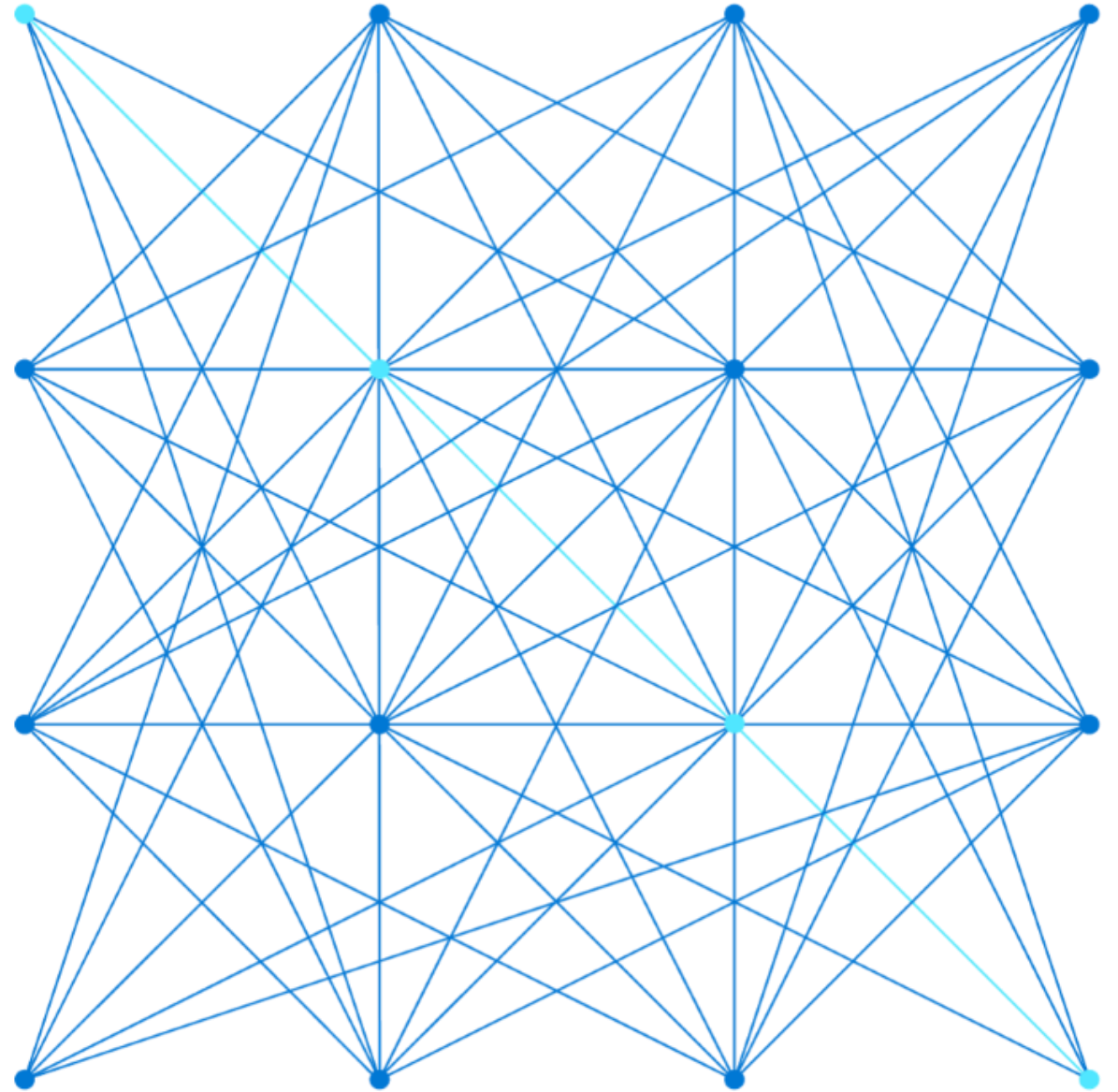


AZ-400.00

Module 7: Managing Application Configuration and Secrets



Lesson 01: Module overview



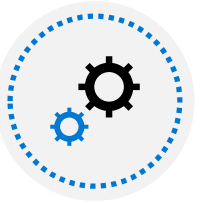
Module overview



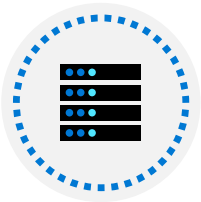
Lesson 1: Module overview



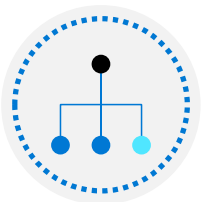
Lesson 2: Introduction to security



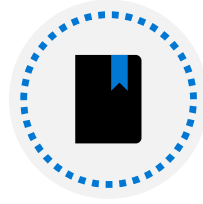
Lesson 3: Implement a secure development process



Lesson 4: Rethinking application configuration data



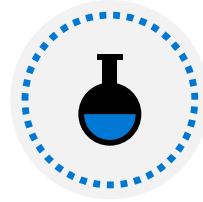
Lesson 5: Manage secrets, tokens, and certificates



Lesson 6: Integrating with identity management systems



Lesson 7: Implementing application configuration



Lesson 8: Lab



Lesson 9: Module review and takeaways

Learning objectives

After completing this module, students will be able to:



Manage application config and secrets



Integrate Azure Key Vault with a pipeline

Lesson 02: Introduction to security



(#mw) DevSecOps

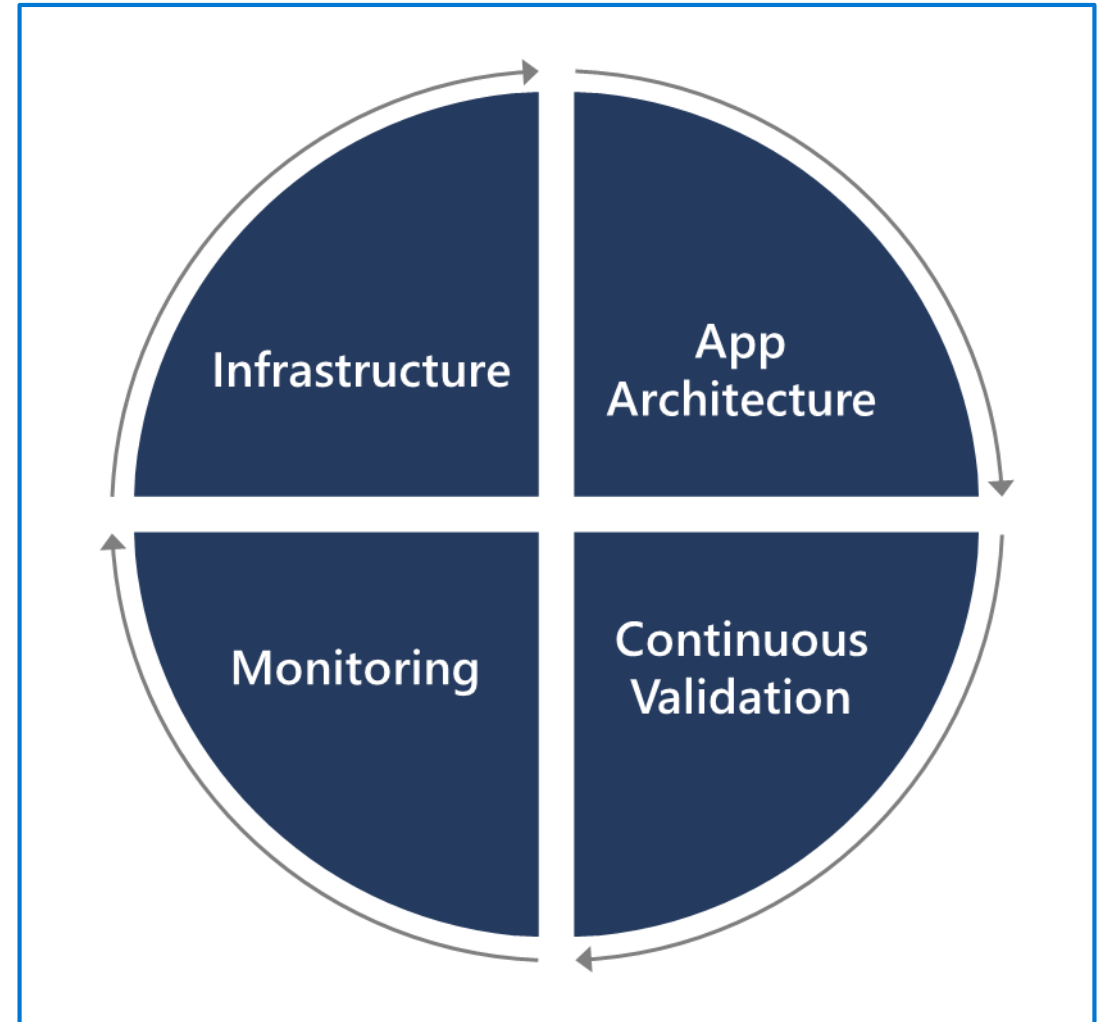
***DevSecOps** incorporates the security team and their capabilities into your DevOps*

Introduction to security

Security is everyone's responsibility.

Securing applications is a continuous process that encompasses the following:

- Secure infrastructure
- Designing an architecture with layered security
- Continuous security validation
- Monitoring for attacks



