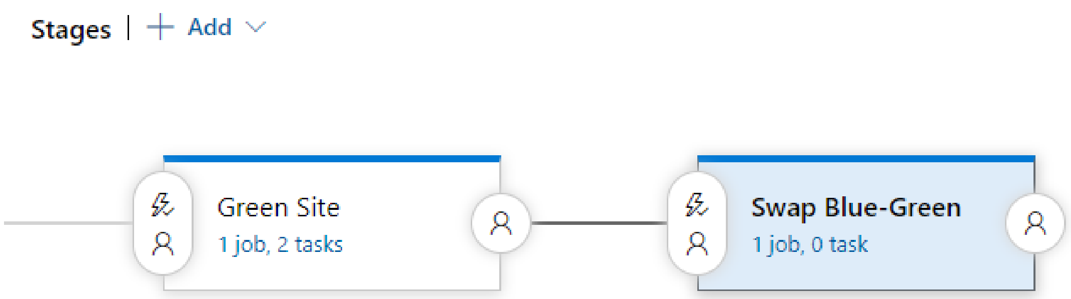
Leave both browser windows open for later in the walkthrough.

Configure blue-green swap and approval

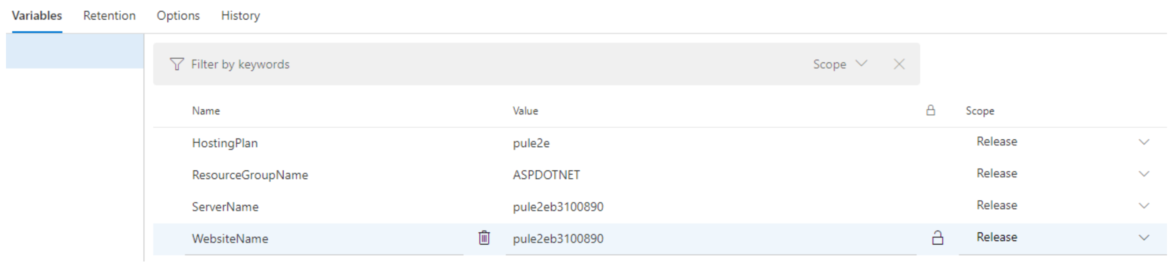
Now that both sites are working, let's configure the release pipeline for blue-green deployment.

17. In **Azure DevOps** , in the main menu for the **PU Hosted** project, click **Pipelines** , then click **Releases** , then click **Edit** to return to edit mode.

18. Click the **Production** stage, click **Delete** , then **Confirm** to remove it. Click **+Add** to add an extra stage and click **Empty job** for the template. Set **Swap Blue-Green** for the **Stage name** .

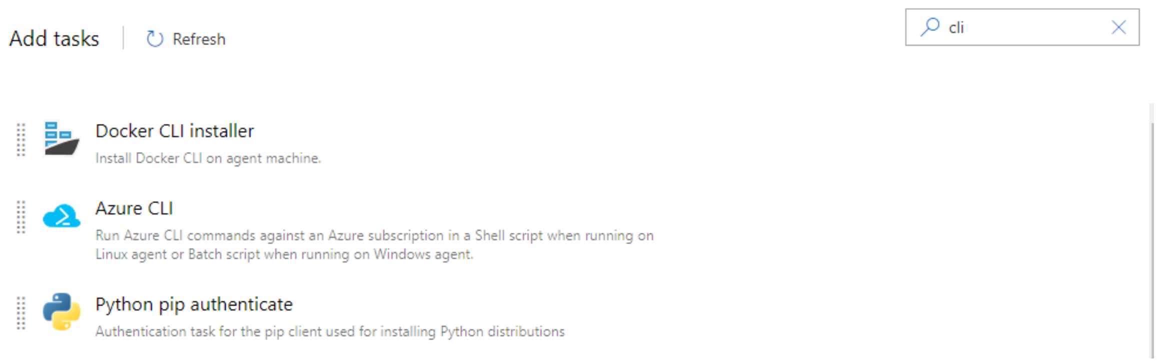


19. Click **Variables** and modify the **Scope** of **WebsiteName** to **Release** .



Name	Value		Scope
HostingPlan	pule2e		Release
ResourceGroupName	ASPDOTNET		Release
ServerName	pule2eb3100890		Release
WebsiteName	pule2eb3100890		Release

20. From the **Tasks** drop-down list, click to select the **Swap Blue-Green** stage. Click the **+** to the right-hand side of **Agent Job** to add a new task. In the **Search** box, type **CLI** .

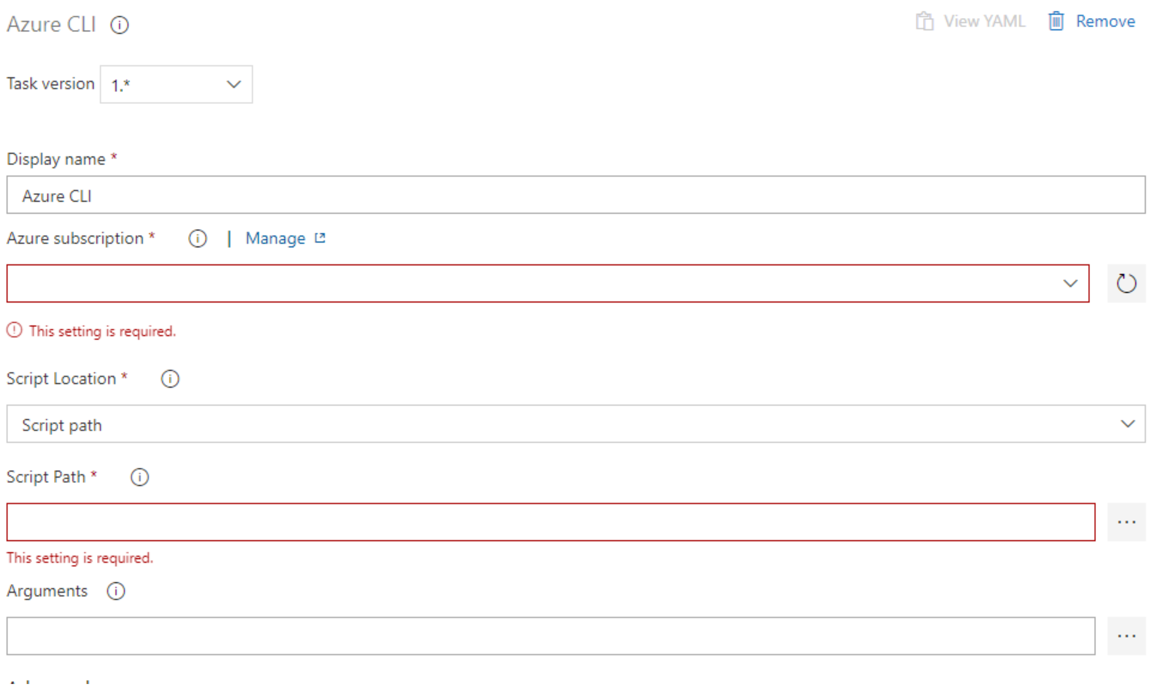


Docker CLI installer
Install Docker CLI on agent machine.

Azure CLI
Run Azure CLI commands against an Azure subscription in a Shell script when running on Linux agent or Batch script when running on Windows agent.

Python pip authenticate
Authentication task for the pip client used for installing Python distributions

21. Hover over the **Azure CLI** template and when the **Add** button appears, click it, then click to select the **Azure CLI** task to open its settings pane.



Task version 1.*

Display name * Azure CLI

Azure subscription *

This setting is required.

Script Location * Script path

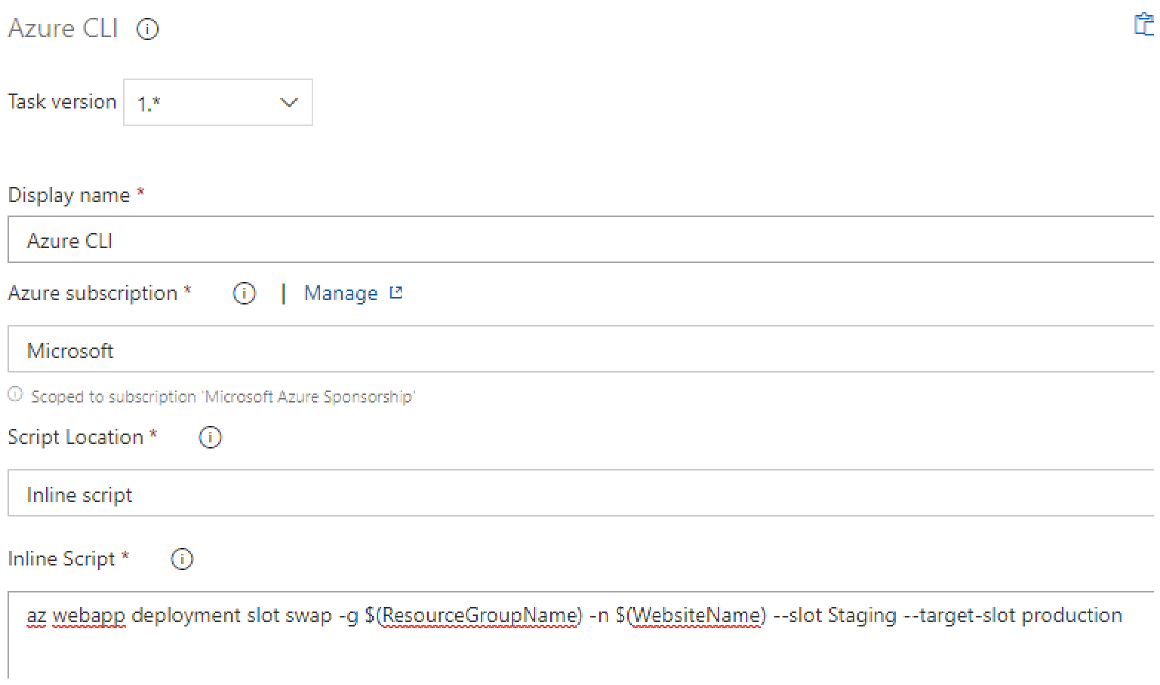
Script Path *

This setting is required.

Arguments

22. Configure the pane as follows, with your subscription, a **Script Location** of **Inline script** , and the **Inline Script** :

```
Az webapp deployment slot swap -g $(ResourceGroupName) -n $(WebsiteName) --slot
Staging --target-slot production
```


Azure CLI

Task version: 1.*

Display name *: Azure CLI

Azure subscription *: Microsoft

Scoped to subscription 'Microsoft Azure Sponsorship'

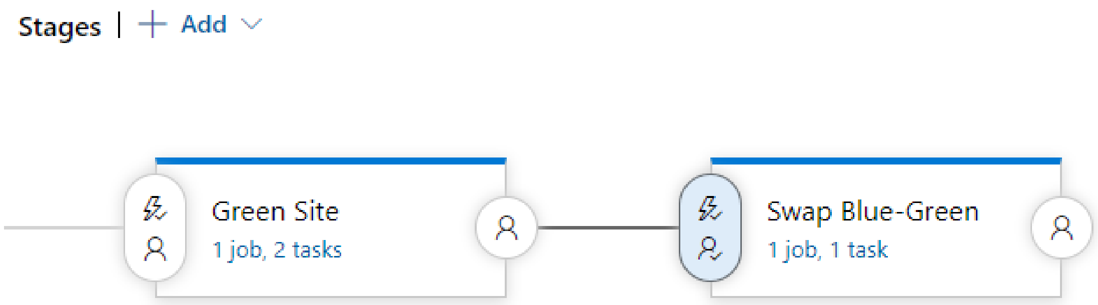
Script Location *: Inline script

Inline Script *:

```
az webapp deployment slot swap -g $(ResourceGroupName) -n $(WebsiteName) --slot Staging --target-slot production
```

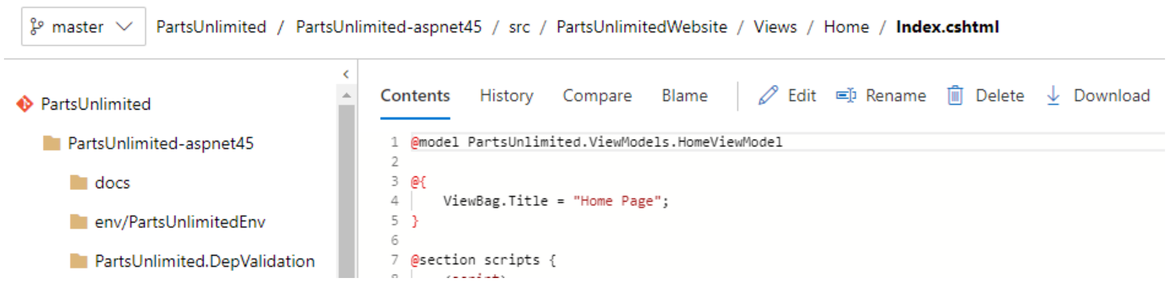
23. From the menu above the task list, click **Pipeline** . Click the **Pre-deployment conditions** icon for the **Swap Blue-Green** stage, then in the **Triggers** pane, enable **Pre-deployment approvals** .

