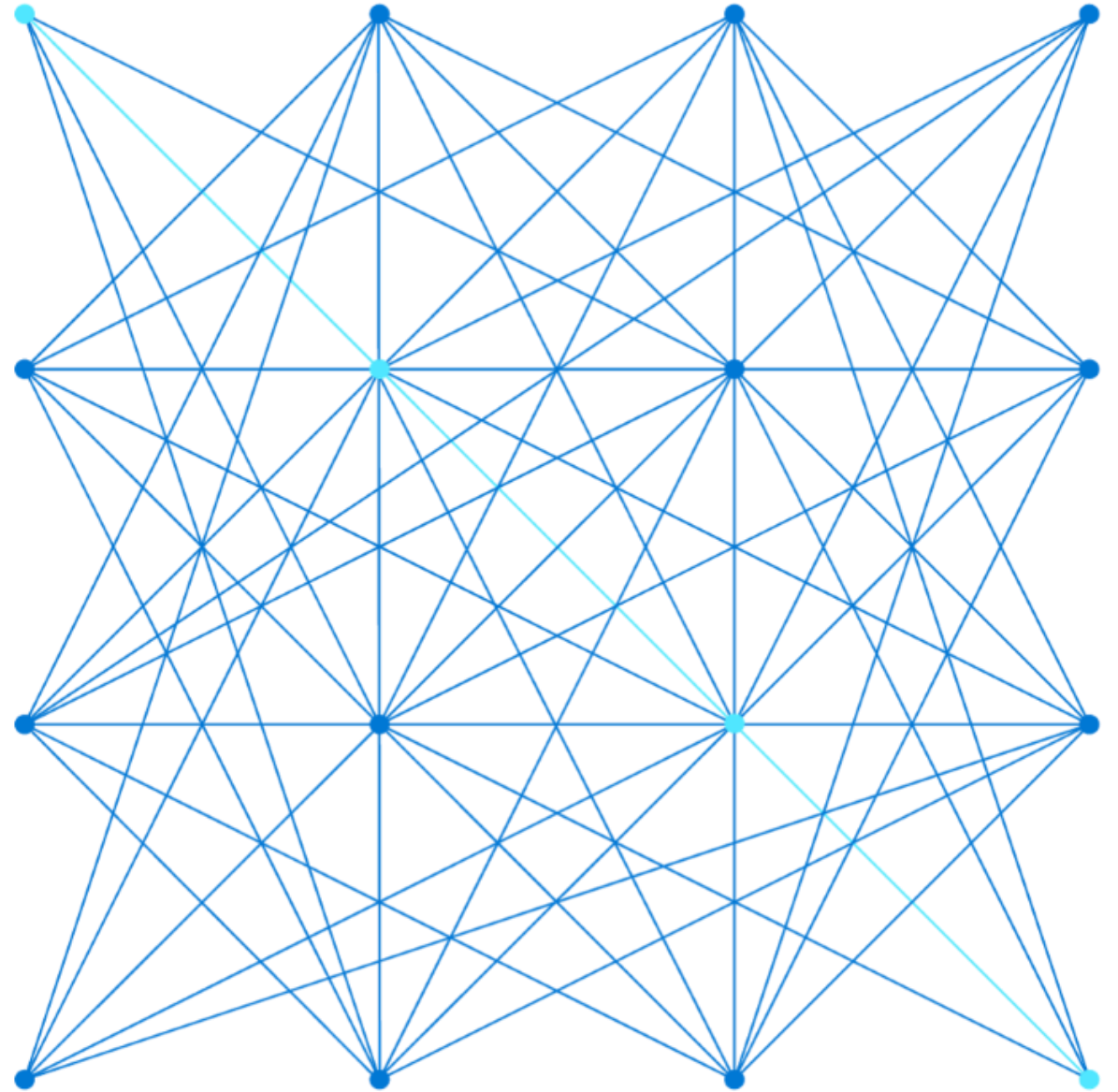
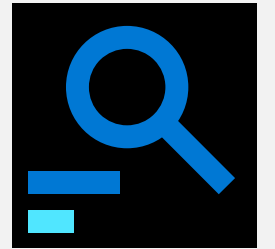