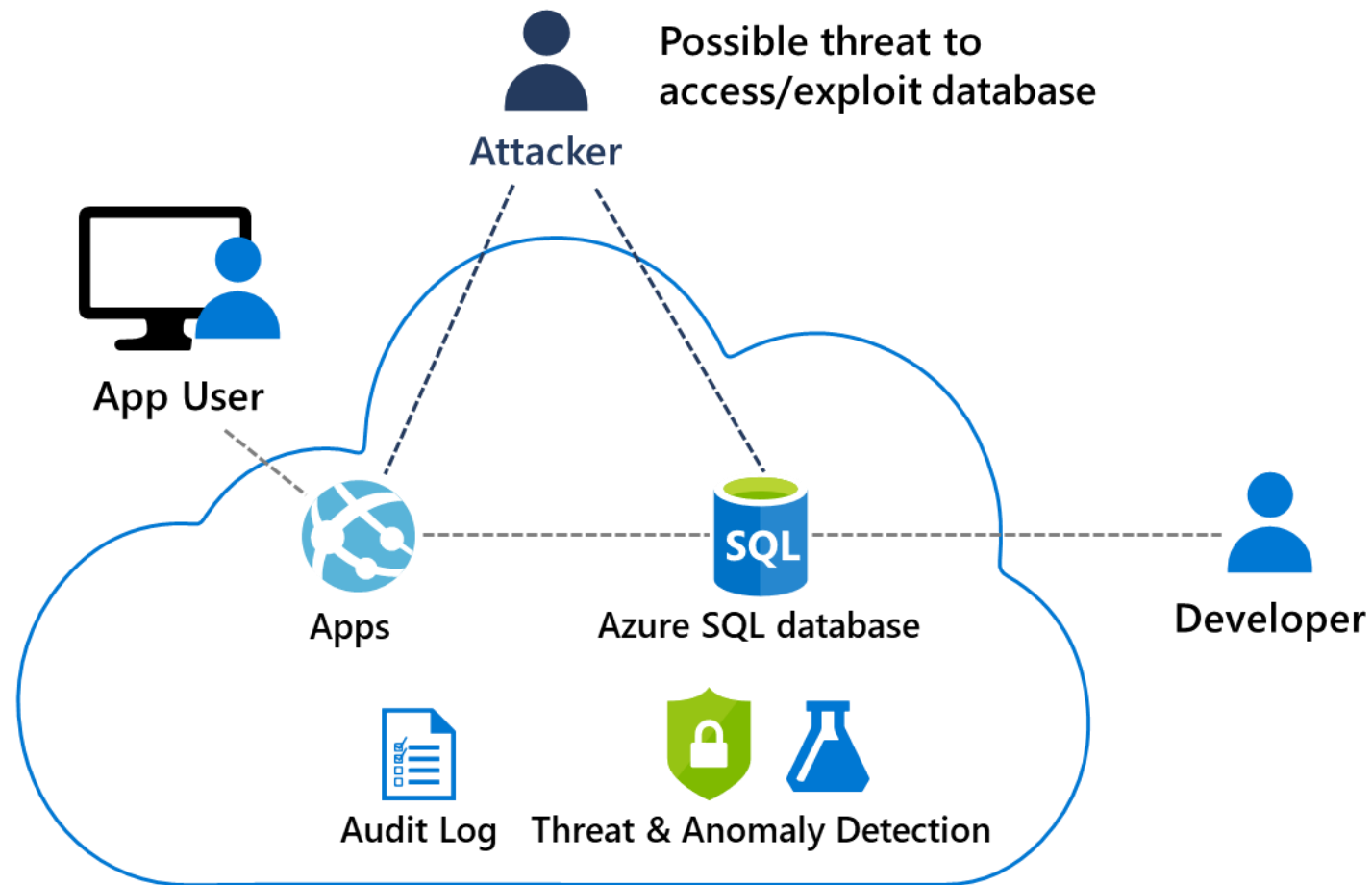
(#mw – Attackers have to find just one hole, defenders have to plug every hole)

SQL injection attack ([example](#))

A simple web app with SQL database

Heard this one before?

Hey John, a business user needs to see the customer data that's stored in the database... Why don't you expose the data on a web page in a table that the user can directly interact with?



(#mw – Ensure application has minimal permissions + use parameter placeholders)

