Microsoft Azure

# Azure Study Group
# AZ-300 - Microsoft Azure Architect Technologies

Jeff Wagner

Partner Technology Strategist

Microsoft Azure

# Implement Authentication and Secure Data (5-10%)

# Agenda

1. Series Agenda
2. Speaker Introduction
3. Feedback Loop
4. Objective Review
5. Open Mic

# Series Agenda

| | |
|---|---|
| 1 | Deploy and Configure Infrastructure (25-30%) |
| 2 | Implement Workloads and Security (20-25%) |
| 3 | Create and Deploy Apps (5-10%) |
| 4 | Implement Authentication and Secure Data (5-10%) |
| 5 | Develop for the Cloud (20-25%) |

https://aka.ms/azurecsg

# Series Agenda

| | |
|---|---|
| 1 | Deploy and Configure Infrastructure (25-30%) |
| 2 | Implement Workloads and Security (20-25%) |
| 3 | Create and Deploy Apps (5-10%) |
| 4 | Implement Authentication and Secure Data (5-10%) |
| 5 | Develop for the Cloud (20-25%) |

https://aka.ms/azurecsg

# Speaker Introduction - Jeff Wagner

- Partner Technology Strategist based in Atlanta
- 21+ years with Microsoft, more in the industry
- Been working with Microsoft Azure when we weren't sure if the was called Windows A*zure* or Windows *Azure*
- Constant learner - *Ancora Imparo*





Microsoft Azure

# Feedback Loop

# Objectives

## Implement authentication

*May include but not limited to:* Implement authentication by using certificates, forms-based authentication, tokens, or Windows-integrated authentication; implement multi-factor authentication by using Azure AD; implement OAuth2 authentication; implement Managed Service Identity (MSI) Service Principal authentication

## Implement secure data solutions

*May include but not limited to:* Encrypt and decrypt data at rest and in transit; encrypt data with Always Encrypted; implement Azure Confidential Compute and SSL/TLS communications; create, read, update, and delete keys, secrets, and certificates by using the KeyVault API

# Implement Authentication

# Implementing Authentication in Applications

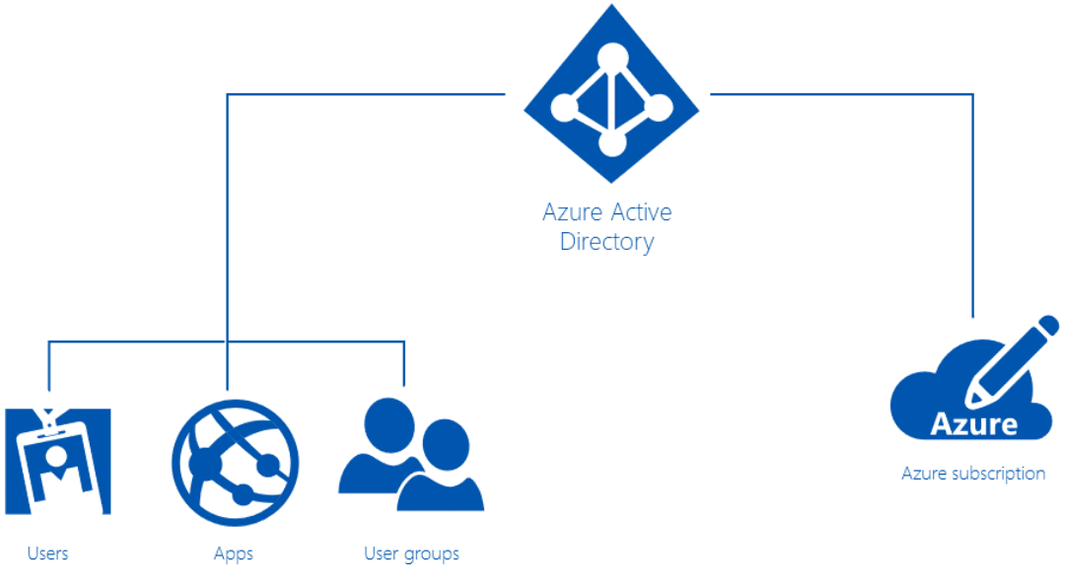
# Certificate-based authentication

- Establishes identity by using a digital certificate:
  - Front-end applications interact with back-end services
  - Securing connections in hybrid scenarios over Transport Layer Security (TLS)
  - API Management protecting access to the back-end service

# Azure Active Directory (Azure AD)

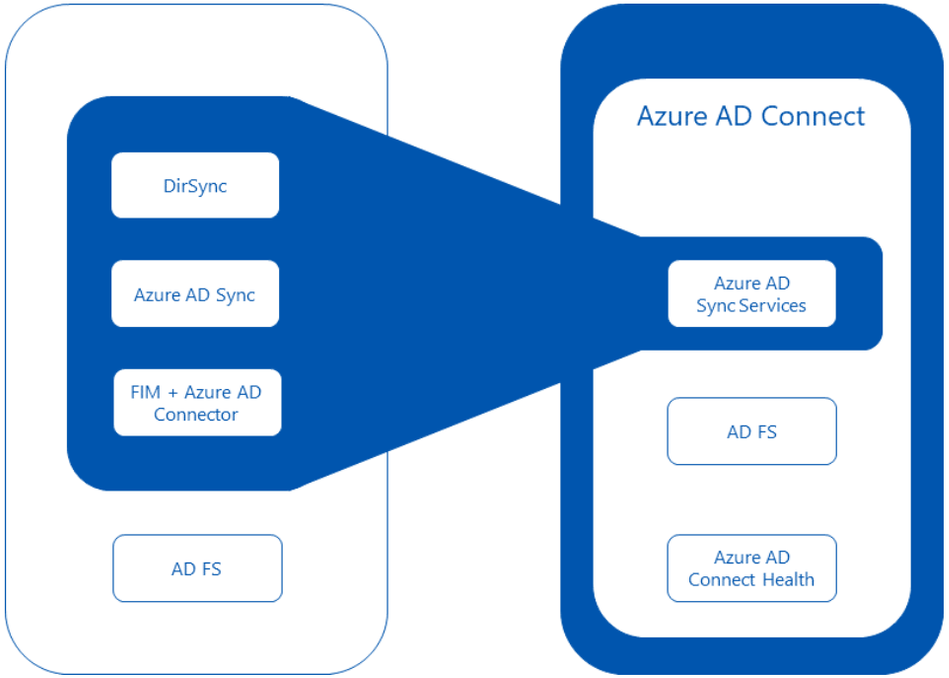
- Microsoft's cloud-based directory and identity management service:
  - Combines directory services, identity governance, and application access management

