Microsoft

AZ-400.00 Learning Path 05: Implement a secure continuous deployment using Azure Pipelines

Agenda

- Module 01: Introduction to deployment patterns.
- Module 02: Implement blue-green deployment and feature toggles.
- Module 03: Implement canary releases and dark launching.
- Module 04: Implement A/B testing and progressive exposure deployment.
- Module 05: Integrate with identity management systems.
- Module 06: Manage application configuration data.
- Labs & Learning Path review and takeaways.

Microsoft

Learning Path overview

Learning objectives

After completing this Learning Path, students will be able to:

1. Describe deployment patterns
2. Implement blue-green deployment
3. Implement canary release
4. Implement progressive exposure deployment
5. Manage application config and secrets
6. Deploy to environment securely using a service connection
7. Integrate Azure Key Vault with a pipeline

Module 01: Introduction to deployment patterns

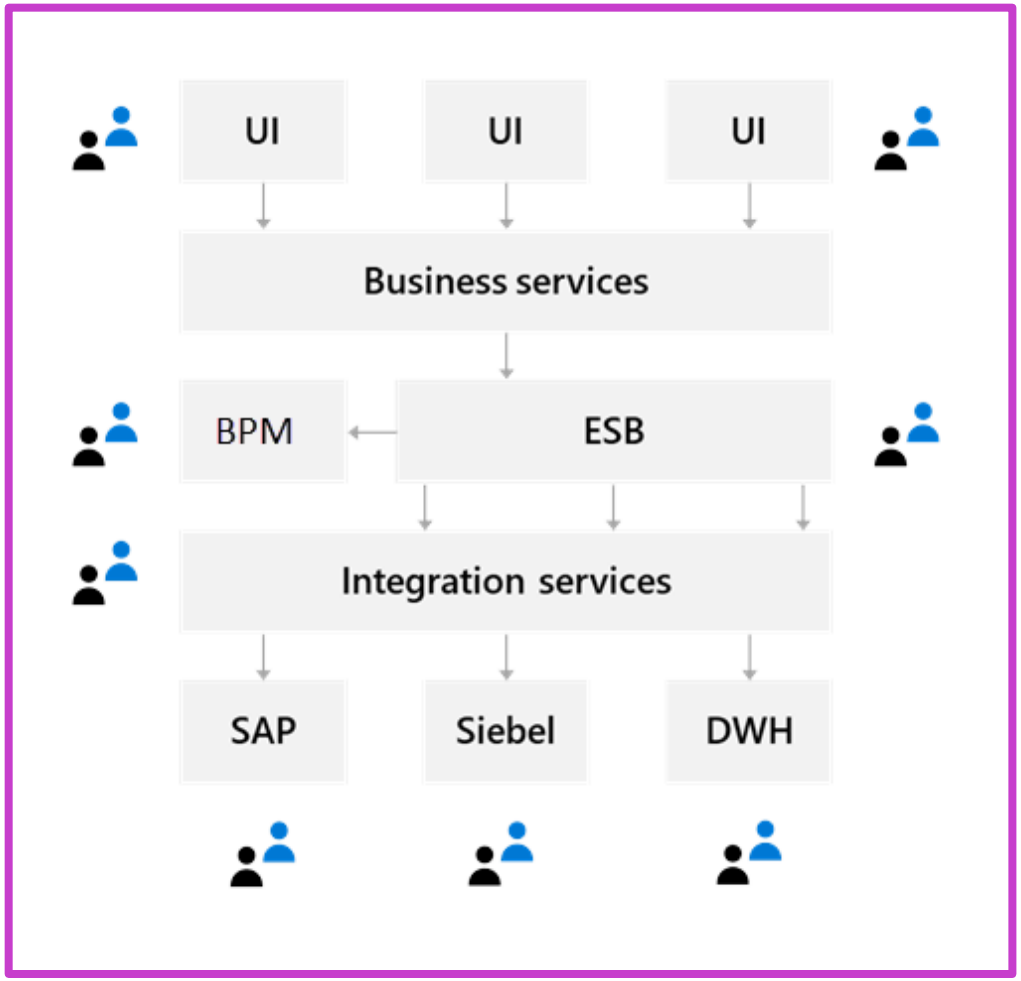
Introduction

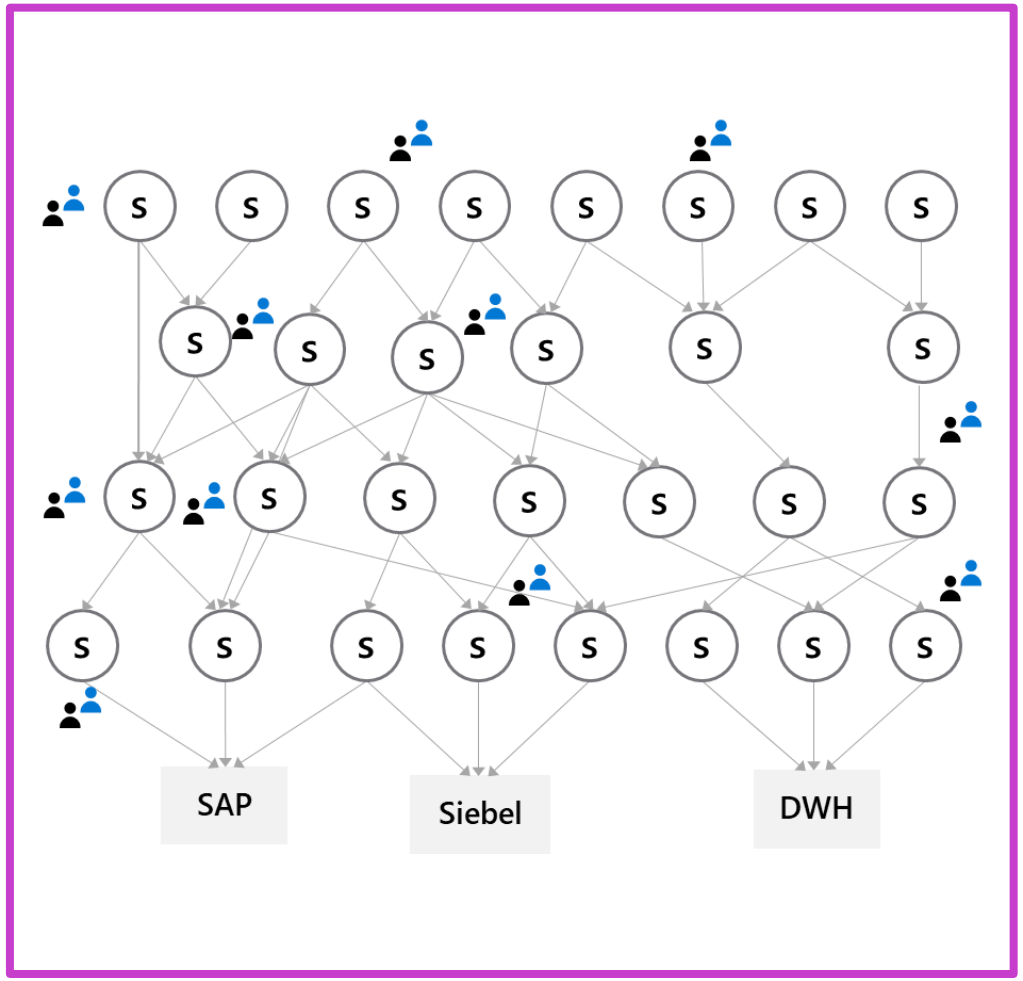
1. Continuous delivery is more than release management

2. **Deployment is only one step:**
   - Testing
   - Safe coding
   - Architecture

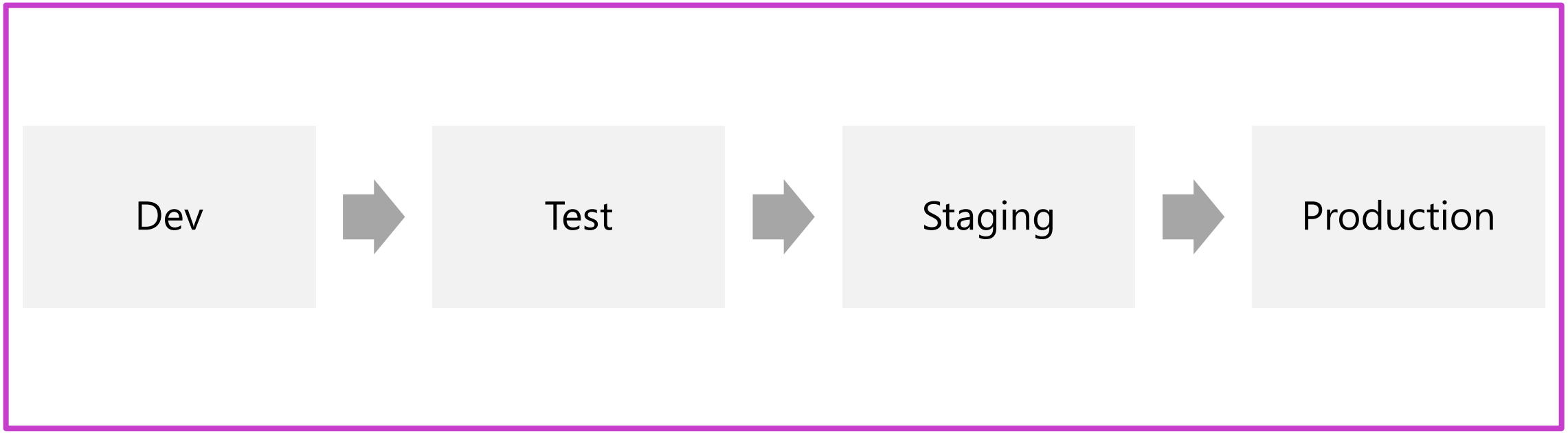
3. **Monoliths are hard to deliver because of all the dependencies:**
   - Break up in smaller pieces
   - Microservices