Lesson 03: Implement a secure development process



Threat modeling

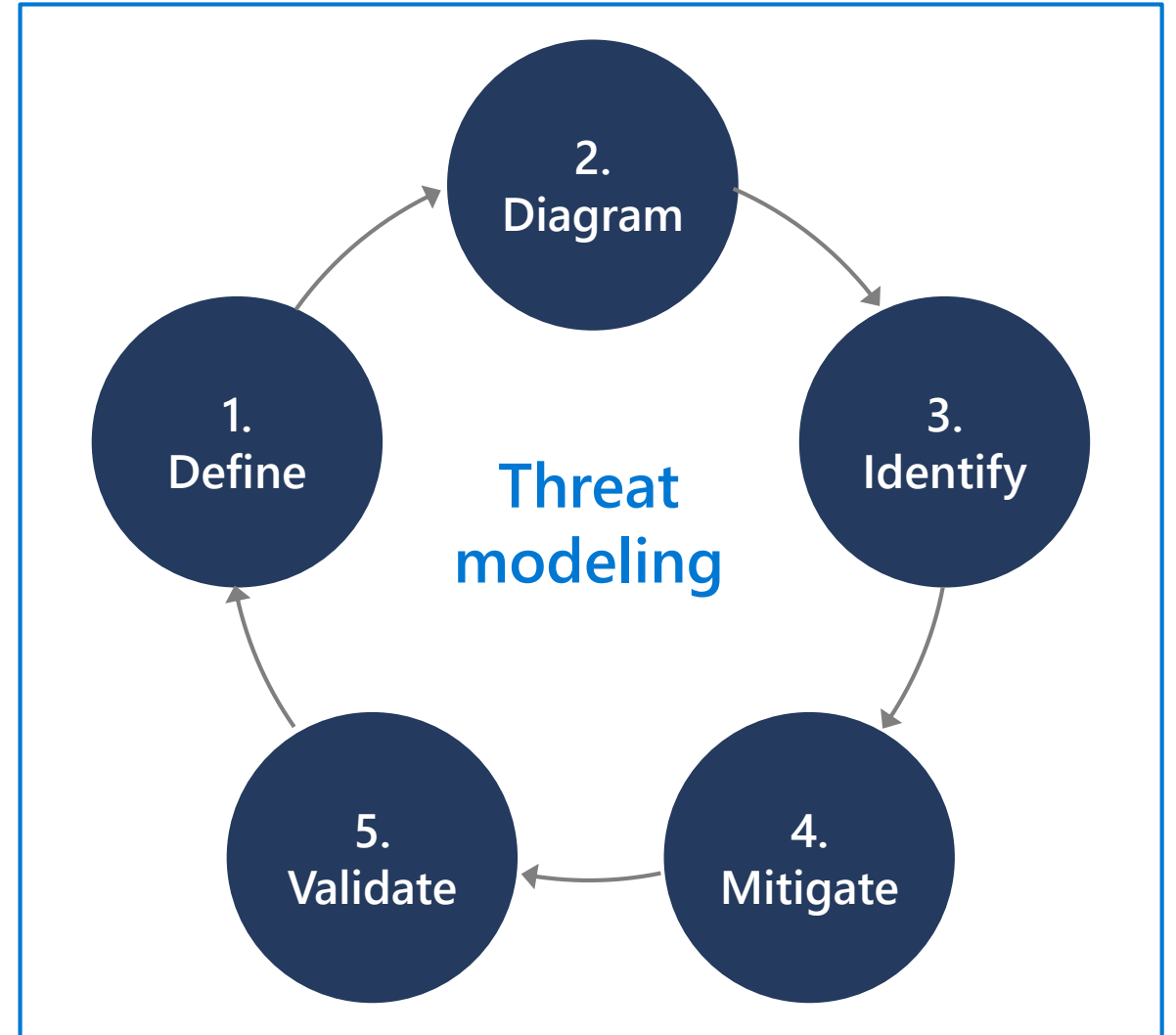
Define security requirements

Create an application diagram

Identify threats

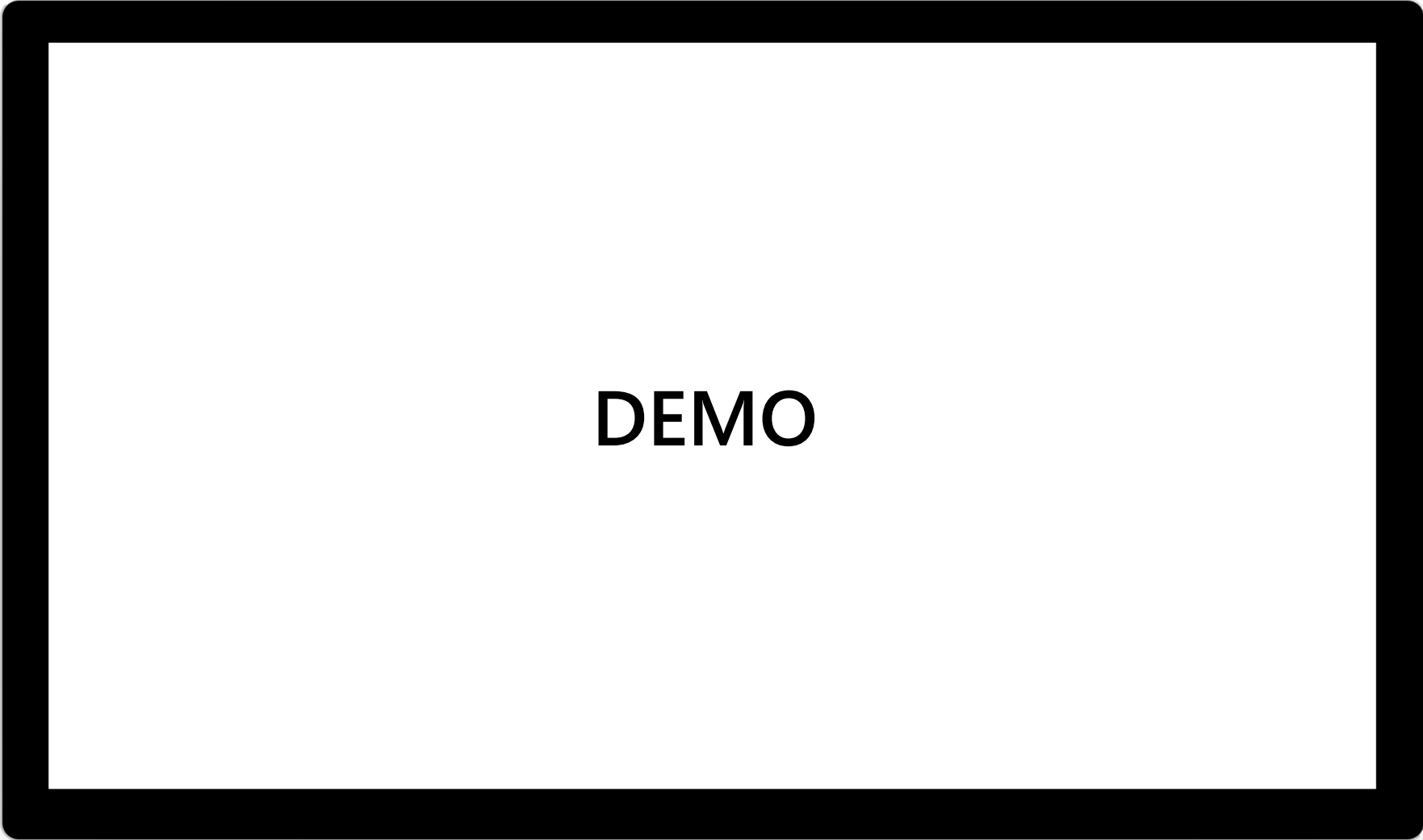
Mitigate threats

Validate that threats have been mitigated



(#mw – It's a cost/benefit thing, also $\text{Risk} = \text{probability} * \text{Impact}$)