- AD DS vs. Azure AD:
  - Similarities:
    - Directory stores of user, group, and application objects.
    - Identity and authentication providers.
  - Differences
    - Single tenant vs. multi-tenant
    - A server role in Windows Server vs. cloud service
    - X.500-based hierarchical structure vs. flat structure
    - LDAP lookups vs. Graph API
    - Kerberos and NTLM vs. SAML, WS-Federation, and OpenID Connect/OAuth
    - Built-in GPO-based management capabilities vs. integration with management products such as Intune
    - Domains and forests trusts vs. federation



# Azure AD Connect

- Integrates AD DS with Azure AD:
  - Implements a common identity for your users across Azure, Office 365, and SaaS apps
- Provides support for:
  - **Sync Services:** synchronize AD DS objects, such as users and groups.
  - **Health Monitoring**: offers centralized monitoring
  - **Federation:** simplifies configuration of AD FS



# Legacy authentication methods

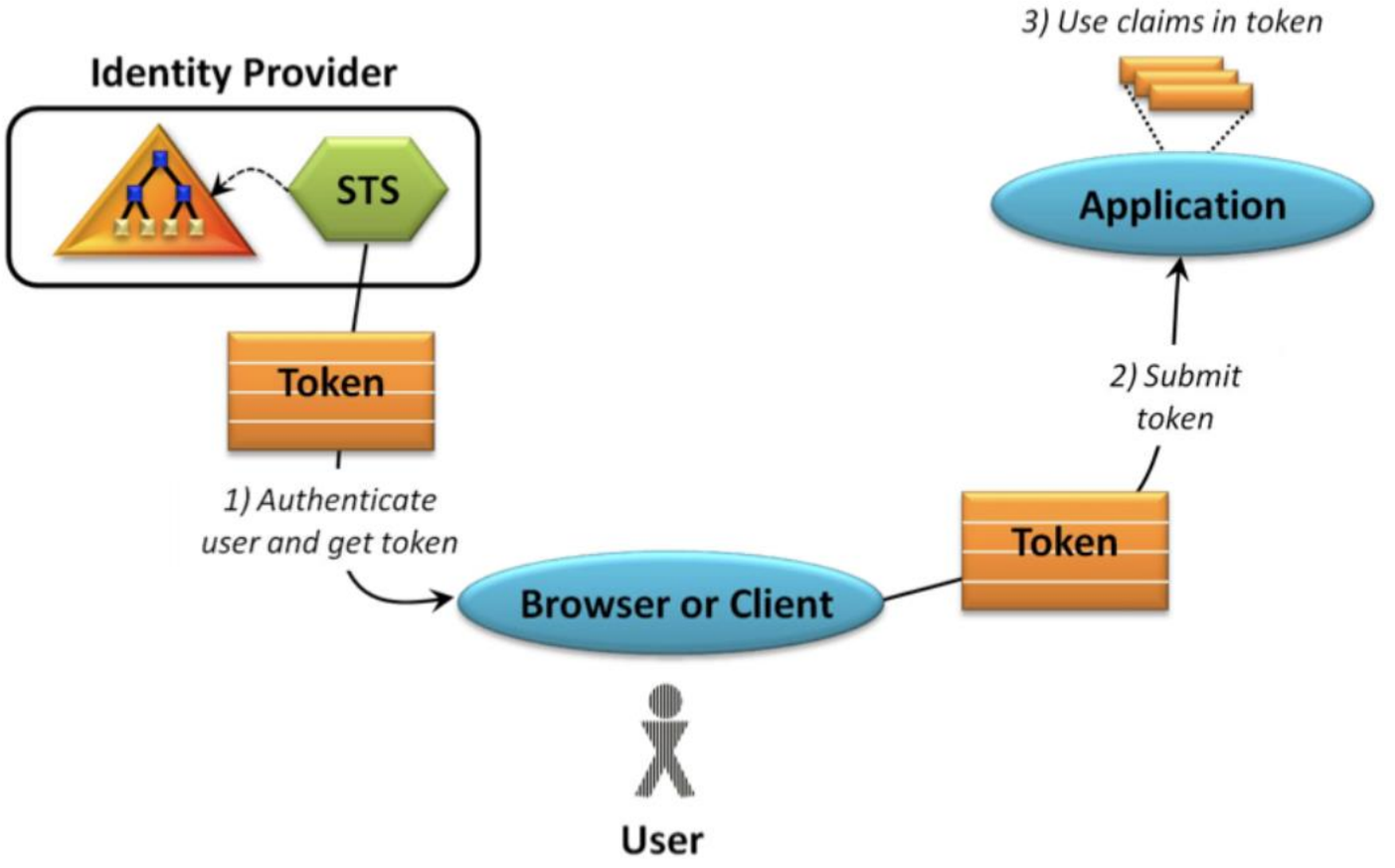
- Forms-based authentication:
  - Requires use of a browser client
  - Requires extra measures to prevent cross-site request forgery
  - Sends user credentials in plaintext

- Windows-based authentication:
  - Not suitable for internet applications
  - Does not support BYOD scenarios
  - Relies on Kerberos or NTLM
  - Requires domain-joined computers

Example:
If you are moving an ASP.NET forms-based auth application to Azure, change connection string pointing to your on-premise SQL Server database used to store forms auth data to Azure SQL Server.