AZ-400.00

Module 12: Implementing an Appropriate Deployment Pattern



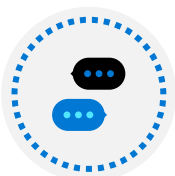
Lesson 01: Module overview



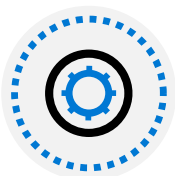
Module overview



Lesson 1: Module overview



Lesson 2: Introduction to deployment patterns



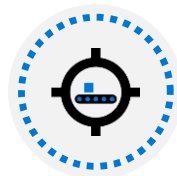
Lesson 3: Implement blue-green deployment



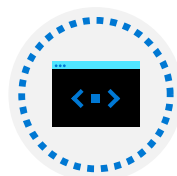
Lesson 4: Feature toggles



Lesson 5: Canary releases



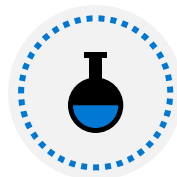
Lesson 6: Dark launching



Lesson 7: A/B testing



Lesson 8: Progressive exposure deployment



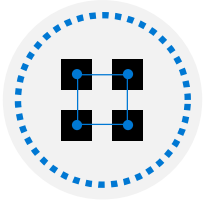
Lesson 9: Lab



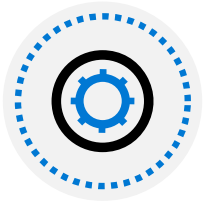
Lesson 10: Module review and takeaways

Learning objectives

After completing this module, students will be able to:



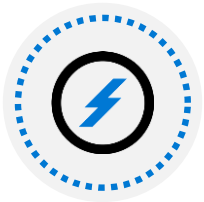
Describe deployment patterns



Implement blue-green deployment

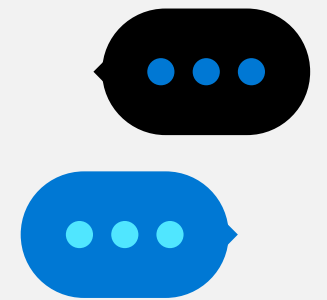


Implement canary release



Implement progressive exposure deployment

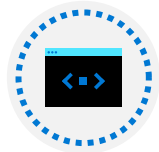
Lesson 02: Introduction to deployment patterns



Introduction to continuous delivery and continuous deployment



Continuous Delivery is an extension of Continuous Integration



Deployment is only one step. We also need to consider:

- Are most tests automated?
 - Is code safe and maintainable?
 - Architecture. Monolith or separation of concerns?
-