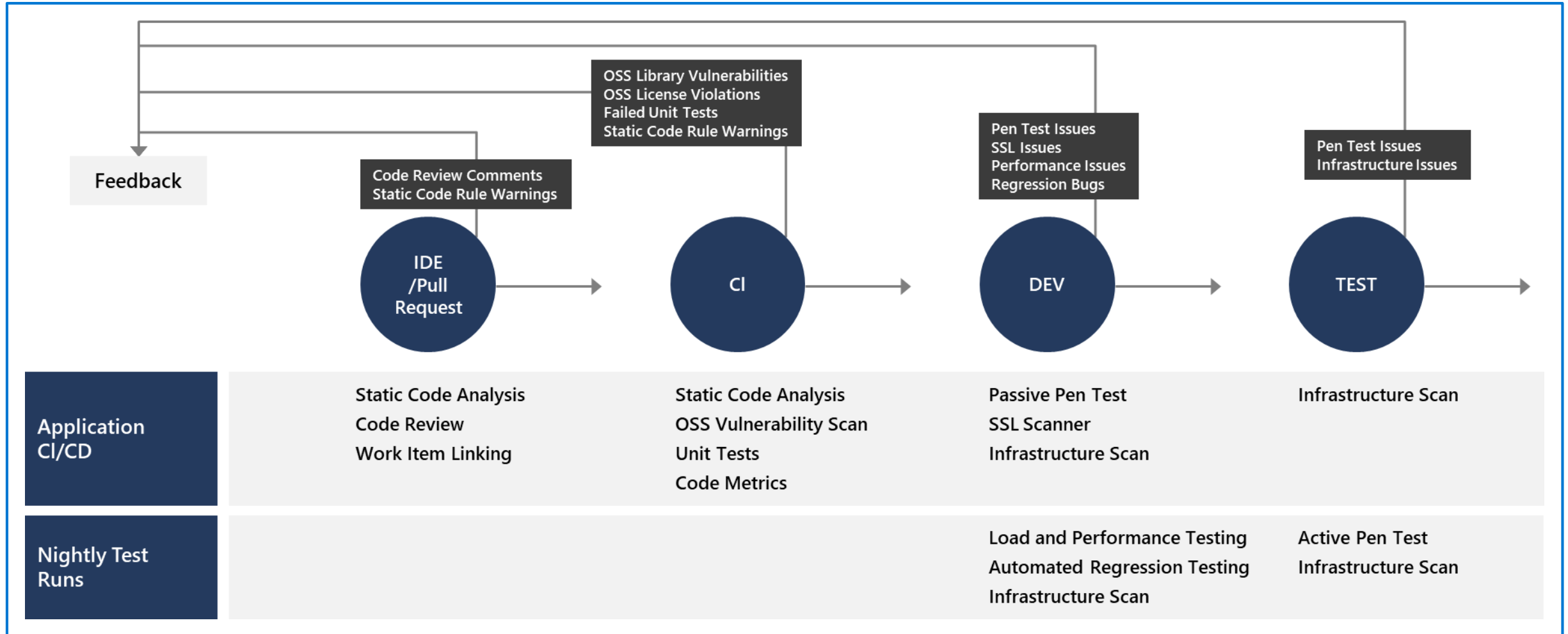
Threat modeling



DEMO

Key validation points

(#mw – [some links](#))



Continuous security validation should be added at each step from development through production

Continuous integration



The CI build should be executed as part of the pull request (PR-CI) process and once the merge is complete



Several tools are available:

SonarQube

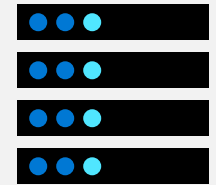
Visual Studio Code Analysis and the Roslyn Security Analyzers

Checkmarx – A Static Application Security Testing (SAST) tool

BinSkim – A binary static analysis tool that provides security and correctness results for Windows portable executables

And many more

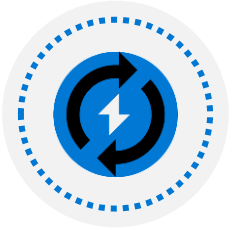
Lesson 04: Rethinking application configuration data



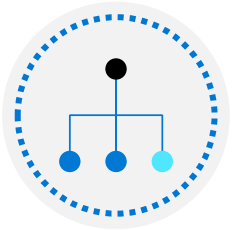
Rethinking application config data



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?

