Microsoft

AZ-203.3 Module 03: Develop solutions that use a relational database

Prashanth Kumar

Microsoft

Topics

- Azure SQL Database
- Create, read, update, and delete database entities by using code

Lesson 01: Azure SQL Database

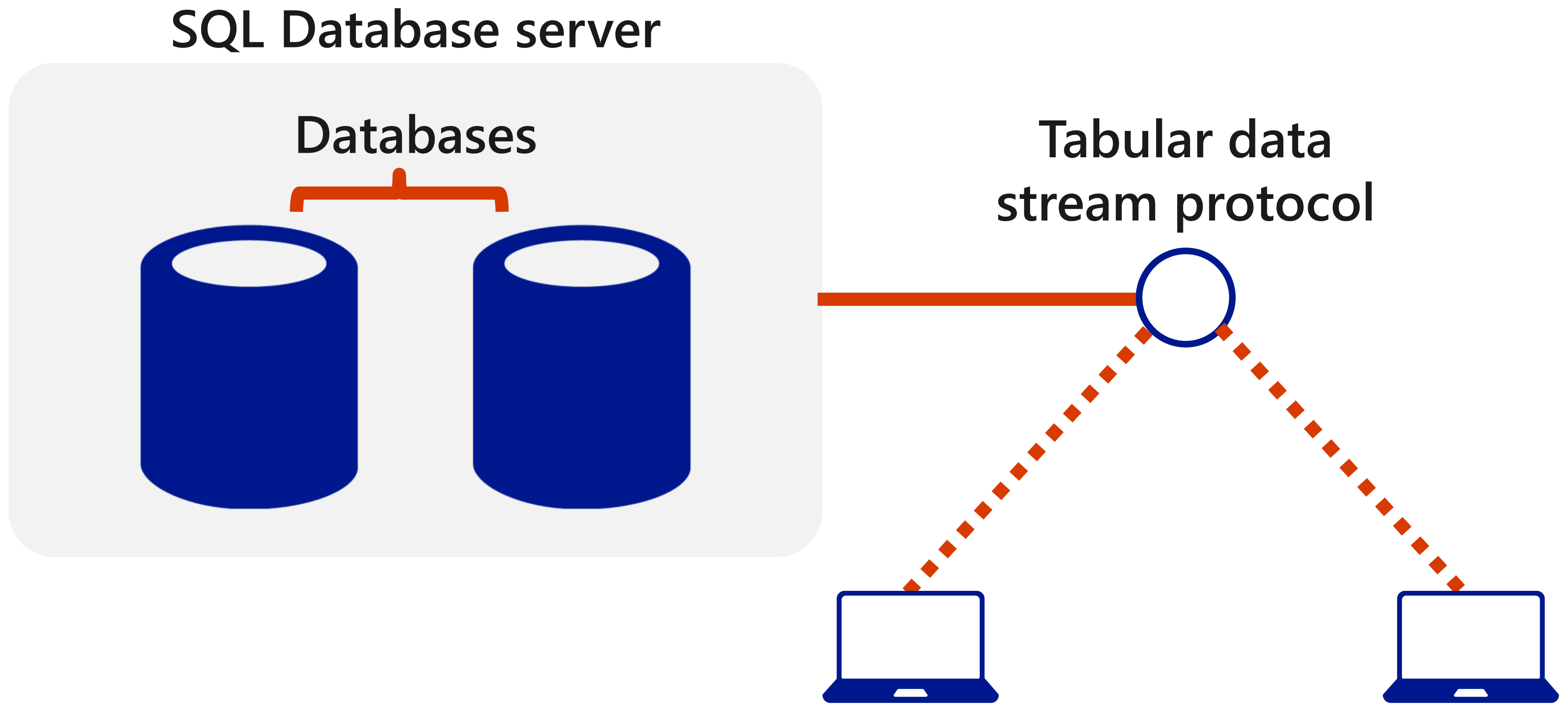
Azure SQL Database

- Relational database managed service
  - Microsoft handles patching and updates
- Shares code and features with SQL Server
- Two purchasing models
  - vCore-based compute purchasing
  - DTU-based throughput purchasing