24. Configure yourself as an approver, click **Save** , then **OK** .



Test the blue-green swap

25. In the **PU Hosted** main menu, click **Repos,** then click **Files** to open the project files. Navigate to the following file.


master
PartsUnlimited / PartsUnlimited-aspnet45 / src / PartsUnlimitedWebsite / Views / Home / **Index.cshtml**
PartsUnlimited
PartsUnlimited-aspnet45
docs
env/PartsUnlimitedEnv
PartsUnlimited.DepValidation
Contents
History
Compare
Blame
Edit
Rename
Delete
Download

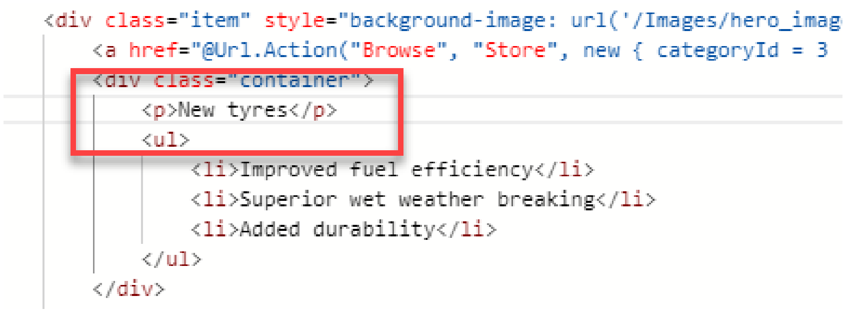
```
@model PartsUnlimited.ViewModels.HomeViewModel

@{
    ViewBag.Title = "Home Page";
}

@section scripts {
```

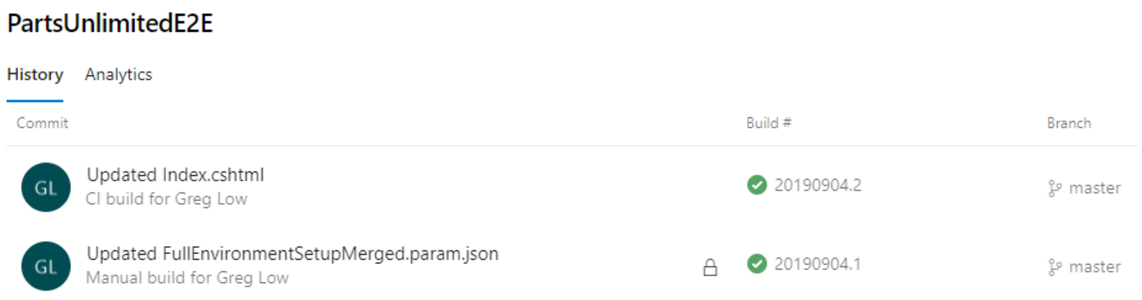
We'll make a cosmetic change to see that the website has been updated. We'll change the word **tires** in the main page rotation to **tyres** to target an international audience.

26. Click **Edit** to allow editing, then find the word **tires** and replace it with the word **tyres** . Click **Commit** and **Commit** to save the changes and trigger a build and release.



```
<div class="item" style="background-image: url('/Images/hero_imag
    <a href="@Url.Action("Browse", "Store", new { categoryId = 3
    <div class="container">
        <p>New tyres</p>
        <ul>
            <li>Improved fuel efficiency</li>
            <li>Superior wet weather breaking</li>
            <li>Added durability</li>
        </ul>
    </div>
```

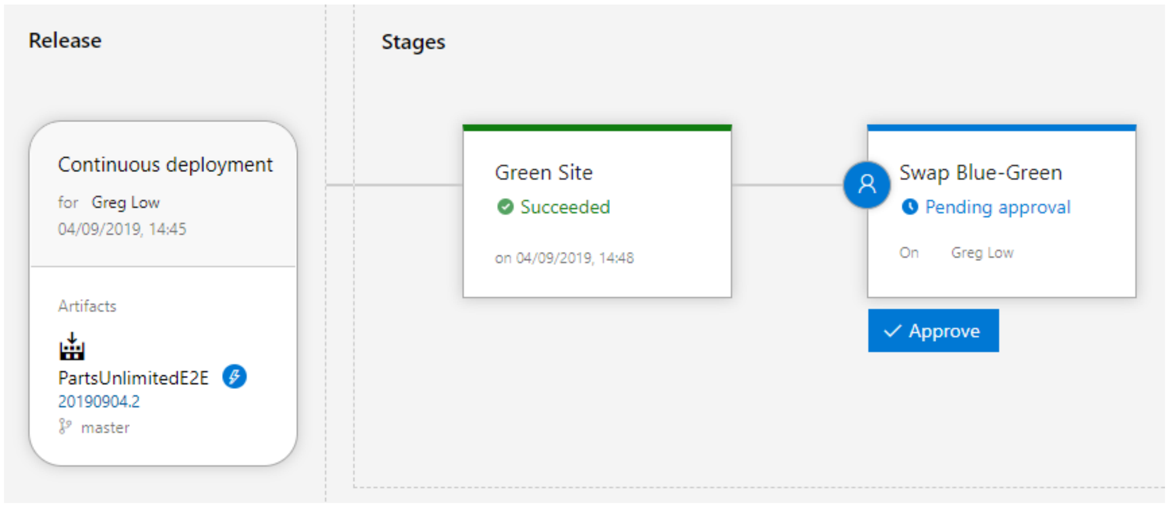
27. From the main menu, click **Pipelines** , then **Builds** . Wait for the continuous integration build to complete successfully.



PartsUnlimitedE2E

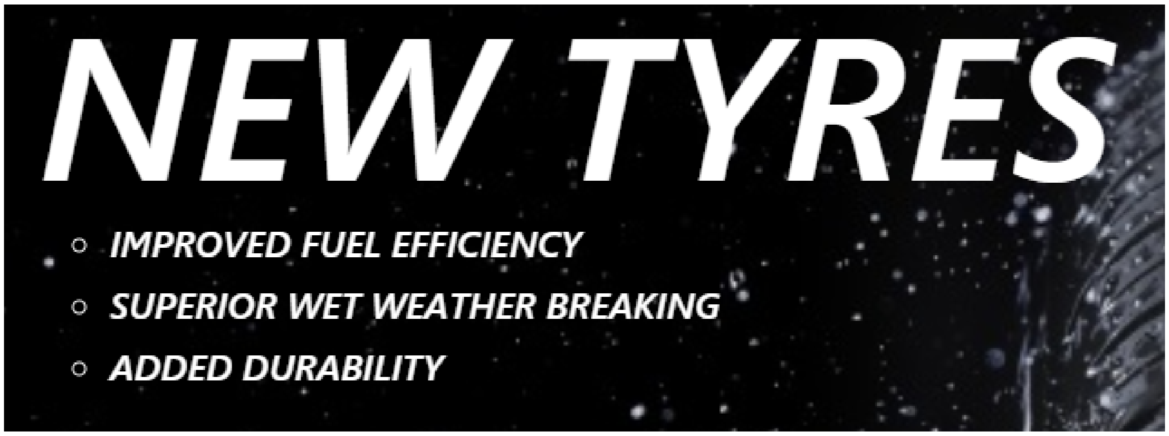
Commit	Build #	Branch
Updated Index.cshtml CI build for Greg Low	✓ 20190904.2	master
Updated FullEnvironmentSetupMerged.param.json Manual build for Greg Low	✓ 20190904.1	master

28. From the main menu, click **Releases** . Click to open the latest release (at the top of the list).

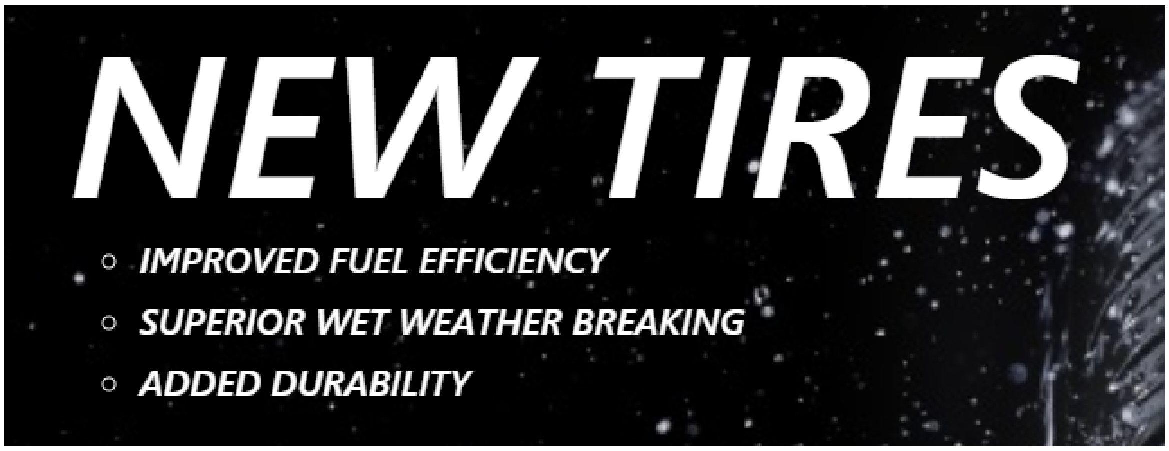


You're now being asked to approve the deployment swap across to Production. We'll check the green deployment first.

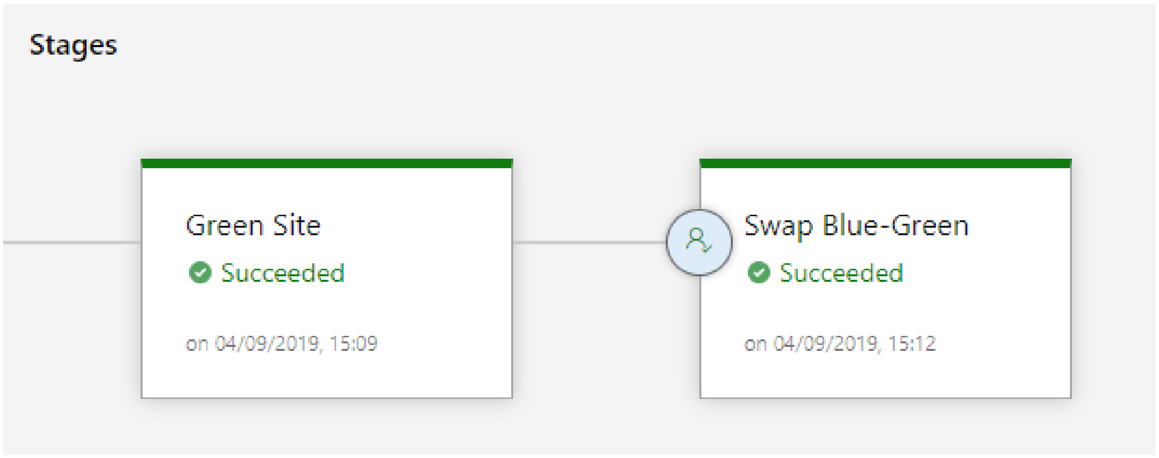
29. Refresh the Green site (that is, Staging slot) browser tab and see if your change has appeared. It now shows the altered word.



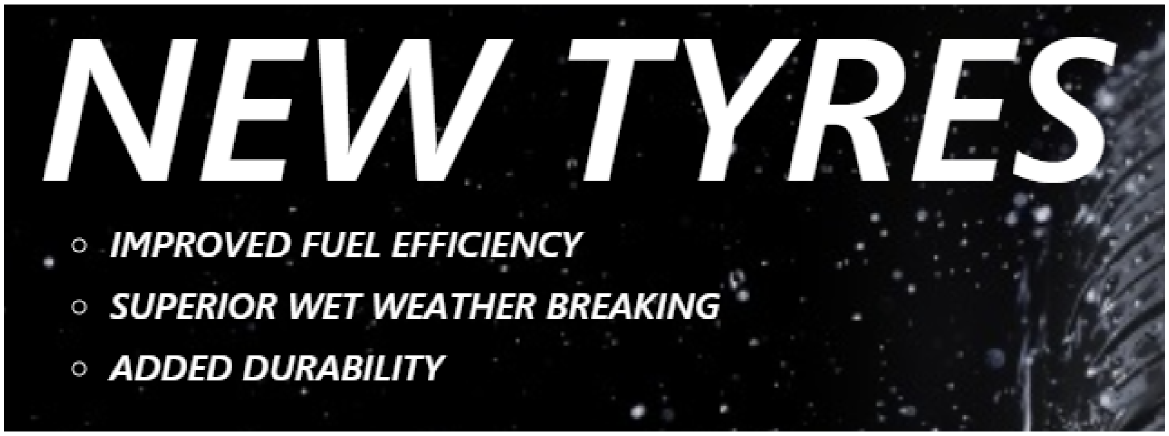
30. Refresh the Production site browser tab and notice that it still isn't updated.



31. As you're happy with the change, in release details, click **Approve** , then **Approve** and wait for the stage to complete.



32. Refresh the Production site browser tab and check that it now has the updated code.



Final notes

If you check the green site, you'll see it has the previous version of the code.

It's the critical difference with Swap, rather than just a typical deployment process from one staged site to another. You have a rapid fallback option by swapping the sites back if needed.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to feature toggles

Completed

- 3 minutes

Feature Flags allow you to change how our system works without making significant changes to the code. Only a small configuration change is required. In many cases, it will also only be for a few users.

Feature Flags offer a solution to the need to push new code into the trunk and deploy it, but it isn't functional yet.

They're commonly implemented as the value of variables used to control conditional logic.

Imagine that your team is all working in the main trunk branch of a banking application.

You've decided it's worth trying to have all the work done in the main branch to avoid messy operations of merge later.

Still, you need to ensure that significant changes to the interest calculations can happen, and people depend on that code every day.

Worse, the changes will take you weeks to complete. You can't leave the main code broken for that period.

A Feature Flag could help you get around it.

You can change the code so that other users who don't have the Feature Flag set will use the original interest calculation code.

The members of your team who are working on the new interest calculations and set to see the Feature Flag will have the new interest calculation code.

It's an example of a business feature flag used to determine business logic.

The other type of Feature Flag is a Release Flag. Now, imagine that after you complete the work on the interest calculation code, you're nervous about publishing a new code out to all users at once.

You have a group of users who are better at dealing with new code and issues if they arise, and these people are often called Canaries.

The name is based on the old use of the Canaries in coal mines.

You change the configuration so that the Canary users also have the Feature Flag set, and they'll start to test the new code as well. If problems occur, you can quickly disable the flag for them again.

Another release flag might be used for AB testing. Perhaps you want to find out if a new feature makes it faster for users to complete a task.

You could have half the users working with the original version of the code and the other half working with the new code version.

You can then directly compare the outcome and decide if the feature is worth keeping. Feature Flags are sometimes called Feature Toggles instead.

By exposing new features by just "flipping a switch" at runtime, we can deploy new software without exposing any new or changed functionality to the end-user.

The question is, what strategy do you want to use in releasing a feature to an end-user.

- Reveal the feature to a segment of users, so you can see how the new feature is received and used.
- Reveal the feature to a randomly selected percentage of users.
- Reveal the feature to all users at the same time.

The business owner plays a vital role in the process, and you need to work closely with them to choose the right strategy.

