PASS 2023 Pre-conference Workshop

Azure SQL and SQL Server: All Things Developers

PASS 2023 Pre-conference Workshop

Azure SQL and SQL Server: All Things Developers

PASS
Data Community
SUMMIT
2023

Today's Speakers

Anagha Todalbagi
Microsoft

Brian Spendolini
Microsoft

Schedule

Pre-Conference Sessions
8:30 AM – 10:30 AM

Refreshment Break
10:30 AM – 10:45 AM

Pre-Conference Sessions
10:45 AM – 12:15 PM

Pre-Conference Lunch
12:15 PM – 1:15 PM

Pre-Conference Sessions
1:15 PM – 2:45 PM

Refreshment Break
2:45 PM – 3:00 PM

Pre-Conference Sessions
3:00 PM – 4:30 PM

Agenda

- Workshop Overview
- Workshop Tasks
- Various Breaks and Lunch

Workshop Goals

- Get comfortable with new development tools
  - Local and cloud
  - New CLIs
- Expose you to REST/GraphQL
- Get started with Azure SQL Database
- Developing with Azure
- Expand your thinking about app development
- No marketing – all hands on

WORKSHOP OVERVIEW

Workshop Overview/Tasks

- Setup your environment
- Create database and objects
- Use Data API builder
- Applications with SWA
- Deploy to Azure SQL Database
- Invoke REST endpoints in the database
- Create a change data stream
- CICD with GitHub Actions

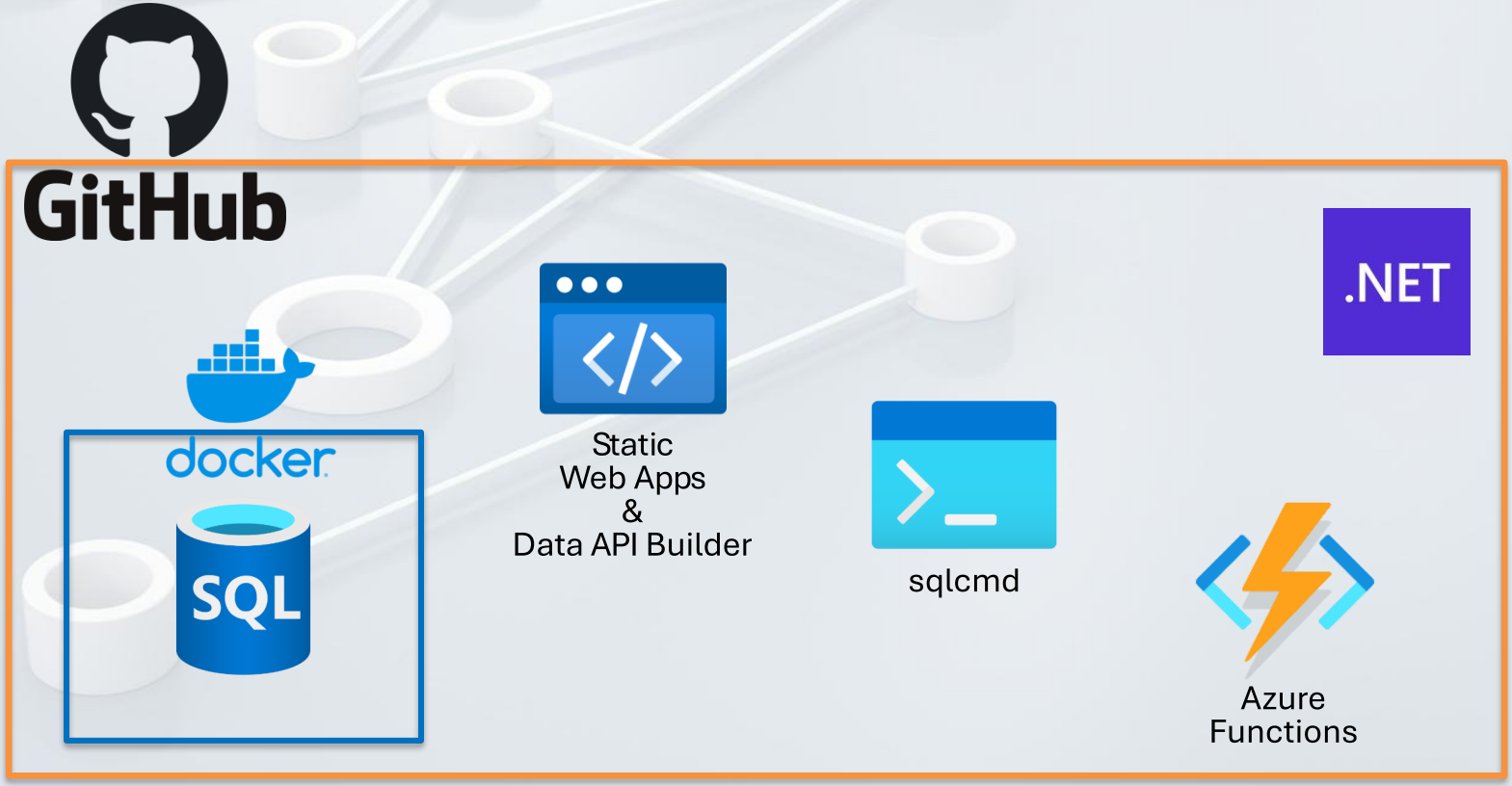
Getting Started

Accounts Needed

- GitHub account
- Azure account
  - Free Azure SQL Database

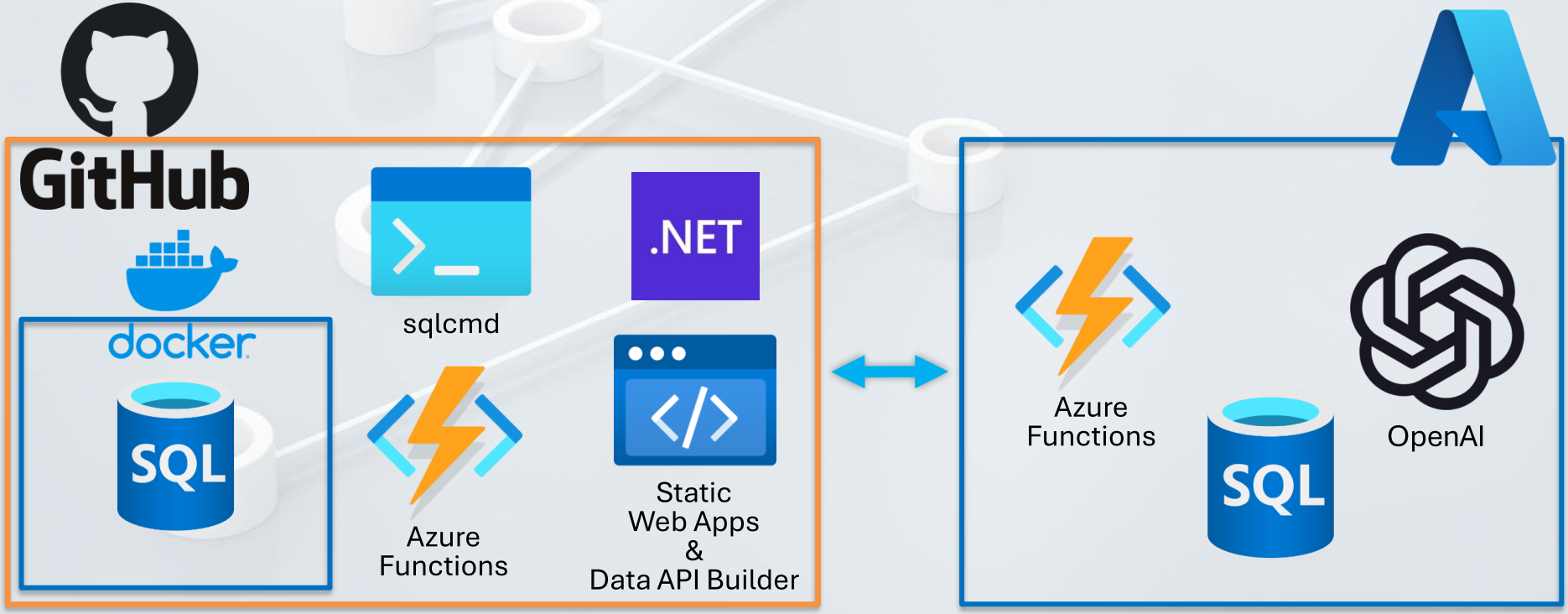
Getting Started

Development Environment



Getting Started

Development Environment

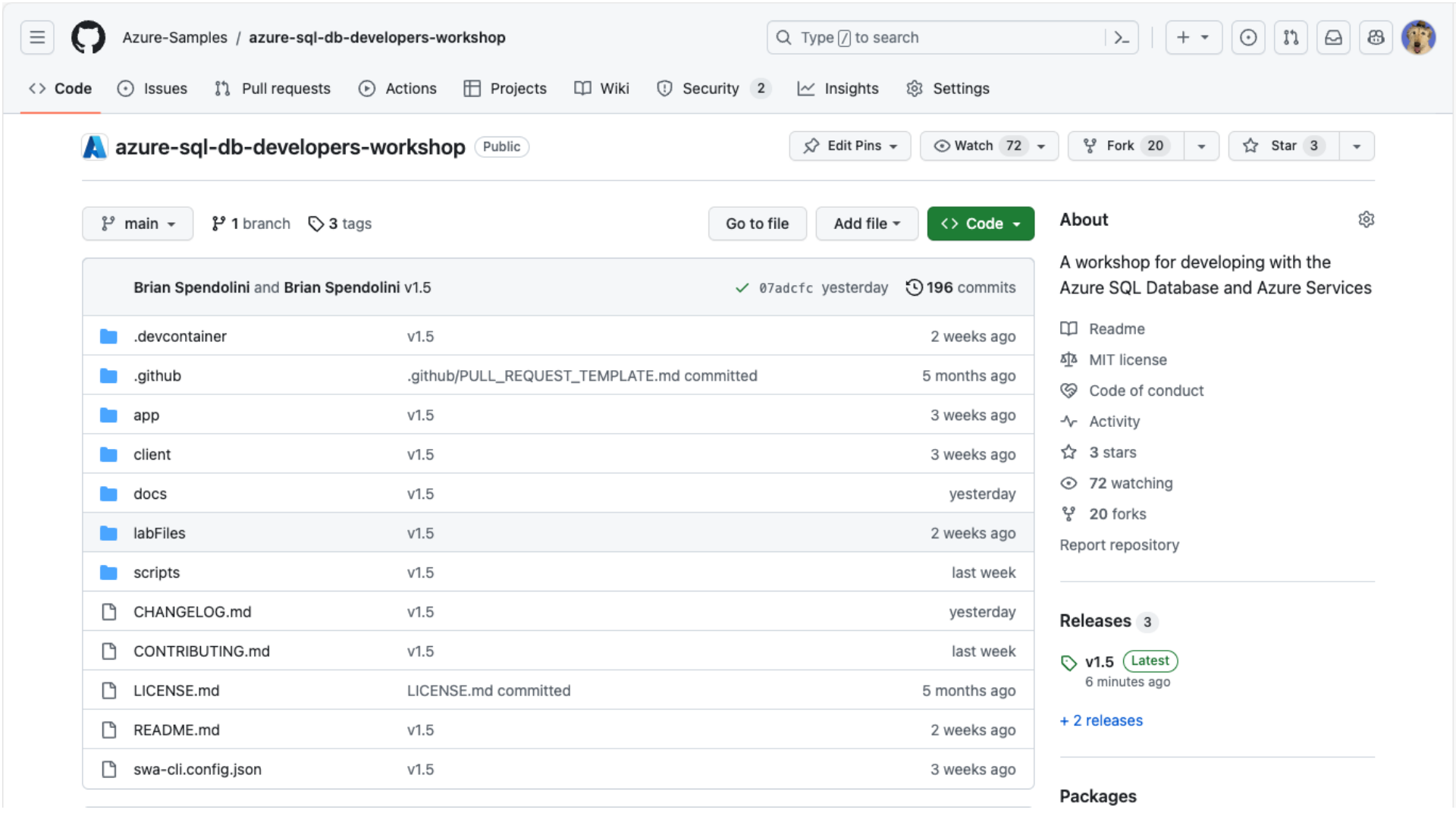


CHAPTER 1: SETUPS

Chapter 1: Setups

- Fork the repository
- Create the codespace

https://aka.ms/forkit


Azure-Samples / azure-sql-db-developers-workshop
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security 2
Insights
Settings
azure-sql-db-developers-workshop
Public
Edit Pins
Watch 72
Fork 20
Star 3
main
1 branch
3 tags
Go to file
Add file
Code
Brian Spendolini and Brian Spendolini v1.5
07adcfc yesterday
196 commits
.devcontainer v1.5 2 weeks ago
.github .github/PULL_REQUEST_TEMPLATE.md committed 5 months ago
app v1.5 3 weeks ago
client v1.5 3 weeks ago
docs v1.5 yesterday
labFiles v1.5 2 weeks ago
scripts v1.5 last week
CHANGELOG.md v1.5 yesterday
CONTRIBUTING.md v1.5 last week
LICENSE.md LICENSE.md committed 5 months ago
README.md v1.5 2 weeks ago
swa-cli.config.json v1.5 3 weeks ago
About
A workshop for developing with the Azure SQL Database and Azure Services
Readme
MIT license
Code of conduct
Activity
3 stars
72 watching
20 forks
Report repository
Releases 3
v1.5 Latest
6 minutes ago
+ 2 releases
Packages