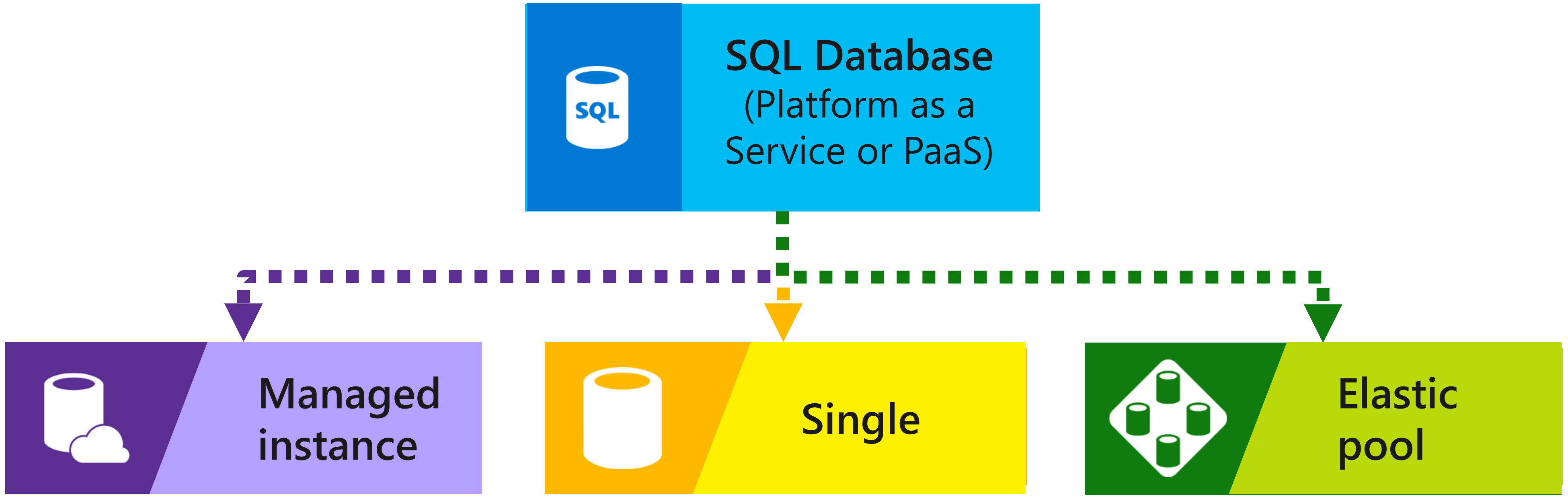
SQL Database

SQL Server

SQL Database Server and SQL Database



Azure SQL Database deployment options



Choosing the right SQL option in Azure

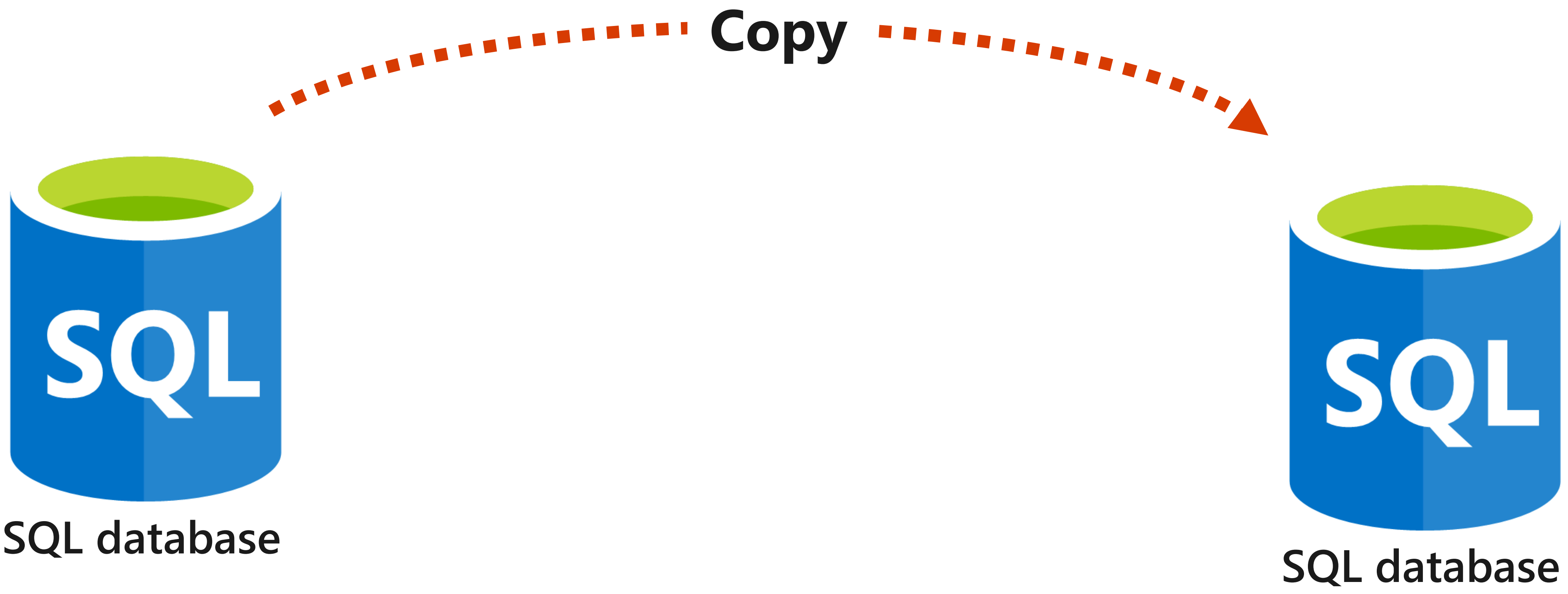
SQL Server on a virtual machine (VM)	Azure SQL Database (Managed Instance)	Azure SQL Database (Logical server)
• You have full control over the SQL Server engine • Up to 99.95% availability • Full parity with the matching version of on-premises SQL Server • Fixed, well-known database engine version • Easy migration from SQL Server on-premises • Private IP address within Azure VNet • You have the ability to deploy application or services on the host where SQL Server is placed	• High compatibility with SQL Server on-premises • 99.99% availability guaranteed • Built-in backups, patching, recovery • Latest stable Database Engine version • Easy migration from SQL Server • Private IP address within Azure VNet • Built-in advanced intelligence and security • Online change of resources (CPU/storage)	• The most commonly used SQL Server features are available • 99.99% availability guaranteed • Built-in backups, patching, recovery • Latest stable Database Engine version • Ability to assign necessary resources (CPU/storage) to individual databases • Built-in advanced intelligence and security • Online change of resources (CPU/storage)