Just as in all the other deployment patterns mentioned in the introduction, the most crucial part is always looking at how the system behaves.

The idea of separating feature deployment from Feature exposure is compelling and something we want to incorporate in our Continuous Delivery practice.

It helps us with more stable releases and better ways to roll back when we run into issues when we have a new feature that produces problems.

We switch it off again and then create a hotfix. By separating deployments from revealing a feature, you make the opportunity to release a day anytime since the new software won't affect the system that already works.

What are feature toggles?

Feature toggles are also known as feature flippers, feature flags, feature switches, conditional features, and so on.

Besides the power they give you on the business side, they also provide an advantage on the development side.

Feature toggles are a great alternative to branching as well. Branching is what we do in our version control system.

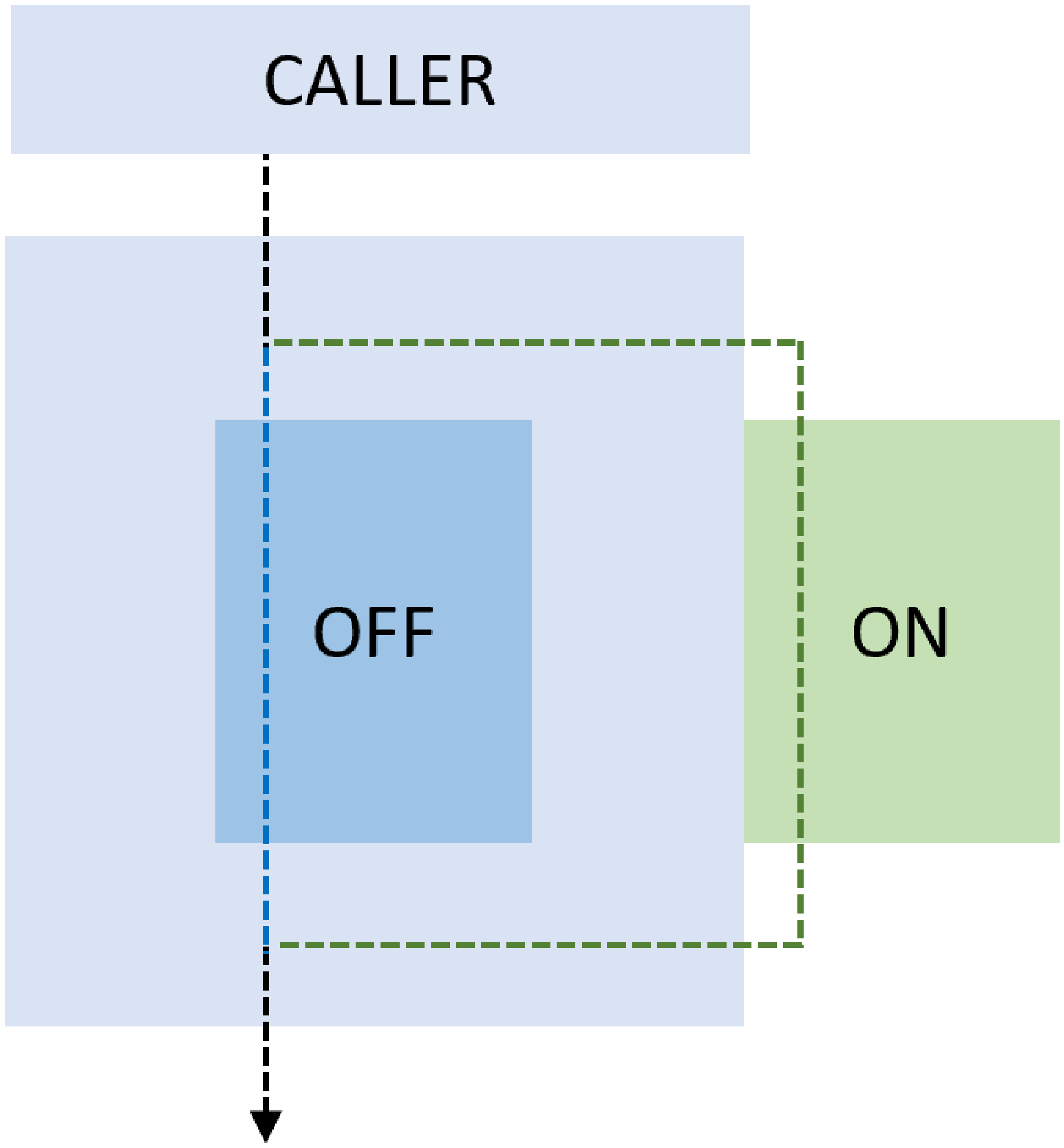
To keep features isolated, we maintain a separate branch.

When we want the software to be in production, we merge it with the release branch and deploy it.

With feature toggles, you build new features behind a toggle. Your feature is "off" when a release occurs and shouldn't be exposed to or impact the production software.

How to implement a feature toggle

In the purest form, a feature toggle is an IF statement.



When the switch is off, it executes the code in the IF, otherwise the ELSE.

You can make it much more intelligent, controlling the feature toggles from a dashboard or building capabilities for roles, users, and so on.

If you want to implement feature toggles, many different frameworks are available commercially as Open Source.

For more information, see also [Explore how to progressively expose your features in production for some or all users](#) .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Describe feature toggle maintenance

Completed

- 2 minutes

A feature toggle is just code. And to be more specific, conditional code. It adds complexity to the code and increases the technical debt.

Be aware of that when you write them, and clean up when you don't need them anymore.

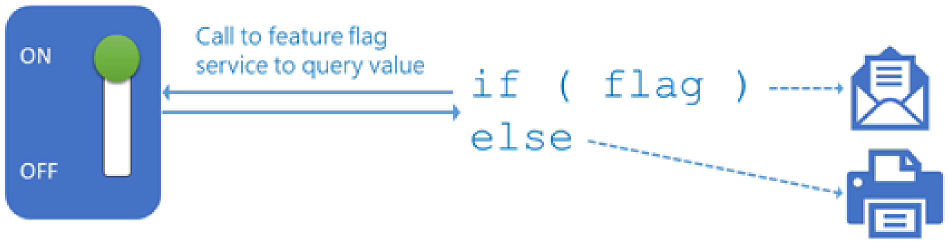
While feature flags can be helpful, they can also introduce many issues of their own.

The idea of a toggle is that it's short-lived and only stays in the software when it's necessary to release it to the customers.

You can classify the different types of toggles based on two dimensions as described by Martin Fowler.

He states that you can look at the dimension of how long a toggle should be in your codebase and, on the other side how dynamic the toggle needs to be.

Planning feature flag lifecycles



The most important thing is to remember that you need to remove the toggles from the software.

If you don't do that, they'll become a form of technical debt if you keep them around for too long.

As soon as you introduce a feature flag, you've added to your overall technical debt.

Like other technical debt, they're easy to add, but the longer they're part of your code, the bigger the technical debt becomes because you've added scaffolding logic needed for the branching within the code.

The cyclomatic complexity of your code keeps increasing as you add more feature flags, as the number of possible paths through the code increases.

Using feature flags can make your code less solid and can also add these issues:

- The code is harder to test effectively as the number of logical combinations increases.
- The code is harder to maintain because it's more complex.
- The code might even be less secure.
- It can be harder to duplicate problems when they're found.

A plan for managing the lifecycle of feature flags is critical. As soon as you add a flag, you need to plan for when it will be removed.

Feature flags shouldn't be repurposed. There have been high-profile failures because teams decided to reuse an old flag that they thought was no longer part of the code for a new purpose.

Tooling for release flag management

The amount of effort required to manage feature flags shouldn't be underestimated. It's essential to consider using tooling that tracks:

- Which flags exist.
- Which flags are enabled in which environments, situations, or target customer categories.
- The plan for when the flags will be used in production.
- The plan for when the flags will be removed.

Using a feature flag management system lets you get the benefits of feature flags while minimizing the risk of increasing your technical debt too high.

Azure App Configuration offers a Feature Manager. See Azure App Configuration Feature Manager .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 5 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices is the easiest way to create a staging environment for an Azure WebApp?

○

Create a deployment slot.

○

Use Application Insights.

○

Create an app close.

2.

Which of the following choices describes a Feature Flag functionality?

○

Feature Flags allow teams to create automatic release notes.

○

Feature Flags can be used to control exposure to new product functionality.

○

Feature Flags help teams control deployment release gates.

3.

Which of the following choices isn't a deployment pattern that allows you to plan to slowly increase the traffic to a newer version of your site?

○

A/B Testing.

○

Blue-Green.

○

Canary Release.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module described the blue-green deployment process and introduced feature toggle techniques to implement in the development process.

You learned how to describe the benefits and usage of:

- Explain deployment strategies.
- Implement blue-green deployment.
- Understand deployment slots.
- Implement and manage feature toggles.

Learn more

- Release Engineering Continuous deployment - Azure Architecture Center | Microsoft Docs .
- Deployment jobs - Azure Pipelines | Microsoft Docs .
- Configure canary deployments for Azure Linux virtual machines - Azure Virtual Machines | Microsoft Docs .
- Progressive experimentation with feature flags - Azure DevOps | Microsoft Docs .
- Set up staging environments - Azure App Service | Microsoft Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Implement canary releases and dark launching

Introduction

Completed

- 1 minute

This module describes deployment strategies around canary releases and dark launching and examines traffic managers.

Learning objectives

After completing this module, students and professionals can:

- Describe deployment strategies.
- Implement canary release.
- Explain traffic manager.
- Understand dark launching.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore canary releases

Completed

- 1 minute

The term canary release comes from the days that miners took a canary with them into the coal mines.

The purpose of the canary was to identify the existence of toxic gasses.

The canary would die much sooner than the miner, giving them enough time to escape the potentially lethal environment.

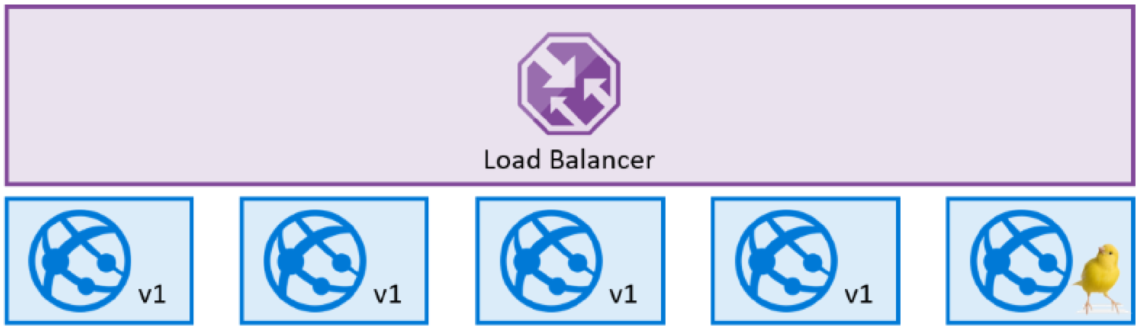
A canary release is a way to identify potential problems without exposing all your end users to the issue at once.

The idea is that you tell a new feature only to a minimal subset of users.

By closely monitoring what happens when you enable the feature, you can get relevant information from this set of users and either continue or rollback (disable the feature).

If the canary release shows potential performance or scalability problems, you can build a fix for that and apply that in the canary environment.

After the canary release has proven to be stable, you can move the canary release to the actual production environment.



Canary releases can be implemented using a combination of feature toggles, traffic routing, and deployment slots.

- You can route a percentage of traffic to a deployment slot with the new feature enabled.
- You can target a specific user segment by using feature toggles.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Examine Traffic Manager

Completed

- 3 minutes

In the previous module, we saw how Deployment slots in Azure Web Apps enable you to swap between two different versions of your application quickly.

Suppose you want more control over the traffic that flows to your other versions. Deployment slots alone aren't enough.

To control traffic in Azure, you can use a component called Azure Traffic Manager.

Azure Traffic Manager

Azure Traffic Manager is a DNS-based traffic load balancer that enables you to distribute traffic optimally to services across global Azure regions while providing high availability and responsiveness.

Traffic Manager uses DNS to direct client requests to the most appropriate service endpoint based on a traffic-routing method and the health of the endpoints.

An endpoint is an Internet-facing service hosted inside or outside of Azure.

Traffic Manager provides a range of traffic-routing methods and endpoint monitoring options to suit different application needs and automatic failover models.

Traffic Manager is resilient to failure, including the breakdown of an entire Azure region.

While the available options can change over time, the Traffic Manager currently provides six options to distribute traffic:

- **Priority** : Select Priority when you want to use a primary service endpoint for all traffic and provide backups if the primary or the backup endpoints are unavailable.
- **Weighted** : Select Weighted when you want to distribute traffic across a set of endpoints, either evenly or according to weights, which you define.
- **Performance** : Select Performance when you have endpoints in different geographic locations, and you want end users to use the "closest" endpoint for the lowest network latency.
- **Geographic** : Select Geographic so that users are directed to specific endpoints (Azure, External, or Nested) based on which geographic location their DNS query originates from. It empowers Traffic Manager customers to enable scenarios where knowing a user's geographic region and routing them based on that is necessary. Examples include following data sovereignty mandates, localization of content & user experience, and measuring traffic from different regions.
- **Multivalue** : Select MultiValue for Traffic Manager profiles that can only have IPv4/IPv6 addresses as endpoints. When a query is received for this profile, all healthy endpoints are returned.
- **Subnet** : Select the Subnet traffic-routing method to map sets of end-user IP address ranges to a specific endpoint within a Traffic Manager profile. The endpoint returned will be mapped for that request's source IP address when a request is received.

When we look at the options the Traffic Manager offers, the most used option for Continuous Delivery is routing traffic based on weights.

Note

Traffic is only routed to endpoints that are currently available.

For more information, see also:

- What is Traffic Manager?
- How Traffic Manager works
- Traffic Manager Routing Methods

Controlling your canary release

Using a combination of feature toggles, deployment slots, and Traffic Manager, you can achieve complete control over the traffic flow and enable your canary release.

You deploy the new feature to the new deployment slot or a new instance of an application, and you enable the feature after verifying the deployment was successful.

Next, you set the traffic to be distributed to a small percentage of the users.