Setups

Repository Contents

.devcontainer

- Codespace docker config

client

- contains the Todo application for use with Data API builder and SWA

app

- contains the sample JavaScript test application for use with Data API builder and SWA

docs

- Has all the chapters/lessons of the workshop

scripts

- Installs all the needed dev tools

labFiles

- contains files to be used with the workshop in various chapters

GitHub Codespaces

- A development environment that's hosted in the cloud
- Each codespace is hosted by GitHub in a Docker container
- Browser based environment
- Config files are contained in the repository
- VS Code extension available

main 1 branch 0 tags

Go to file | Add file | Code

This branch is up to date with Azure-Samples/azure-sql-db-developers-workshop:main. Contribute Sync fork

JetterMcTedder v1.5	eadcad5 yesterday	167 commits
.devcontainer	v1.5	last week
.github	.github/PULL_REQUEST_TEMPLATE.md committed	4 months ago
app	v1.5	5 days ago
client	v1.5	5 days ago
docs	v1.5	yesterday
labFiles	v1.5	5 days ago
scripts	v1.5	2 days ago
CHANGELOG.md	v1.5	2 days ago
CONTRIBUTING.md	v1.0	4 months ago
LICENSE.md	LICENSE.md committed	4 months ago
README.md	Updated README	3 weeks ago
swa-cli.config.json	v1.5	last week

README.md

About

A workshop for developing with the Azure SQL Database and Azure Services

- Readme
- MIT license
- Activity
- 0 stars
- 0 watching
- 21 forks

Releases

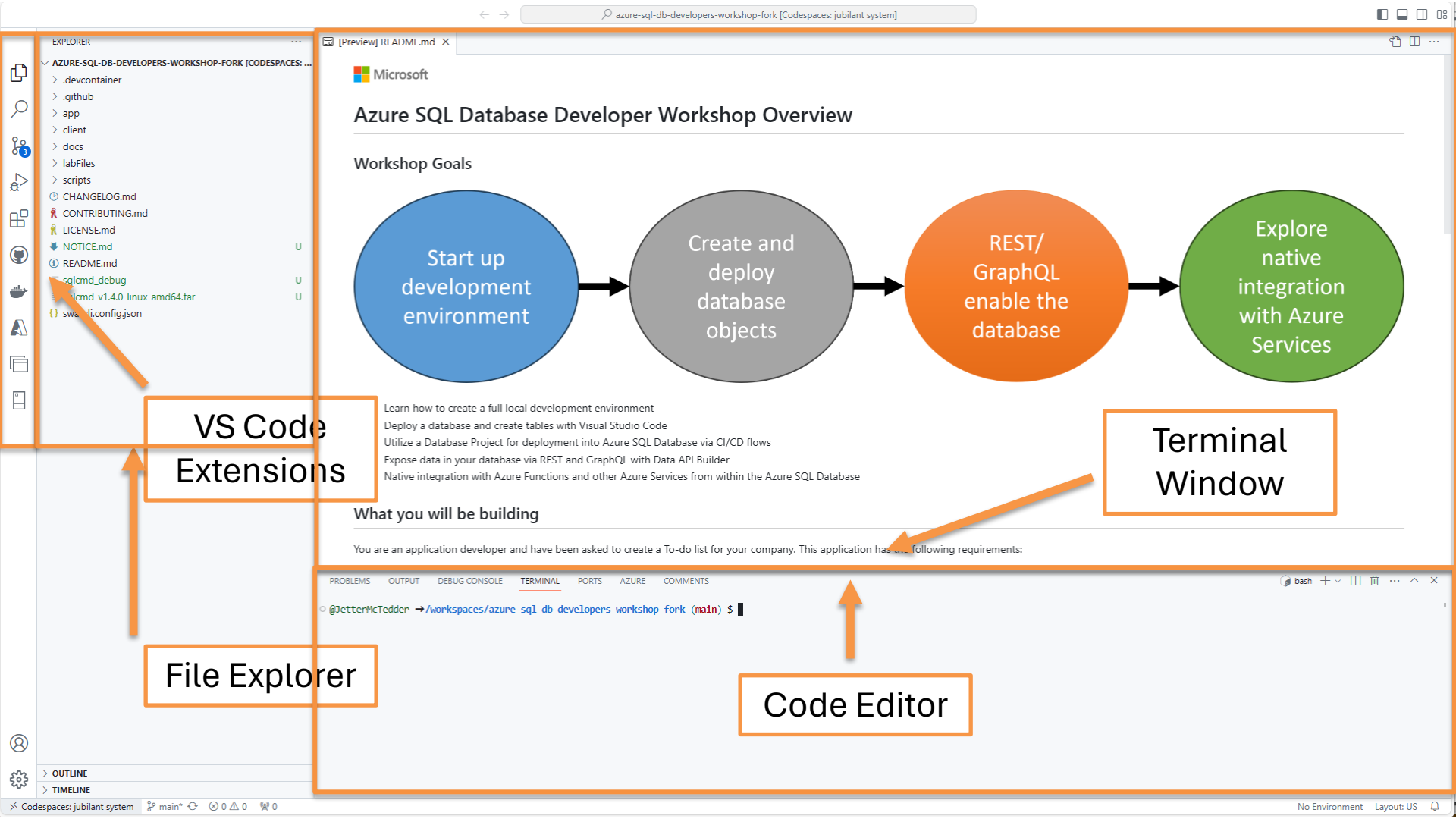
No releases published
Create a new release

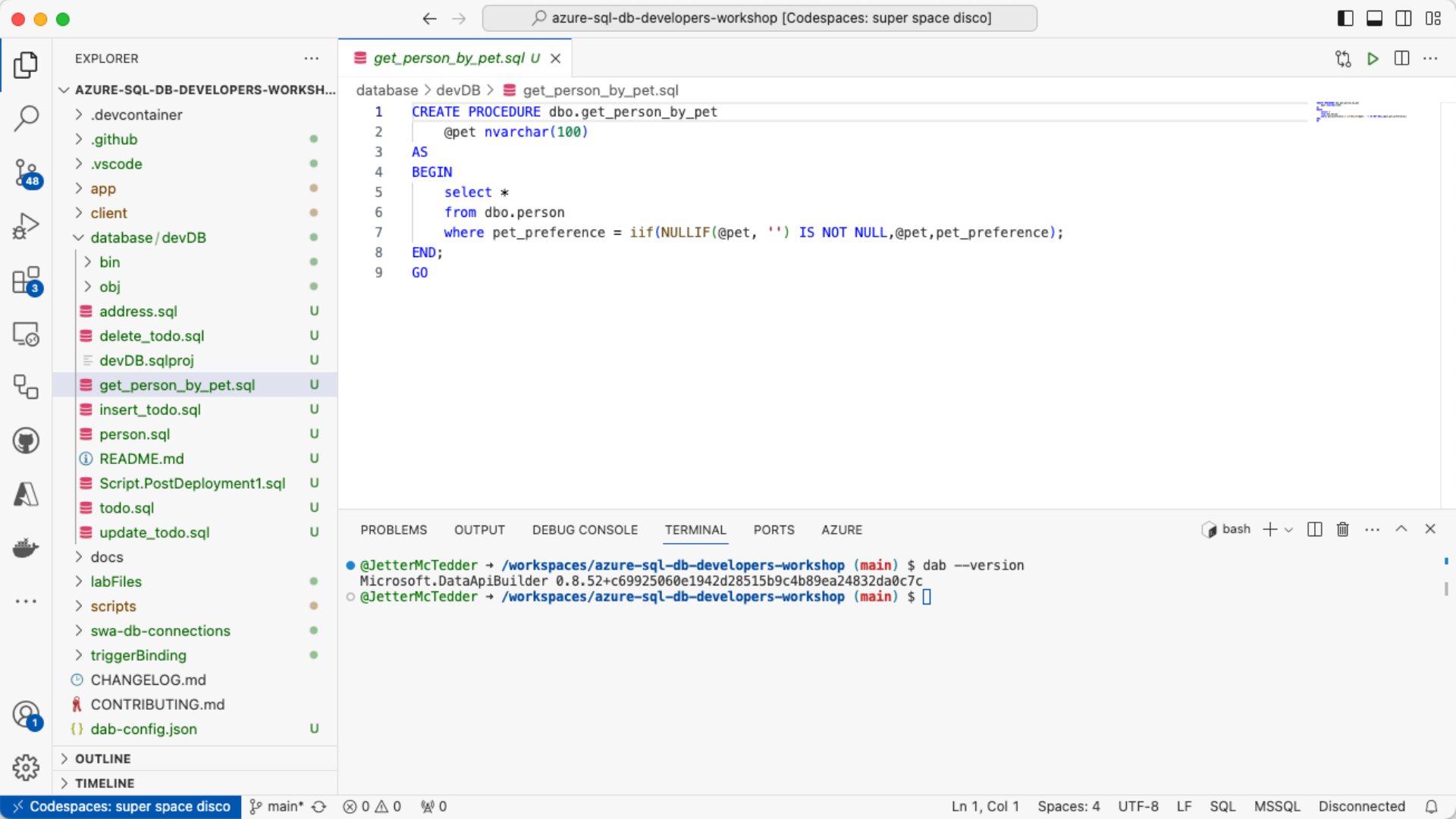
Packages

No packages published
Publish your first package

Languages

- Vue 39.7%
- SourcePawn 34.0%
- HTML 11.5%
- CSS 5.7%
- Shell 5.0%
- JavaScript 4.1%


azure-sql-db-developers-workshop-fork [Codespaces: jubilant system]
EXPLORER
AZURE-SQL-DB-DEVELOPERS-WORKSHOP-FORK [CODESPACES: ...
.devcontainer
.github
app
client
docs
labFiles
scripts
CHANGELOG.md
CONTRIBUTING.md
LICENSE.md
NOTICE.md
README.md
sqlcmd_debug
sqlcmd-v1.4.0-linux-amd64.tar
swa-cli.config.json
[Preview] README.md
Microsoft
Azure SQL Database Developer Workshop Overview
Workshop Goals
Start up development environment
Create and deploy database objects
REST/ GraphQL enable the database
Explore native integration with Azure Services
Learn how to create a full local development environment
Deploy a database and create tables with Visual Studio Code
Utilize a Database Project for deployment into Azure SQL Database via CI/CD flows
Expose data in your database via REST and GraphQL with Data API Builder
Native integration with Azure Functions and other Azure Services from within the Azure SQL Database
What you will be building
You are an application developer and have been asked to create a To-do list for your company. This application has the following requirements:
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
COMMENTS
bash
@JetterMcTedder ➜ /workspaces/azure-sql-db-developers-workshop-fork (main) $
OUTLINE
TIMELINE
Codespaces: jubilant system
main*
No Environment
Layout: US
VS Code Extensions
File Explorer
Terminal Window
Code Editor


azure-sql-db-developers-workshop [Codespaces: super space disco]
EXPLORER
AZURE-SQL-DB-DEVELOPERS-WORKSH...
.devcontainer
.github
.vscode
app
client
database / devDB
bin
obj
address.sql U
delete_todo.sql U
devDB.sqlproj U
get_person_by_pet.sql U
insert_todo.sql U
person.sql U
README.md U
Script.PostDeployment1.sql U
todo.sql U
update_todo.sql U
docs
labFiles
scripts
swa-db-connections
triggerBinding
CHANGELOG.md
CONTRIBUTING.md
dab-config.json U
OUTLINE
TIMELINE
get_person_by_pet.sql U
database > devDB > get_person_by_pet.sql
1 CREATE PROCEDURE dbo.get_person_by_pet
2 @pet nvarchar(100)
3 AS
4 BEGIN
5 select *
6 from dbo.person
7 where pet_preference = iif(NULLIF(@pet, '') IS NOT NULL,@pet,pet_preference);
8 END;
9 GO
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
bash
@JetterMcTedder → /workspaces/azure-sql-db-developers-workshop (main) $ dab --version
Microsoft.DataApiBuilder 0.8.52+c69925060e1942d28515b9c4b89ea24832da0c7c
@JetterMcTedder → /workspaces/azure-sql-db-developers-workshop (main) $
Codespaces: super space disco
main*
Ln 1, Col 1
Spaces: 4
UTF-8
LF
SQL
MSSQL
Disconnected

Utilities Check

Checks to run once the post-build scripts are finished:

Functions Core Tools
- `func --version`

Data API builder
- `dab --version`

sqlcmd
- `sqlcmd --version`

Static Web Apps
- `swa --version`

```
@JetterMcTedder → /workspaces/azure-sql-db-developers-workshop (main) $ func --version
4.0.5390
@JetterMcTedder → /workspaces/azure-sql-db-developers-workshop (main) $ dab --version
Microsoft.DataApiBuilder 0.8.52+c69925060e1942d28515b9c4b89ea24832da0c7c
@JetterMcTedder → /workspaces/azure-sql-db-developers-workshop (main) $ sqlcmd --version
v1.4.0
Legal docs and information: aka.ms/SqlcmdLegal
Third party notices: aka.ms/SqlcmdNotices
@JetterMcTedder → /workspaces/azure-sql-db-developers-workshop (main) $ swa --version
1.1.4
@JetterMcTedder → /workspaces/azure-sql-db-developers-workshop (main) $
```

Knowledge Check

What cloud-based development tool are you going to use today?