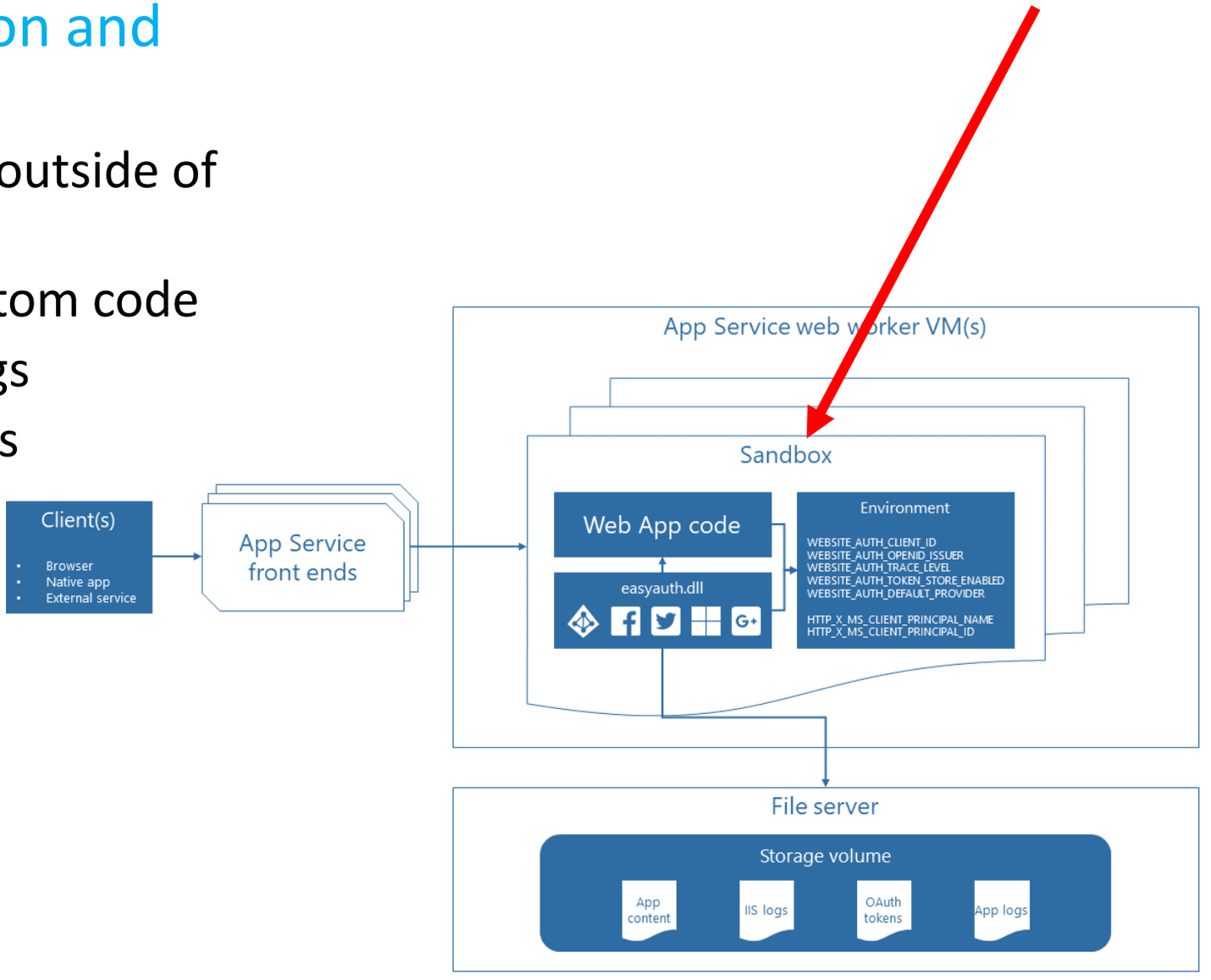
# Token-based authentication

- Claims-based authentication in .NET:
  - ASP.NET Identity provides a unified identity platform for ASP.NET applications
    - Can be used with web, phone, store and hybrid applications
    - Ideal for token-based auth because:
      - Provider model for logins
      - Supports claims-based authentication



# Token-based authentication

- Azure App Service authentication and authorization:
  - Runs in the worker sandbox and outside of the web app code
  - Available with minimal or no custom code
  - Configurable by using app settings
  - Injects claim into request headers
  - Provides built-in token store



# Implement Multi-factor Authentication

# Multi-factor authentication

- Enforces two or more factors when authenticating users:
  - **Knowledge:** something that only the user knows (security questions, password, or PIN).
  - **Possession:** something that only the user has (corporate badge, mobile device, or security token).
  - **Inherence:** something that only the user is (fingerprint, face, voice, or iris).