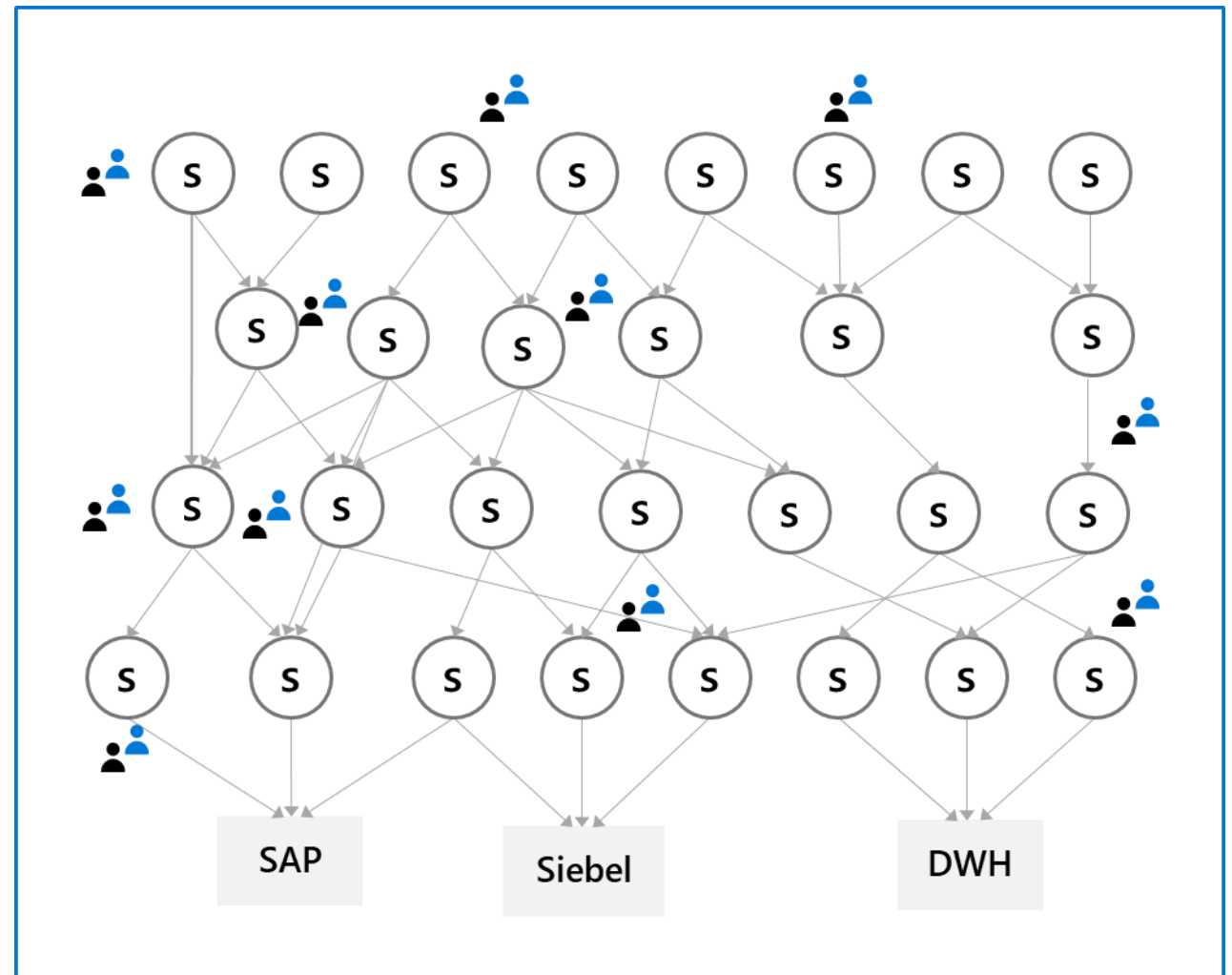