A. GitHub CICD

B. Azure DevOps

C. GitHub Codespaces

D. Microsoft Teams

E. A and D

Chapter 1: Setups

- Fork the repository
- Create the codespace


START CHAPTER 1

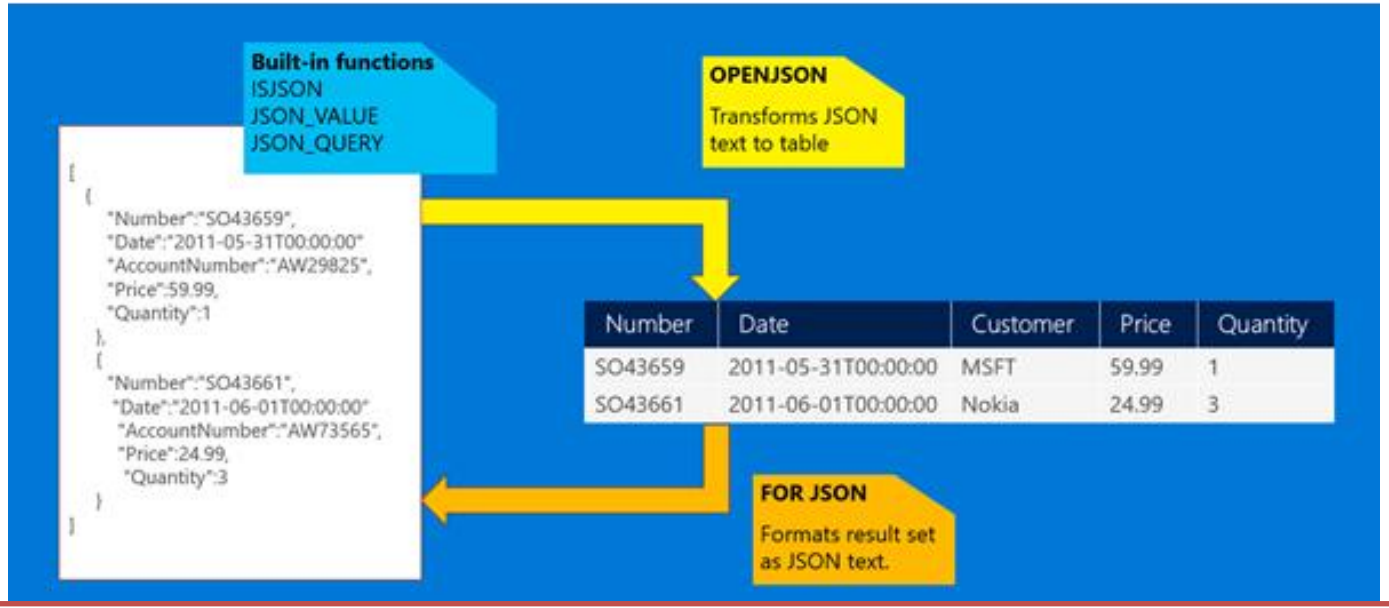
CHAPTER 2: CREATE A DATABASE AND OBJECTS

Azure SQL Database

- Features for developers
  - Multi-model support
    - Relational / JSON / Graph / Geospatial / XML
  - Row-Store and Column-Store
    - on the same table
  - Ledger Tables
  - Replicas / High-Availability / Scale-Out / Scale-Up

JSON Support

Azure SQL fully support storage, manipulation and creation of JSON documents



https://docs.microsoft.com/sql/relational-databases/json/json-data-sql-server

JSON Support

ISJSON: Tests whether a string contains valid JSON.

JSON_PATH_EXISTS: Tests whether a specified SQL/JSON path exists in the input JSON string.

JSON_MODIFY: Updates the value of a property in a JSON string and returns the updated JSON string.

JSON_VALUE: Extracts a scalar value from a JSON string.

JSON Support

JSON_QUERY: Extracts an object or an array from a JSON string.

JSON_OBJECT: Constructs JSON object text from zero or more expressions.

JSON_ARRAY: Constructs JSON array text from zero or more expressions.

FOR JSON: Converts SQL Server data types to JSON types.

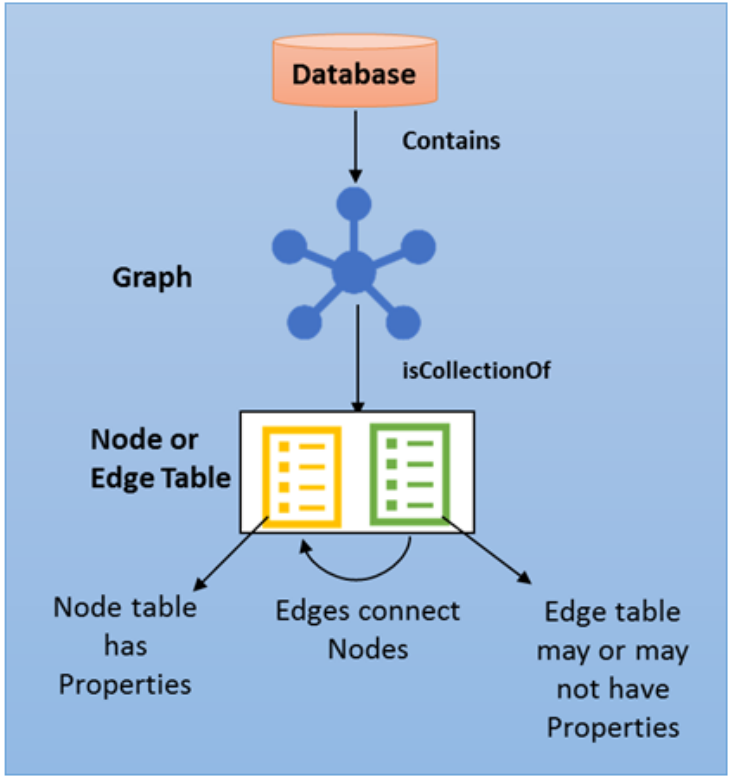
OPENJSON: Converts JSON text into a set of rows and columns.

JSON Support

```
SELECT Name,
        Surname,
        JSON_VALUE(jsonCol, '$.info.address.PostCode') AS PostCode,
        JSON_VALUE(jsonCol, '$.info.address."Address Line 1"')
            + ' ' + JSON_VALUE(jsonCol, '$.info.address."Address Line 2"')
            AS Address,
        JSON_QUERY(jsonCol, '$.info.skills') AS Skills
FROM People
WHERE ISJSON(jsonCol) > 0
    AND JSON_VALUE(jsonCol, '$.info.address.Town') = 'Belgrade'
    AND STATUS = 'Active'
ORDER BY JSON_VALUE(jsonCol, '$.info.address.PostCode');
```

Graph Models

- Create EDGE and NODE objects
  - For simplicity they look like tables
- Use MATCH to find rows that satisfy some relationship condition
  - Syntax is close to Cypher language (supported by Neo4J)
- Getting started:
  - https://aka.ms/azuresqlgraph

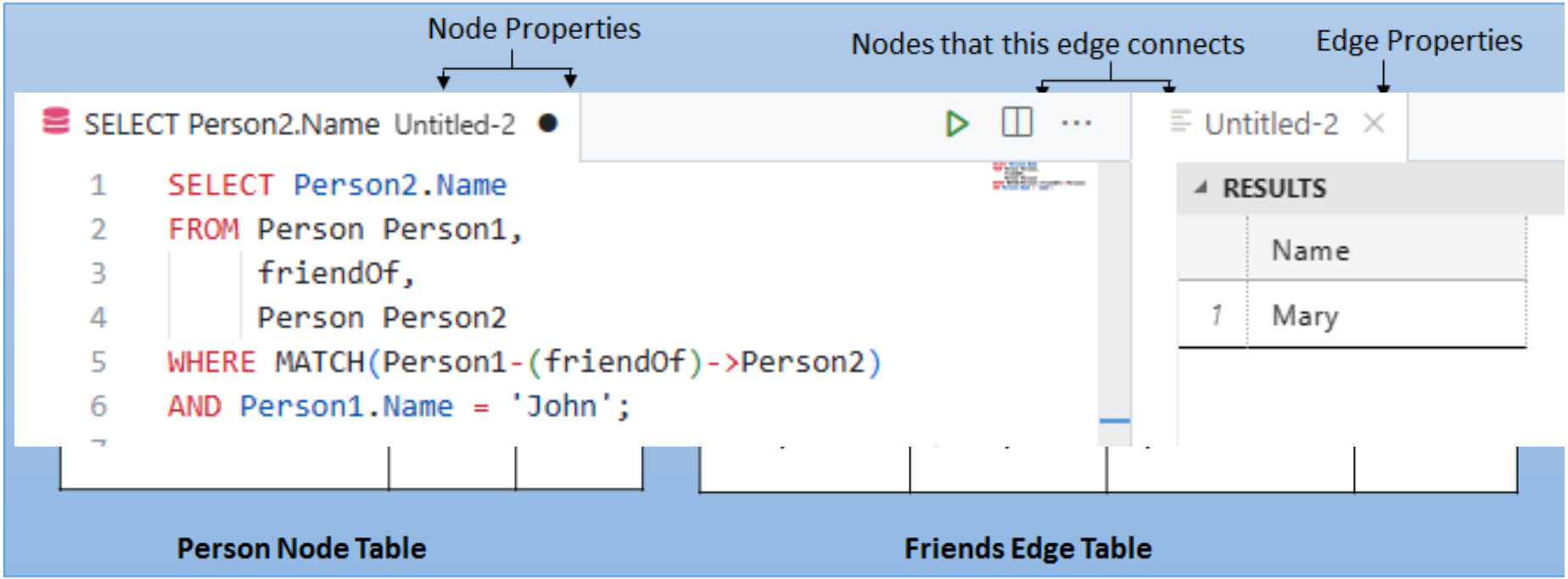


Graph Models

```
CREATE TABLE Person (
      ID INTEGER PRIMARY KEY,
      Name VARCHAR(100),
      Age INT)
AS NODE;

CREATE TABLE Friends (
      StartDate date)
AS EDGE;
```

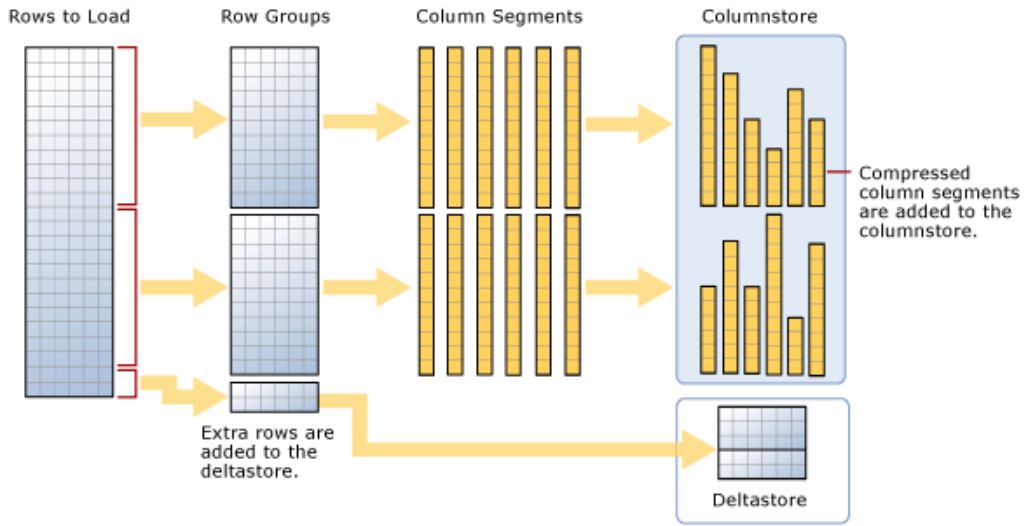
```
--Find friends of John
SELECT Person2.Name
FROM Person Person1,
      Friends,
      Person Person2
WHERE MATCH(Person1-(Friends)->Person2)
AND Person1.Name = 'John';
```

Graph Models



Columnstore

- Store data by columns instead of by rows
- Great for analytical queries
  - Aggregations / Projections
- Columnstore is updatable
  - available from 2016
  - “Deltastore”



https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview

Reasons for Columnstore Indexes

- Columns store values that commonly have similar values, which result in high compression rates.
- High compression rates improve query performance by using a smaller in-memory footprint
- Queries often select only a few columns from a table, which reduces total I/O from the physical media.

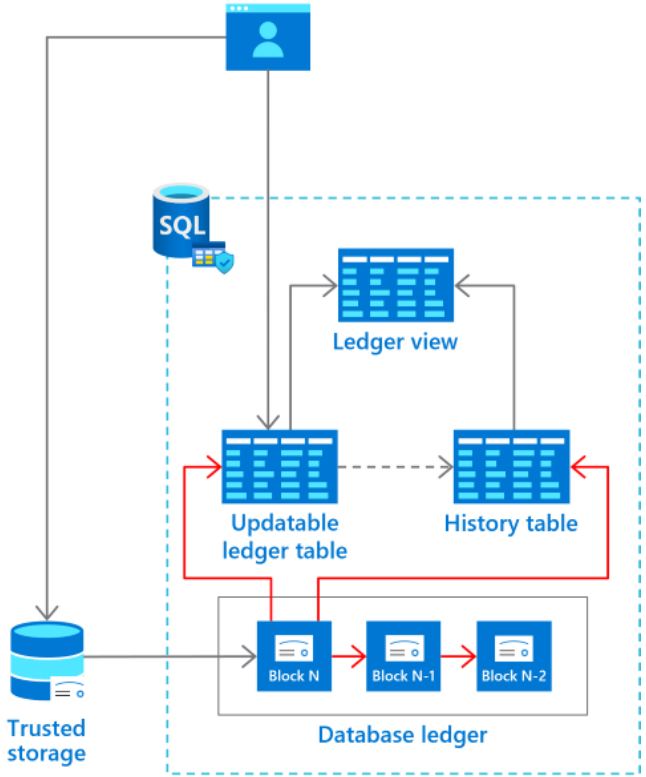
Ledger Tables

Updatable ledger tables

Updatable ledger tables are system-versioned tables on which users can perform updates and deletes while also providing tamper-evidence capabilities.

When updates or deletes occur, all earlier versions of a row are preserved in a secondary table, while the latest version of the row remains in the ledger table.

Ideal for systems/tables where auditing and record accuracy are vital.