SQL option weaknesses

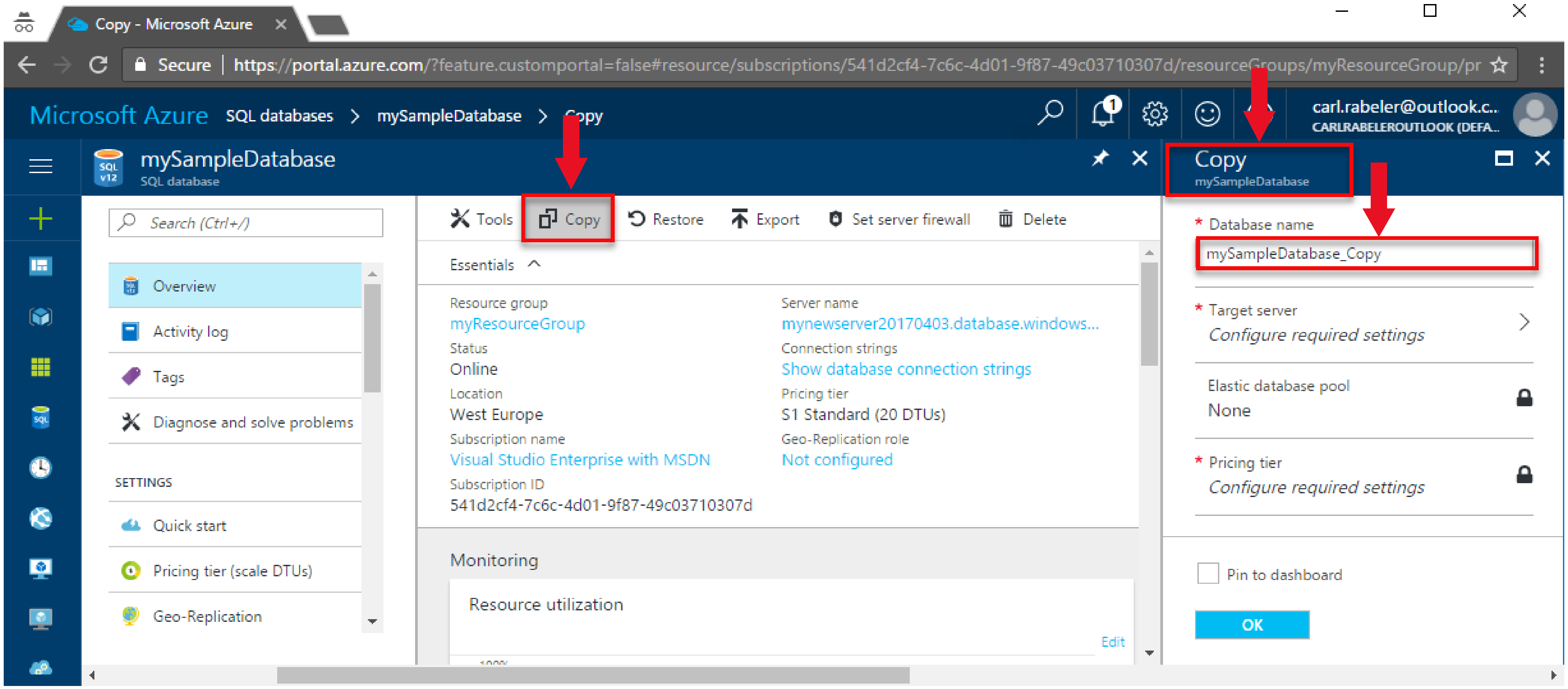
SQL Server on VM	Azure SQL Database (Managed Instance)	Azure SQL Database (Logical server)
• You need to manage your backups and patches • You need to implement your own high-availability solution • There is downtime while changing the resources (CPU/storage)	• There is still a minimal number of SQL Server features that are not available • No guaranteed exact maintenance time (but nearly transparent) • Compatibility with the SQL Server version can be achieved only by using database compatibility levels	• Migration from SQL Server might be difficult • Some SQL Server features are not available • No guaranteed exact maintenance time (but nearly transparent) • Compatibility with the SQL Server version can be achieved only by using database compatibility levels • Private IP address cannot be assigned (you can limit the access using firewall rules)

Demo: Creating an Azure SQL Database

Copying a SQL database



Copy an Azure SQL database - Azure portal


Copy - Microsoft Azure

Secure | https://portal.azure.com/?feature.customportal=false#resource/subscriptions/541d2cf4-7c6c-4d01-9f87-49c03710307d/resourceGroups/myResourceGroup/pr

Microsoft Azure SQL databases > mySampleDatabase > Copy

carl.rabeler@outlook.c...
CARLRABELEROUTLOOK (DEFA...

mySampleDatabase
SQL database

Search (Ctrl+/)

Overview
Activity log
Tags
Diagnose and solve problems

SETTINGS

Quick start
Pricing tier (scale DTUs)
Geo-Replication

Tools | Copy | Restore | Export | Set server firewall | Delete

Essentials

Resource group
myResourceGroup
Status
Online
Location
West Europe
Subscription name
Visual Studio Enterprise with MSDN
Subscription ID
541d2cf4-7c6c-4d01-9f87-49c03710307d

Server name
mynewserver20170403.database.windows...
Connection strings
Show database connection strings
Pricing tier
S1 Standard (20 DTUs)
Geo-Replication role
Not configured