You carefully watch the application's behavior, for example, by using application insights to monitor the performance and stability of the application.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Understand dark launching

Completed

- 1 minute

Dark launching is in many ways like canary releases.

However, the difference here's that you're looking to assess users' responses to new features in your frontend rather than testing the performance of the backend.

The idea is that rather than launch a new feature for all users, you instead release it to a small set of users.

Usually, these users aren't aware they're being used as test users for the new feature, and often you don't even highlight the new feature to them, as such the term "Dark" launching.

Another example of dark launching is launching a new feature and using it on the backend to get metrics.

Let me illustrate with a real-world "launch" example.

As Elon Musk describes in his biography, they apply all kinds of Agile development principles in SpaceX.

SpaceX builds and launches rockets to launch satellites. SpaceX also uses dark launching.

When they have a new version of a sensor, they install it alongside the old one.

All data is measured and gathered both by the old and the new sensor.

Afterward, they compare the outcomes of both sensors.

Only when the new one has the same or improved results the old sensor is replaced.

The same concept can be applied to software. You run all data and calculations through your new feature, but it isn't "exposed" yet.

How to implement dark launching

In essence, dark launching doesn't differ from a canary release or the implementation and switching of a feature toggle.

The feature is released and only exposed at a particular time.

As such, the techniques, as described in the previous chapters, do also apply for dark launching.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices is an Azure-based tool can you use to divert a percentage of your web traffic to a newer version of an Azure website?

○

Load Balancer.

○

Application Gateway.

○

Traffic Manager.

2.

Which of the following choices is a characteristic that makes users suitable for working with Canary deployments?

○

Uses the app irregularly.

○

High tolerance for issues.

○

It depends highly on the app working all the time.

3.

Which of the following choices describes the difference Dark Launching has from Canary releases?

○

You're looking to assess users' responses to new features in your frontend rather than testing the performance of the backend.

○

You're looking to assess users' responses to new features in your backend and frontend.

○

You're looking to assess users' responses to new features in your backend rather than testing the performance of the frontend.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module described deployment strategies around canary releases and dark launching and examined traffic managers.

You learned how to describe the benefits and usage of:

- Describe deployment strategies.
- Implement canary release.
- Explain traffic manager.
- Understand dark launching.

Learn more

- Release Engineering: Deployment .
- Canary deployment strategy for Kubernetes deployments .
- What is Traffic Manager? .
- Progressive experimentation with feature flags .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Implement A/B testing and progressive exposure deployment

Introduction

Completed

- 1 minute

This module introduces A/B testing and progressive exposure deployment concepts and explores CI/CD with deployment rings--ring-based deployment.

Learning objectives

After completing this module, students and professionals can:

- Implement progressive exposure deployment.
- Implement A/B testing.
- Implement CI/CD with deployment rings.
- Identify the best deployment strategy.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

What is A/B testing?

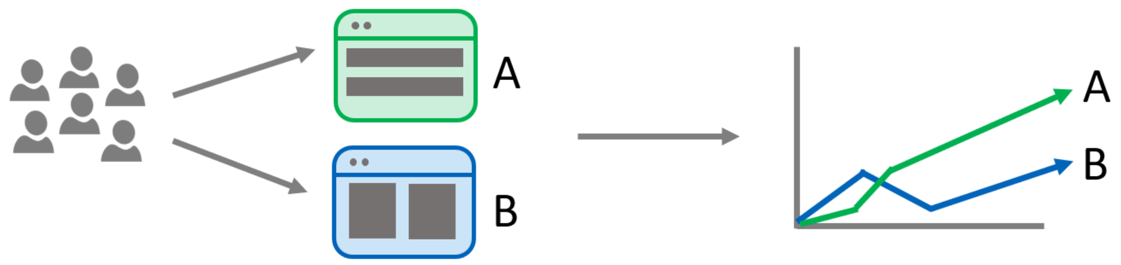
Completed

- 1 minute

A/B testing (also known as split testing or bucket testing) compares two versions of a web page or app against each other to determine which one does better.

A/B testing is mainly an experiment where two or more page variants are shown to users at random.

Also, statistical analysis is used to determine which variation works better for a given conversion goal.



A/B testing isn't part of continuous delivery or a pre-requisite for continuous delivery. It's more the other way around.

Continuous delivery allows you to deliver MVPs to a production environment and your end-users quickly.

Common aims are to experiment with new features, often to see if they improve conversion rates.

Experiments are continuous, and the impact of change is measured.

A/B testing is out of scope for this course.

But because it's a powerful concept that is enabled by implementing continuous delivery, it's mentioned here to dive into further.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore CI-CD with deployment rings

Completed

- 2 minutes

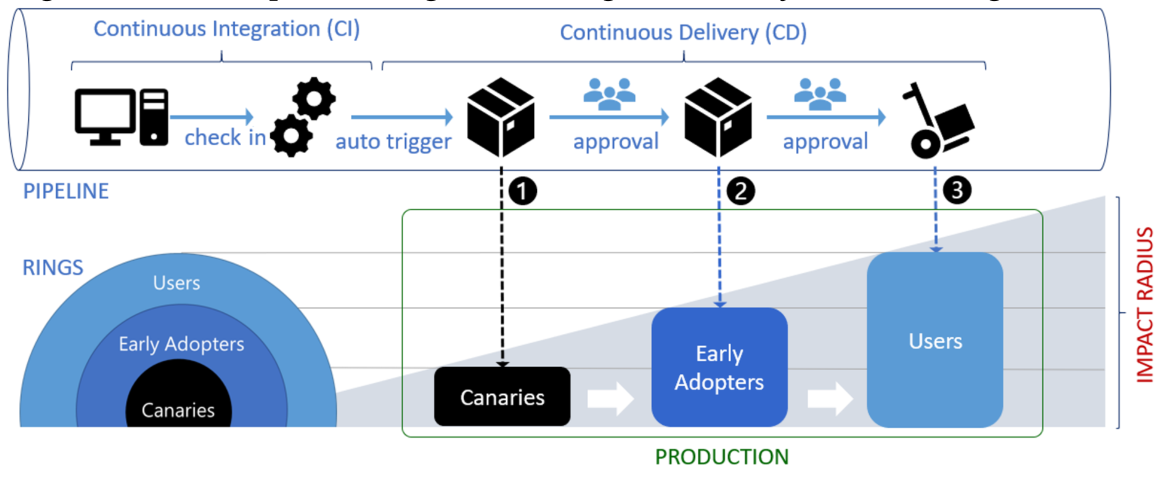
Progressive exposure deployment, also called ring-based deployment, was first discussed in Jez Humble's Continuous Delivery book.

They support the production-first DevOps mindset and limit the impact on end users while gradually deploying and validating changes in production.

Impact (also called blast radius) is evaluated through observation, testing, analysis of telemetry, and user feedback.

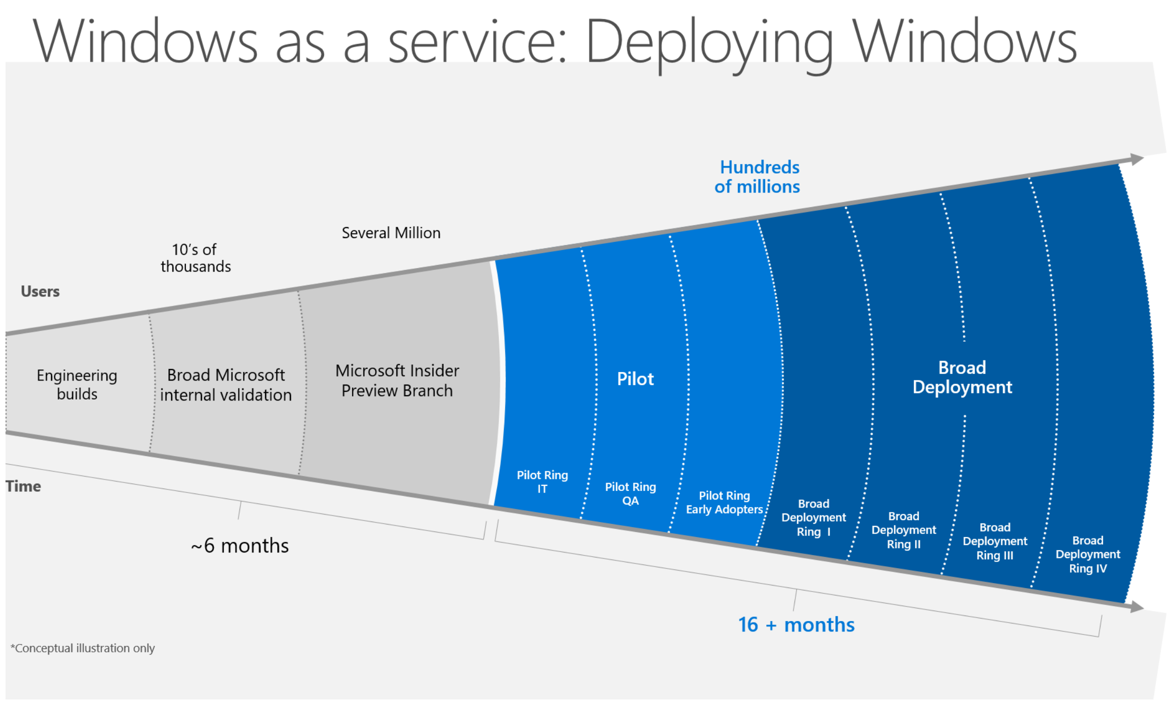
In DevOps, rings are typically modeled as stages.

Rings are, in essence, an extension of the canary stage. The canary release releases to a stage to measure impact. Adding another ring is essentially the same thing.



With a ring-based deployment, you first deploy your changes to risk-tolerant customers and progressively roll out to a more extensive set of customers.

The Microsoft Windows team, for example, uses these rings.



When you have identified multiple groups of users and see value in investing in a ring-based deployment, you need to define your setup.

Some organizations that use canary releasing have multiple deployment slots set up as rings.

The first release of the feature to ring 0 targets a well-known set of users, mostly their internal organization.

After things have been proven stable in ring 0, they propagate the release to the next ring. It's with a limited set of users outside their organization.

And finally, the feature is released to everyone. It is often done by flipping the switch on the feature toggles in the software.

As in the other deployment patterns, monitoring and health checks are essential.

By using post-deployment release gates that check a ring for health, you can define an automatic propagation to the next ring after everything is stable.

When a ring isn't healthy, you can halt the deployment to the following rings to reduce the impact.

For more information, see also Explore how to progressively expose your Azure DevOps extension releases in production to validate before impacting all users .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Exercise - Ring-based deployment

Completed

- 5 minutes

In this exercise, you'll investigate ring-based deployment.

Steps

Let's look at how a release pipeline can stage features using ring-based deployments.

When I have a new feature, I might want to release it to a few users first, just in case something goes wrong.

I could do it in authenticated systems by having those users as members of a security group and letting members of that group use the new features.

However, on a public website, I might not have logged-in users. Instead, I might want to direct a small percentage of the traffic to use the new features.

Let's see how that's configured.

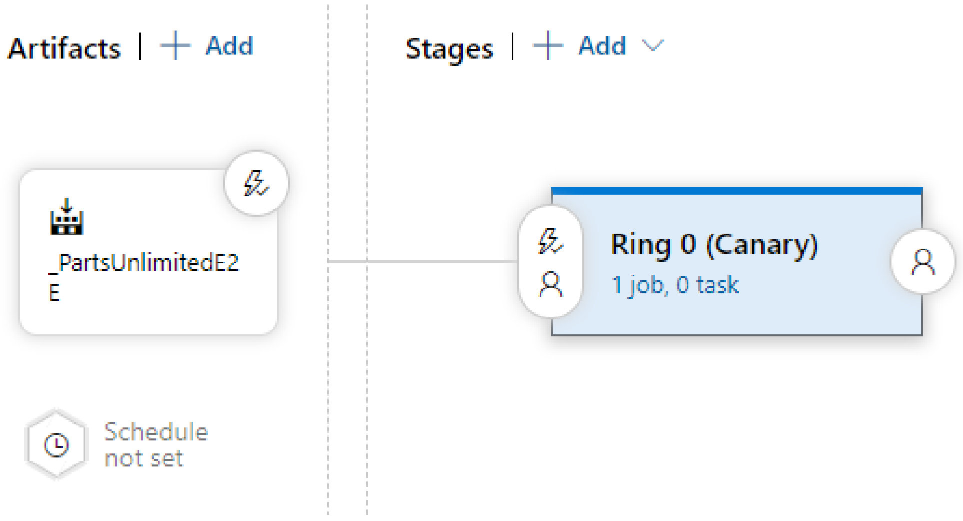
We'll create a new release pipeline that isn't triggered by code changes but manually when we slowly release a new feature.

We start by assuming that a new feature has already been deployed to the Green site (the staging slot).

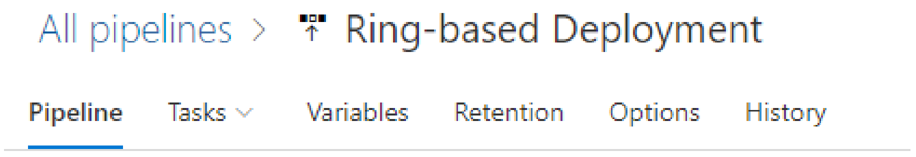
1. In the main menu for the **PU Hosted** project, click **Pipelines** , then click **Release** , click **+New** , then click **New release pipeline** .