Ledger Tables

```
CREATE TABLE [dbo].[HighScores]
(               [ScoreID] INT IDENTITY,
                [PlayerID] INT NOT NULL,
                [FirstName] VARCHAR (50) NOT NULL,
```

CommitTime	UserN…	Sco…	Pla…	Firs…	LastName	Game	Score	Date	Operation
2023-04-20 23:20:49…	bill	2	2	Bill	Mickelson	Horse Monkey	11099913	2023-04-20	INSERT
2023-04-20 23:20:49…	bill	2	2	Bill	Mickelson	Horse Monkey	11099911	2023-04-20	DELETE
2023-04-20 23:20:35…	sqladmin	1	1	Steve	Wobble	Horse Monkey	11099912	2023-04-20	INSERT
2023-04-20 23:20:35…	sqladmin	3	3	John	Hill	Universe In…	218870	2023-04-20	INSERT
2023-04-20 23:20:35…	sqladmin	2	2	Bill	Mickelson	Horse Monkey	11099911	2023-04-20	INSERT

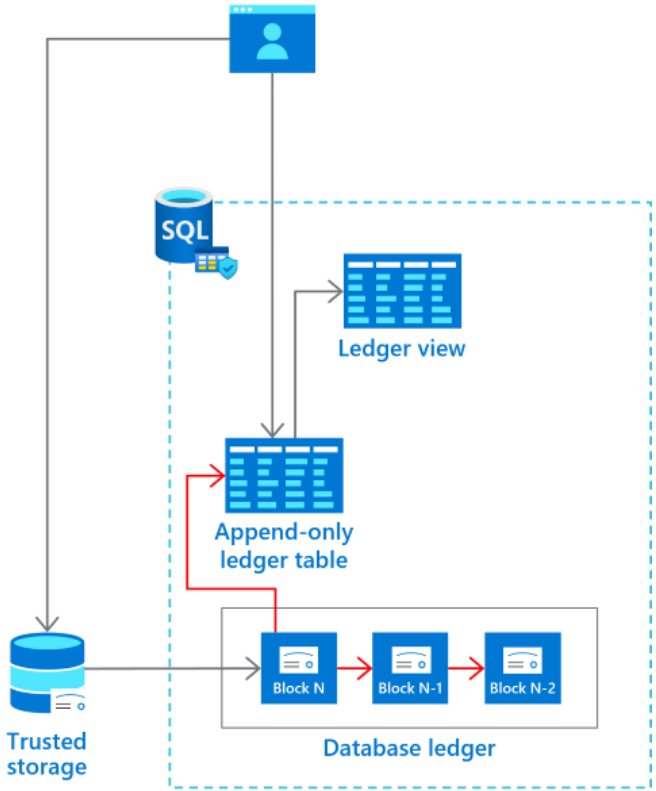
```
LEDGER = ON );

INSERT INTO [dbo].[HighScores]
VALUES      (1, 'Steve', 'Wobble', 'Horse Monkey', 11099912, GETDATE()),
            (2, 'Bill', 'Mickelson', 'Horse Monkey', 11099911, GETDATE()),
            (3, 'John', 'Hill', 'Universe Invaders', 218870, GETDATE());
```

Ledger Tables

Append-only ledger tables

Append-only ledger tables allow only INSERT operations on your tables, which ensure that privileged users such as database administrators can't alter data through traditional Data Manipulation Language operations.

Ideal for security information event and management systems such as badge in/out or ID readers.



Ledger Tables

```
UPDATE [dbo].[ArcadeRoomAccess]
SET Timestamp = GETDATE()
WHERE AccessID = 4;
```

```
Msg 37359, Level 16, State 1, Line 1
Updates are not allowed for the
append only Ledger table 'dbo.ArcadeRoomAccess’.
```

Azure SQL DB Hyperscale

Azure SQL Database Hyperscale provides the flexibility and performance modern applications require:

- Independently scalable compute and storage, without sacrificing performance
- Rapid, auto-scaling storage support up to 100TB
- Low latency, high throughput
- Snapshot-based backups that do not impact query performance
- Available serverless compute and resource-sharing elastic pools

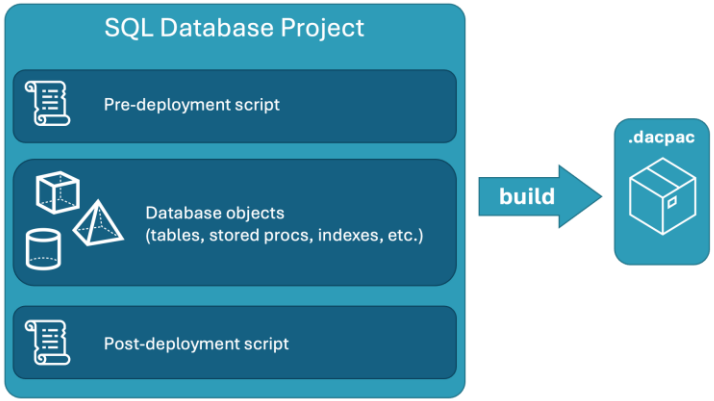
Now with lower pricing – save up to 35%

Chapter 2: Database Tasks

- Create a database SQL project
- Create a SQL 2022 database in the codespace
- Create database objects
- Publish objects to the local SQL 2022 database

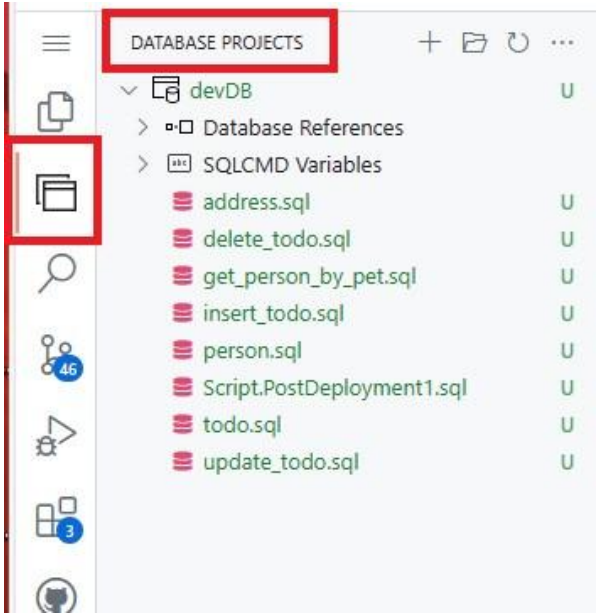
SQL Database Projects

- An Azure Data Studio and Visual Studio Code extension
- For developing SQL databases and objects in a project-based development environment
- Update/Create project from database
- Compare schemas
- Publish all or just changes
- Participate in CI/CD flows
- Deployment time configurations



SQL Database Projects

- `dotnet new sqlproj -n "devDB" -tp "SqlAzureV12“`



Create a SQL Database

- SQLCMD CLI
  - Now written in Go
- Use the go-mssqldb driver
- More Azure Active Directory (Entra ID) authentication options
- New commands
  - sqlcmd create mssql --using https://url.com/backup.bak

Create a SQL Database

- `sqlcmd create mssql -u devDB --accept-eula`
- `sqlcmd config connection-strings`

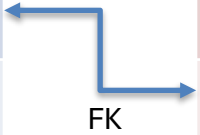
```
@JetterMcTedder ➜/workspaces/azure-sql-db-developers-workshop/database (main) $ sqlcmd create mssql -u devDB --accept-eula
Downloading mcr.microsoft.com/mssql/server:latest
Starting mcr.microsoft.com/mssql/server:latest
Created context "mssql" in "/home/vscode/.sqlcmd/sqlconfig", configuring user account...
Disabled "sa" account (and rotated "sa" password). Creating user "vscode"
Creating default database [devDB]
Now ready for client connections on port 1433

HINT:
  1. Run a query:               sqlcmd query "SELECT @@version"
  2. Start interactive session: sqlcmd query
  3. View sqlcmd configuration: sqlcmd config view
  4. See connection strings:    sqlcmd config connection-strings
  5. Remove:                    sqlcmd delete
```

Create Database Objects

- Tables
  - ID columns
  - Foreign/Primary Keys
- Table APIs - Stored procedures for
  - Insert
  - Update
  - Delete

Person

Name	Data Type
person_id	int (IDENTITY)
person_name	nvarchar(200)
person_email	nvarchar(200)
pet_preference	nvarchar(100)



Address

Name	Data Type
address_id	int (IDENTITY)
person_id	int
address	nvarchar(200)

Todo

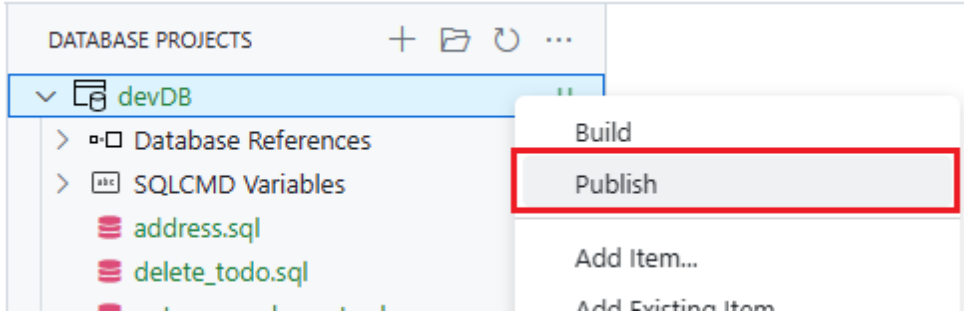
Name	Data Type
id	uniqueidentifier
title	nvarchar(1000)
completed	bit
owner_id	nvarchar(126)
position	int

Products

Name	Data Type
product_id	int
product_name	nvarchar(1000)
ListPrice	money

Publish objects to the local database

- Use SQL Database Projects
  - Publish to the local database created with sqlcmd
- Deploy the tables, stored procedures and run a post deploy script
- Verify the deployment



Testing the Table APIs Reminder

You can also test out the stored procedures with the following code:

```
exec dbo.insert_todo @title = 'My Test Todo', @owner_id = '1001001', @order = 1;
select * from dbo.todo;
GO
```

Copy and paste the resulting ID from the result set into the area marked **COPIED ID FROM RESULT SET** and use it in the next stored procedure example:

```
exec dbo.update_todo @id = 'COPIED ID FROM RESULT SET', @title = 'zzzz', @owner_id =
'1001001';
select * from dbo.todo;
GO
```

And again, copy and paste the resulting ID from the result set into the area marked **COPIED ID FROM RESULT SET** and use it in the next stored procedure example:

```
exec dbo.delete_todo @id = 'COPIED ID FROM RESULT SET', @owner_id = '1001001';
select * from dbo.todo;
GO
```

Knowledge Check

Which of the following can be used in Azure SQL Database

A. JSON

B. XML

C. Geospatial

D. Graph

E. All of the above

I have an application that logs cars entering and exiting a secured area, which ledger table should I use?

A. Updatable

B. Insert Only

C. Append Only

D. Delete Updatable

E. A, B, and D

Knowledge Check

Are SQL Database Projects good for CICD?

A. Yes

B. No

Chapter 2: Database Tasks

- Create a database SQL project
- Create a SQL 2022 database in the codespace
- Create database objects
- Publish objects to the local SQL 2022 database

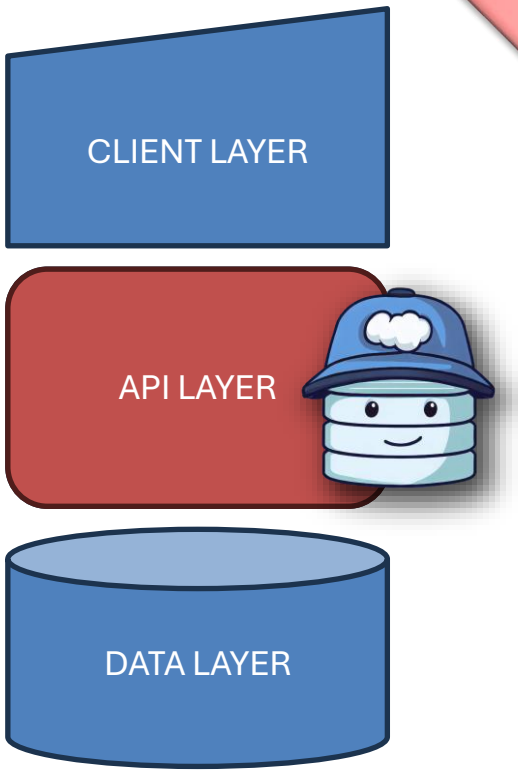
START CHAPTER 2

CHAPTER 3:
DATA API BUILDER

Data API builder

public preview

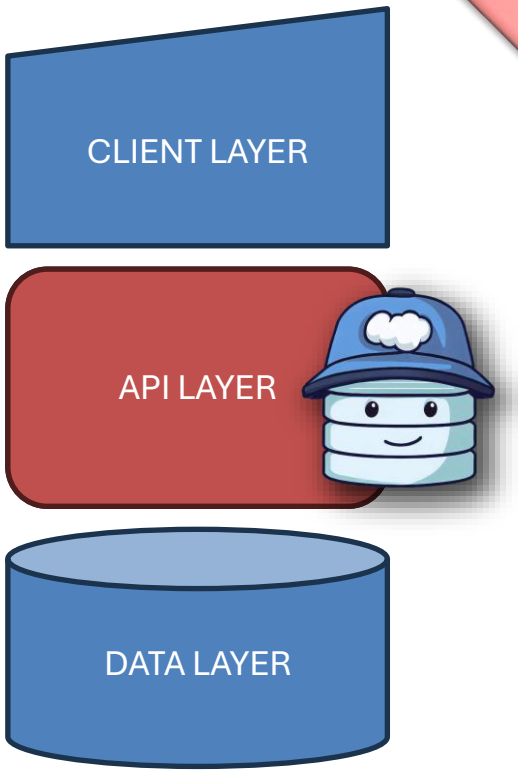
- Open Source and Free (https://aka.ms/dab)
- Container-based
- Zero code
  - Configuration-driven
- Supported authentication:
  - EasyAuth, EntraID/AAD, JWT tokens
- Supported endpoints:
  - Standard REST API and Standard GraphQL
- Supported databases:
  - Azure SQL DB & SQL Server
  - Cosmos DB (document)
  - PostgreSQL
  - MySQL