Monitoring

Resource utilization

Edit

Copy
mySampleDatabase

* Database name
mySampleDatabase_Copy

* Target server
Configure required settings

Elastic database pool
None

* Pricing tier
Configure required settings

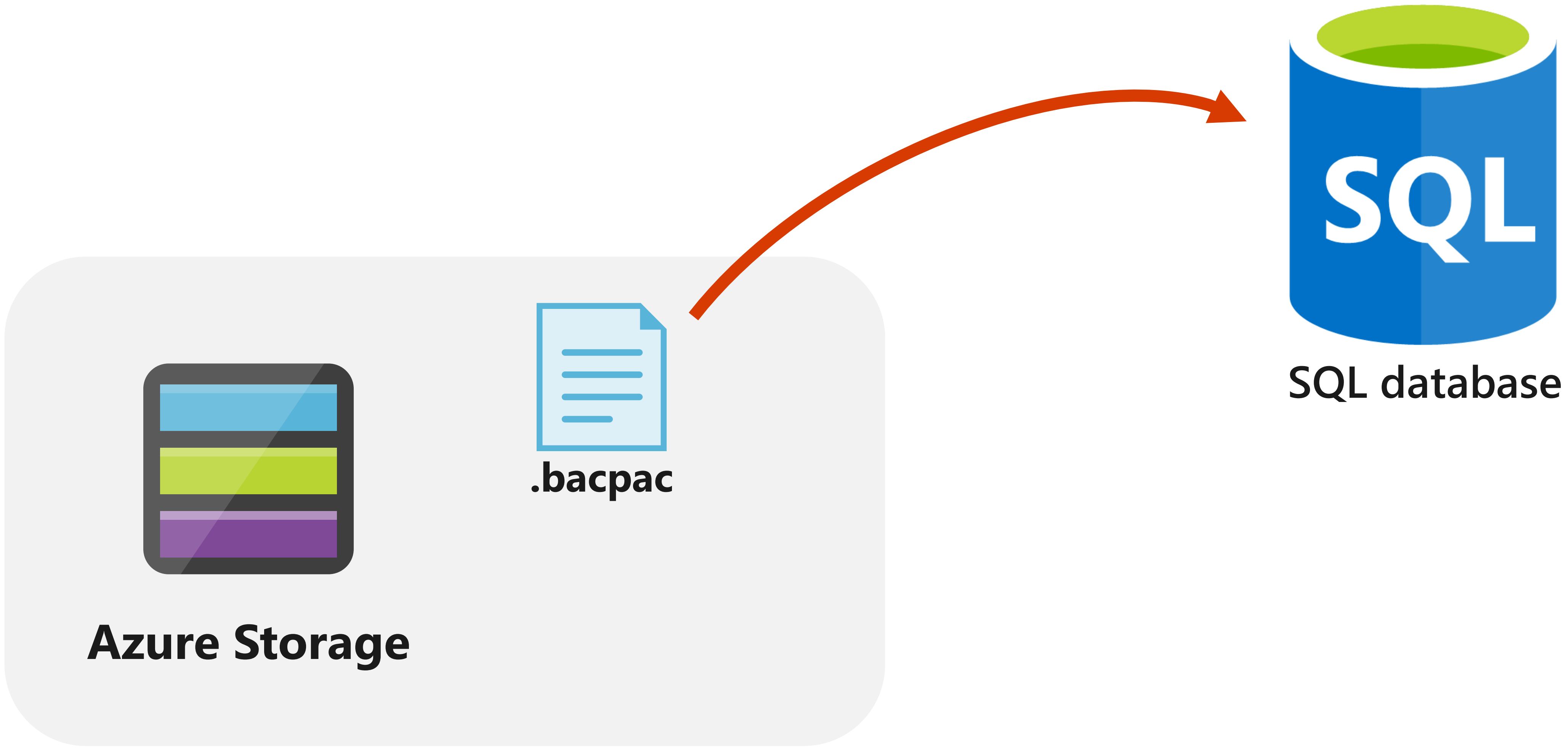
Pin to dashboard

OK

Copy an Azure SQL database – Azure PowerShell

```
New-AzSqlDatabaseCopy `
    -ResourceGroupName "myResourceGroup" `
    -ServerName $sourceserver `
    -DatabaseName "MySampleDatabase" `
    -CopyResourceGroupName "myResourceGroup" `
    -CopyServerName $targetserver `
    -CopyDatabaseName "CopyOfMySampleDatabase"
```