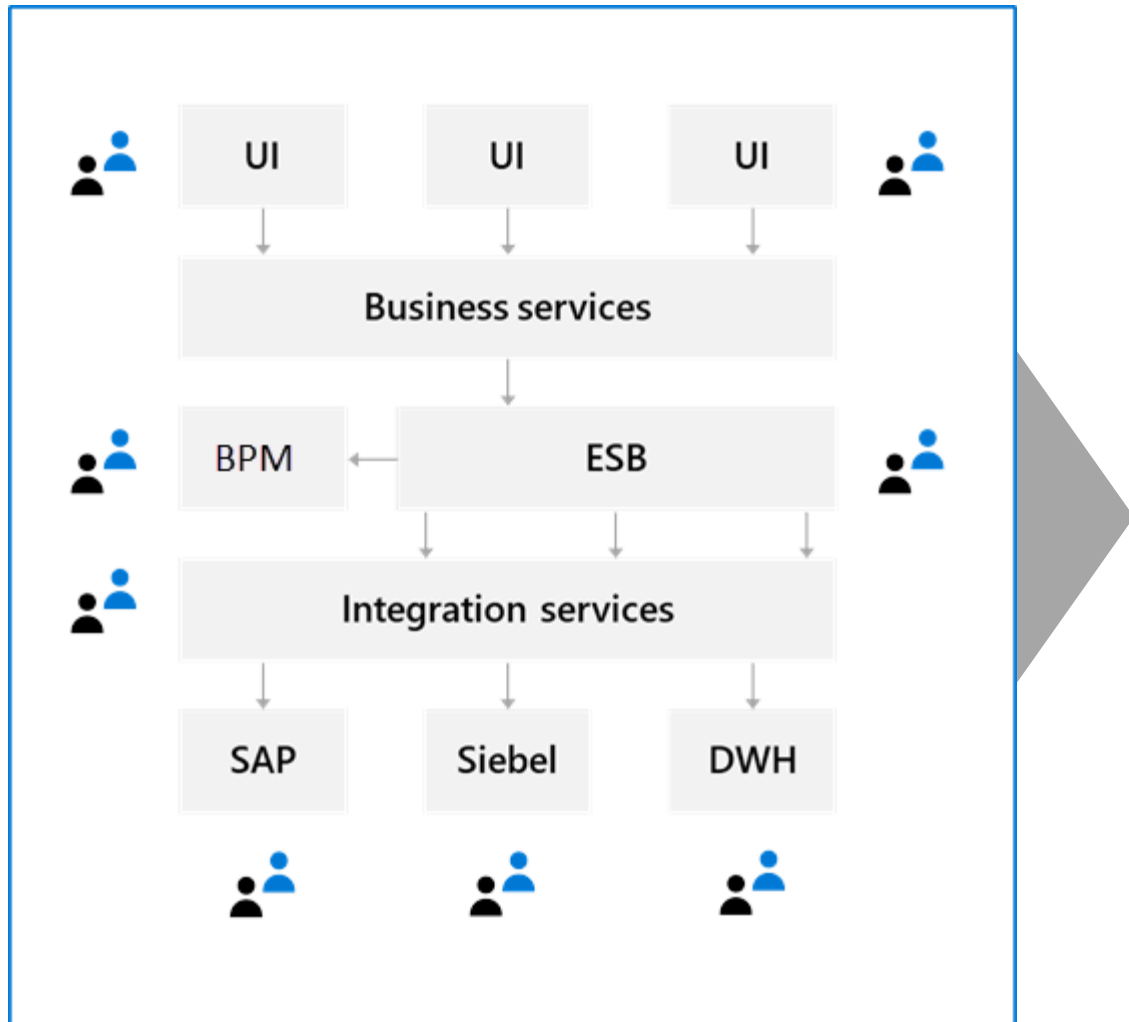