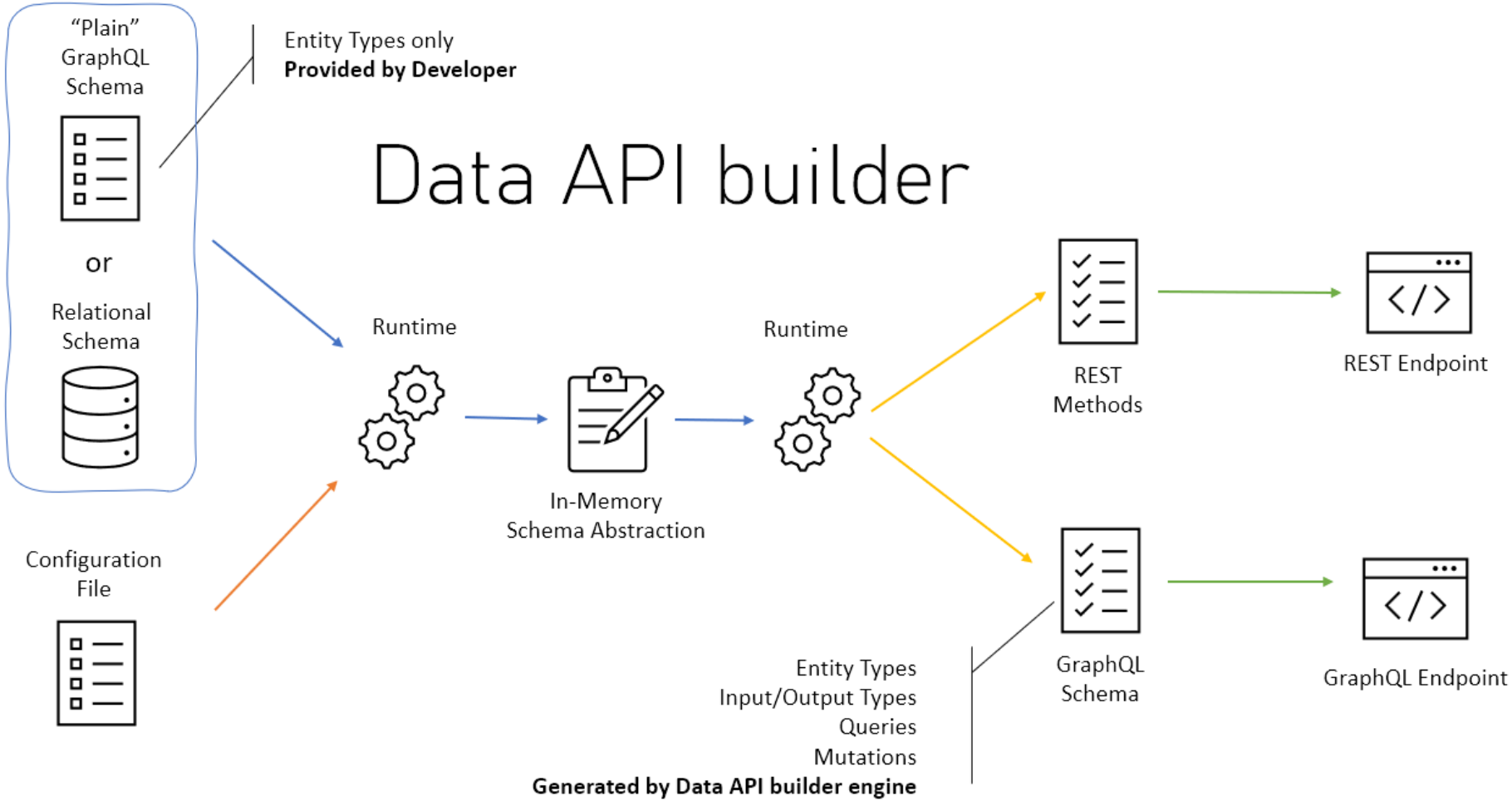
Data API builder

public preview

- Built in developer tools and clients
  - OpenAPI, Swagger, Banana Cake Pop
  - DAB CLI
- Database Objects Supported
  - Tables
  - Views
  - Stored Procedures
- GraphQL Relationships
  - Foreign Keys
  - One-to-Many/Many-to-One/Many-to-Many
- Authorization
  - Database Context
  - Role-based authorization using received claims
  - Item-level security via policy expressions



Data API builder



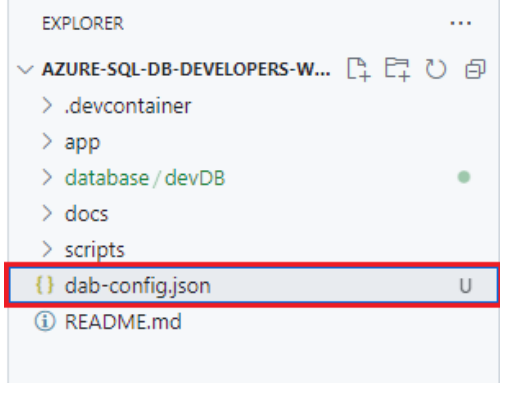
Chapter 3: Data API Builder

- Setup Data API Builder
- Create REST/GraphQL entities
- Test the endpoints

Setup Data API Builder

Create the Data API Builder initialization file

```
dab init --database-type "mssql" --connection-string
"Server=localhost;Database=devDB;User ID=vscode;Password=
"'XXXXXXXX'";TrustServerCertificate=true" --host-mode
"Development" --rest.path "rest" --set-session-context
true
```

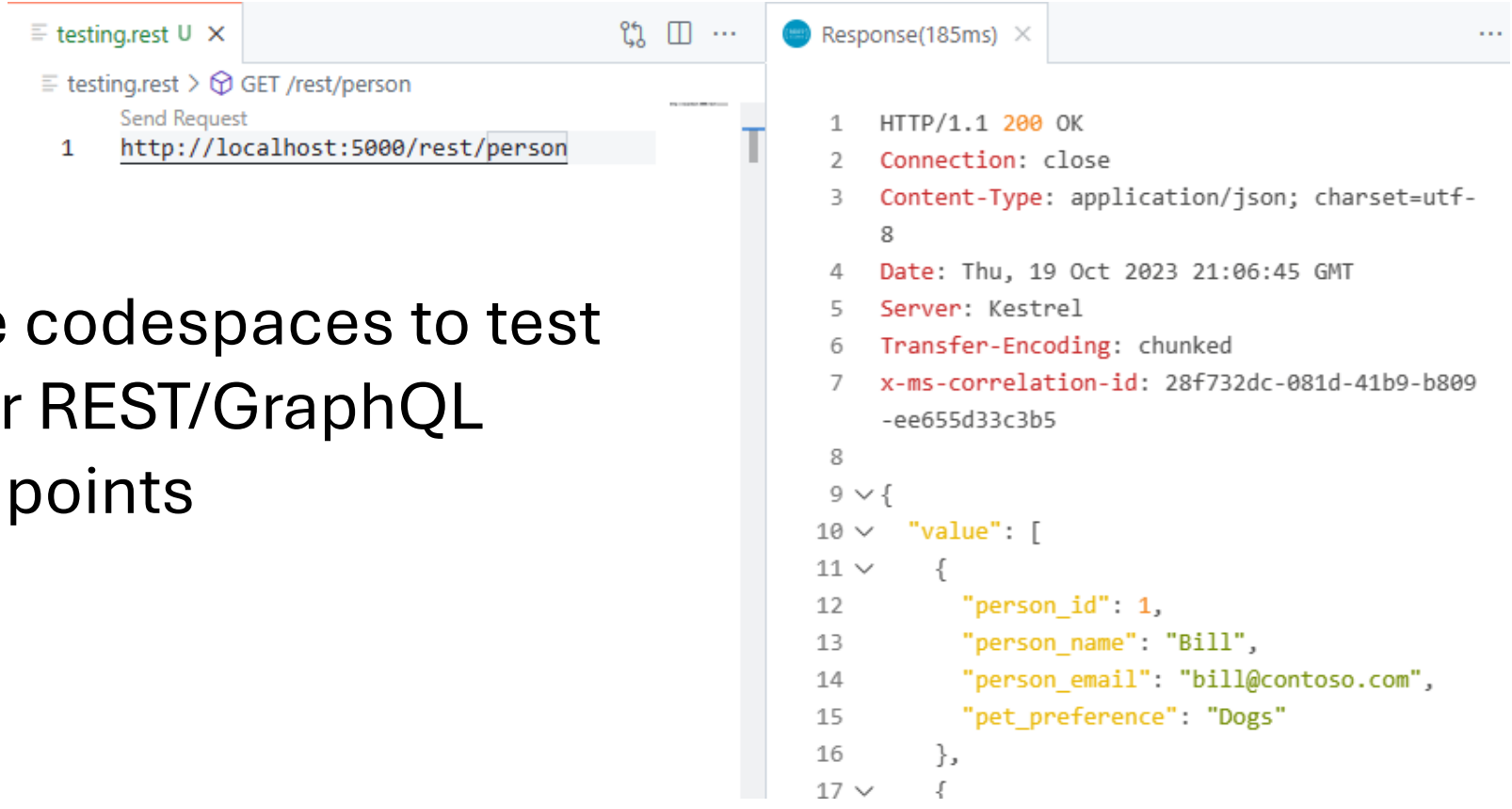


Create REST/GraphQL entities

Use DAB CLI to add the tables, stored procedures, and relationships to the dab-config.json file

- **dab add person** --source dbo.person --permissions "anonymous:*"
- **dab update person** --relationship "address" --cardinality "many" --target.entity "address"
- **dab add getPersonByPet** --source dbo.get_person_by_pet --source.type "stored-procedure" --source.params "pet:" --permissions "anonymous:execute" --rest.methods "get" --graphql.operation "query"

Test the endpoints



Use codespaces to test your REST/GraphQL endpoints

Knowledge Check

Data API builder lets you REST/GraphQL enable?

A. Tables

B. Views

C. Stored Procedures

D. A and B

E. A, B, and C

On average, how many lines of code do I need to write to GraphQL enable a database object with Data API Builder

A. 77

B. 4000

C. None

D. 4 or 5

Chapter 3: Data API Builder

- Setup Data API Builder
- Create REST/GraphQL entities
- Test the endpoints

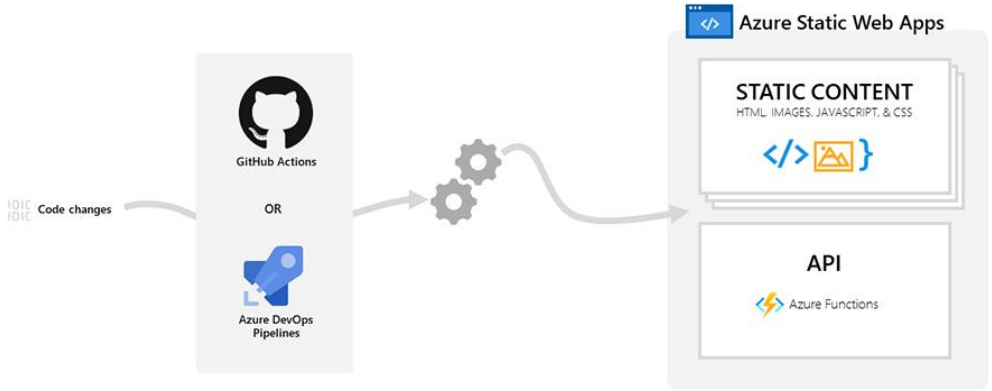

START CHAPTER 3

CHAPTER 4: AZURE STATIC WEB APPS

Azure Static Web Apps

Automatically build and deploy full stack web apps to Azure from a code repository.

- **Build modern web applications** with JavaScript frameworks and libraries like Angular, React, Svelte, Vue, or using Blazor to create WebAssembly applications, with an Azure Functions back-end.
- **Publish static sites** with frameworks like Gatsby, Hugo, VuePress.
- **Deploy web applications** with frameworks like Next.js and Nuxt.js.



Key Features

- **Web hosting** for static content like HTML, CSS, JavaScript, and images.
- **Integrated API** support provided by managed Azure Functions, with the option to link an existing function app, web app, container app, or API Management instance using a standard account. If you need your API in a region that doesn't support managed functions, you can bring your own functions to your app.
- **First-class GitHub and Azure DevOps integration** that allows repository changes to trigger builds and deployments.
- **Globally distributed** static content, putting content closer to your users.
- **Free SSL certificates**, which are automatically renewed.
- **Custom domains** to provide branded customizations to your app.