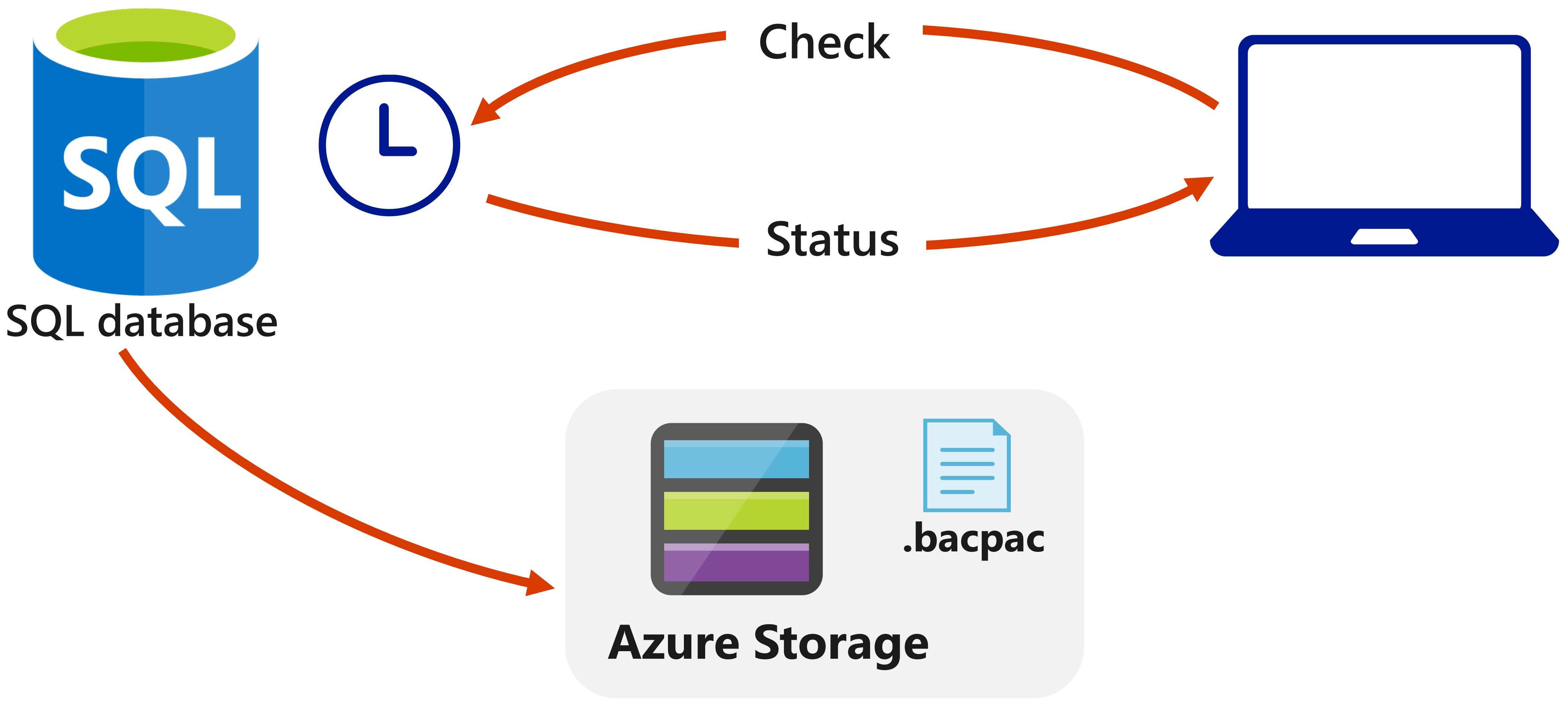
Importing a .bacpac file



Importing a .bacpac file - PowerShell

```
$importRequest = New-AzSqlDatabaseImport
    -ResourceGroupName "<your_resource_group>" `
    -ServerName "<your_server>" `
    -DatabaseName "<your_database>" `
    -DatabaseMaxSizeBytes "<database_size_in_bytes>" `
    -StorageKeyType "StorageAccessKey" `
    -StorageKey "<storage_access_key>"
    -StorageUri "https://[account].blob.core.windows.net/example/demo.bacpac" `
    -Edition "Standard" `
    -ServiceObjectiveName "P6" `
    -AdministratorLogin "<your_server_admin_account_user_id>" `
    -AdministratorLoginPassword $(ConvertTo-SecureString –String `
        "<your_server_admin_account_password>" -AsPlainText -Force)
```