Explore microservices architecture





Examine classical deployment patterns



Understand modern deployment patterns

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

Discussion: A critical look at your architecture

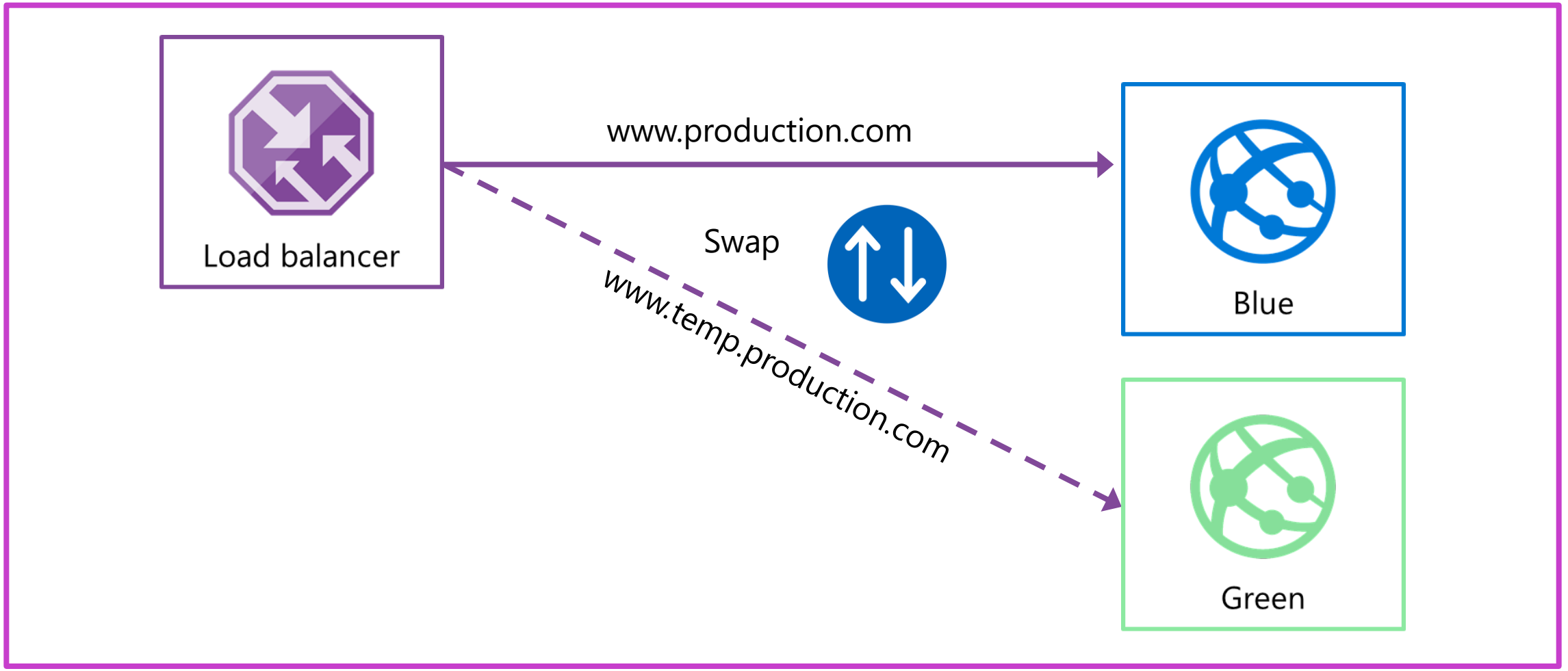
Are your architecture and the current state of your software ready for continuous delivery?

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?

Module 02: Implement blue-green deployment and feature toggles

What is blue-green deployment?



Explore deployment slots

What is it?

A way to set up multiple environments and swap between environments

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

Swap

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

Demonstration: Set up a blue–green deployment

DEMO

Introduction to feature toggles

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

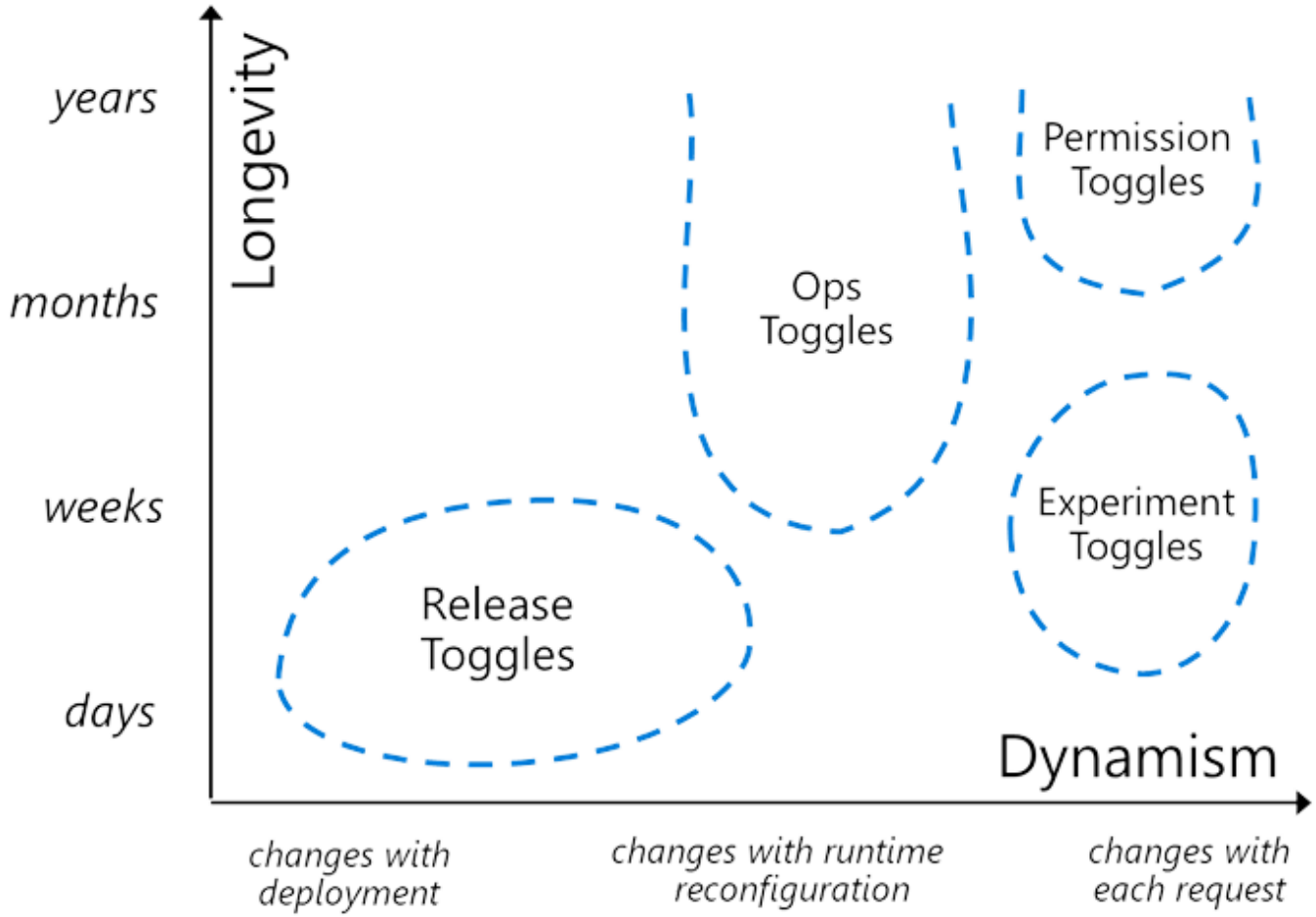
Enables A/B testing, canary releases and dark launching