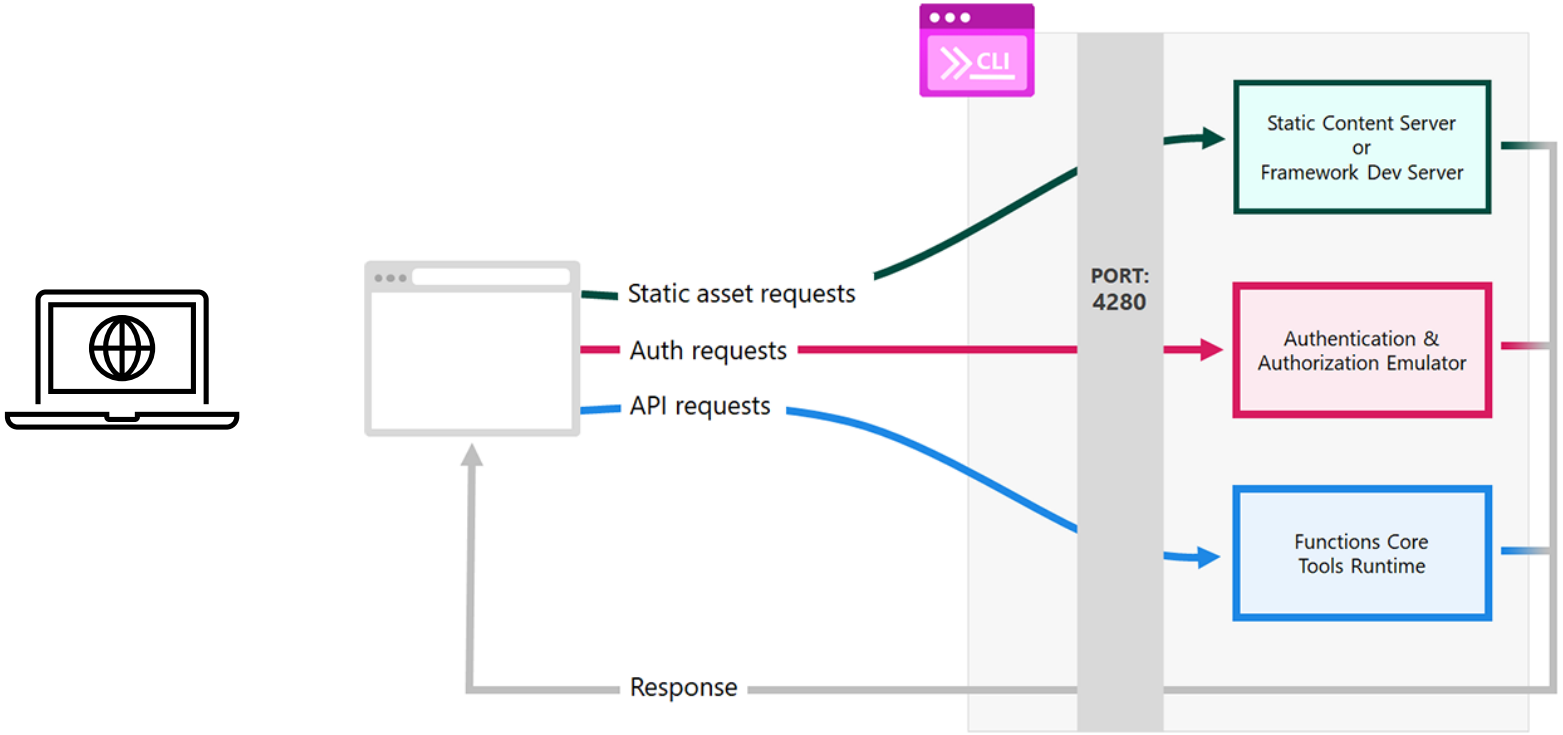
Key Features

- **Seamless security model** with a reverse-proxy when calling APIs, which requires no CORS configuration.
- **Authentication provider integrations** with Microsoft Entra ID and GitHub.
- **Customizable authorization role definition** and assignments.
- **Back-end routing rules** enabling full control over the content and routes you serve.
- **Generated staging versions** powered by pull requests enabling preview versions of your site before publishing.
- **CLI support** through the Azure CLI to create cloud resources, and via the Azure Static Web Apps CLI for local development.

Azure Static Web Apps CLI

- Serve static app assets, or proxy to your app dev server
- Serve API requests, or proxy to APIs running in Azure Functions Core Tools
- Emulate authentication and authorization
- Emulate Static Web Apps configuration, including routing and ACL roles
- Deploy your app to Azure Static Web Apps

Azure Static Web Apps CLI



Chapter 4: SWA

- Initialize SWA
- Test the web applications
- Use SWA authentication with Data API builder

But First....

Creating a mapping in Data API Builder

```
dab update todo --map "position:order"
```

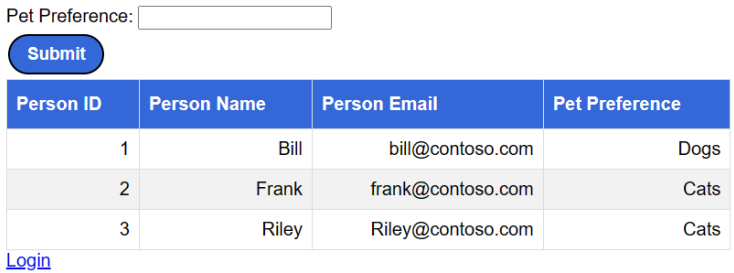
To map a UI element called order to the position column (in dab-config.json)

```
"mappings": {
  "position": "order"
}
```

Initialize SWA and Test

Create the connections file and start with the sample JavaScript application

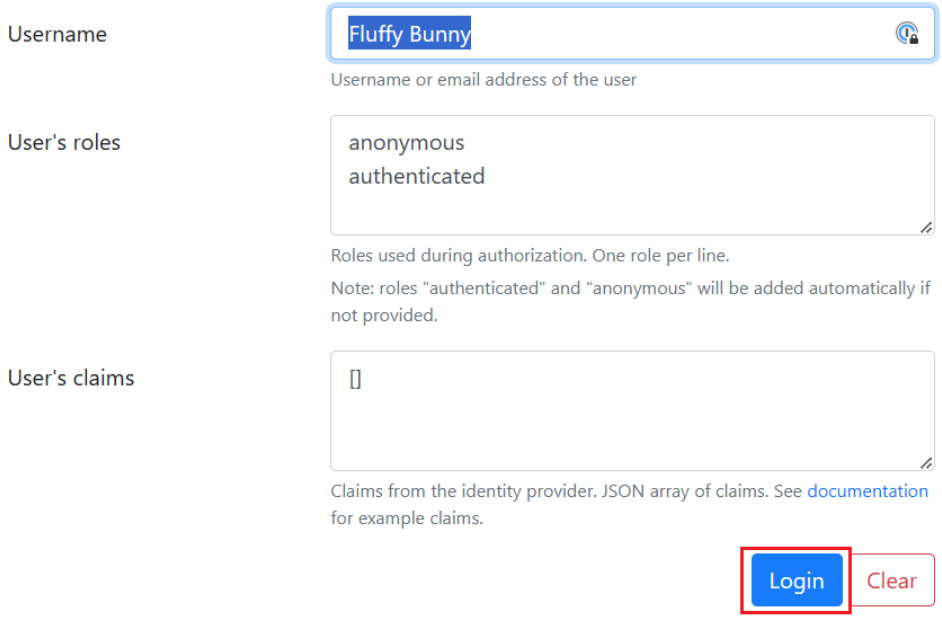
1) `cp dab-config.json ./swa-db-connections/staticwebapp.database.config.json`
2) `swa start --app-location ./app --data-api-location ./swa-db-connections`



Use SWA authentication with Data API builder

In the permissions section, change the role from **anonymous** to **authenticated**

```
"permissions": [
   {
     "role": "authenticated",
     "actions": [
       "execute"
     ]
   }
 ],
```



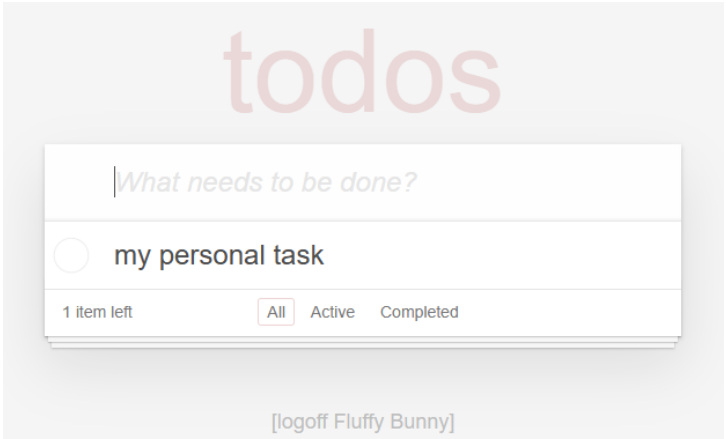
Working with the ToDo Application

- Change the `swa-cli.config.json`
- Build the app (`swa build`)
- Start swa and test the ToDo application

```
swa start --data-api-location ./swa-db-connections
```

Working with the ToDo Application

In the permissions section, change replace the permissions section for the Todo table with the following code:




```
"permissions": [
   {
     "role": "anonymous",
     "actions": [
       {
         "action": "*",
         "policy": {
           "database": "@item.owner_id eq 'public'"
         }
       }
     ]
   },
   {
     "role": "authenticated",
     "actions": [
       {
         "action": "*",
         "policy": {
           "database": "@item.owner_id eq @claims.userId"
         }
       }
     ]
   }
 ],
```

Working with the ToDo Application

Move from the REST endpoints to stored procedures

1. `mv client/src/components/ToDoList.vue labFiles/ToDoList.vue.DAB1`

2. `mv labFiles/ToDoList.vue.SP client/src/components/ToDoList.vue`

3. `swa start --data-api-location ./swa-db-connections`

Knowledge Check

Static Web Apps is a cloud only service/tool?

A. True

B. False

A major issue with using Static Web Apps is that I have to deploy an instance in each region I want to use?

A. True

B. False

Chapter 4: SWA

- Initialize SWA
- Test the web applications
- Use SWA authentication with Data API builder

START CHAPTER 4

Announcing…

Azure Database Fleet Manager - Private Preview

- New service designed to help SaaS solution builders create multi-tenant database tiers.
  - Offers a simple and intuitive API to provision and manage tenant databases, seamlessly scaling from few to millions.
  - Tenants are organized in tiers. A tier represents a grouping of settings the platform enforces on all databases in a tier.
  - Fleet Manager dynamically load balances resource utilization for databases in a tier, optimizing costs and performance for all tenants.
- Available with Azure SQL Database Elastic Pools and single databases now, more options will follow.
- More details: https://aka.ms/dbfleetmanager-prpr

CHAPTER 5:
DEPLOY TO AZURE SQL DATABASE

Azure SQL Database free of charge

Get 100,000 vCore seconds of serverless compute and 32 GB of storage every month!

Azure SQL Database with serverless compute

Flexible compute automatically scales to meet demand.

No time limits

Apply this free offer for the life of your subscription.

Need more? No problem.

Stick with the default auto-pause option or continue usage for additional charges.

What's included:

One Azure SQL Database with serverless compute per Azure Subscription with 100,000 vCore seconds every month.

32 GB data storage +

32 GB backup storage.

Learn More: aka.ms/sqlfreeoffer

Want to try Azure SQL Database for free? Create a free serverless database with the first 100,000 vCore seconds, 32GB of data, and 32GB of backup storage free per month for the lifetime of the subscription. Learn more

Apply offer (Preview)

Extra Credit

AZURE DATABASE MIGRATION SERVICE

Azure Database Migration Service

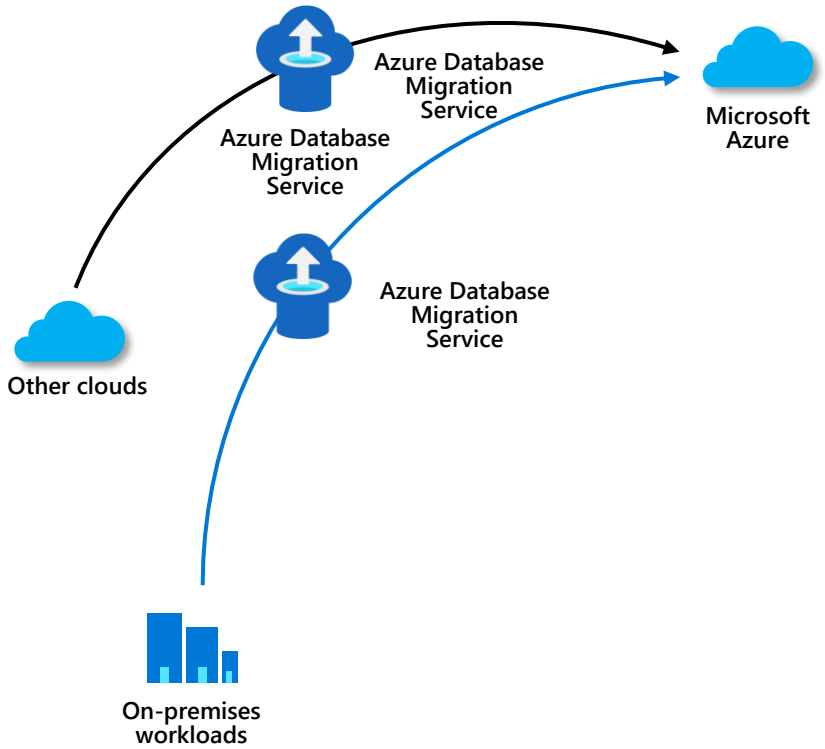
Migrate your databases to Azure at scale

A fully managed service designed to streamline the migration of SQL Server databases, whether running on-premises or in other cloud environments

Highly resilient and self-healing migration experience, with near-zero downtime

Seamlessly migrate to Azure using familiar tools like the Azure Portal and Azure Data Studio.