Exporting a .bacpac file



Create a .bacpac export job - PowerShell

```
$exportRequest = New-AzSqlDatabaseExport `
    -ResourceGroupName $ResourceGroupName `
    -ServerName $ServerName `
    -DatabaseName $DatabaseName `
    -StorageKeytype $StorageKeytype `
    -StorageKey $StorageKey `
    -StorageUri $BacpacUri `
    -AdministratorLogin $creds.UserName `
    -AdministratorLoginPassword $creds.Password
```

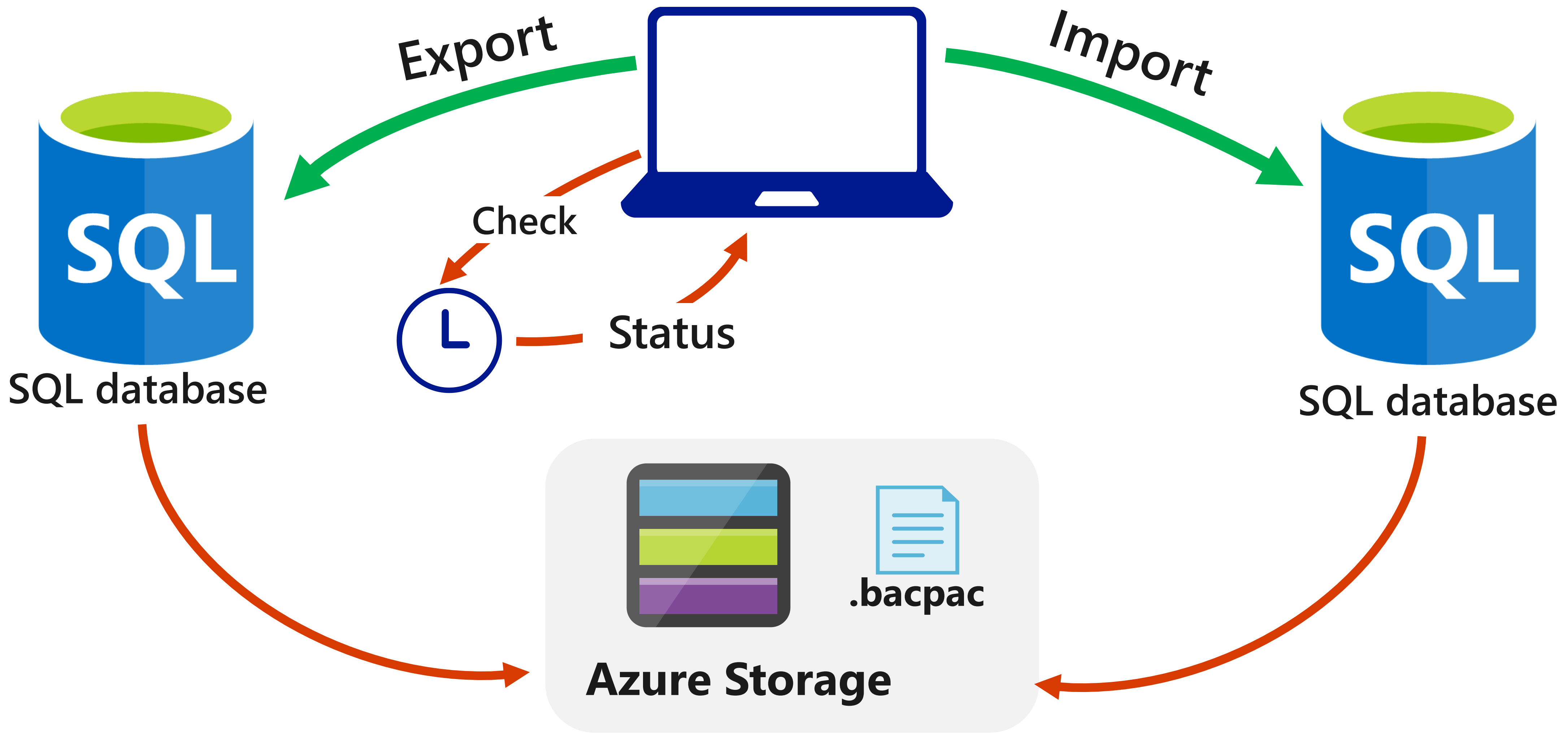
Observing a .bacpac export - PowerShell

```
$exportStatus = Get-AzSqlDatabaseImportExportStatus `
    -OperationStatusLink $exportRequest.OperationStatusLink

while ($exportStatus.Status -eq "InProgress")
{
    Start-Sleep -s 10
    $exportStatus = Get-AzSqlDatabaseImportExportStatus `
        -OperationStatusLink $exportRequest.OperationStatusLink
}

$exportStatus
```