2. When prompted to select a template, click **Empty job** from the top of the pane.

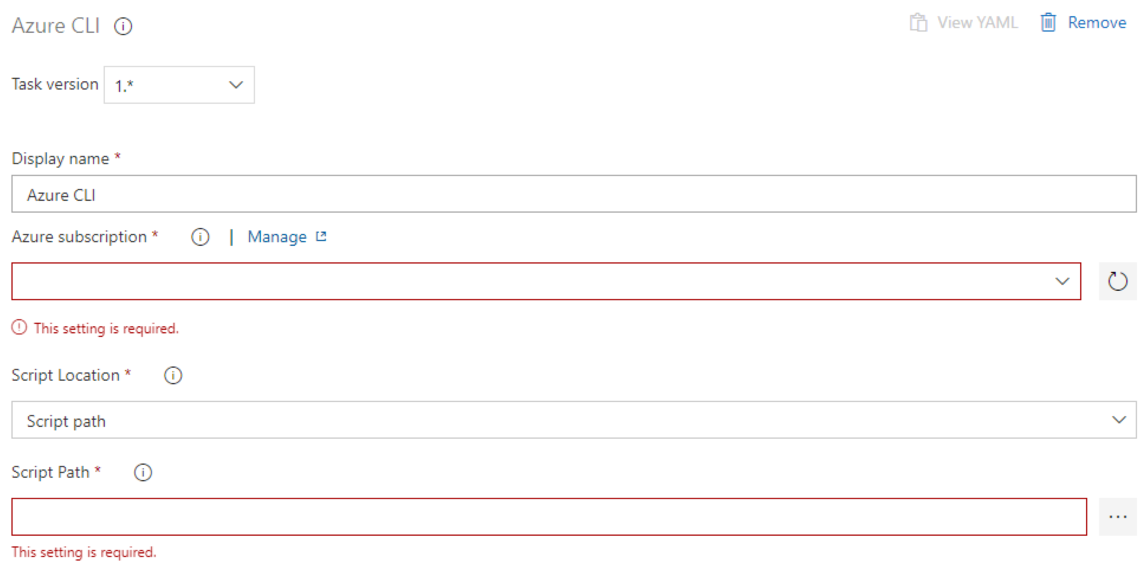
3. Click on the **Stage 1** stage and rename it to **Ring 0 (Canary)** .



4. Hover over the **New release pipeline** name at the top of the page, and when a pencil appears, click it, and change the pipeline name to **Ring-based Deployment** .



5. Select the **Ring 0 (Canary)** stage from the Tasks drop-down list. Click the **+** to add a new task, and from the list of tasks, hover over **Azure CLI** when the **Add** button appears, click it, then click to select the **Azure CLI** task in the task list for the stage.



Task version: 1.*

Display name *: Azure CLI

Azure subscription *:

This setting is required.

Script Location *: Script path

Script Path *:

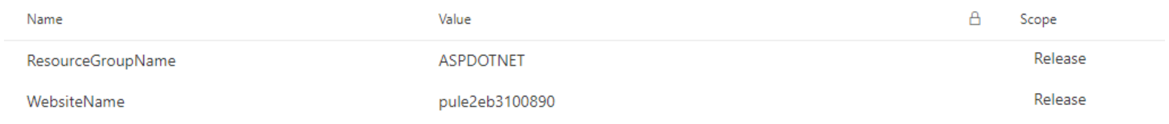
This setting is required.

6. In the **Azure CLI** settings pane, select your **Azure subscription** , set **Script Location** to **Inline script** , set the **Inline Script** to the following, then click **Save** and **OK** .

```
az webapp traffic-routing set --resource-group $(ResourceGroupName) --name
$(WebsiteName) --distribution staging=10
```

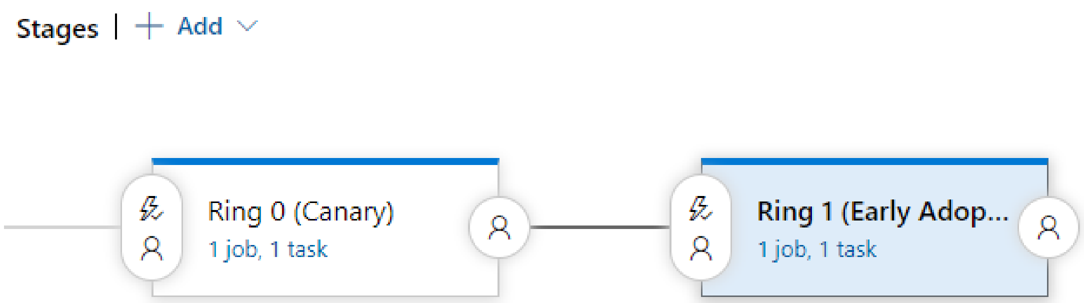
   This distribution will cause 10% of the web traffic to be sent to the new feature Site (currently the staging slot).

7. From the menu above the task list, click **Variables** . Create two new variables as shown. (Make sure to use your correct website name).

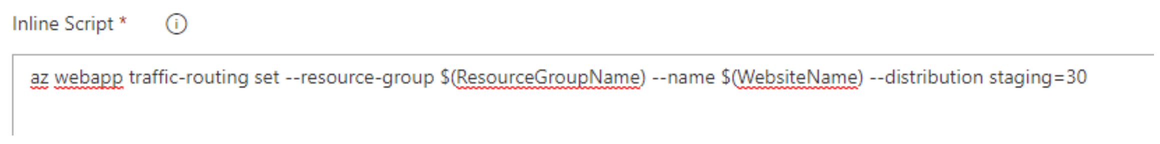


Name	Value	Scope
ResourceGroupName	ASPDOTNET	Release
WebsiteName	pule2eb3100890	Release

8. From the menu above the variables, click **Pipeline** to return to editing the pipeline. Hover over the **Ring 0 (Canary)** stage and click the **Clone** icon when it appears. Select the new stage and rename it to **Ring 1 (Early Adopters)** .



9. Select the **Ring 1 (Early Adopters)** stage from the Tasks drop-down list and select the **Azure CLI** task. Modify the script by changing the value from **10** to **30** to cause 30% of the traffic to go to the new feature site.

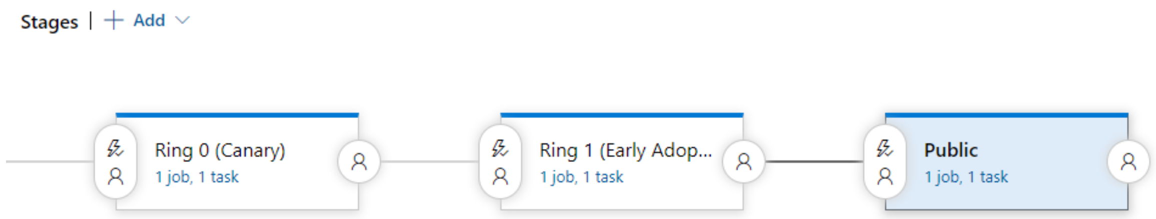


az webapp traffic-routing set --resource-group $(ResourceGroupName) --name $(WebsiteName) --distribution staging=30

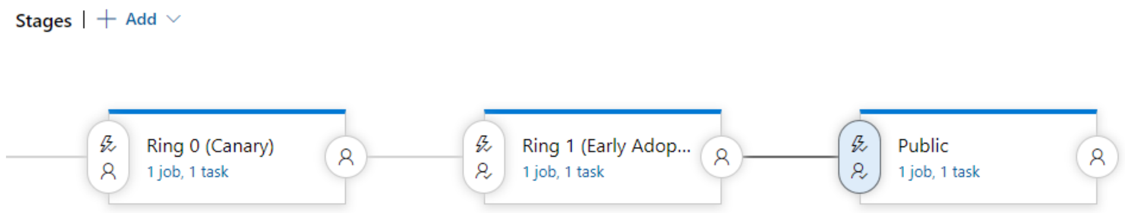
```
az webapp traffic-routing set --resource-group $(ResourceGroupName) --name
$(WebsiteName) --distribution staging=30
```

It allows us to move the new feature into broader distribution if it works ok in smaller users.

10. From the menu above the tasks, click **Pipeline** to return to editing the release pipeline. Hover over the **Ring 1 (Early Adopters)** stage and when the **Clone** icon appears, click it. Click to select the new stage and rename it to **Public** . Click **Save** and **OK** .



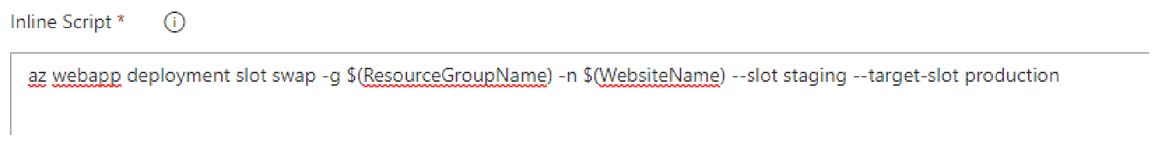
11. Click the **Pre-deployment conditions** icon for the **Ring 1 (Early Adopters)** stage and add yourself as a pre-deployment approver. Do the same for the **Public** stage—Click **Save** and **OK** .



The first step in releasing the new code to the public is to swap the new feature site (that is, the staging site) with the production so that production is now running the new code.

12. From the **Tasks** drop-down list, select the **Public** stage. Select the **Azure CLI** task, change the **Display name** to **Swap sites** and change the **Inline Script** to the following command:

```
az webapp deployment slot swap -g $(ResourceGroupName) -n $(WebsiteName) --slot
staging --target-slot production
```


Inline Script *

```
az webapp deployment slot swap -g $(ResourceGroupName) -n $(WebsiteName) --slot staging --target-slot production
```

az webapp deployment slot swap -g $(ResourceGroupName) -n $(WebsiteName) --slot staging --target-slot production

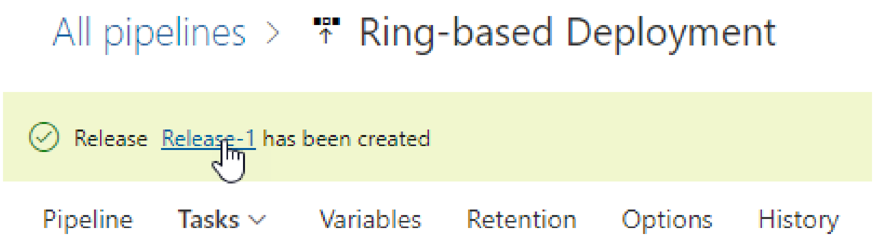
Next, we need to remove any traffic from the staging site.

13. Right-click the **Swap sites** task and click **Clone tasks(s)** . Select the **Swap sites copy** task, change its **Display name** to **Stop staging traffic** , and set the **Inline Script** to the following:

```
az webapp traffic-routing set --resource-group $(ResourceGroupName) --name $(WebsiteName) -
-distribution staging=0
```

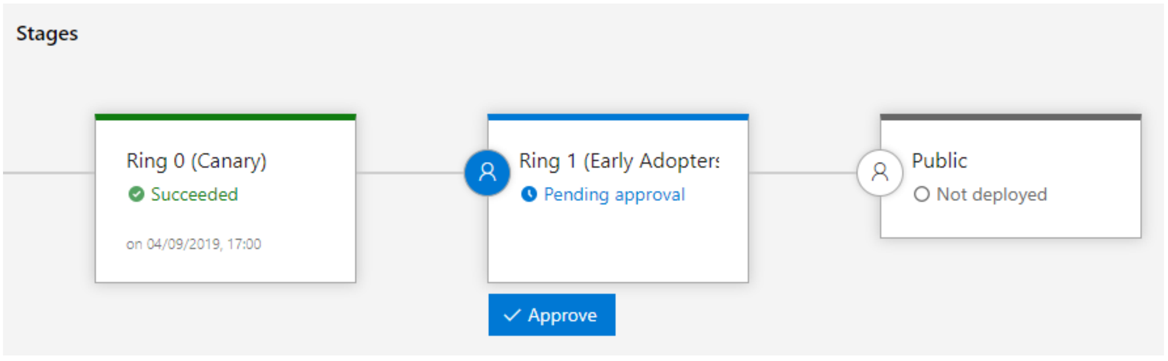
1. Click **Save,** then **OK** to save your work.

2. Click **Create release** and **Create** to start a release. Click the release link to see the progress.


All pipelines > Ring-based Deployment

Release Release-1 has been created

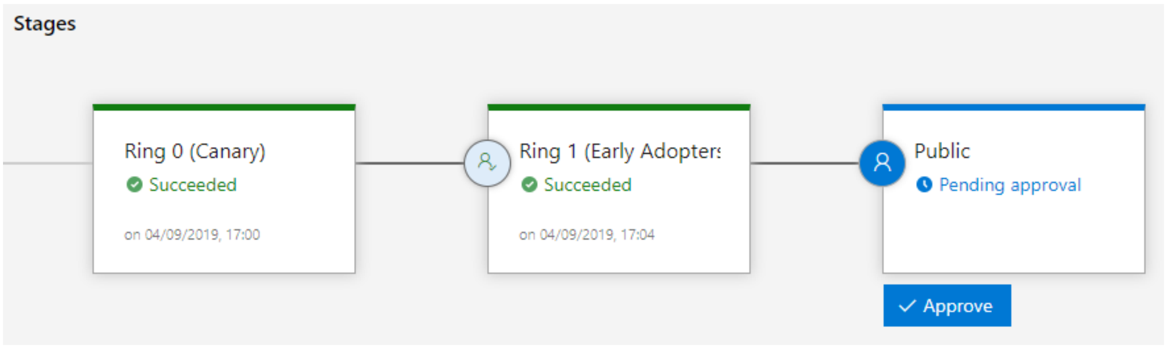
Wait until Ring 0 (Canary) has succeeded.



Currently, 10% of the traffic will go to the new feature site.

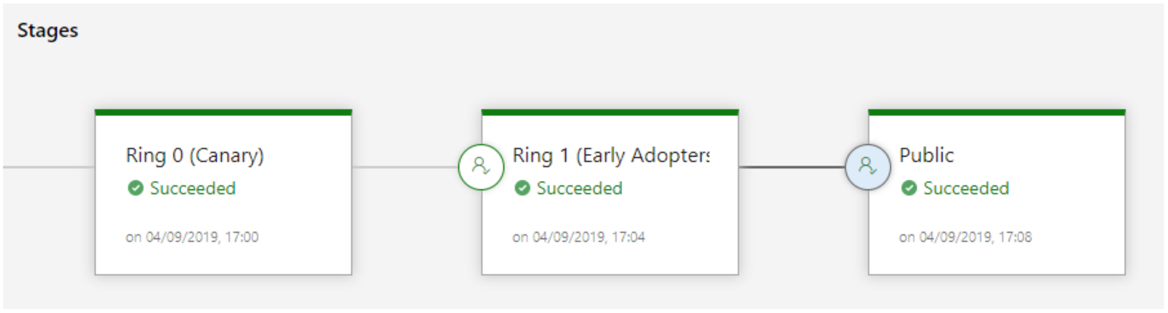
3. Click **Approve** on the **Ring 1 (Early Adopters)** stage, and then **Approve** .