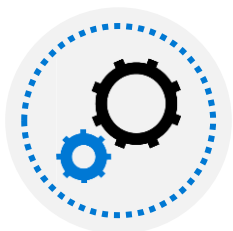
Separation of concerns



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

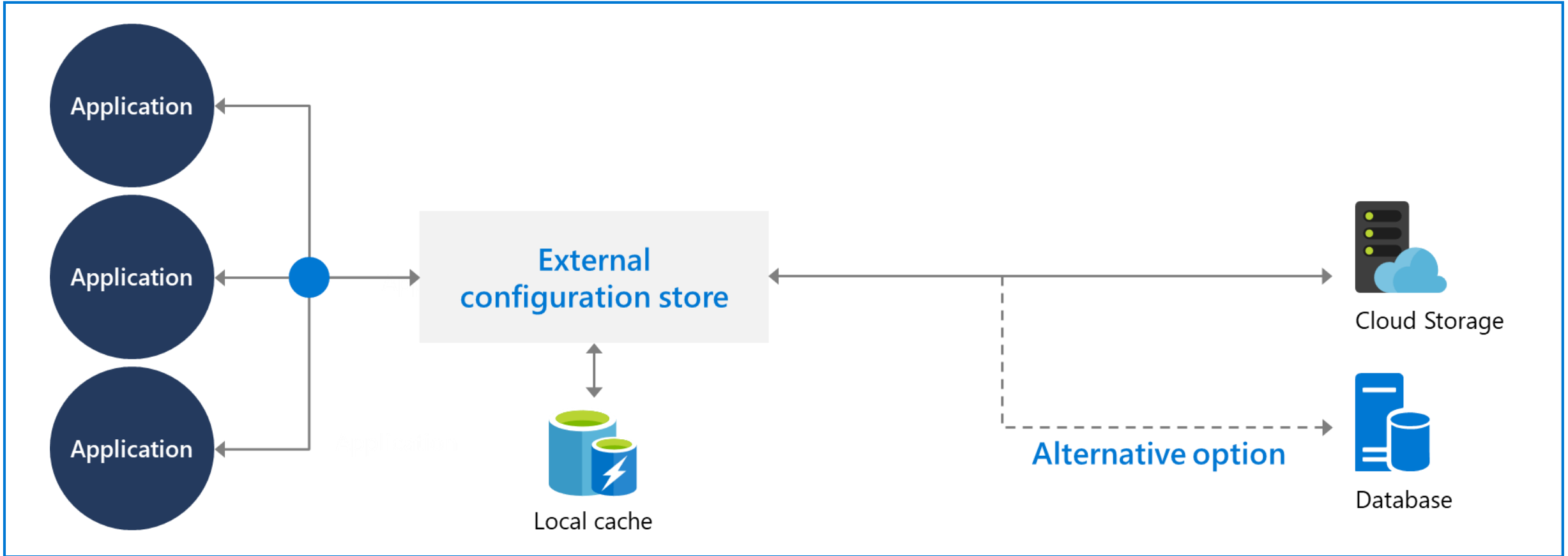


Configuration store: The underlying store that is leveraged to store the configuration



Secret store: A separate store for persisting secrets

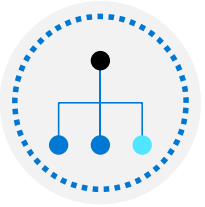
External configuration store patterns



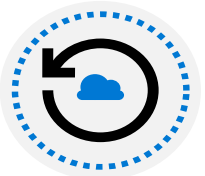
Store the configuration information in external storage.

Provide an interface to quickly and efficiently read and update.

Integrating Azure Key Vault with Azure Pipeline

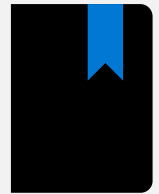


Azure Key Vault allows you to manage your organization's secrets and certificates in a centralized repository.



Azure Key Vault stores secrets, keys, and certificates.