Exporting and importing databases



Lesson 02: Create, read, update, and delete database entities by using code

Entity Framework

- Object-relational mapper library for .NET
- Designed to reduce the impedance mismatch between the relational and object-oriented world
- Goal is to enable developers to interact with data stored in relational databases by using strongly typed .NET objects that represent the application's domain
- Eliminates the need for a large portion of the data access "plumbing" code that they usually need to write to access data in a database

Entity Framework Core and Entity Framework

- Entity Framework Core (EF Core) is a recent rewrite of the entire Entity Framework library to target .NET standard
- Entity Framework Core can be used with .NET Framework applications and .NET Core applications
- Entity Framework Core was built to be more lightweight and agile than the full Entity Framework, by dropping many of the earlier features from Entity Framework and implementing new, modern, and extensible features at an agile pace
- For new applications, we recommend considering using Entity Framework Core over Entity Framework

Entity framework providers

There are many database providers in the current market, including:

SQL Server

SQLite

PostgreSQL

MySQL™

MariaDB Foundation

MyCAT

Microsoft® SQL Server®

Firebird

DB2

Informix

Microsoft Access

MySQL providers

- MySql.Data.EntityFrameworkCore
  - https://aka.ms/AA4gbio
- Pomelo.EntityFrameworkCore.MySql
  - https://aka.ms/AA4gbih

PostgreSQL providers

- Npgsql.EntityFrameworkCore.PostgreSQL
  - https://aka.ms/AA4gbii
- Devart.Data.PostgreSql.EFCore
  - https://aka.ms/AA4gj9r

PostgreSQL

Modeling a database by using Entity Framework Core

BlogId	URL	Description
1	/first-post	This is my first post on this platform
2	/follow-up-post	NULL

```
public class Blog
{
        public int BlogId { get; set; }
        public string Url { get; set; }
        public string Description { get; set; }
}
```

Modeling classes

```csharp
public class BlogDatabase
{
    public IEnumerable<Blog> Blogs { get; set; }
}
```

Entity Framework fluent API

```csharp
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Blog>()
        .HasKey(c => c.BlogId)
        .Property(b => b.Url)
        .IsRequired()
        .Property(b => b.Description);
}
```

Entity Framework data annotations

```
public class Blog
{
    [Key]
    public int BlogId { get; set; }

    [Required]
    public string Url { get; set; }

    public string Description { get; set; }
}
```

Entity Framework DbContext implementation

```csharp
public class BlogContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
}
```

Querying databases by using Entity Framework Core

```
List<Blog> allblogs = context.Blogs.ToList();

IEnumerable<Blog> someblogs = context.Blogs
    .Where(b => b.Url.Contains("dotnet"))

Blog specificblog = context.Blogs
    .Single(b => b.BlogId == 1);
```

Creating records by using Entity Framework

```csharp
using (var context = new BlogContext())
{
    var blog = new Blog { Url = "/sample-post", Description = "This is an
example of a post." };

    context.Blogs.Add(blog);

    context.SaveChanges();

    Console.WriteLine(blog.BlogId + ": " + blog.Url);
}
```

Modifying records by using Entity Framework

```
using (var context = new BlogContext())
{
    var blog = context.Blogs.First();

    blog.Url = "/original-post";

    context.SaveChanges();
}
```

Demo: Writing Entity Framework code by using C#

Microsoft

Review

- Azure SQL Database
- Create, read, update, and delete database entities by using code

Microsoft