When this stage completes, 30% of the traffic will go to the early adopters in ring 1.



4. Click **Approve** on the **Public** stage, and then **Approve** .

   When this stage completes, all the traffic will go to the swapped production site, running the new code.



The new feature has been fully deployed.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following choices is a deployment pattern extension of Canary Release?

○

Blue-Green.

○

A/B Testing.

○

Progressive Exposure.

2.

Which of the following choices is another way A/B testing is called?

○

Smoke testing.

○

Split testing or Bucket testing.

○

Integration testing.

3.

Which of the following choices is a correct statement about A/B testing?

○

A/B testing isn't part of Continuous Delivery or a pre-requisite for Continuous Delivery.

○

A/B testing can be implemented using Azure Artifacts and multiple environments.

○

A/B testing is a pre-requisite and part of Continuous Delivery.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module introduced A/B testing and progressive exposure deployment concepts and explored CI/CD with deployment rings--ring-based deployment.

You learned how to describe the benefits and usage of:

- Implement progressive exposure deployment.
- Implement A/B testing.
- Implement CI/CD with deployment rings.
- Identify the best deployment strategy.

Learn more

- Progressively expose your releases using deployment rings - Azure DevOps | Microsoft Docs .
- Testing your app and Azure environment - Azure Architecture Center | Microsoft Docs .
- What is Continuous Delivery? - Azure DevOps | Microsoft Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Integrate with identity management systems

Introduction

Completed

- 1 minute

This module describes the integration with GitHub and single sign-on (SSO) for authentication, service principal, and managed service identities.

Learning objectives

After completing this module, students and professionals can:

- Integrate Azure DevOps with identity management systems.
- Integrate GitHub with single sign-on (SSO).
- Understand and create a service principal.
- Create managed service identities.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Integrate GitHub with single sign-on (SSO)

Completed

- 2 minutes

To use SSO, you need to connect your identity provider to GitHub at the organization level.

GitHub offers both **SAML** and **SCIM** support.

Provider	Available Support
Active Directory Federation Services (ADFS)	SAML
Microsoft Entra ID	SAML and SCIM
Okta	SAML and SCIM

Provider	Available Support
OneLogin	SAML and SCIM
PingOne	SAML
Shibboleth	SAML

For more information, see:

- Enforcing SAML single sign-on for your organization .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore service principals

Completed

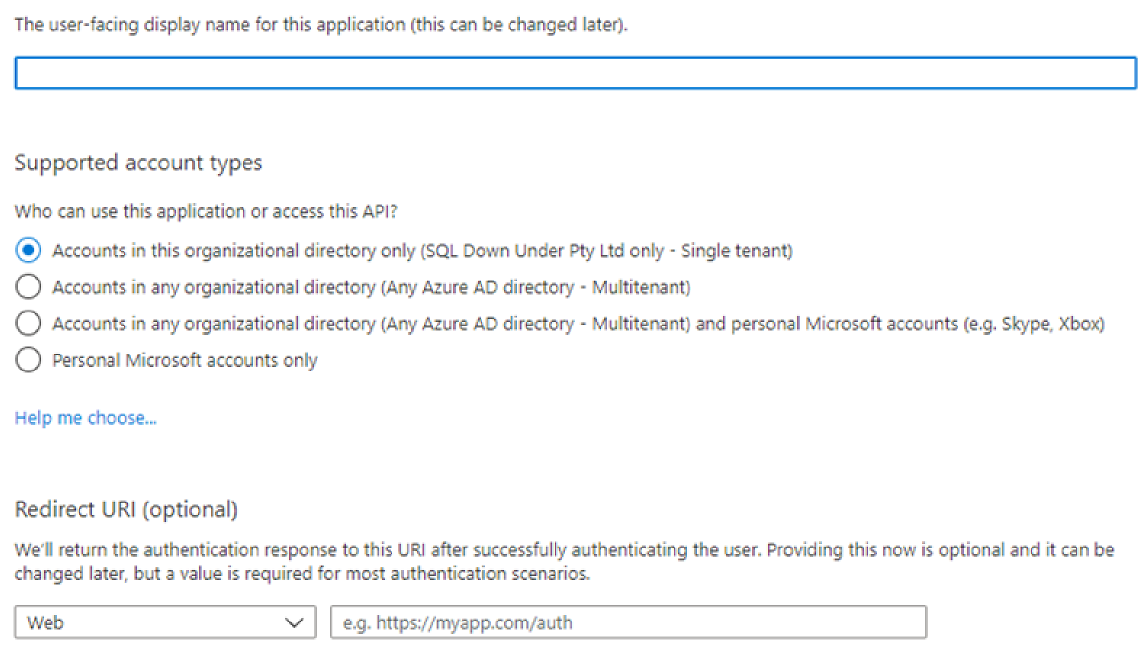
- 2 minutes

Microsoft Entra ID offers different kinds of mechanisms for authentication. In DevOps Projects, though, one of the most important is the use of Service Principals.

Microsoft Entra applications

Applications are registered with a Microsoft Entra tenant within Microsoft Entra ID. Registering an application creates an identity configuration. You also determine who can use it:

- Accounts in the same organizational directory.
- Accounts in any organizational directory.
- Accounts in any organizational directory and Microsoft Accounts (personal).
- Microsoft Accounts (Personal accounts only).



The user-facing display name for this application (this can be changed later).

Supported account types

Who can use this application or access this API?

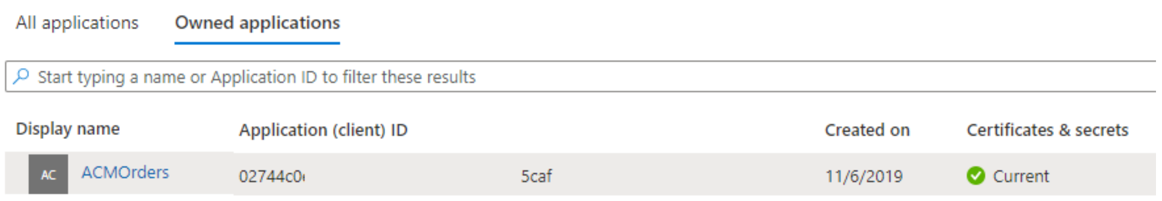
- [✓ Accounts in this organizational directory only (SQL Down Under Pty Ltd only - Single tenant)]
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web | e.g. https://myapp.com/auth



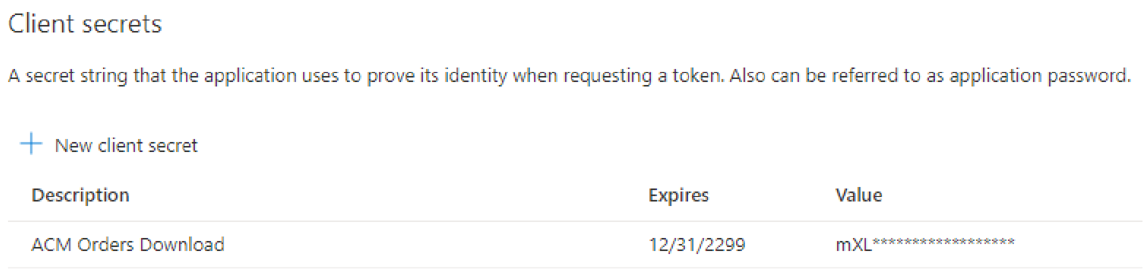
All applications | **Owned applications**

Start typing a name or Application ID to filter these results

Display name	Application (client) ID	Created on	Certificates & secrets
AC ACMOrders	02744c0 [redacted] 5caf	11/6/2019	✓ Current

Client secret

Once the application is created, you then should create at least one client secret for the application.



Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value
ACM Orders Download	12/31/2299	mXL******************

Grant permissions

The application identity can then be granted permissions within services and resources that trust Microsoft Entra ID.

Service principal

To access resources, an entity must be represented by a security principal. To connect, the entity must know:

- TenantID.
- ApplicationID.
- Client Secret.

For more information on Service Principals, see [App Objects and Service Principals].

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

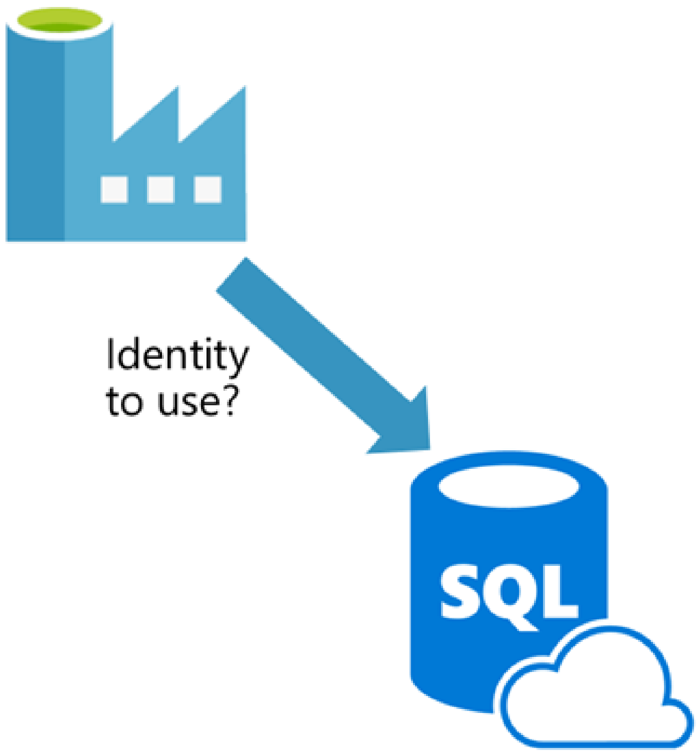
[Explore Managed Identity]

Completed

- 2 minutes

Another authentication mechanism offered by Microsoft Entra ID is Managed identities.

Imagine that you need to connect from an Azure Data Factory (ADF) to an Azure SQL Database. What identity should ADF present to the database?



The traditional answer would have been to use SQL Authentication with a username and password. It leaves yet another credential that needs to be managed on an ongoing basis.

Identity of the service

Many Azure services expose their own identity. It isn't an identity that you need to manage. For example, you don't need to worry about password policies and so on.

You can assign permissions to that identity, as with any other Microsoft Entra identity.

In the ADF example, you can add the ADF MSI as an Azure SQL Database user and add it to roles within the database.

Managed identity types

There are two types of managed identities:

- System-assigned - It's the types of identities described above. Many, but not all, services expose these identities.
- User-assigned - you can create a managed identity as an Azure resource. It can then be assigned to one or more instances of a service.

For more information, see: What are managed identities for Azure resources?

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 4 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following level choices do you need to connect your identity provider to GitHub to use SSO?

○

At the organization level.

○

At the project level.

○

At the repository level.

2.

Applications are registered with a Microsoft Entra tenant within Microsoft Entra ID. Registering an application creates an identity configuration. Which of the following choices isn't correct about who can use it?

○

Accounts in the same organizational directory.

○

Azure DevOps groups.

○

Accounts in any organizational directory and Microsoft Accounts (personal).

3.

Which of the following choices isn't a type of managed identities?

○

Organization-assigned.

○

System-assigned.

○

User-assigned.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module described the integration with GitHub and single sign-on (SSO) for authentication, service principal, and managed service identities.

You learned how to describe the benefits and usage of:

- Integrate Azure DevOps with identity management systems.
- Integrate GitHub with single sign-on (SSO).
- Understand and create a service principal.
- Create managed service identities.

Learn more

- About security, authentication, authorization, and security policies - Azure DevOps | Microsoft Docs .
- Azure Identity and Access Management Solutions | Microsoft Azure .
- About authentication with SAML single sign-on - GitHub Docs .
- Connect to Microsoft Azure - Azure Pipelines | Microsoft Docs .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Manage application configuration data

Introduction

Completed

- 1 minute

This module explores ways to rethink application configuration data and the separation of concerns method. It helps you understand configuration patterns and how to integrate Azure Key Vault with Azure Pipelines.

Learning objectives

After completing this module, students and professionals can:

- Rethink application configuration data.
- Understand separation of concerns.
- Integrate Azure Key Vault with Azure Pipelines.
- Manage secrets, tokens, and certificates.

Prerequisites

- Understanding of what DevOps is and its concepts.
- Familiarity with version control principles is helpful but isn't necessary.
- Beneficial to have experience in an organization that delivers software.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Rethink application configuration data

Completed

- 4 minutes

Configuration information in files

Most application runtime environments include configuration information that is held in files deployed with the application. In some cases, it's possible to edit these files to change the application behavior after deploying.

However, changes to the configuration require the application to be redeployed, often resulting in unacceptable downtime and other administrative overhead.

Local configuration files also limit the configuration to a single application, but sometimes it would be helpful to share configuration settings across multiple applications.

Examples include database connection strings, UI theme information, or the URLs of queues and storage used by a related set of applications.

It's challenging to manage changes to local configurations across multiple running instances of the application, especially in a cloud-hosted scenario.

It can result in instances using different configuration settings while the update is being deployed. Also, updates to applications and components might require changes to configuration schemas.

Many configuration systems don't support different versions of configuration information.

Example

It's 2:00 AM. Adam is done making all changes to his super fantastic code piece.

The tests are all running fine. They hit commit -> push -> all commits pushed successfully to git. Happily, Adam drives back home. 10 mins later, they get a call from the SecurityOps engineer, "Adam, did you push the Secret Key to our public repo?"

YIKES! That blah.config file, Adam thinks. How could I've forgotten to include that in .gitignore? The nightmare has already begun.

We can surely blame Adam for sinning, checking in sensitive secrets, and not following the recommended practices of managing configuration files.

Still, the bigger question is that if the underlying toolchain had abstracted out the configuration management from the developer, this fiasco would have never happened!

History

The virus was injected a long time ago. Since the early days of .NET, the app.config and web.config files have the notion that developers can make their code flexible by moving typical configuration into these files.

When used effectively, these files are proven to be worthy of dynamic configuration changes. However, much time, we see the misuse of what goes into these files.

