Testing! We're Talking About (Unit) Testing

Dan Mallott

THANKS TO ALL OUR SPONSORS!

BizStream

Atomic Object

THE C2 GROUP

KL&A
KUNZ, LEIGH AND ASSOCIATES

MICHIGAN SOFTWARE LABS

WEST MICHIGAN TECH TALENT

TEKTON

KFORCE

Leading EDJE

Mutually human

Scout APM

Sweetwater
Music Instruments & Pro Audio

DAN MALLOTT

- Senior Principal in Product Engineering at West Monroe
- Based in Chicago, USA
- > 10 years experience as a DBA and Developer
- Mostly Microsoft technologies (.NET, SQL Server, Azure)
- Ice Hockey Official

What is Unit Testing?

In computer programming, unit testing is a software testing method by which individual unit of source code – sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures – are tested to determine whether they are fit for use.

A Unit Test is an automated piece of code that invokes a unit of work in a system and then checks a single assumption about the behavior of that unit of work.

A unit test is a way of testing a unit – the smallest piece of code that can be logically isolated in a system.

Unit Testing, also known as Component Testing, is a level of software testing where individual units/components of a software are tested

Why Unit Testing? Why Not Unit Testing?

Why Unit Testing?

- Find logical issues early
- Creates a code contract that protects against changes
- Ensures adherence to acceptance criteria
- Provides living documentation of how the code is meant to be consumed

Why Not Unit Testing?

- Does not catch integration issues
- Can be difficult to set up realistic unit tests
- Adds extra development effort – sometimes more than twice as much as the initial code
- Requires tests to be run to be effective

What Are The Characteristics of Unit Tests?

1	Unit Tests are ***automated***
2	Unit Tests are ***granular***
3	Unit Tests ***isolate*** their target
4	Unit Tests are ***deterministic***
5	Unit Tests are ***independent***
6	Unit Tests are ***fast***

So, What Does a Unit Test *Look* Like?

Say We Have a Basic Function:

```
public string GetMessage(string name) => $"Hello, {name}";
```

We Can Unit Test It Like This:

```
[Fact]
0 references | Run Test | Debug Test
public void Test_GetMessage()
{
    // Arrange
    var testName = "World";
    var expected = "Hello, World";
    var generator = new MessageGenerator();

    // Act
    var result = generator.GetMessage(testName);

    // Assert
    Assert.Equal(expected, result);
}
```

But That Obviously Wasn't T-SQL

Should We Be Unit Testing Our T-SQL Code?

Database Unit Testing?

Do You Have Business Logic In Your Database?

Could Be Stored Procedures, Defaults, Computed Columns…

Is It Tested When You First Develop It?

How About Against Regressions?

You Should (Strongly) Consider Unit Testing!

What Does a Database Unit Test Look Like?

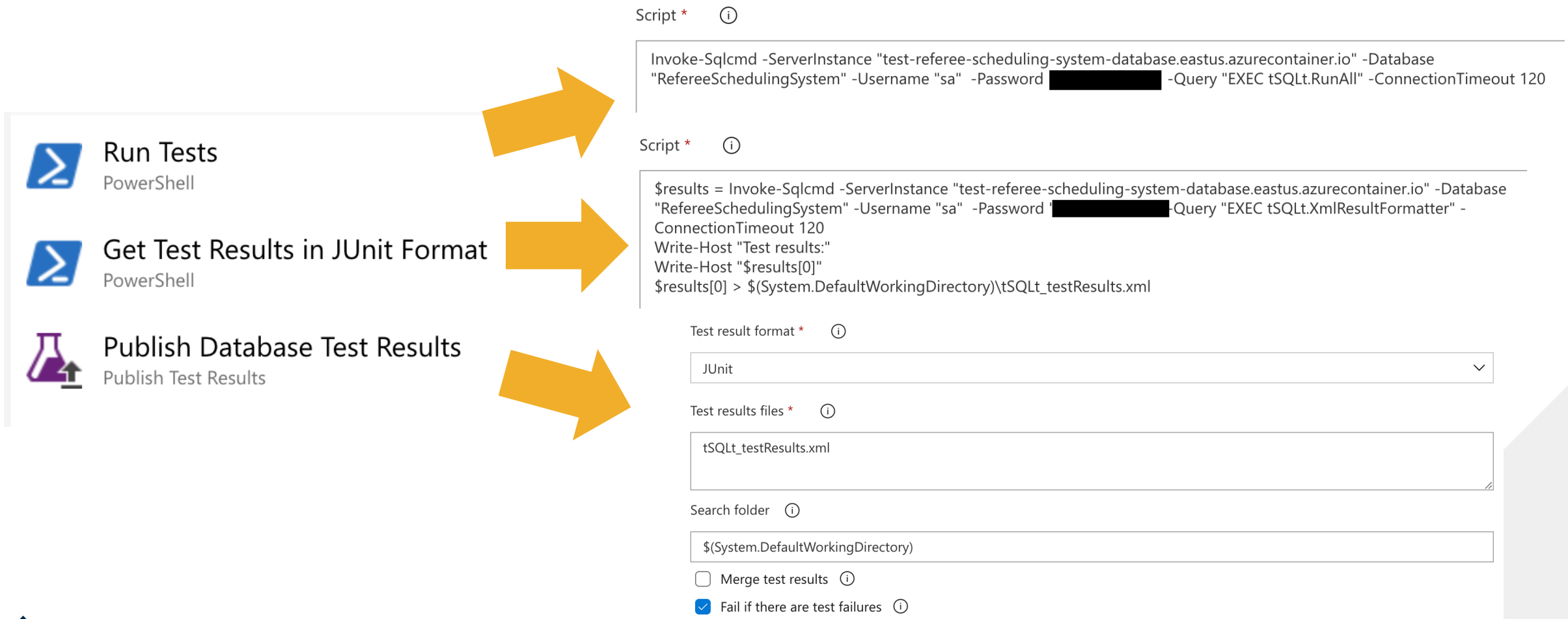
(AKA Demo Time)

Now That We Have a Framework, Can We Automate This?

Absolutely!

Just Add Your Favorite CI/CD Tool (and Maybe a Dash of PowerShell)!

What About Automation?



Questions?

Thank You!

@danielmallott
linkedin.com/in/danielmallott
dmallott@westmonroe.com