Monoliths are hard to deliver because of all the dependencies:

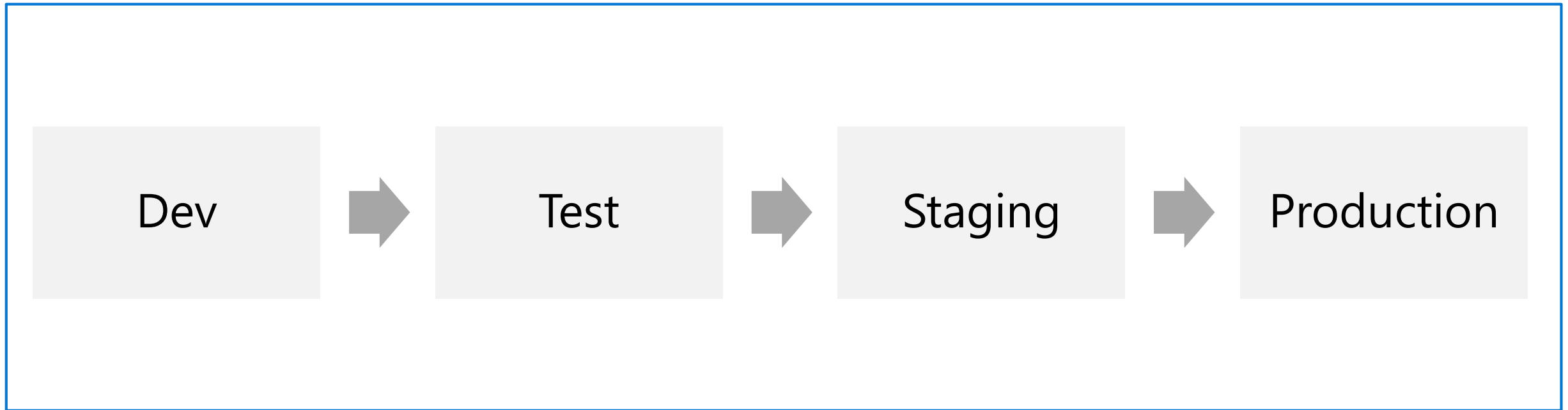
- Break up in smaller pieces
- Microservices

Microservices architecture

Each microservice has its own lifecycle and Continuous Delivery pipeline

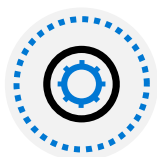


Classical deployment patterns (Big Bang)



What has your experience been with this?
(Maybe it works in test, then fails in production?)