Lesson 05: Manage secrets, tokens, and certificates



Manage secrets, tokens and certificates

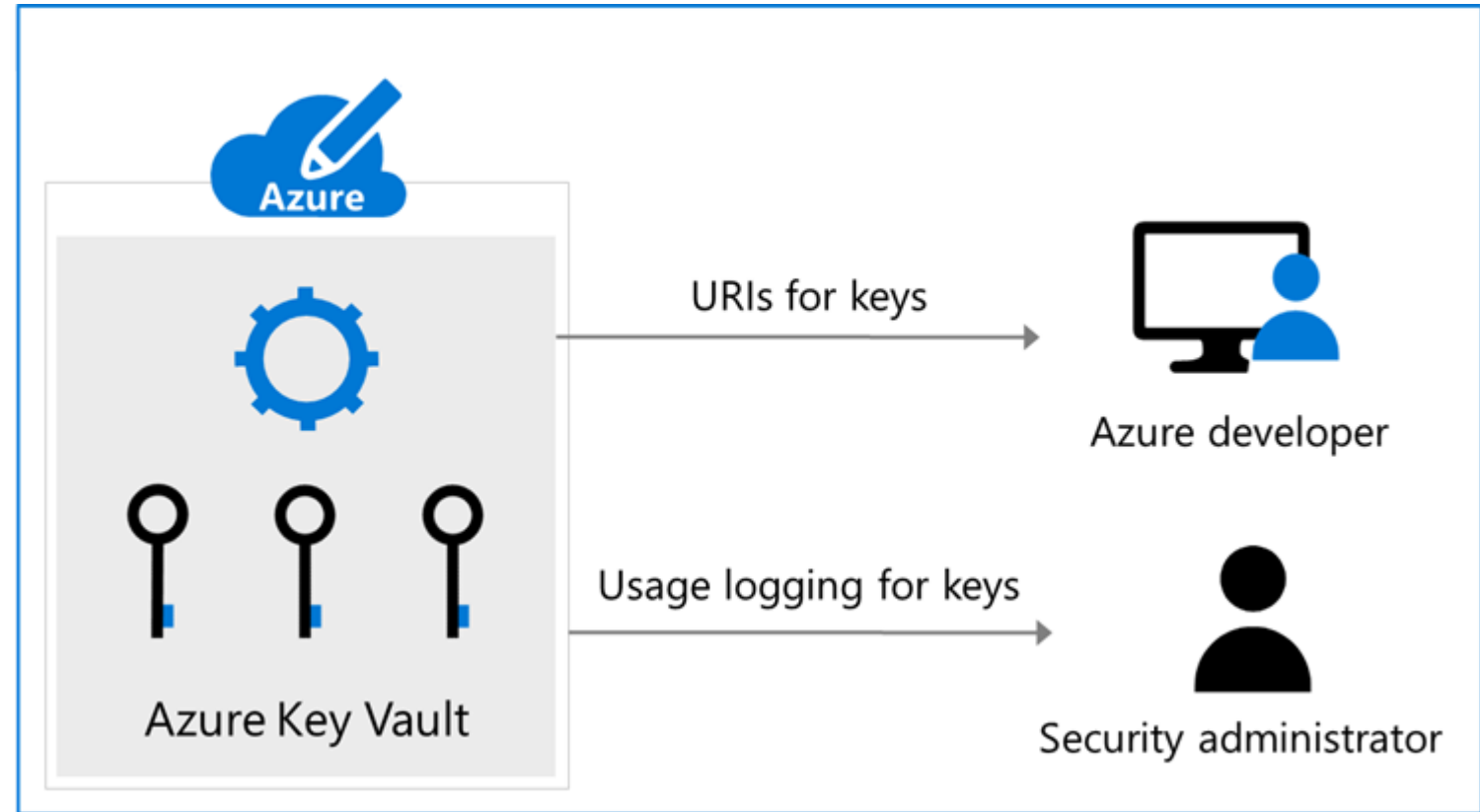
Centralize application secrets

Securely store secrets and keys

Monitor access and use

Simplified administration of application secret

Integrate with other Azure services

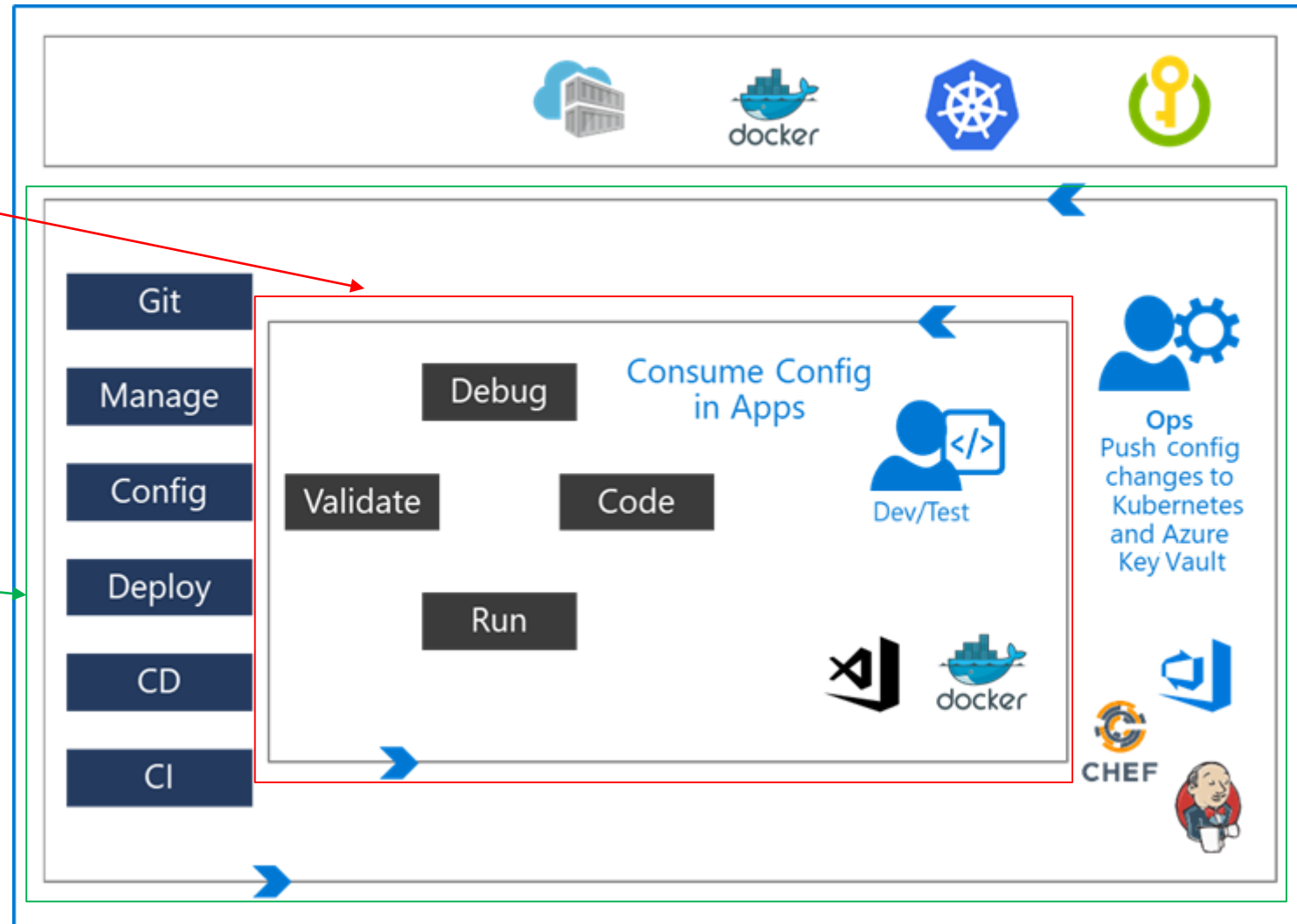


DevOps inner and outer loop

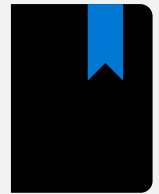
The **inner loop** is focused on the developer teams iterating over their solution development.

Developer teams consume the configuration published by the **outer loop**.

The Ops Engineer governs the configuration management and pushes the changes.



Lesson 06: Integrating with identity management systems



Integrating **GitHub** with single sign-on (SSO)

Connect identity provider to GitHub at organization level

Both SAML and [SCIM](#) support available

Provider	Available Support
Active Directory Federation Services (ADFS)	SAML
Azure Active Directory (Azure AD)	SAML and SCIM
Okta	SAML and SCIM
OneLogin	SAML and SCIM
PingOne	SAML
Shibboleth	SAML

(#mw – why is this here? AzureDevOps needs access to resources under the control of AAD, e.g. KeyVault)

Service principals

- Register an AD application
- Create a client secret
- Grant permissions to the identity

To connect, application presents: -

- TenantID
- Application ID
- Client Secret

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	02744c0e-5caf-11e6-80d6-56b636570000	11/6/2019	 Current

 New client secret

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

Supported account types

Who can use this application or access this API?