It provides an alternative to keeping multiple branches in version control

Enables change without redeployment

Describe feature toggle maintenance

- Feature toggles need to be maintained
- Remove them when you can
- It is technical debt if you keep them around



Source: http://martinfowler.com/articles/feature-toggles.html

Module 03: Implement canary releases and dark launching

Explore 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)

Examine 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

Understand 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

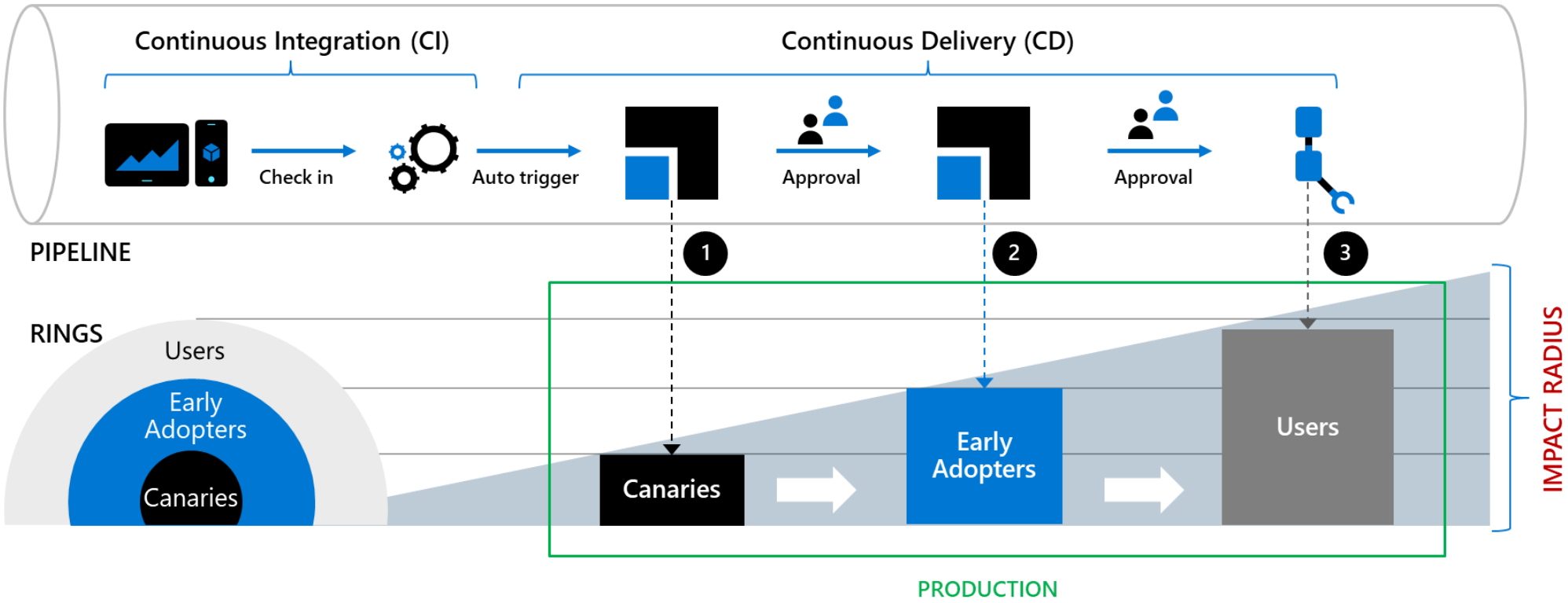
Users are often not aware they are being used to test the new feature

Module 04: Implement A/B testing and progressive exposure deployment

What is 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

Explore CI/CD with ring-based deployment



Demonstration: Ring-based deployment

DEMO

Module 05: Integrate with identity management systems

Integrate GitHub with single sign-on (SSO)