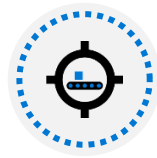
Modern deployment patterns (Overview)



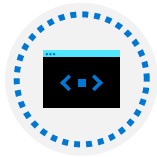
Blue-green deployments



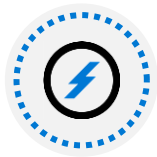
Canary releases



Dark launching



A/B testing



Progressive exposure deployment/ring-based deployments



Feature toggles

We'll look at these soon



Thoughts: A critical look at your architecture

Are your **architecture** and the current state of your software **ready for continuous delivery**?

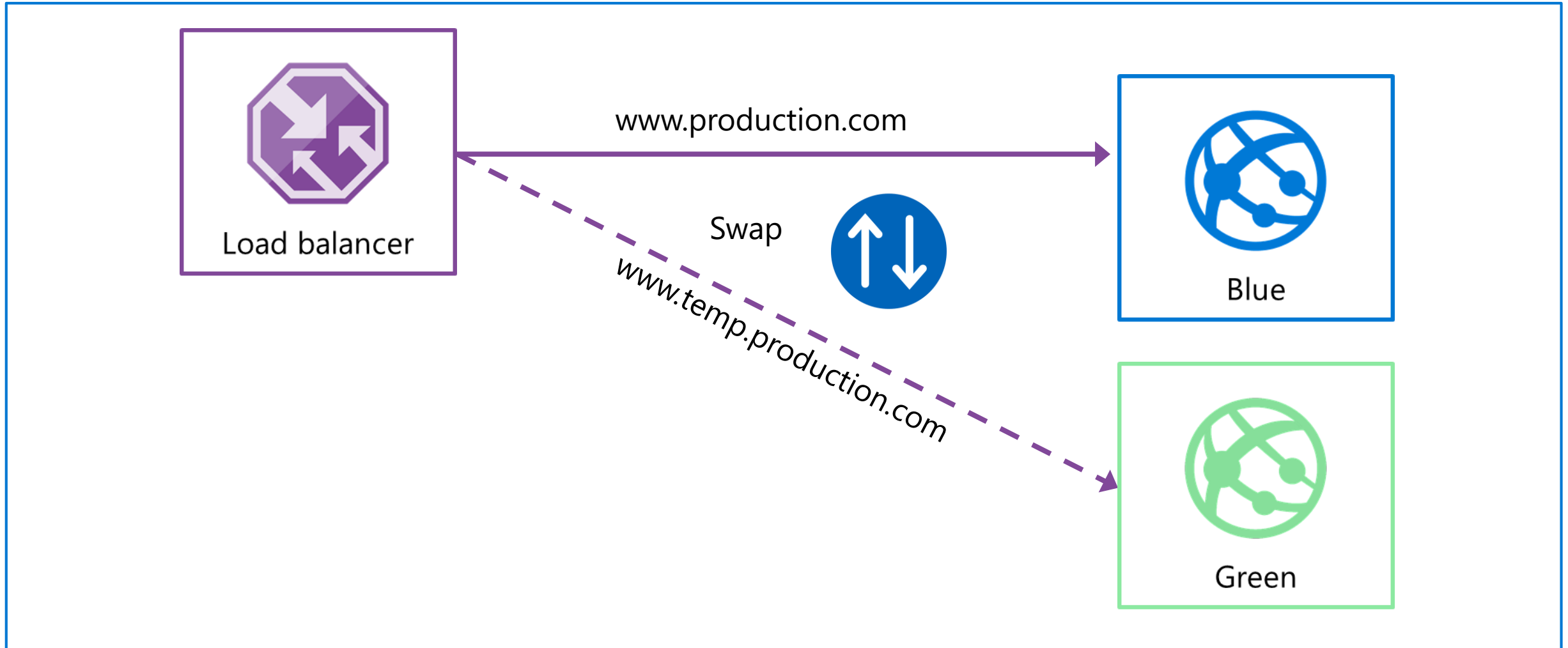
Topics you might want to consider are:

-  Is your software built as one big 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 need to improve to implement continuous delivery?