Automate with ease using PowerShell and Azure CLI



On-premises

Microsoft Azure

Azure Database Migration Service

Assessment

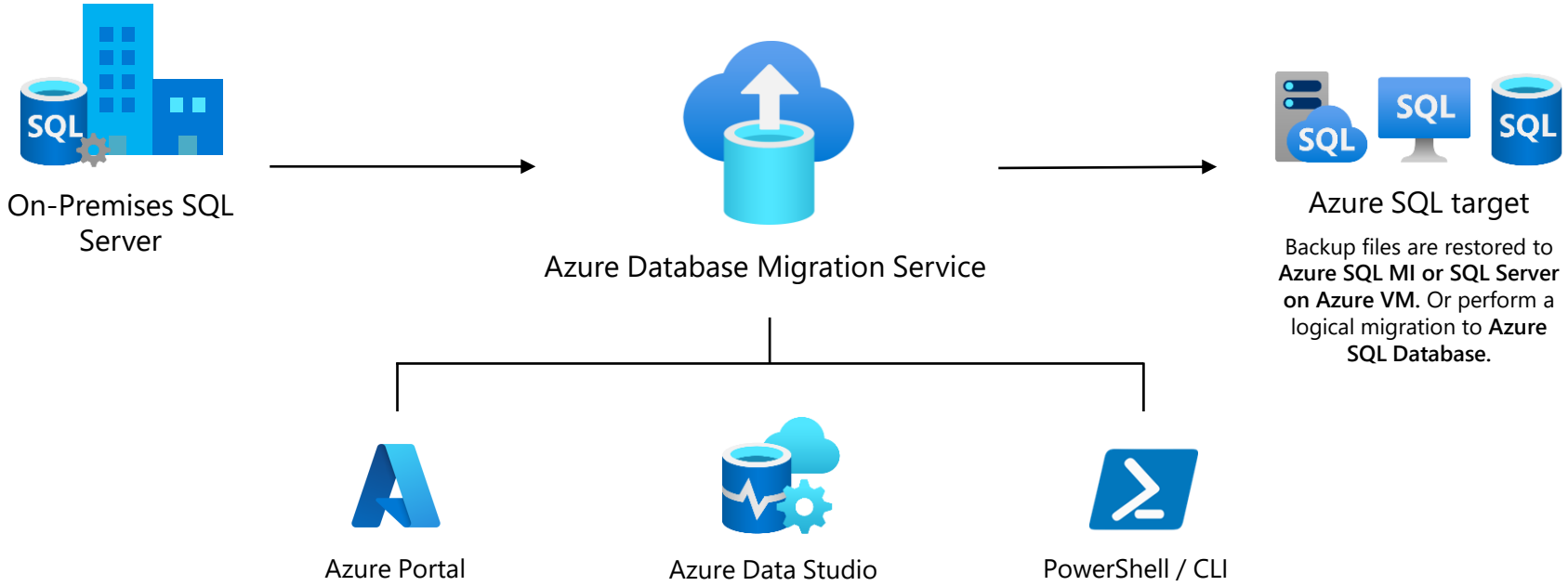
Azure DMS
(ADS, Azure portal, PowerShell, CLI)
Azure PostgreSQL Migration (ADS)
Azure Migrate

Migration

Azure Database
Migration Service

Azure Database Migration Service

Migration tools

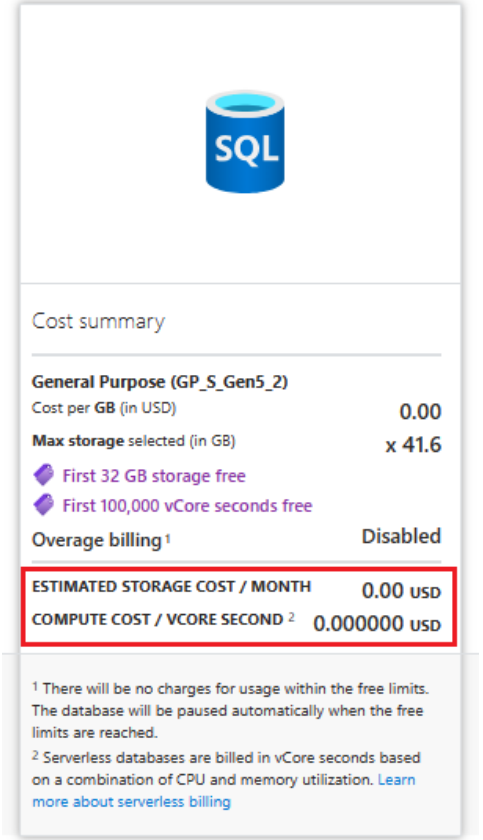


Chapter 5: Deploy to Azure

- Create a Free Azure SQL Database
- Connect to the Azure SQL Database
- Publish objects to the Azure SQL Database

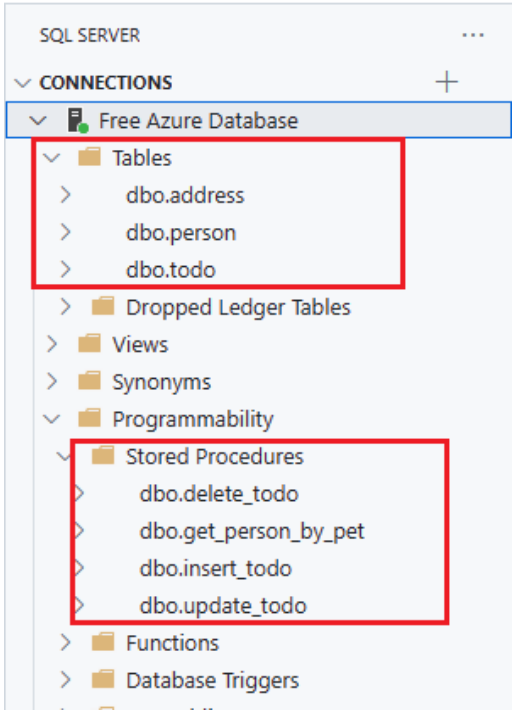
Create a Free Azure SQL Database

- Login to the Azure Portal
- Create a new SQL Database
- Apply the promotion
- Create!



Connect and Publish to the Azure SQL Database

- Create a connection in codespace
- Deploy the database project to the free Azure SQL Database
- Verify the deployment and post-install script



Knowledge Check

Azure SQL Database is expensive as a development platform?

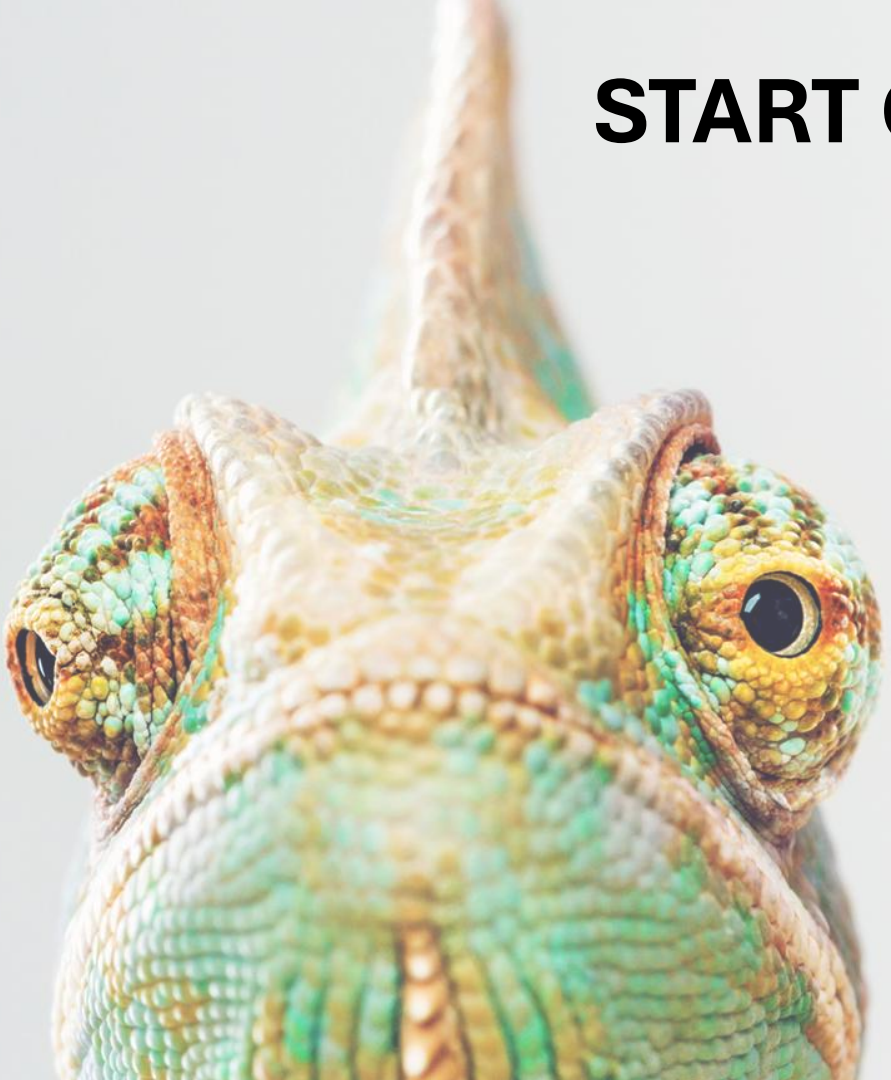
A. True

B. False

Chapter 5: Deploy to Azure

- Create a Free Azure SQL Database
- Connect to the Azure SQL Database
- Publish objects to the Azure SQL Database

START CHAPTER 5



CHAPTER 6: INVOKE REST ENDPOINTS

Services Integration

I really want to use other Azure products and services from within my Azure SQL Database without having to write complex code, functions, or scheduled events. Is this even possible?

External REST Endpoint Invocation

Using a single stored procedure, call REST/GraphQL service endpoints in Azure directly from within an Azure SQL Database.

Use Cases

- Activate workflows
- Data enrichment
- Cache invalidation / update
- Augment business/process logic
- Update websites
- Integrate with event-based architectures
- Data streams

Security Included

In the Database

- Requires EXECUTE ANY EXTERNAL ENDPOINT database permission.

Endpoint Authentication

- Header
- Query String/URL
- Managed Identity

Easy to use

One simple and easy call to a REST endpoint directly from the Database

```
EXEC @returnValue = sp_invoke_external_rest_endpoint
    [ @url = ] N'url'
    [ , [ @payload = ] N'request_payload' ]
    [ , [ @headers = ] N'http_headers_as_json_array' ]
    [ , [ @method = ] 'GET' | 'POST' | 'PUT' | 'PATCH' | 'DELETE' | 'HEAD' ]
    [ , [ @timeout = ] seconds ]
    [ , [ @credential = ] credential ]
    [ , @response OUTPUT ]
```

Chapter 6: Invoke REST endpoints

- Invoke an Azure Function endpoint
- Invoke an OpenAI endpoint
- Add translation logic to the todo application with OpenAI

Invoke an Azure Function endpoint

- Use a function to augment/enhance data
- Get the conversion rate for currency X and see what the price is
- Call endpoint and extract rate from JSON into a sql variable

```
set @priceConversion = (select JSON_VALUE(
@response,'$.result.priceConversion') AS priceConversion);


select product_id, product_name, ListPrice,
cast(round(ListPrice*@priceConversion,2,1) as money) AS
convertedPriceInYen from dbo.products
```

Invoke an OpenAI endpoint

Use AI to get generated content from data in the database

```
set @adcopy =
(select 'You are an experienced marketing expert named Don Chase
Katz. Generate 200 letters of ad copy to '
+ person_name +
' to convince them to love ' +
case when pet_preference = 'Cats' then 'Dogs'
     when pet_preference = 'Dogs' then 'Cats'
end
from person
where person_id = 1);
```

Point SWA at the Azure SQL Database

Change the connection string in the **staticwebapp.database.config.json** file to point to the Azure SQL Database

staticwebapp.database.config.json 3, U

swa-db-connections > staticwebapp.database.config.json > data-source

```
{
  "$schema": "https://github.com/Azure/data-api-builder/releases/download/v0.8.52/dab.draft.schema.json",
  "data-source": {
    "database-type": "mssql",
    "connection-string": "Server=freedbsqlserver.database.windows.net;Database=freeDB;User ID=sqladmin;Password=PASSWORD;TrustServerCertificate=true",
    "options": {
      "set-session-context": false
    }
  },
```

Add translations with OpenAI

Alter the insert_todo stored procedure to call OpenAI to translate the text of a ToDo before an insert is done

```
declare @payload nvarchar(max) =
N'{"messages":[{"role":"system","content":"Translate
\"'+(@title)+'\" into german, only respond with the
translation"}]}'
```

Knowledge Check

When calling REST endpoints from the database, the authentication method(s) not available is?

A. Managed Identity

B. Mangled Identity

C. Header

D. Query String/URL

E. B and D

Chapter 6: Invoke REST endpoints

- Invoke an Azure Function endpoint
- Invoke an OpenAI endpoint
- Add translation logic to the todo application with OpenAI


START CHAPTER 6

CHAPTER 7: AZURE SQL BINDINGS

Azure Functions Core Tools

- Develop Azure Functions locally using Core Tools
- Deploy your code project to Azure
  - Function, App Container,
- Work with app settings locally (local.settings.json)

Change Detection & Change Stream

How can I leverage the change tracking features in Azure SQL Database (or SQL Server/MI/VM) to expose data changes via an outgoing change stream without the need for custom code and scheduler functions?

Azure SQL bindings for Azure Functions

- **Azure SQL**
  - encompasses all Azure SQL products (DB, MI, VM) + SQL Server with cloud connectivity
- **Azure Functions**
  - serverless runtime for standalone use or integrated with Azure Static Web Apps
- **Popular Languages**
  - C# (in proc and out of proc), JavaScript, Python, Java, and PowerShell

Three Azure SQL bindings for Azure Functions

Input binding

object in function

SQL query results

Azure SQL Input bindings take a SQL query or stored procedure to run and returns the output to the function.

Output binding

object in function

SQL table

Azure SQL Output bindings take a list of rows and upserts them to the user table.