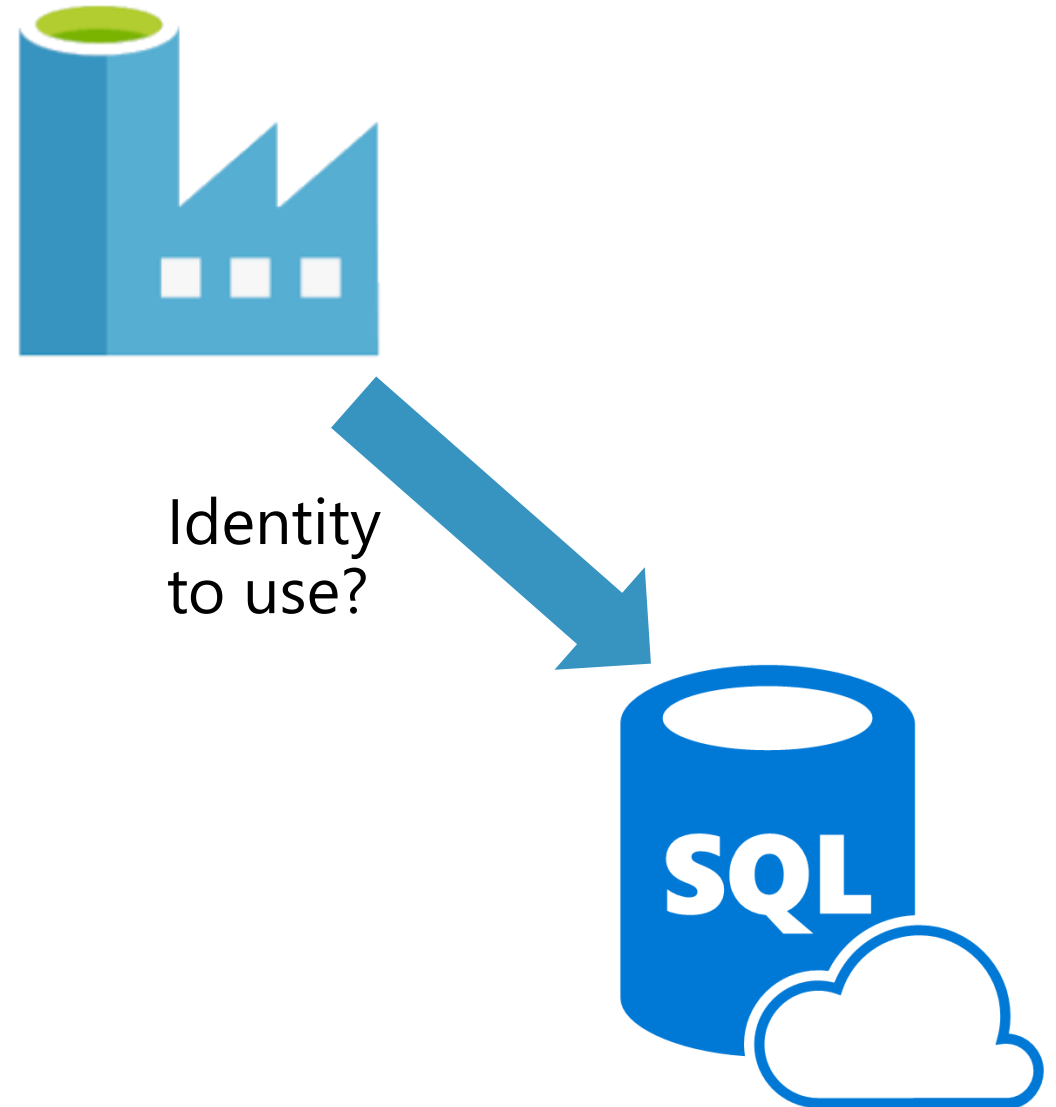
- ☒ Accounts in this organizational directory only (Pty Ltd only - Single tenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

Managed service identities

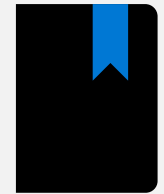
Two types of managed identity:

- 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.



Lesson 07: Implementing application configuration



Azure App configuration service



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

Key-value pairs

- 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

```
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
```

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

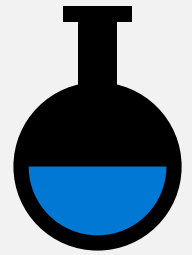
Filters are use cases for when an application feature should be enabled

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

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

NOTE: Feature flags are discussed later in the course.

Lesson 08: Lab



Integrating Azure Key Vault with Azure DevOps

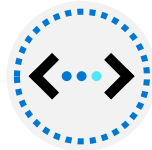


Lab overview:

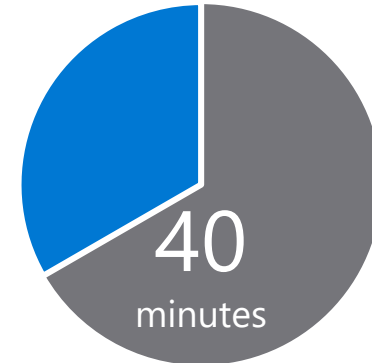
In this lab, you will see how you can integrate Azure Key Vault with an Azure DevOps pipeline.

Objectives:

- Create an Azure Active Directory (Azure AD) service principal
- Create an Azure key vault
- Track pull requests through the Azure DevOps pipeline



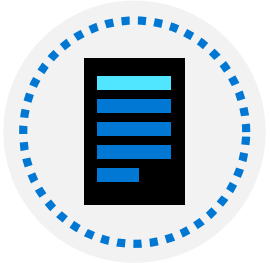
Duration:



Lesson 09: Module review questions



What did you learn?



Manage application config and secrets



Integrate Azure Key Vault with a pipeline

Module review questions

1

What are the five stages of threat modeling?

2

What is the Azure Key Vault and why would you use it?