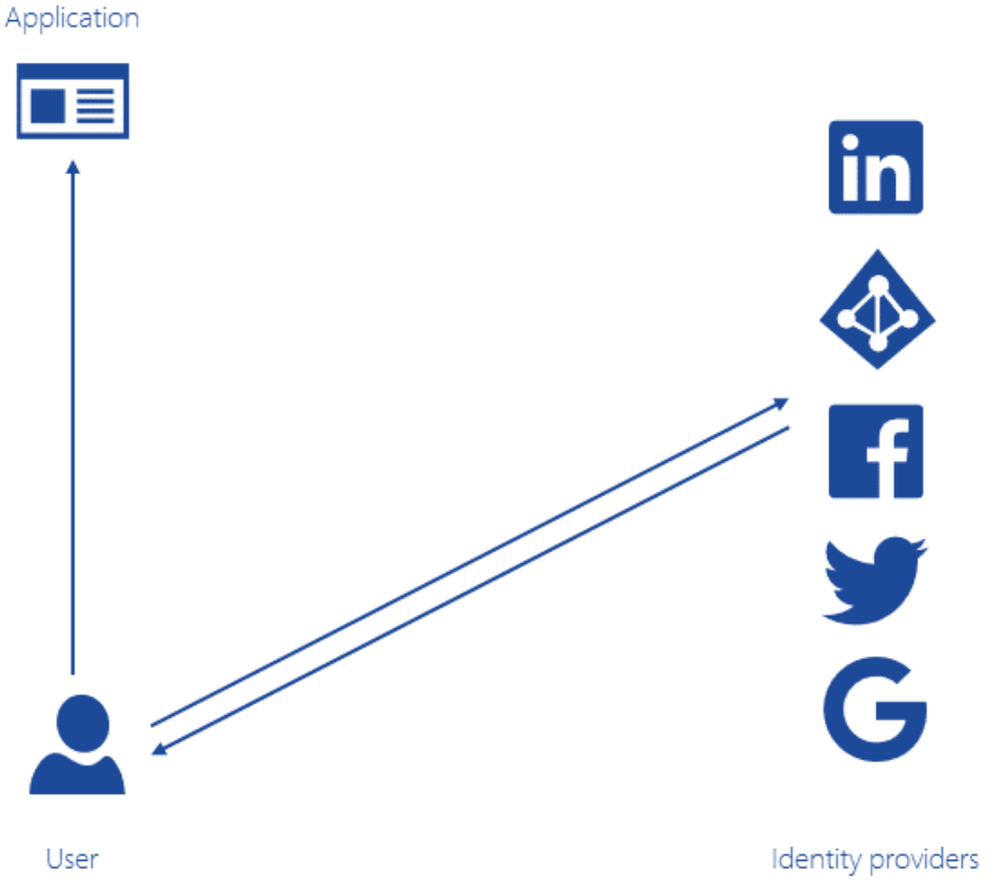
# Multi-factor authentication with Azure AD

- Can be enabled by:
  - Enabling users for MFA to trigger two-step verification each time they sign in:
  - Setting up a conditional access policy requiring two-step verification
- Offers several authentication methods:
  - Call to phone
  - Text message to phone
  - Notification through mobile app
  - Verification code from mobile app (Microsoft Authenticator app)
- Can be customized by using the Multi-Factor Authentication SDK:
  - Allows building MFA into the sign-in or transaction processes of applications
  - Supports C#, Visual Basic (.NET), Java, Perl, PHP, and Ruby

# Claims-based Authorization

# Claims

- A name/value pair representing an identity and its properties
  - Generated by an identity provider:
    - Azure AD, Facebook, Google, LinkedIn, Twitter, etc.
  - Serves as the basic for authorization:
    - Handled by a resource provider
    - Determines access to resources



# Claims-based authorization

- To implement claims-based auth in ASP.NET:
  - Build and register the policy:
  - Apply the policy:

Probably good to know the formatting of this type of claim

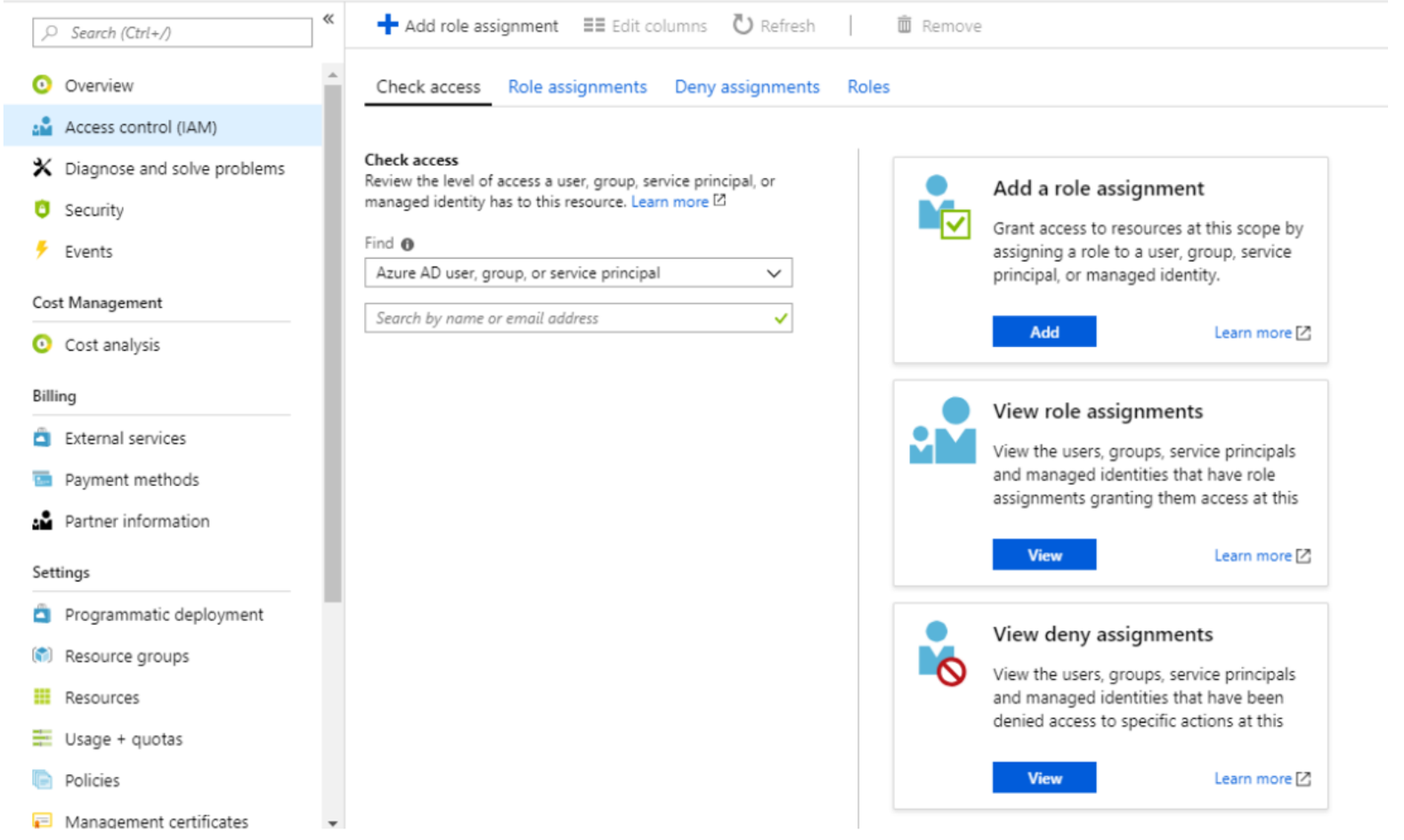
```
public void ConfigureServices(IServiceCollection services)
{
        services.AddMvc();
        services.AddAuthorization(options =>
        {
                options.AddPolicy("EmployeeOnly", policy => policy.RequireClaim("EmployeeNumber"));
        });
}
```

```
[Authorize(Policy = "EmployeeOnly")]
public IActionResult VacationBalance()
{
        return View();
}
```

# Role-based access control (RBAC) authorization

# Role-based authorization

- Manages and enforces permissions based on user roles:
  - An identity may belong to one or more roles



# Role-based authorization

- In ASP.NET-based implementation:

```
[Authorize(Roles = "Administrator,
PowerUser")] public class
ControlPanelController : Controller
{
        public ActionResult SetTime()
        {
        } [Authorize(Roles =
"Administrator")]      public
ActionResult ShutDown()
        {
        }
}
```

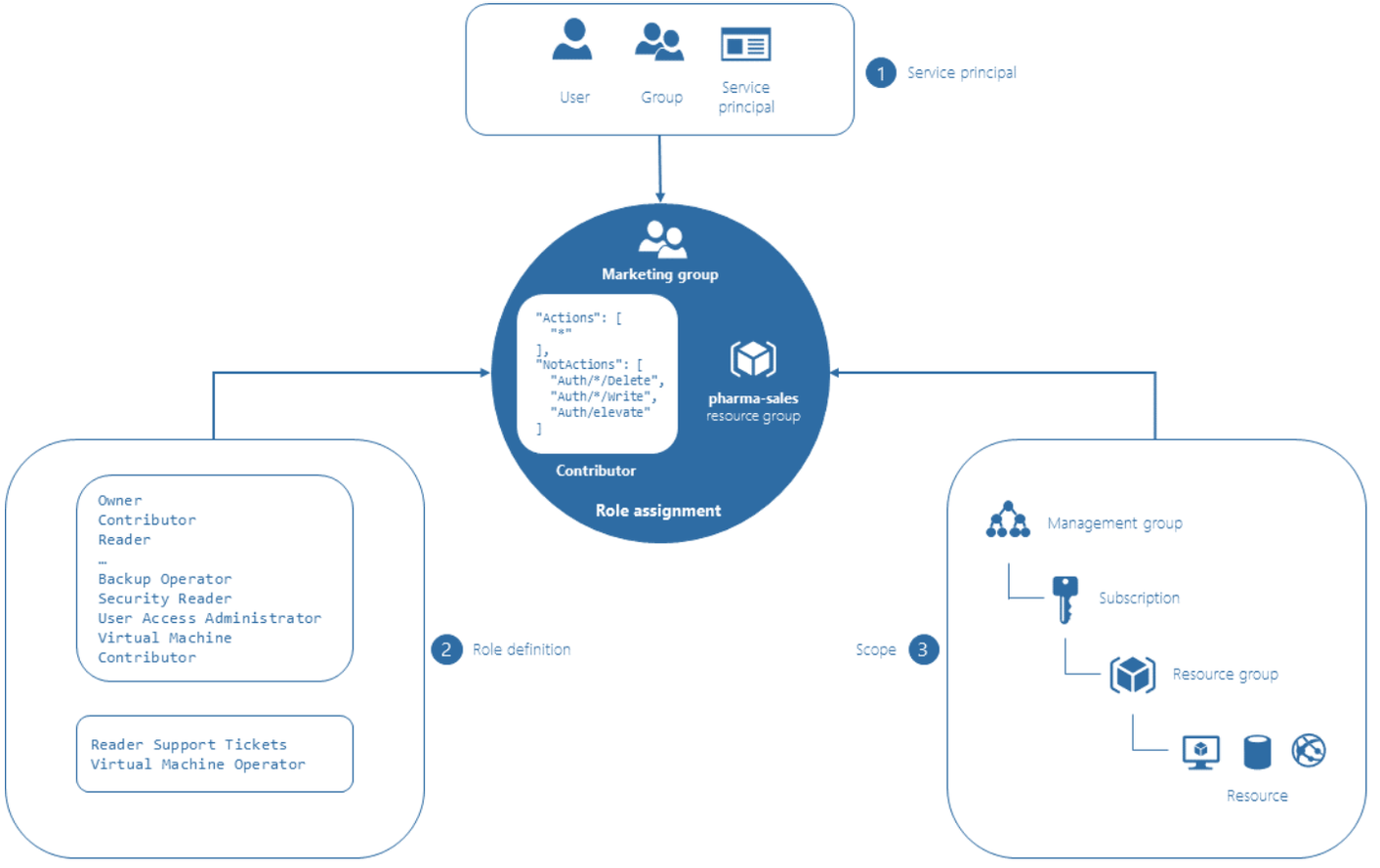
Members of either the **Administrator** role or the **PowerUser** role can access the controller and the `SetTime` action, but only members of the **Administrator** role can access the `ShutDown` action.

```
public void ConfigureServices(IServiceCollection
services)
{
        services.AddMvc();
        services.AddAuthorization(options =>
        {
        options.AddPolicy("RequireAdministratorR
        ole",   policy =>
        policy.RequireRole("Administrator")); });
}
```