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

Connect identity provider to GitHub at organization level

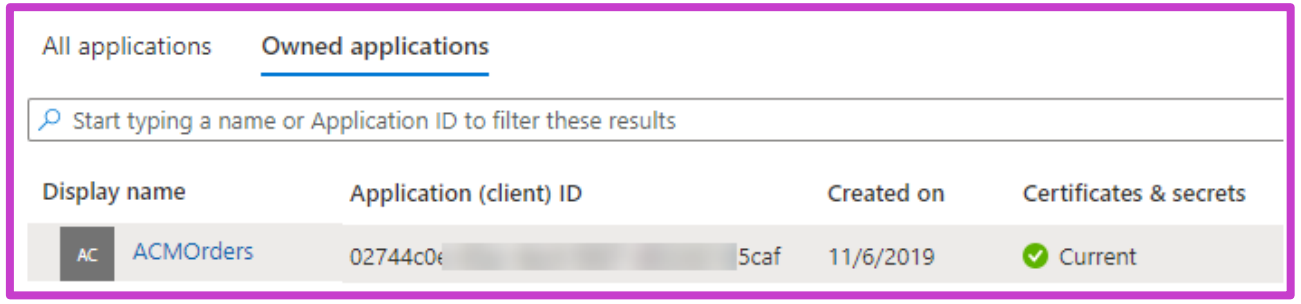
Both SAML and SCIM support available

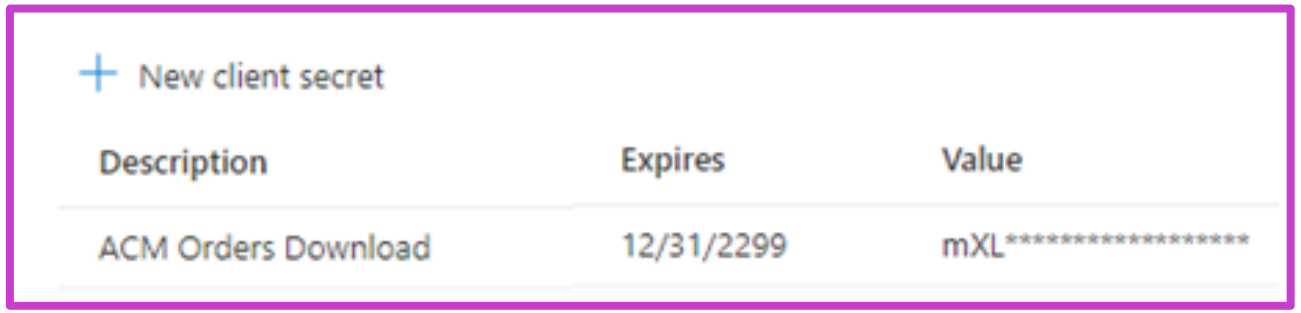
Explore service principals

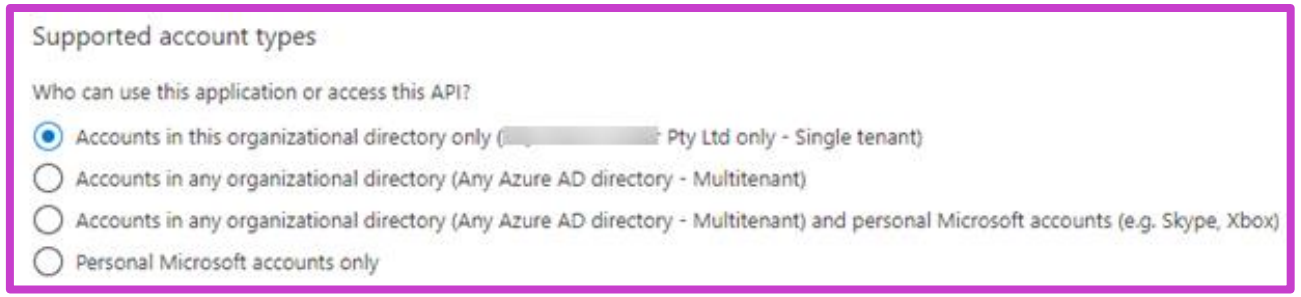
- Register an Entra ID Application
- Create a client secret
- Grant permissions to the identity

To connect, application presents:

- TenantID
- Application ID
- Client Secret





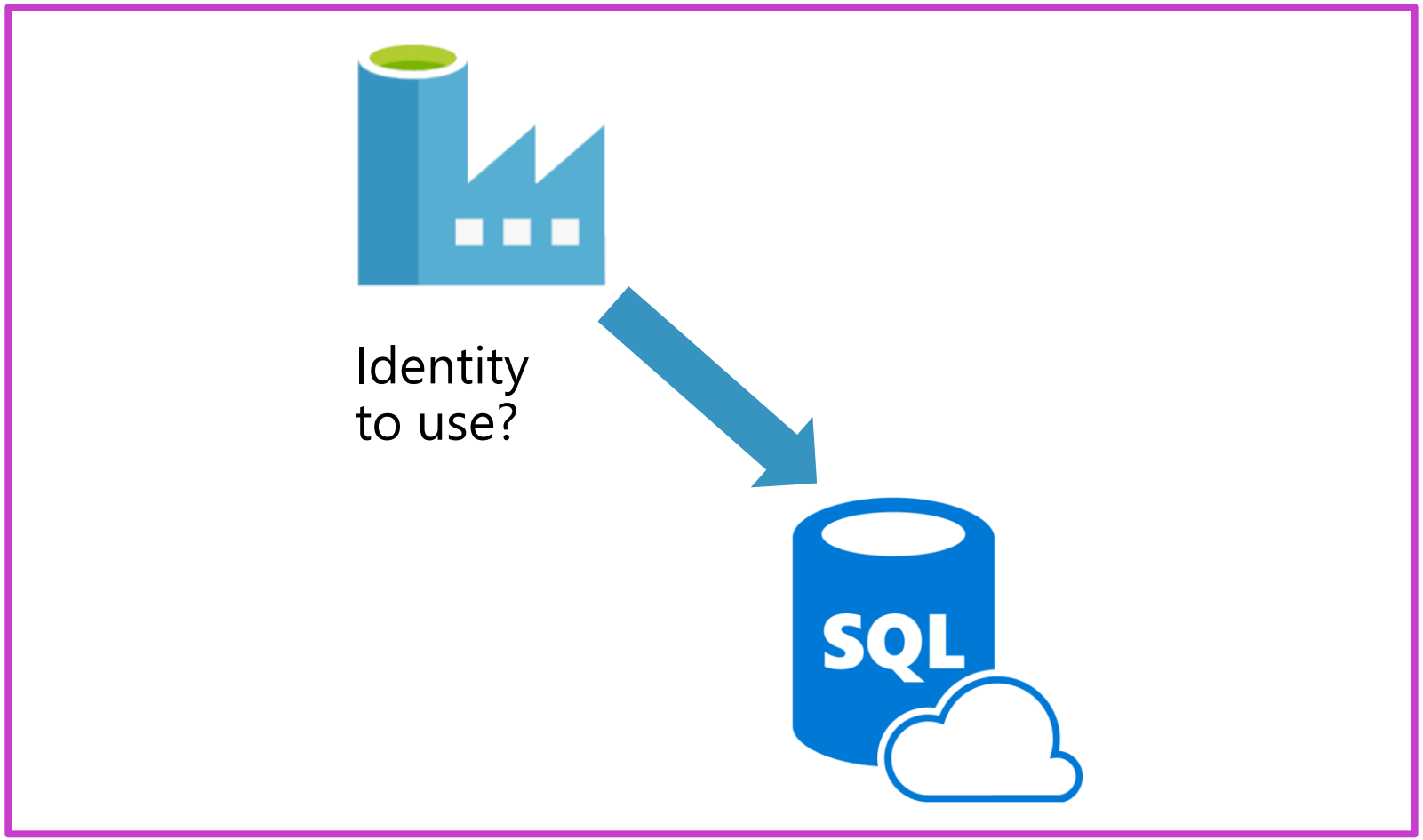


Explore Managed Identity

Two types of managed identity:

- System-assigned (many services expose these identities)
- User-assigned (create a managed identity and assign it to services)

Managed identities remove the need for the user to provide ongoing management of a credential



Module 06: Manage application configuration data

Rethink application configuration data

Configuration information is stored in files

Changes can require downtime and administrative overhead

Challenging to manage changes to local configurations across multiple running instances of the application

Discussion: How will you secure this information?

Explore separation of concerns

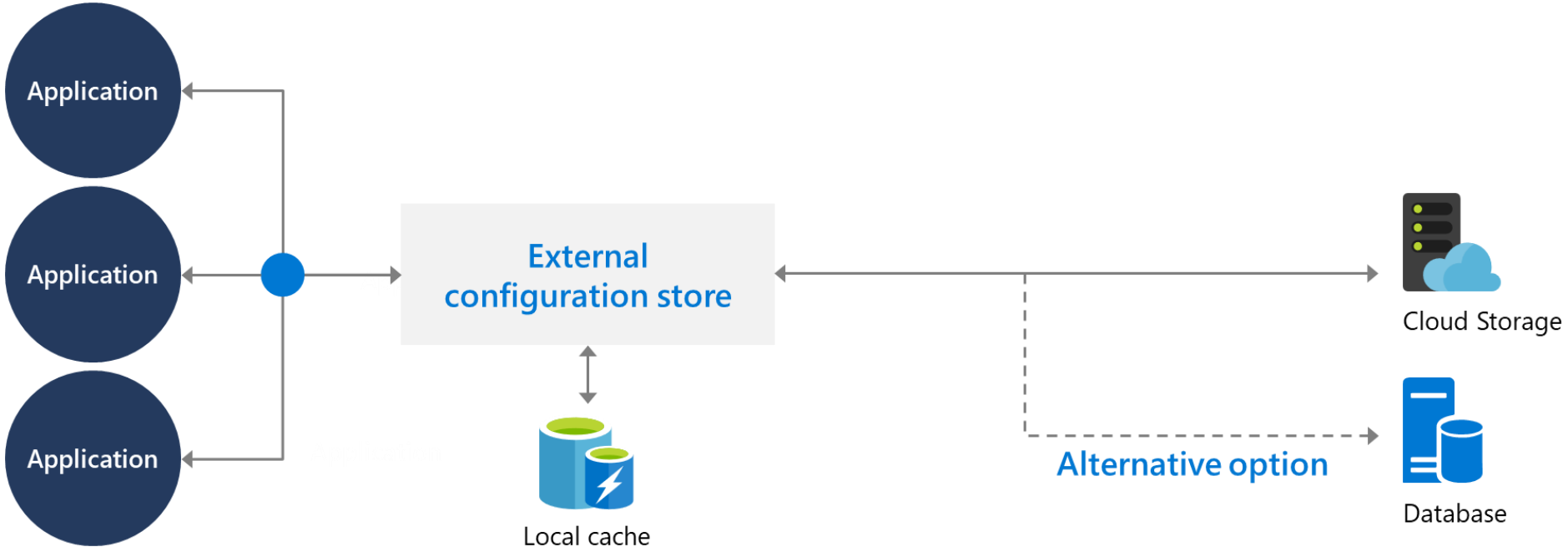
Configuration custodian: Responsible for generating and maintaining the life cycle of configuration values

Configuration consumer: Responsible for defining the schema (loose term) for the configuration and then consuming the configuration values in the application or library code

Configuration store: The underlying store that is leveraged to store the configuration

Secret store: A separate store for persisting secrets

Understand external configuration store patterns



Store the configuration information in external storage.

Provide an interface to quickly and efficiently read and update.