A common culprit is how samples and documentation have been written. Most samples on the web would usually use these config files to store key elements such as ConnectionStrings and even passwords.

The values might be obfuscated but what we are telling developers is that "hey, It's a great place to push your secrets!".

So, in a world where we're preaching using configuration files, we can't blame the developer for not managing its governance.

We aren't challenging the use of Configuration here. It's an absolute need for an exemplary implementation. Instead, we should debate using multiple JSON, XML, and YAML files to maintain configuration settings.

Configs are great for ensuring the flexibility of the application, config files. However, they aren't straightforward, especially across environments.

A ray of hope: The DevOps movement

In recent years, we've seen a shift around following some excellent practices around effective DevOps and some great tools (Chef, Puppet) for managing Configuration for different languages.

While these have helped inject values during CI/CD pipeline and greatly simplified configuration management, the blah.config concept hasn't moved away.

Frameworks like ASP.NET Core support the notion of appSettings.json across environments.

The framework has made it practical to use across environments through interfaces like IHostingEnvironment and IConfiguration, but we can do better.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Explore separation of concerns

Completed

- 2 minutes

One of the key reasons we would want to move the configuration away from source control is to outline responsibilities.

Let's define some roles to elaborate on them. None of those are new concepts but rather a high-level summary:

- **Configuration custodian** : Responsible for generating and maintaining the life cycle of configuration values. These include CRUD on keys, ensuring the security of secrets, regeneration of keys and tokens, defining configuration settings such as Log levels for each environment. This role can be owned by operation engineers and security engineering while injecting configuration files through proper DevOps processes and CI/CD implementation. They do not define the actual configuration but are custodians of their management.

- **Configuration consumer** : Responsible for defining the schema (loose term) for the configuration that needs to be in place and then consuming the configuration values in the application or library code. It's the Dev. And Test teams shouldn't be concerned about the value of keys but rather what the key's capability is. For example, a developer may need a different ConnectionString in the application but not know the actual value across different environments.

- **Configuration store** : The underlying store used to store the configuration, while it can be a simple file, but in a distributed application, it needs to be a reliable store that can work across environments. The store is responsible for persisting values that modify the application's behavior per environment but aren't sensitive and don't require any encryption or HSM modules.

- **Secret store** : While you can store configuration and secrets together, it violates our separation of concern principle, so the recommendation is to use a different store for persisting secrets. It allows a secure channel for sensitive configuration data such as ConnectionStrings, enables the operations team to have Credentials, Certificate, Token in one repository, and minimizes the security risk if the Configuration Store gets compromised.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Understand external configuration store patterns

Completed

- 3 minutes

These patterns store the configuration information in an external location and provide an interface that can be used to quickly and efficiently read and update configuration settings.

The type of external store depends on the hosting and runtime environment of the application.

A cloud-hosted scenario is typically a cloud-based storage service but could be a hosted database or other systems.

The backing store you choose for configuration information should have an interface that provides consistent and easy-to-use access.

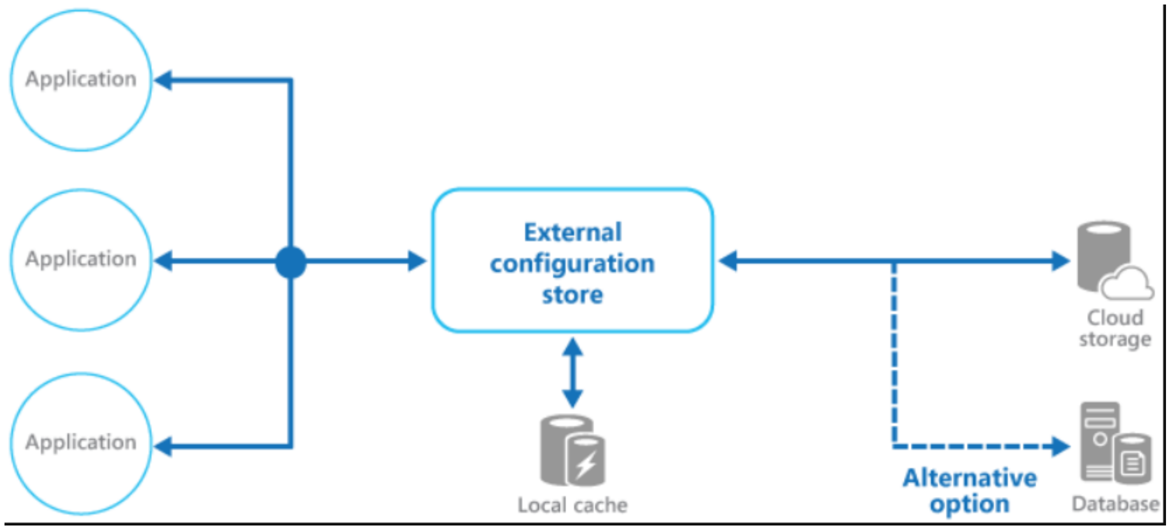
It should expose the information in a correctly typed and structured format.

The implementation might also need to authorize users' access to protect configuration data and be flexible enough to allow storage of multiple configuration versions (such as development, staging, or production, including many release versions of each one).

Many built-in configuration systems read the data when the application starts up and cache the data in memory to provide fast access and minimize the impact on application performance.

Depending on the type of backing store used and its latency, it might be helpful to implement a caching mechanism within the external configuration store.

For more information, see the Caching Guidance. The figure illustrates an overview of the External Configuration Store pattern with optional local cache.



This pattern is helpful for:

- Configuration settings are shared between multiple applications and application instances, or where a standard configuration must be enforced across various applications and application instances.
- A standard configuration system doesn't support all the required configuration settings, such as storing images or complex data types.
- As a complementary store for some application settings, they allow applications to override some or all the centrally stored settings.
- To simplify the administration of multiple applications and optionally monitor configuration settings by logging some or all types of access to the configuration store.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Introduction to Azure App Configuration

Completed

- 3 minutes

Azure App Configuration is a service for central management of application settings and feature flags.

Modern programs include distributed components, each that needs its settings.

It's prevalent with microservice-based applications and with serverless applications.

Distributed configuration settings can lead to hard-to-troubleshoot deployment errors.

Azure App Configuration service stores all the settings for your application and secures their access in one place.

Azure App Configuration service provides the following features:

- A fully managed service that can be set up in minutes.
- Flexible key representations and mappings.
- Tagging with labels.
- A point-in-time replay of settings.
- Dedicated UI for feature flag management.
- Comparison of two sets of configurations on custom-defined dimensions.
- Enhanced security through Azure managed identities.
- Complete data encryptions, at rest or in transit.
- Native integration with popular frameworks.

App Configuration complements Azure Key Vault, which is used to store application secrets. App Configuration makes it easier to implement the following scenarios:

- Centralize management and distribution of hierarchical configuration data for different environments and geographies.
- Dynamically change application settings without the need to redeploy or restart an application.
- Control feature availability in real time.

Use App Configuration

The easiest way to add an App Configuration store to your application is through one of Microsoft's client libraries.

Based on the programming language and framework, the following best methods are available to you.

Programming language and framework	How to connect

Programming language and framework	How to connect
.NET Core and ASP.NET Core	App Configuration provider for .NET Core
.NET Framework and ASP.NET	App Configuration builder for .NET
Java Spring	App Configuration client for Spring Cloud
Others	App Configuration REST API

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Examine Key-value pairs

Completed

- 4 minutes

Azure App Configuration stores configuration data as key-value pairs.

Keys

Keys serve as the name for key-value pairs and are used to store and retrieve corresponding values.

It's common to organize keys into a hierarchical namespace by using a character delimiter, such as `/` or `:` . Use a convention that's best suited for your application.

App Configuration treats keys as a whole. It doesn't parse keys to figure out how their names are structured or enforce any rule on them.

Keys stored in App Configuration are case-sensitive, Unicode-based strings.

The keys *app1* and *App1* are distinct in an App Configuration store.

When you use configuration settings within an application, keep it in mind because some frameworks handle configuration keys case-insensitively.

You can use any Unicode character in key names entered into App Configuration except for `*` , `,` , and `\` .

These characters are reserved. If you need to include a reserved character, you must escape it by using `\{Reserved Character}` .

There's a combined size limit of 10,000 characters on a key-value pair.

This limit includes all characters in the key, its value, and all associated optional attributes.

Within this limit, you can have many hierarchical levels for keys.

Design key namespaces

There are two general approaches to naming keys used for configuration data: flat or hierarchical.

These methods are similar from an application usage standpoint, but hierarchical naming offers several advantages:

- Easier to read. Instead of one long sequence of characters, delimiters in a hierarchical key name function as spaces in a sentence.
- Easier to manage. A key name hierarchy represents logical groups of configuration data.
- Easier to use. It's simpler to write a query that pattern-matches keys in a hierarchical structure and retrieves only a portion of configuration data.

Below are some examples of how you can structure your key names into a hierarchy:

- Based on component services

  ```
  AppName:Service1:ApiEndpoint
  AppName:Service2:ApiEndpoint
  ```

- Based on deployment regions

  ```
  AppName:Region1:DbEndpoint
  AppName:Region2:DbEndpoint
  ```

Label keys

Key values in App Configuration can optionally have a label attribute.

Labels are used to differentiate key values with the same key.

A key *app1* with labels *A* and *B* forms two separate keys in an App Configuration store.

By default, the label for a key value is empty or null.

Label provides a convenient way to create variants of a key. A common use of labels is to specify multiple environments for the same key:

```
Key = AppName:DbEndpoint & Label = Test
Key = AppName:DbEndpoint & Label = Staging
Key = AppName:DbEndpoint & Label = Production
```

Version key values

App Configuration doesn't version key values automatically as they're modified.

Use labels as a way to create multiple versions of a key value.

For example, you can input an application version number or a Git commit ID in labels to identify key values associated with a particular software build.

Query key values

Each key value is uniquely identified by its key plus a label that can be `null` . You query an App Configuration store for key values by specifying a pattern.

The App Configuration store returns all key values that match the pattern and their corresponding values and attributes.

Values

Values assigned to keys are also Unicode strings. You can use all Unicode characters for values.

There's an optional user-defined content type associated with each value.

Use this attribute to store information, for example, an encoding scheme, about a value that helps your application process it properly.

Configuration data stored in an App Configuration store, which includes all keys and values, is encrypted at rest and in transit.

App Configuration isn't a replacement solution for Azure Key Vault. Don't store application secrets in it.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Examine App configuration feature management

Completed

- 4 minutes

Feature management is a modern software development practice that decouples feature release from code deployment and enables quick changes to feature availability on demand.

Feature Flags are discussed in another module, but at this point, it's worth noting that Azure App Configuration Service can be used to store and manage feature flags. (It's also known as feature toggles, feature switches, and other names).

Basic concepts

Here are several new terms related to feature management:

- **Feature flag** : A feature flag is a variable with a binary state of *on* or *off* . The feature flag also has an associated code block. The state of the feature flag triggers whether the code block runs or not.
- **Feature manager** : A feature manager is an application package that handles the lifecycle of all the feature flags in an application. The feature manager typically provides more functionality, such as caching feature flags and updating their states.
- **Filter** : A filter is a rule for evaluating the state of a feature flag. A user group, a device or browser type, a geographic location, and a time window are all examples of what a filter can represent.

Effective implementation of feature management consists of at least two components working in concert:

- An application that makes use of feature flags.
- A separate repository that stores the feature flags and their current states.

How these components interact is illustrated in the following examples.

Feature flag usage in code

The basic pattern for implementing feature flags in an application is simple. You can think of a feature flag as a Boolean state variable used with an `if` conditional statement in your code:

```
if (featureFlag) {
    // Run the following code.
}
```

In this case, if `featureFlag` is set to `True` , the enclosed code block is executed; otherwise, it's skipped. You can set the value of `featureFlag` statically, as in the following code example:

```
bool featureFlag = true;
```

You can also evaluate the flag's state based on certain rules:

```
bool featureFlag = isBetaUser();
```

A slightly more complicated feature flag pattern includes an `else` statement as well:

```
if (featureFlag) {
    // This following code will run if the featureFlag value is true.
} else {
    // This following code will run if the featureFlag value is false.
}
```

Feature flag declaration

Each feature flag has two parts: a name and a list of one or more filters used to evaluate if a feature's state is *on* (that is when its value is `True`).

A filter defines a use case for when a feature should be turned on.

When a feature flag has multiple filters, the filter list is traversed until one of the filters determines the feature should be enabled.

At that point, the feature flag is *on* , and any remaining filter results are skipped. If no filter indicates the feature should be enabled, the feature flag is *off* .

The feature manager supports *appsettings.json* as a configuration source for feature flags.

The following example shows how to set up feature flags in a JSON file:

```
"FeatureManagement": {
    "FeatureA": true, // Feature flag set to on
    "FeatureB": false, // Feature flag set to off
    "FeatureC": {
        "EnabledFor": [
            {
                "Name": "Percentage",
                "Parameters": {
                    "Value": 50
                }
            }
        ]
    }
}
```

Feature flag repository

To use feature flags effectively, you need to externalize all the feature flags used in an application.

This approach allows you to change feature flag states without modifying and redeploying the application itself.

Azure App Configuration is designed to be a centralized repository for feature flags.

You can use it to define different kinds of feature flags and manipulate their states quickly and confidently.

You can then use the App Configuration libraries for various programming language frameworks to easily access these feature flags from your application.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Integrate Azure Key Vault with Azure Pipelines

Completed

- 1 minute

Applications contain many secrets, such as:

- Connection strings.
- Passwords.
- Certificates.
- Tokens, which, if leaked to unauthorized users, can lead to a severe security breach.

It can result in severe damage to the organization's reputation and compliance issues with different governing bodies.

Azure Key Vault allows you to manage your organization's secrets and certificates in a centralized repository.

The secrets and keys are further protected by Hardware Security Modules (HSMs).

It also provides versioning of secrets, full traceability, and efficient permission management with access policies.

For more information on Azure Key Vault, visit What is Azure Key Vault .