Role requirements can be expressed using the `Policy` syntax, where a developer registers a policy at startup as part of the authorization service configuration. This normally occurs in `ConfigureServices()` in your `Startup.cs` file:

# Role-based access control (RBAC)

- Provides fine-grained access management of resources in Azure
- Facilitates segregation of duties
- Role assignments bind a role definition to a security principal, at a specific scope (or boundary) for the purpose of granting access.
- Supports four scope types:
  - A management group
  - A subscription
  - A resource group
  - A resource
- Includes built-in roles

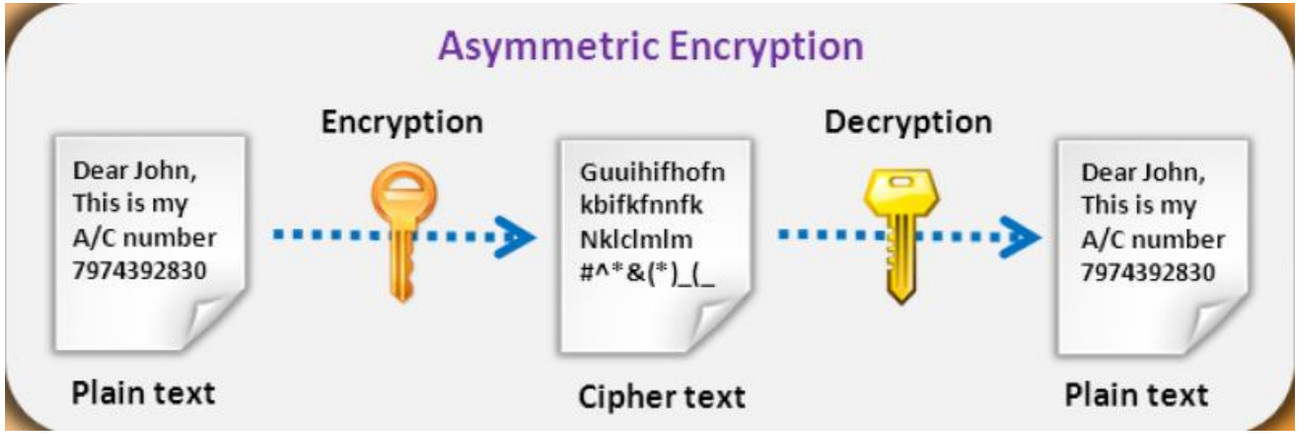


# Secure Data Solutions

# Encrypt and decrypt data at rest

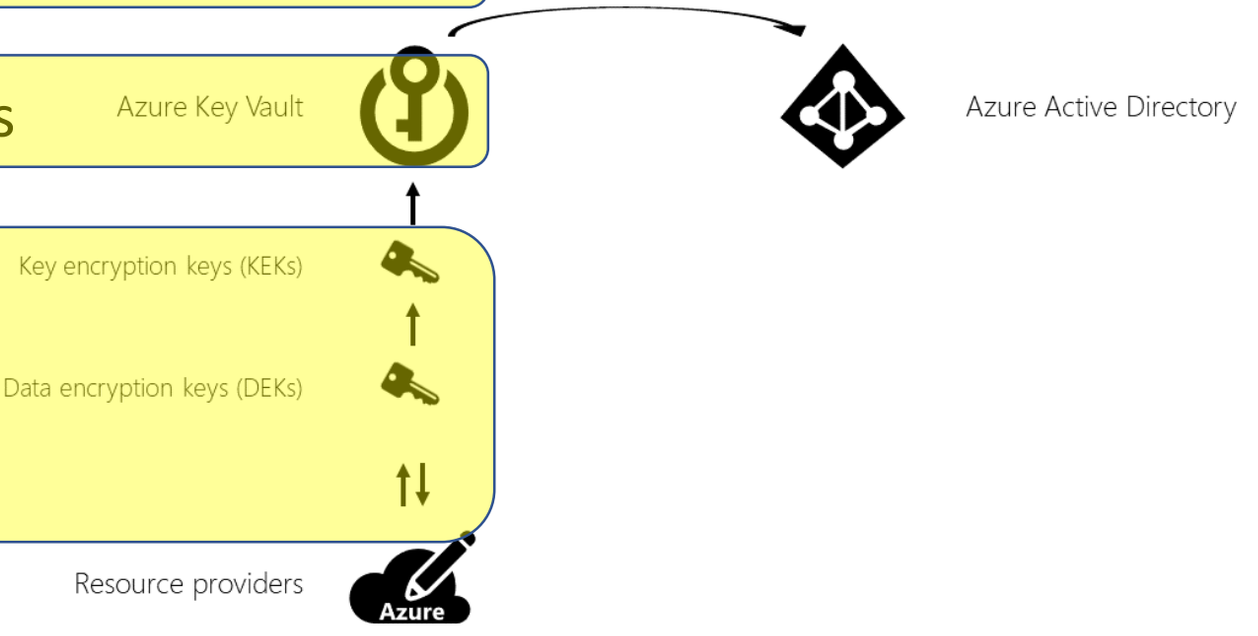
# Encryption

- The process of translating plain text into ciphertext:
- Uses an encryption algorithm and one or two keys:
  - In symmetric encryption:
    - The same key is used for encryption and decryption
    - Intended for encryption of large amounts of data
  - In asymmetric encryption:
    - Different key for encryption (public) and decryption (private)
    - Intended for small amounts of data or for encryption of a symmetric key



# Encryption at rest

- Encryption of data when it is persisted
  - Protects against attempts to obtain access to data when you have physical access to the hardware
  - Used in scenarios with compliance and security requirements
- Encryption at rest in Azure:
  - Dynamically encrypts/decrypts during writes/reads
  - Uses symmetric encryption keys
  - Uses different keys across partitions
  - Stores keys in a secure location
  - Includes:
    - Azure Storage encryption
    - Azure SQL Database encryption
    - Azure Cosmos DB encryption



# Encrypt Data with Always Encrypted

# Encrypt data with Transparent Data Encryption (TDE)

- Primary TDE characteristics:
  - Encryption at rest for data and log files of:
    - SQL Server
    - Azure SQL Database