SQL trigger

SQL data change

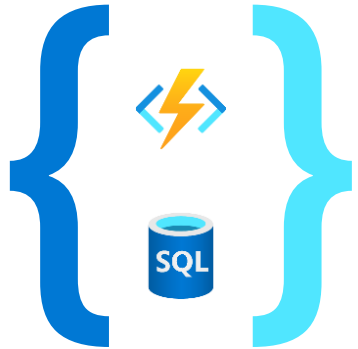
starts function

The Azure SQL trigger uses SQL change tracking functionality to monitor a SQL table for changes

Azure SQL bindings for Azure Functions

Use Cases

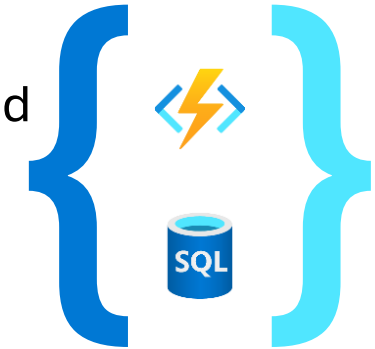
- **Data enrichment**
  - Determine if a value is an outlier or not using Azure Cognitive services/OpenAI
  - Perform reverse geocoding using Azure Functions
  - Call a REST/GraphQL service from a function with data from Azure SQL
- **Start complex processing**
  - Call a Function to kick of some complex process and return the data



Azure SQL bindings for Azure Functions

Use Cases

- **Update websites**
  - Broadcast a SignalR message
- **Integrate with event-based architectures**
  - Send data to Event Hubs for further integration options
- **Create a change data stream**
  - Send data to Stream Analytics for further investigation/fraud detection



SQL bindings + Azure SQL compatibility

- The bindings use the OPENJSON statement which requires a **database compatibility level of 130 or higher (2016 or higher)**.
- Databases on **SQL Server, Azure SQL Database, or Azure SQL Managed Instance** which meet the compatibility level requirement above are supported.

Connecting to the database

- SQL bindings connect to the target database by using **a Connection String configured in the app settings**. This will require a login be created that the function will use to access the server.
- For local testing and development using a SQL (username/password) or Azure Active Directory Login is typically the easiest, but for deployed function apps it is **recommended to use Azure Active Directory Managed Authentication**.

SQL trigger app settings

- **Sql_Trigger_BatchSize**: controls the number of changes processed at once before being sent to the triggered function.

- **Sql_Trigger_PollingIntervalMs**: controls the delay in milliseconds between processing each batch of changes.

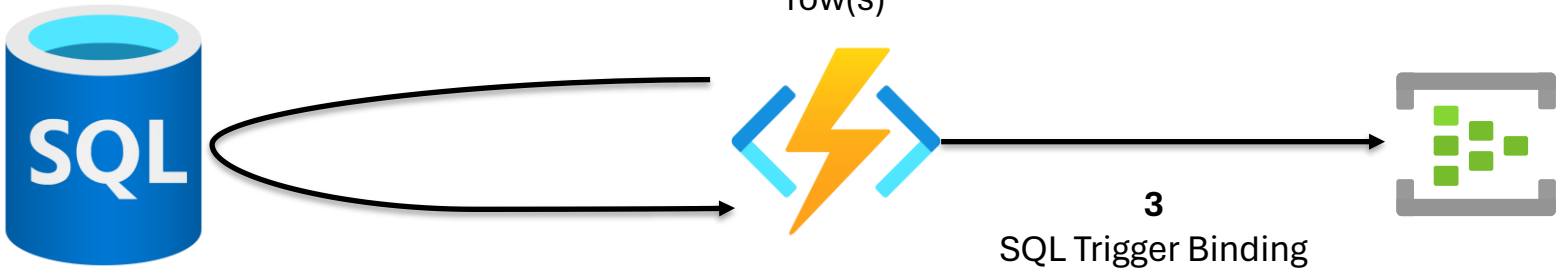
- **Sql_Trigger_MaxChangesPerWorker**: controls the upper limit on the number of pending changes in the user table that are allowed per application-worker. If the count of changes exceeds this limit, it may result in a scale out. (only applies for Azure Function Apps with runtime driven scaling enabled)

Change stream scenario

Requirement is to capture changes from a table and put them onto an Azure Event Hub for down stream subscribers (microservices).

Also, I don’t want to write a lot of code.

Change stream architecture



2
SQL Trigger Binding sees change and gets row(s)

1
Update on Products table

```
UPDATE products
SET product_price = 3.99
  WHERE product_id = 2;
```

3
SQL Trigger Binding sends row(s) over to Azure Event Hub

Chapter 7: Azure SQL Bindings

- Create a function project
- Add a SQL trigger binding
- Run the function and test on the Azure SQL Database

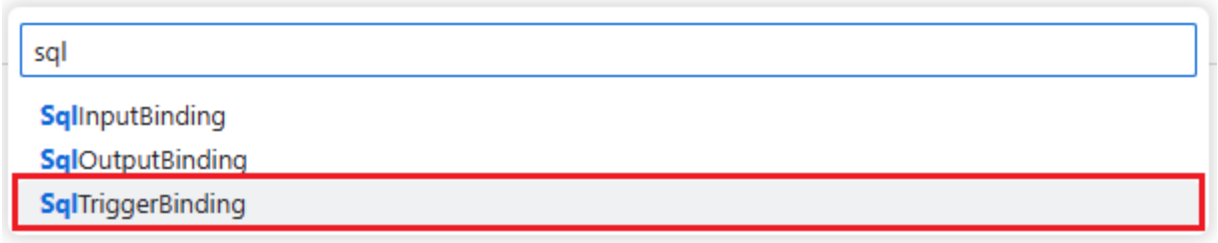
Create a function project

- Create a .NET function using core tools

```
func init triggerBinding --worker-runtime dotnet
```

Add a SQL trigger binding

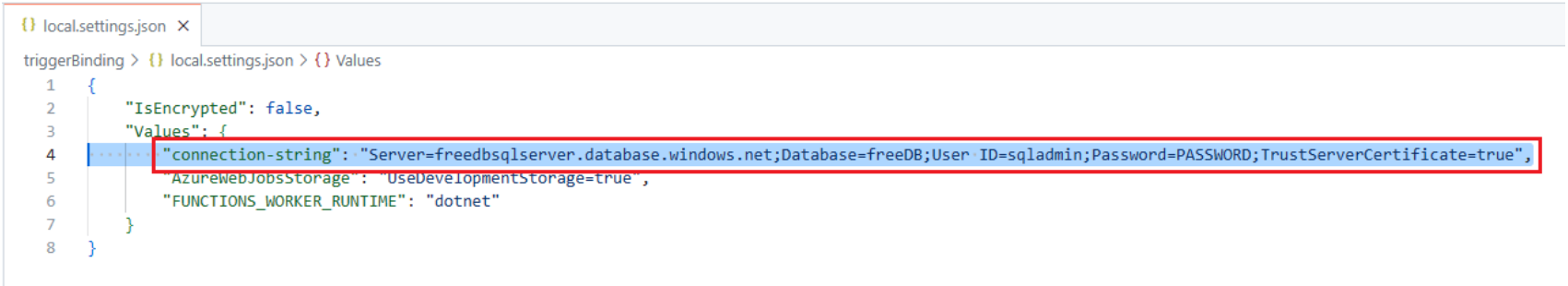
- Use the Create Function wizard to create the SQL Trigger Binding
  - can also be used to create input and output bindings



Run and Test the Function

- Use core tools to run locally
- Connect to a local or cloud SQL Database

```
func host start
```



Knowledge Check

Functions can only be developed in the cloud?

A. True

B. False

Azure SQL bindings for Azure Functions can only be used with Azure SQL Database?

A. True

B. False

Chapter 7: Azure SQL Bindings

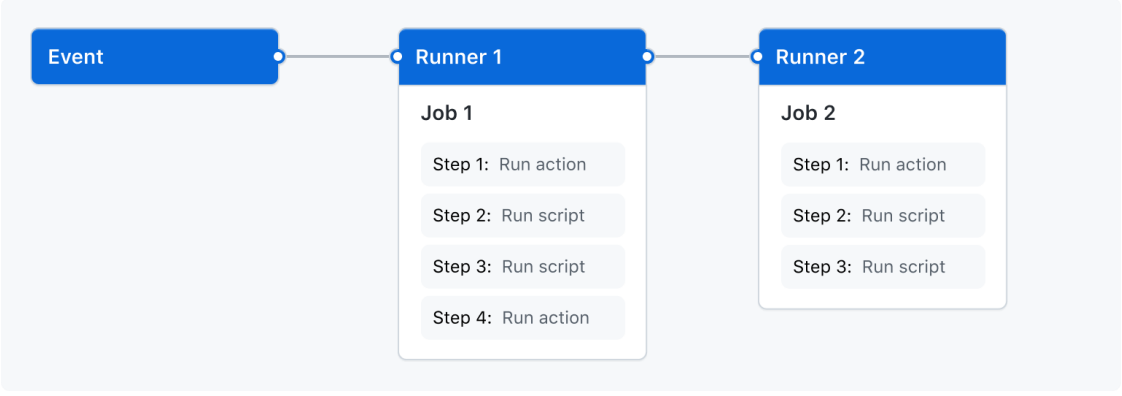
- Create a function project
- Add a SQL trigger binding
- Run the function and test on the Azure SQL Database


START CHAPTER 7

CHAPTER 8:
CI/CD WITH GITHUB ACTIONS

GitHub Actions

- Are a continuous integration and continuous delivery (CI/CD) platform
- Allows you to run workflows based on repository events
- Test or deploy code with workflows

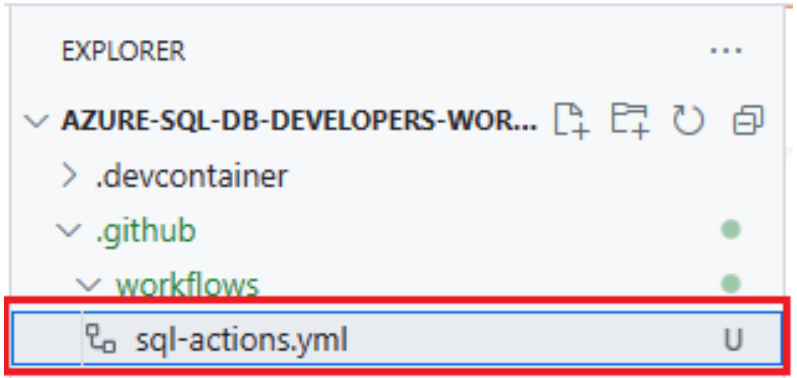


Chapter 8: CI/CD with GitHub Actions

- Create a workflow folder and actions file
- Use sql-actions for DB creation and testing
- Push changes to start the action

Create a workflow folder

- GitHub actions are in the .github/workflows folder
- Can contain multiple action workflows



Use sql-actions

- Sql-actions are SQL specific activities that can be run in a workflow
- Publish (.dacpac) and run scripts on database
- Integrated with GitHub Secrets

Push to Start

- Push code back to your fork to start the workflow
- Workflow will:
  - Create a SQL 2022 DB in docker
  - Create a test DB and set server parameters
  - Deploy the database project to the test database
  - Log all steps for review
  - Automatically deletes environment when done

sql-actions.yml

```yaml
# .github/workflows/sql-actions.yml

name: SQL Server container in deployment
test pipeline
on: [push]
```

Name of the workflow

Runs on a git push

sql-actions.yml

```yaml
jobs:
  build-and-deploy:
    # Containers must run in Linux based operating
systems
    runs-on: ubuntu-latest

    # service/sidecar container for azure-sql-2022
    services:
      mssql:
        image: mcr.microsoft.com/mssql/server:2022-
latest
        env:
          ACCEPT_EULA: 1
          SA_PASSWORD: XXXXXXXX
        ports:
          - 1433:1433
```

Groups together all the jobs that run in the workflow.

OS the container is going to run in

A docker container that will run SQL 2022

sql-actions.yml

```
    steps:
      - name: 'Checkout GitHub Action'
        uses: actions/checkout@v4

      - name: 'wait for sql to be ready'
        run: |
          set +o pipefail +e
          for i in {1..60};
          do
              sqlcmd -S localhost -U sa -P XXXXXXXX -d master -Q
"select getdate()"
              if [ $? -eq 0 ]
              then
                  echo "sql server ready"
                  break
              else
                  echo "not ready yet..."
                  sleep 1
              fi
          done
          set -o pipefail -e
```

Check out the repository code

Wait for the database to be up and running

sql-actions.yml

```
      - name: 'Create and setup database'
        uses: azure/sql-action@v2
        with:
          connection-string: "Server=localhost;Initial
Catalog=master;User ID=sa;Password=
XXXXXXXX;Encrypt=False;TrustServerCertificate=False;"  # the local
connection string
          path: './labFiles/setupDatabase.sql' # the sql script to
create db and configure for clr

      - name: 'Deploy Database Project'
        uses: azure/sql-action@v2.2
        with:
          connection-string: "Server=localhost;Initial
Catalog=testingDB;User ID=sa;Password=
XXXXXXXX;Encrypt=False;TrustServerCertificate=False;"  # the local
connection string
          path: './database/devDB/devDB.sqlproj' # the SQLproj file
          action: 'Publish'
```

Once the DB is ready, run the following script.
(creates testDB and run env config options)

Deploy the database project we created to the testDB

Knowledge Check

If I use a developer edition of SQL Server 2022 in my GitHub action, it will cost me?

A. $100,000,000

B. $15

C. $0

D. The cost of a cup of coffee a day for 30 days

Chapter 8: CI/CD with GitHub Actions

- Create a workflow folder and actions file
- Use sql-actions for DB creation and testing
- Push changes to start the action

START CHAPTER 8

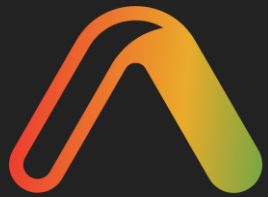

THANK YOU!

Session evaluation

Your feedback is important to us

Evaluate this session at:

www.PASSDataCommunitySummit.com/evaluation



PASS
Data Community
SUMMIT
2021