Introduction to Azure App Configuration

Useful for most applications but particularly applicable to:

Distributed applications (particularly microservice applications) Serverless applications

- 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

Examine Key-value pairs

AppName: Service1:ApiEndpoint

AppName: Service2:ApiEndpoint

AppName: Region1:DbEndpoint

AppName: Region2:DbEndpoint

Key = AppName:DbEndPoint & Label = Test

Key = AppName:DbEndPoint & Label = Staging

Key = AppName:DbEndPoint & Label = Production

- Service stores key-value pairs
- Keys and values are all Unicode strings
- Recommend using hierarchical naming for keys
- Labels can be used to store different values for a particular key
- Labels can provide a way of versioning keys

Examine App configuration feature management

Azure App Configuration can also act as a central repository for feature flags.

Feature flags have two parts:

- Name
- List of one or more filters

Filters are use cases for when an application feature should be enabled

Supports **appsettings.json** as a configuration source for feature flags

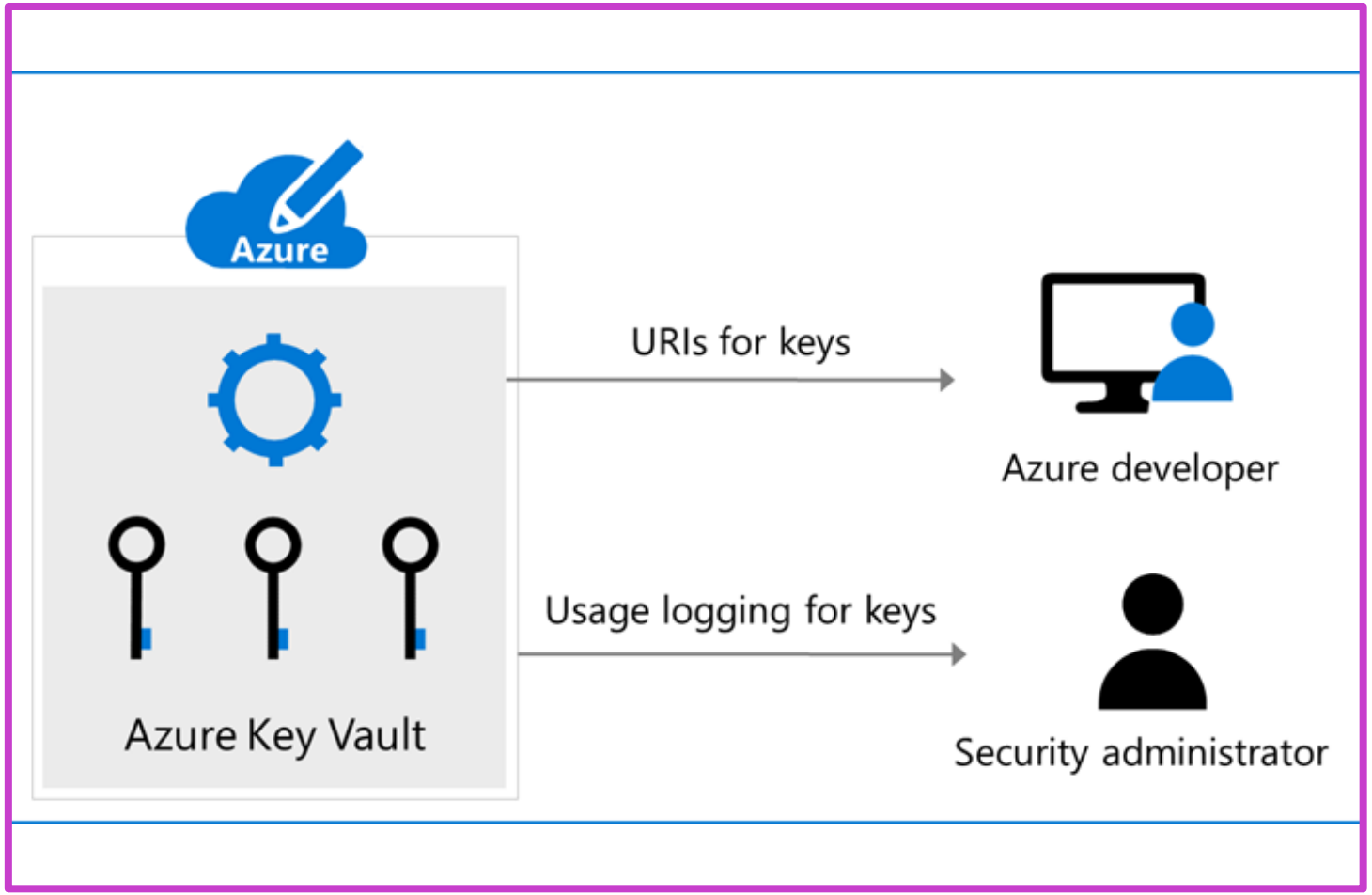
Integrate Azure Key Vault with Azure Pipelines