Lesson 03: Implement blue-green deployment



Blue-green deployment



Deployment slots



What is it?

A way to set up multiple environments (blue/green) and swap.



Why do you need it?

- When you want to deploy with zero downtime
- When you need to test in production
- When you want easy rollback

Demonstration: Set up a blue-green deployment

mw-demo-BlueGreen

DEMO

Lesson 04: Feature toggles



Introduction to feature toggles

D-AppConfigExample



What is it?

Mechanism to separate feature deployment from feature exposure

A.k.a. feature flippers, feature flags, feature switch, conditional feature, etc.



Why do you need it?

- It enables you to give control back to the business on when to release the feature
- Enables A/B testing, canary releases and dark launching
- It provides an alternative to keeping multiple branches in version control
- Enables change without redeployment

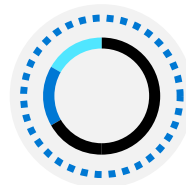
Feature toggle maintenance



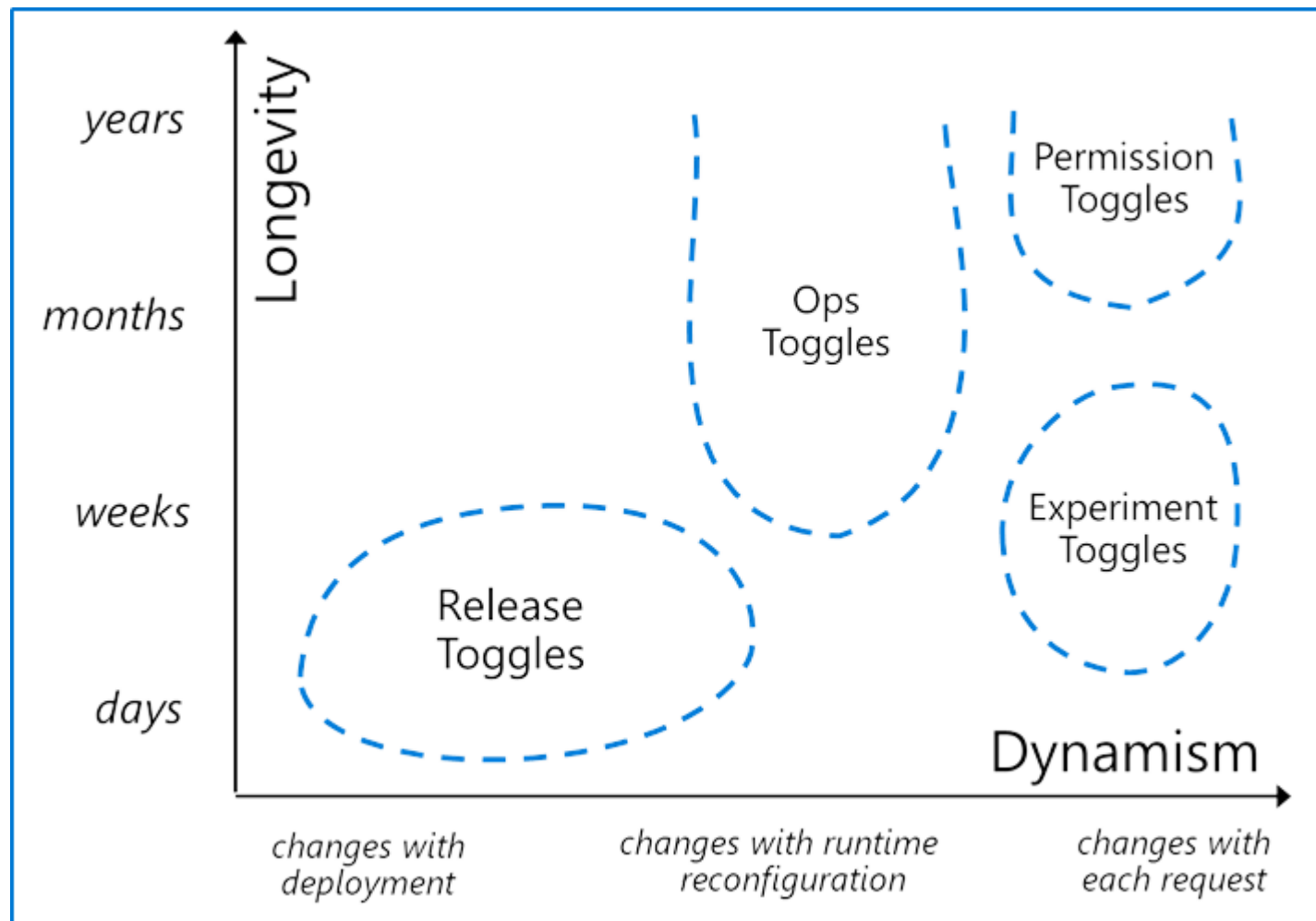
Feature toggles need to be maintained



Remove them when you can



It is technical debt if you keep them around



Lesson 05: Canary releases



Canary releases

What is it?

Releasing a feature to a limited subset of end users

When you want to gradually roll out a feature to ensure success

How to implement it

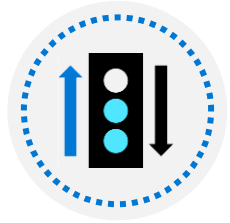
Use a combination of feature toggles, traffic routing and deployment slots

Route traffic % to a deployment slot with the new feature enabled

Target specific user segment (via feature toggles)

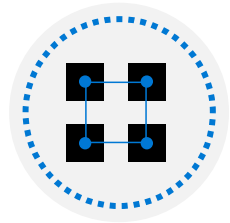


Traffic manager



What is it?

Traffic manager provides the ability to control how requests from web clients are distributed to apps in Azure App Service. Within an app service, it routes traffic between deployment slots



Why do you need it?

It enables failover and load distribution capabilities

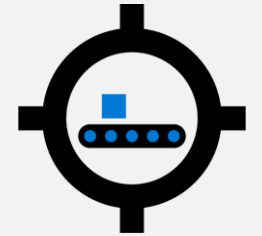
It enables you to deploy to a slot and then slowly move traffic over to the other slot



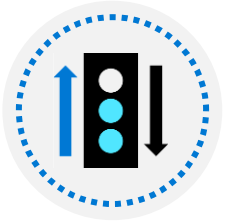
What does it offer?

It provides several options for how to distribute traffic, all based on availability