    - Azure SQL Data Warehouse
  - Real-time I/O encryption and decryption on the page level
  - The use of a database encryption key (DEK):
    - The key is stored in the database boot record for availability during recovery.
    - The key is implemented in one of two ways:
      - Symmetric key secured by using a certificate stored in the master database of the server
      - Asymmetric key protected by an Extensible Key Management (EKM) module.

Watch for key words in questions that would indicate a requirement for TDE

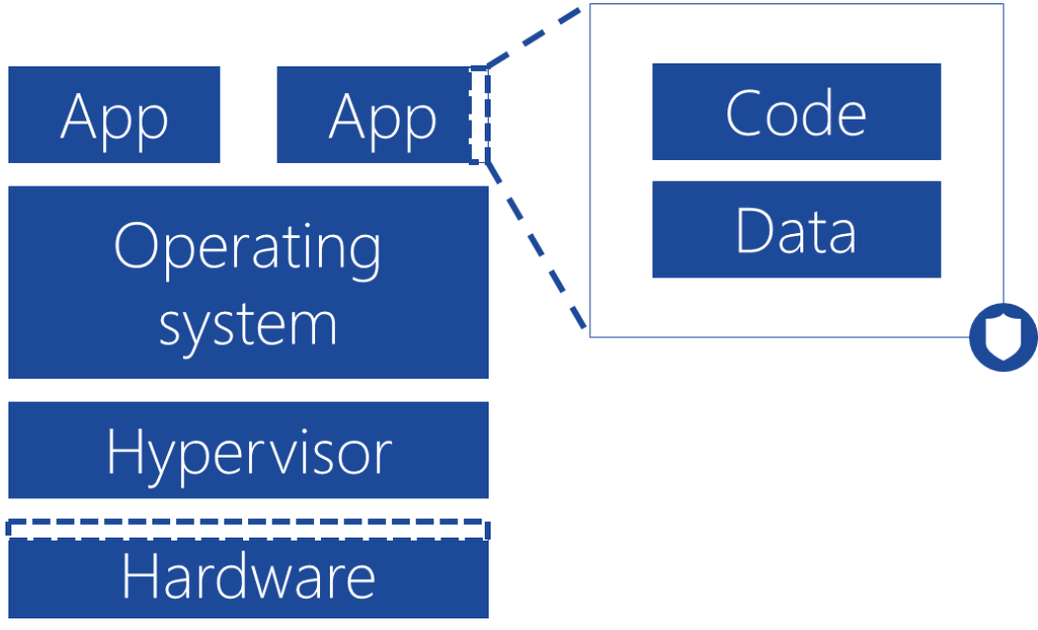
# Encrypt data with Always Encrypted

- Encryption technology in Azure SQL Database and SQL Server:
  - Ensures that sensitive data never appears as plaintext inside the database system.
  - Allows clients to encrypt sensitive data inside client applications
  - Helps protect sensitive data:
    - At rest on the server
    - In transit between client and server
    - While the data is in use
  - Provides a separation between:
    - Who owns the data (and can view it)
    - Who manages the data (but should have no access).
  - Requires a specialized driver installed on client computers to automatically encrypt and decrypt sensitive data in the client application

# Implement Azure Confidential Compute and SSL/TLS Communications

# Azure confidential computing

- Refers to features of many Azure services that encrypt data in use
  - Designed for scenarios where sensitive data needs to be processed in the cloud.
  - Ensures that when data is decrypted, it is protected inside a Trusted Execution Environment
  - TEE provides additional layer of protection:
    - Ensures that there is no way to view data or operations from the outside, even with a debugger
    - Ensures that only authorized code is permitted access
    - Automatically disables the environment if it detects signs of tampering
  - TEE is implemented by a combination of:
    - Hardware
    - Software
    - Services
    - Framework