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

2. Azure Key Vault stores secrets, keys, and certificates.

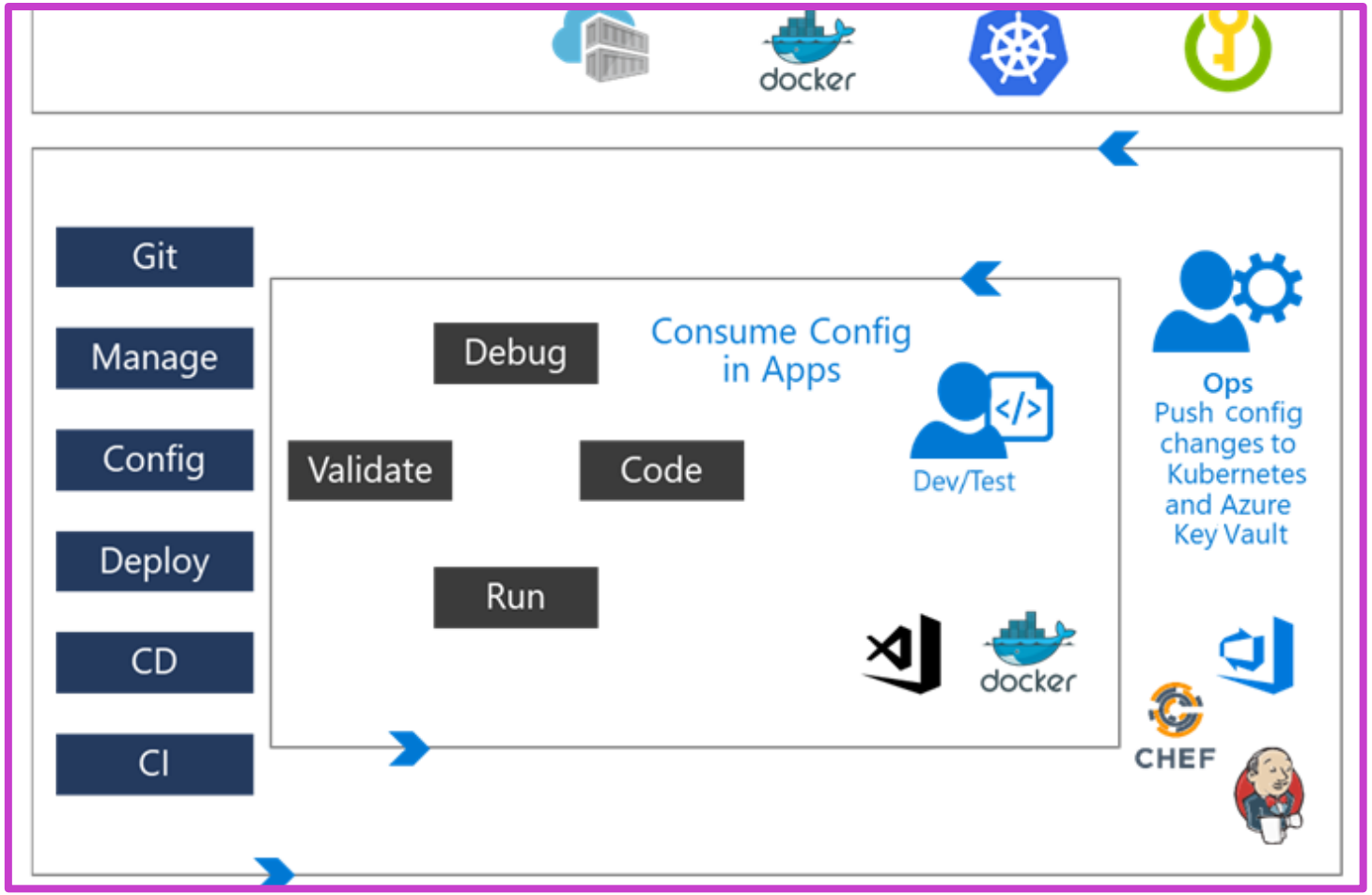
Manage secrets, tokens and certificates

- Centralize application secrets
- Securely store secrets and keys
- Monitor access and use
- Simplified administration of application secret
- Integrate with other Azure services



Examine DevOps inner and outer loop

- The inner loop is focused on the developer teams iterating over their solution development.
- Developer teams consume the configuration published by the outer loop.
- The Ops Engineer governs the configuration management and pushes the changes.



Labs

Lab: Configuring pipelines as code with YAML

Lab overview:

Azure DevOps also provides default templates for popular project types, as well as a YAML designer to simplify the process of defining build and release tasks.

Objectives:

- Configure CI/CD pipelines as code with YAML in Azure DevOps

Duration:

Lab: Setting up and running functional tests

Lab overview:

In this lab, you will learn how to execute Selenium test cases on a C# web application, as part of the Azure DevOps Release pipeline.

Objectives:

- Configure a self-hosted Azure DevOps agent
- Configure release pipeline
- Trigger build and release
- Run tests in Chrome and Firefox

Duration:

Lab: Integrating Azure Key Vault with Azure DevOps

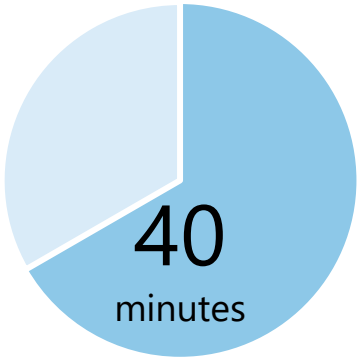
Lab overview:

In this lab, you will see how you can integrate Azure Key Vault with an Azure DevOps pipeline.

Objectives:

- Create a Microsoft Entra service principal
- Create an Azure key vault
- Track pull requests through the Azure DevOps pipeline

Duration:



Lab: Enable Dynamic Configuration and Feature Flags

Lab overview:

Azure App Configuration provides a service to centrally manage application settings and feature flags. Use App Configuration to store all the settings for your application and secure their accesses in one place.

Objectives:

- Enable dynamic configuration
- Manage feature flags

Duration:

Learning Path review and takeaways

What did you learn?

1. Describe deployment patterns
2. Implement blue-green deployment
3. Implement canary release
4. Implement progressive exposure deployment
5. Manage application config and secrets
6. Deploy to environment securely using a service connection
7. Integrate Azure Key Vault with a pipeline

Learning Path 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?
6. What is the Azure Key Vault and why would you use it?