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Manage secrets, tokens and certificates

Completed

- 4 minutes

Azure Key Vault helps solve the following problems:

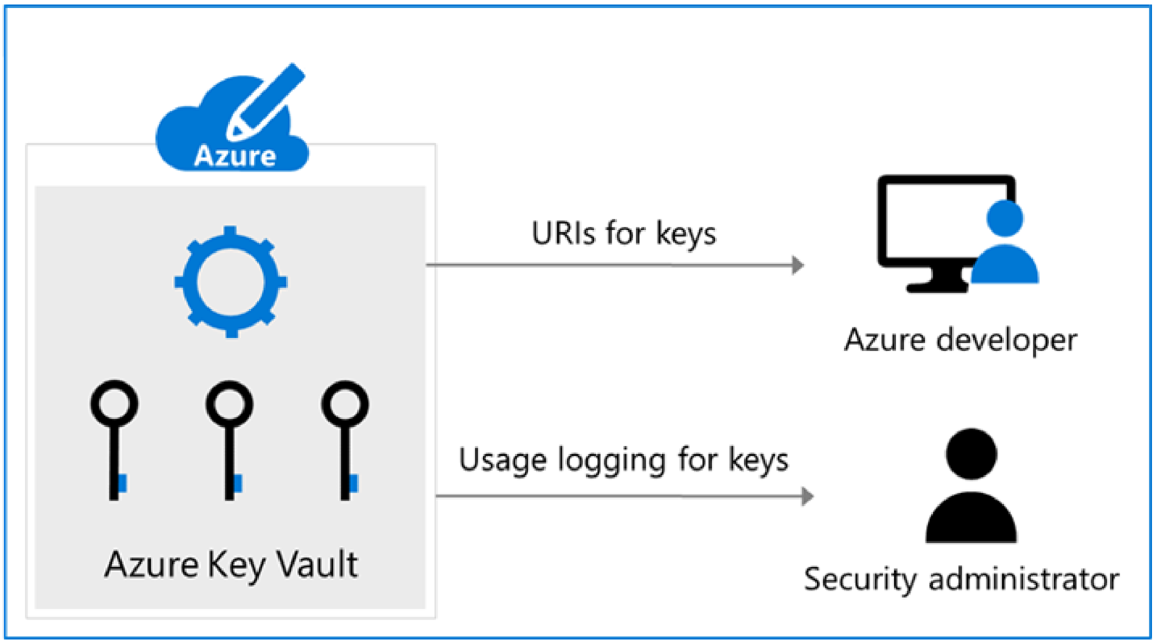
- **Secrets management** - Azure Key Vault can be used to store securely and tightly control access to tokens, passwords, certificates, API keys, and other secrets.

- **Key management** - Azure Key Vault can also be used as a key management solution. Azure Key Vault makes it easy to create and control the encryption keys used to encrypt your data.

- **Certificate management** - Azure Key Vault is also a service that lets you easily provision, manage, and deploy public and private Secure Sockets Layer/Transport

Layer Security (SSL/TLS) certificates for use with Azure. And your internal connected resources.

- **Store secrets backed by hardware security modules** - The secrets and keys can be protected by software or FIPS 140-2 Level 2 validates HSMs.



Why use Azure Key Vault?

Centralize application secrets

Centralizing the storage of application secrets in Azure Key Vault allows you to control their distribution.

Key Vault dramatically reduces the chances that secrets may be accidentally leaked.

When using Key Vault, application developers no longer need to store security information in their applications. It eliminates the need to make this information part of the code.

For example, an application may need to connect to a database. Instead of storing the connection string in the app codes, store it securely in Key Vault.

Your applications can securely access the information they need by using URIs that allow them to retrieve specific versions of a secret after the application's key or secret is stored in Azure Key Vault.

It happens without having to write custom code to protect any of the secret information.

Securely store secrets and keys

Secrets and keys are safeguarded by Azure, using industry-standard algorithms, key lengths, and hardware security modules (HSMs).

The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated.

Access to a key vault requires proper authentication and authorization before a caller (user or application) can get access.

Authentication establishes the identity of the caller, while authorization determines the operations that they can do.

Authentication is done via Microsoft Entra ID. Authorization may be done via role-based access control (RBAC) or Key Vault access policy.

RBAC is used when dealing with the management of the vaults, and a key vault access policy is used when attempting to access data stored in a vault.

Azure Key Vaults may be either software- or hardware-HSM protected.

You can import or generate keys in hardware security modules (HSMs) that never leave the HSM boundary when you require added assurance.

Microsoft uses Thales hardware security modules. You can use Thales tools to move a key from your HSM to Azure Key Vault.

Finally, Azure Key Vault is designed so that Microsoft doesn't see or extract your data.

Monitor access and use

Once you've created a couple of Key Vaults, you'll want to monitor how and when your keys and secrets are accessed.

You can do it by enabling logging for Key Vault. You can configure Azure Key Vault to:

- Archive to a storage account.
- Stream to an Event Hubs.
- Send the logs to Log Analytics.

You have control over your logs, and you may secure them by restricting access, and you may also delete logs that you no longer need.

Simplified administration of application secrets

When storing valuable data, you must take several steps. Security information must be secured. It must follow a lifecycle. It must be highly available.

Azure Key Vault simplifies it by:

- Removing the need for in-house knowledge of Hardware Security Modules.
- Scaling up on short notice to meet your organization's usage spikes.
- Replicating the contents of your Key Vault within a region and to a secondary region. It ensures high availability and takes away the need for any action from the administrator to trigger the failover.
- Providing standard Azure administration options via the portal, Azure CLI and PowerShell.
- Automating specific tasks on certificates that you purchase from Public CAs, such as enrollment and renewal.

Also, Azure Key Vaults allow you to segregate application secrets.

Applications may access only the vault they can access, and they can only do specific operations.

You can create an Azure Key Vault per application and restrict the secrets stored in a Key Vault to a particular application and team of developers.

Integrate with other Azure services

As a secure store in Azure, Key Vault has been used to simplify scenarios like Azure Disk Encryption, the always encrypted functionality in SQL Server and Azure SQL Database, Azure web apps.

Key Vault itself can integrate with storage accounts, Event Hubs, and log analytics.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Examine DevOps inner and outer loop

Completed

- 2 minutes

While you can store configuration and secrets together, it violates our separation of concern principle, so the recommendation is to use a different store for persisting secrets.

It allows a secure channel for sensitive configuration data, such as ConnectionStrings.

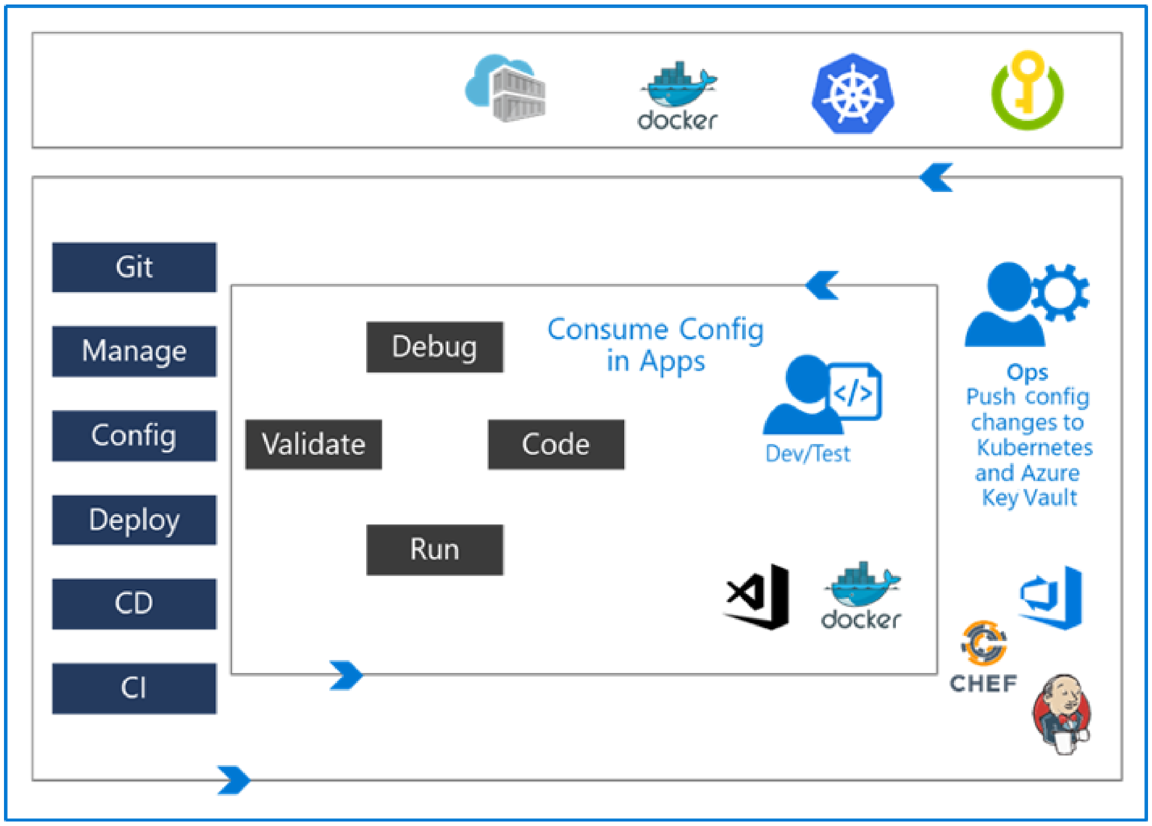
It enables the operations team to have credentials, certificates, and tokens in one repository and minimizes the security risk if the Configuration Store gets compromised.

The following diagram shows how these roles play together in a DevOps inner and outer loop.

The inner loop is focused on the developer teams iterating over their solution development; they consume the configuration published by the Outer Loop.

The Ops Engineer governs the Configuration management.

They push changes into Azure KeyVault and Kubernetes that are further isolated per environment.



Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Integrate Azure Key Vault with Azure DevOps

Completed

- 40 minutes

Estimated time: 40 minutes.

Lab files: none.

Scenario

Azure Key Vault provides secure storage and management of sensitive data, such as keys, passwords, and certificates. Azure Key Vault includes support for hardware security modules and a range of encryption algorithms and key lengths. Using Azure Key Vault can minimize the possibility of disclosing sensitive data through source code, a common mistake developers make. Access to Azure Key Vault requires proper authentication and authorization, supporting fine-grained permissions to its content.

In this lab, you'll see how you can integrate Azure Key Vault with an Azure Pipeline by using the following steps:

- Create an Azure Key vault to store an ACR password as a secret.
- Create an Azure Service Principal to access Azure Key Vault's secrets.
- Configure permissions to allow the Service Principal to read the secret.
- Configure the pipeline to retrieve the password from the Azure Key Vault and pass it on to subsequent tasks.

Objectives

After completing this lab, you'll be able to:

- Create a Microsoft Entra service principal.
- Create an Azure Key Vault.

Requirements

- This lab requires **Microsoft Edge** or an Azure DevOps-supported browser .
- **Set up an Azure DevOps organization:** If you don't already have an Azure DevOps organization that you can use for this lab, create one by following the instructions available at Create an organization or project collection .
- Identify an existing Azure subscription or create a new one.
- Verify that you have a Microsoft or Microsoft Entra account with the Owner role in the Azure subscription and the Global Administrator role in the Microsoft Entra tenant associated with the Azure subscription. For details, refer to List Azure role assignments using the Azure portal and View and assign administrator roles in Microsoft Entra ID .

Exercises

During this lab, you'll complete the following exercises:

- Exercise 0: Configure the lab prerequisites.
- Exercise 1: Setup CI Pipeline to build eShopOnWeb container.
- Exercise 2: Remove the Azure lab resources.

Launch Exercise

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Enable Dynamic Configuration and Feature Flags

Completed

- 60 minutes

Estimated time: 60 minutes.

Lab files: none.

Scenario

Azure App Configuration provides a service to manage application settings and feature flags centrally. Modern programs, especially those running in a cloud, generally have many distributed components. Spreading configuration settings across these components can lead to hard-to-troubleshoot errors during application deployment. Use App Configuration to store all the settings for your application and secure their accesses in one place.

Objectives

After completing this lab, you'll be able to:

- Enable dynamic configuration.
- Manage feature flags.

Requirements

- This lab requires **Microsoft Edge** or an Azure DevOps-supported browser .
- **Set up an Azure DevOps organization:** If you don't already have an Azure DevOps organization that you can use for this lab, create one by following the instructions available at Create an organization or project collection .
- Identify an existing Azure subscription or create a new one.
- Verify that you have a Microsoft or Microsoft Entra account with the Contributor or the Owner role in the Azure subscription. For details, refer to List Azure role assignments using the Azure portal and View and assign administrator roles in Microsoft Entra ID .

Exercises

During this lab, you'll complete the following exercises:

- Exercise 0: Configure the lab prerequisites.
- Exercise 1: Import and run CI/CD Pipelines.
- Exercise 2: Manage Azure App Configuration.

- Exercise 3: Remove the Azure lab resources.

Launch Exercise

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Knowledge check

Completed

- 5 minutes

Choose the best response for each question. Then select **Check your answers** .

Check your knowledge

1.

Which of the following roles is responsible for generating and maintaining the life cycle of configuration values?

○

Configuration Custodian.

○

Configuration Consumer.

○

Developer.

2.

Which of the following choices enables applications to have no direct access to the secrets, which helps improve the security & control?

○

Library.

○

Azure Key Vault.

○

Azure Pipelines.

3.

Which of the following choices isn't a benefit of Azure Key Vault?

○

Code secure files.

○

Certificate management.

○

Secrets management.

Check your answers

You must answer all questions before checking your work.

Continue

Need help? See our troubleshooting guide or provide specific feedback by reporting an issue .

Summary

Completed

- 1 minute

This module explored ways to rethink application configuration data and the separation of concerns method. It helped you understand configuration patterns and how to integrate Azure Key Vault with Azure Pipelines.

You learned how to describe the benefits and usage of:

- Rethink application configuration data.
- Understand separation of concerns.
- Integrate Azure Key Vault with Azure Pipelines.
- Manage secrets, tokens, and certificates.

Learn more

- Use Azure Key Vault secrets in Azure Pipelines - Azure Pipelines | Microsoft Docs .
- Define variables - Azure Pipelines | Microsoft Docs .
- Integrate Azure App Configuration using a continuous integration and delivery pipeline | Microsoft Docs .

Continue