# SSL and TLS overview

- TLS and SSL are cryptographic protocols
- TLS is a successor of SSL
  - Support for TLS 1.2 is integrated into majority of Azure services, including:
    - Azure SQL Database
    - Azure Database for MySQL
    - Azure Storage
    - Azure Application Gateway
    - Azure App Service
  - To take advantage of TLS 1.2, you need to enable it on the client side

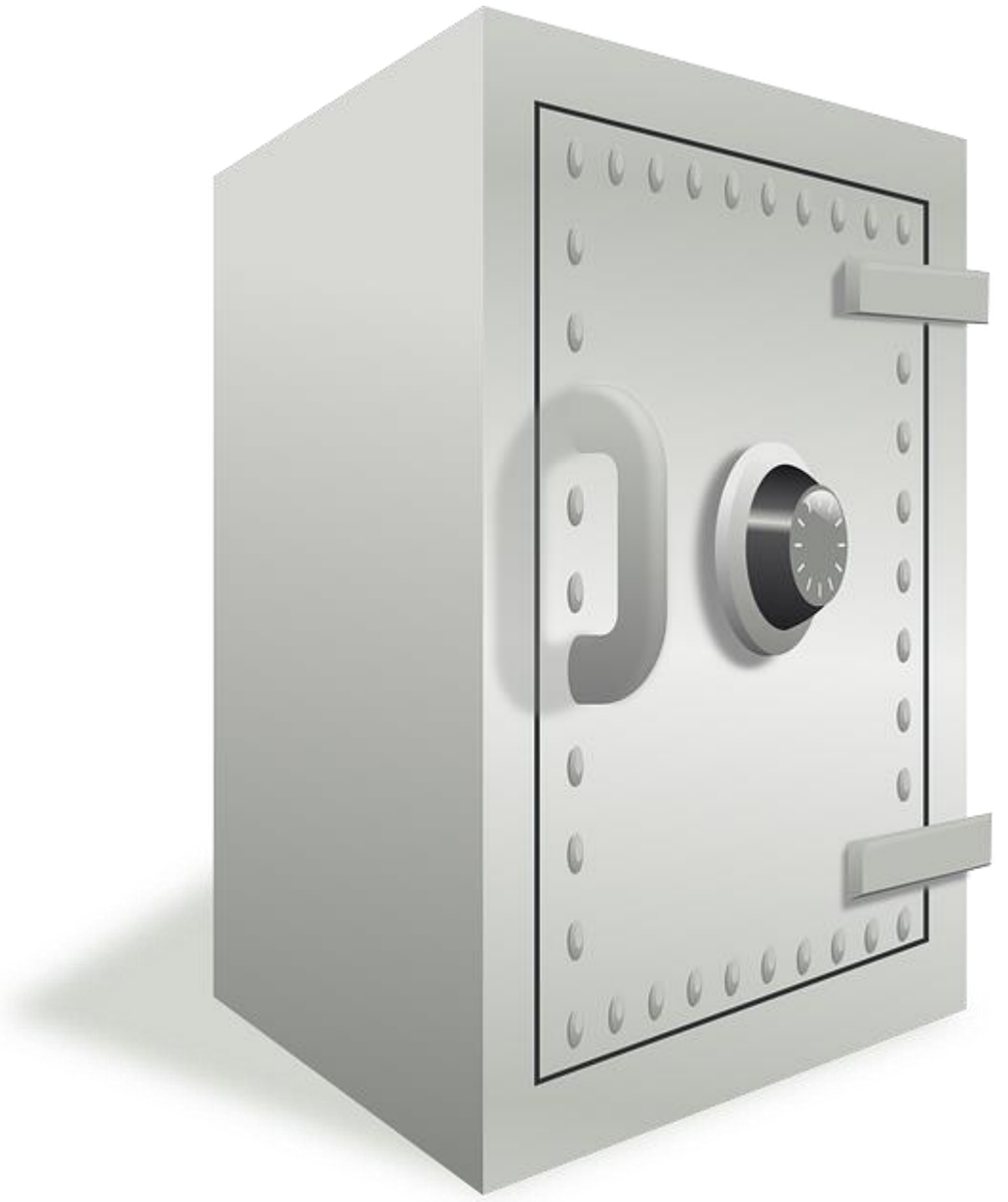
Microsoft.NET: `System.Net.ServicePointManager.SecurityProtocol = System.Net.SecurityProtocolType.Tls12;`

PowerShell: `[System.Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]::Tls12;`

# Manage cryptographic keys in the Azure Key Vault

# Azure key vault

- Cloud service that works as a security-enhanced secrets store:
  - Allows you to create multiple security-enhanced containers, called vaults
  - Main vault characteristics:
    - Support for secrets, such as a password, keys, and certificate.
    - The use of hardware security modules (HSMs) for key storage and cryptographic operations
    - The ability to request and renew TLS certificates
    - Logging of all operations.



# Accessing Key Vault in Azure CLI

- To create a vault by using the Azure CLI, run:
  - `az keyvault create --name contosovault --resource-group SecurityGroup --location westus`
- To add a secret to the vault, run:
  - `az keyvault secret set --vault-name contosovault --name DatabasePassword --value 'Pa5w.rd‘`
- To view the secret value, run:
  - `az keyvault secret show --vault-name contosovault --name DatabasePassword`

# Questions?

# Homework Assignment

https://aka.ms/AZ300

Microsoft Azure

# Open Mic