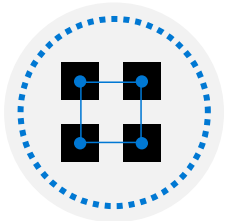
Lesson 06: Dark launching



Dark launching



Like canary releases, but the aim is to assess response of users to new features

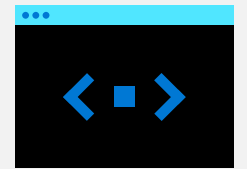


Testing of the back end is not the primary aim



Users are often not aware they are being used as guinea pigs for the new feature

Lesson 07: A/B testing



A/B testing

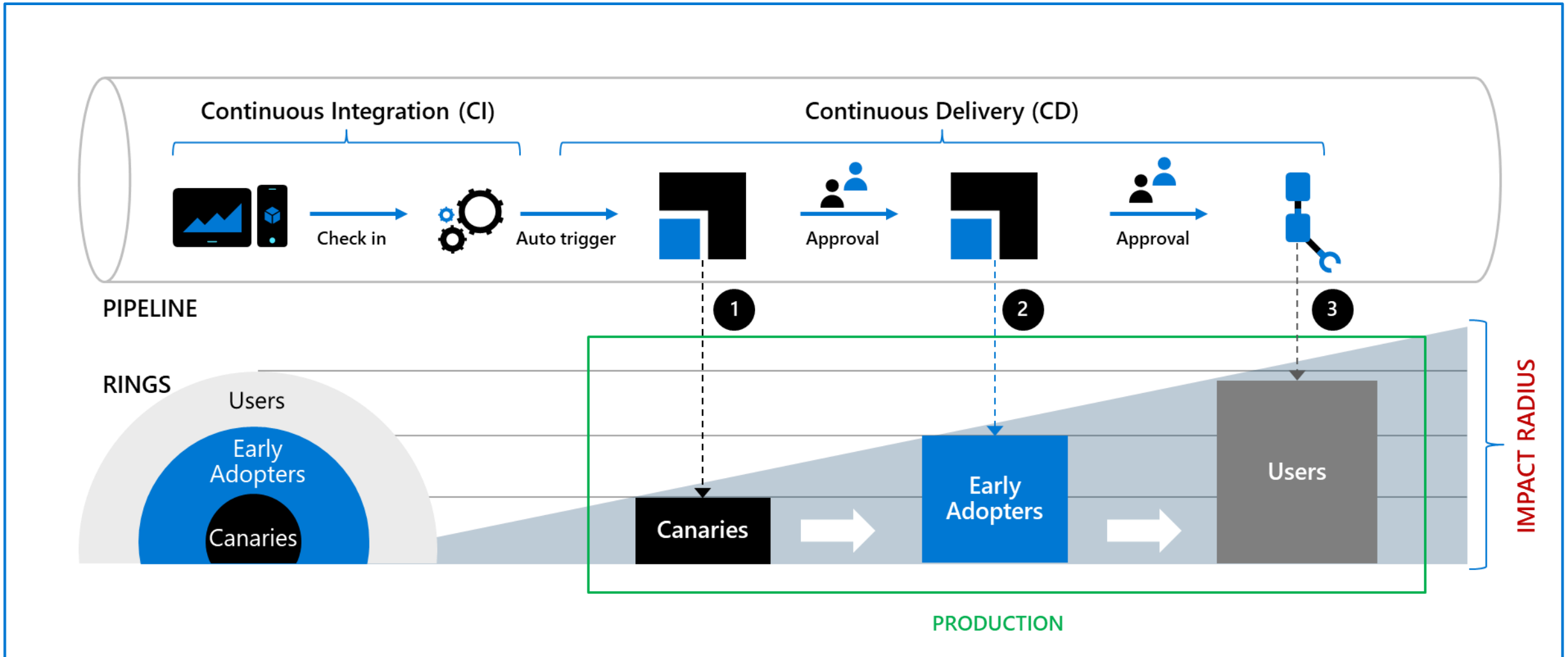
A/B testing is an experiment where two or more variants are shown to users at random, and statistical analysis is used to determine which variation performs better for a given conversion goal



Lesson 08: Progressive exposure deployment



CI/CD with deployment **rings** (aka *progressive disclosure*)



Demonstration: Ring-based deployment

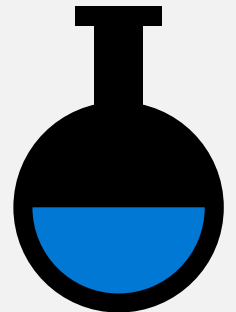
DEMO

az-p-RingBasedDeploymentDemo

Mw-Demo: Ring Based Deployment

Lesson 09: Lab

We're skipping these.
Can do outside class time



Feature flag management with LaunchDarkly and Azure DevOps

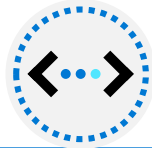


Lab overview:

In this lab, you will create and configure wiki in an Azure DevOps, including managing markdown content and creating a Mermaid diagram.

Objectives:

- Create feature flags in LaunchDarkly
- Integrate LaunchDarkly with Web applications
- Automatically roll-out LaunchDarkly feature flags as part of Azure DevOps release pipelines



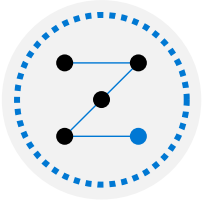
Duration:



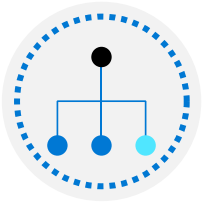
Lesson 10: Module review and takeaways



What did you learn?



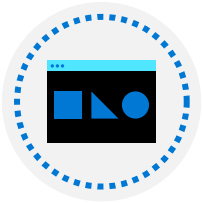
Describe deployment patterns



Implement blue-green deployment



Implement canary release



Implement progressive exposure deployment

Module review questions

1 What is the easiest way to create a staging environment for an Azure WebApp?

2 What Azure-based tool can you use to divert a percentage of your web traffic to a newer version of an Azure website?

3 What characteristics make users suitable for working with canary deployments?

4 What is a potential disadvantage of using canary deployments?

5 Apart from the traffic routing method, what else does Azure Traffic Manager consider when making routing decisions?