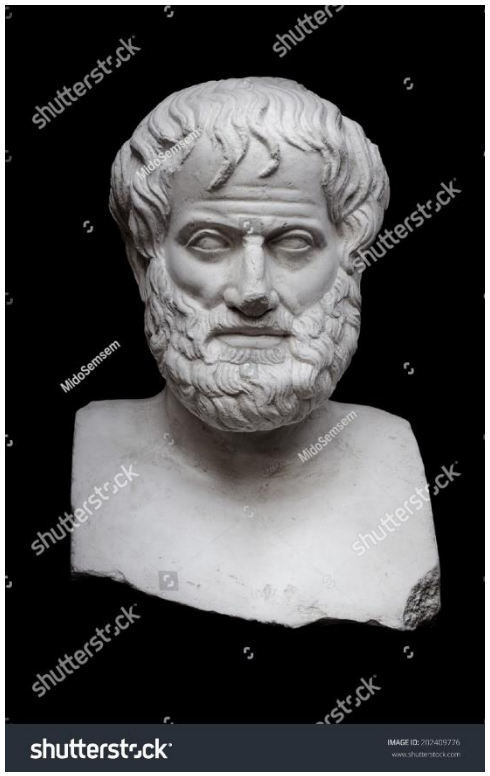
Testing and Verifying an SSIS Package

Chris B. Behrens
SOFTWARE ARCHITECT
@chrisbbehrens

What We Mean By “Testing”

Aristotle and Unit Testing



The Nichomachean Ethics

Both his son and his dad were named Nichomachus

The measure of a thing is how well it fulfills its function

A good knife is a knife that cuts well

This philosophy applied to code tells us how to test

Testing with Databases

This approach is more difficult to apply to databases

We shouldn't waste our time testing the product specification

We should rely on the manufacturer

If that doesn't sound like a good idea, we should find another vendor

Verifying Availability

This isn't about whether the database is up and running

We're concerned that the database works *correctly*

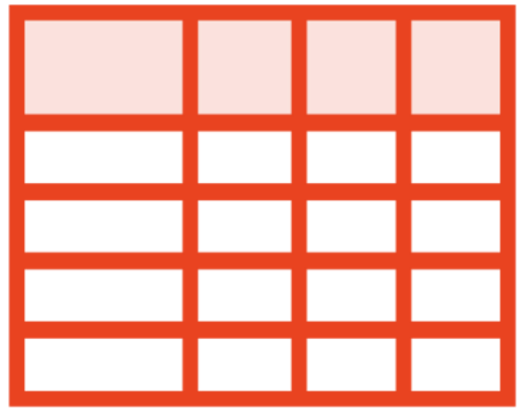
Verifying availability is important – just not what we're talking about now

What We ARE Testing For

Stored procs are easy(er) to verify

Execute the proc and then measure the result

We're going to verify changes with a measurement



By verifying the state of a table when we're done

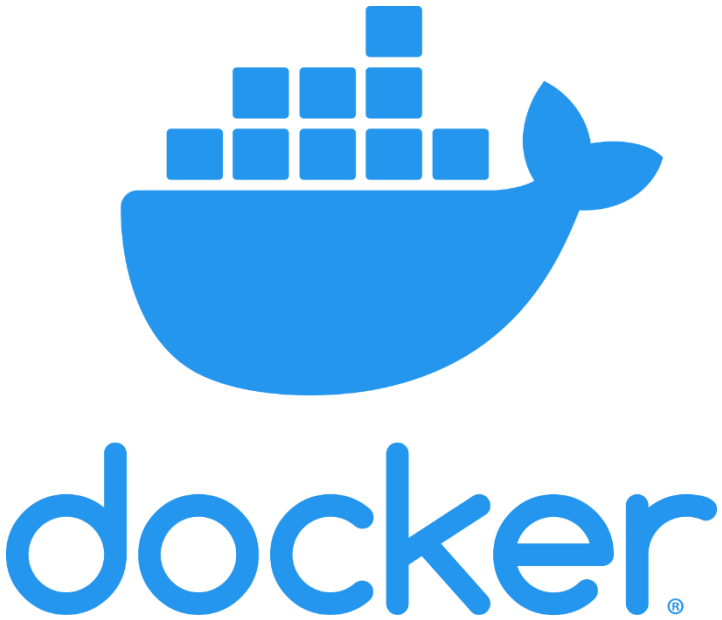
A Database Test Harness

A tricky data layer

I would restore the database before each test...

And then tear it down for the next test

Where Our Target DB Is Running



Don’t sweat it if you’re not too deep into Docker yet

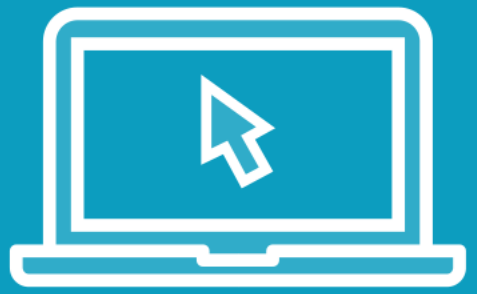
We’re going to use Docker to build up and tear down our test environment

With this, you could do it hundreds of times per day if you needed to

Sql Server is available as a Docker image

On both Windows and Linux

Demo



Take a look at running container database

The command I used to create it

Provision an entirely new database from scratch

- Using a modified version of that command
- Connect to it in SSMS

A Testing and Verification Architecture

Order of Operations

1. Developer makes the change to the package
2. Developer checks in the change
3. The check-in triggers a build
4. The build pulls the new package from version control
5. The build executes the docker run command and creates a new instance of Sql server
6. The build executes a script to construct the test target db
7. The build executes any necessary schema upgrades
8. The build executes the package against the db
9. The build verifies that the results match expectations

Order of Operations

The build executes the docker run command and creates a new instance of Sql server

Order of Operations

The build executes any necessary schema upgrades

https://app.pluralsight.com/library/courses/deploying-databases-octopus

Order of Operations

The build verifies that the results match expectations

Demo

Build up the simple verification within our package

- With a row count transformation
- A script step to validate it
- Execute our package

Break it by changing the data file

Demo

Take the assertion that we've just created

Build a test platform to run it on

Using our build process

Execute our build

Verify our package

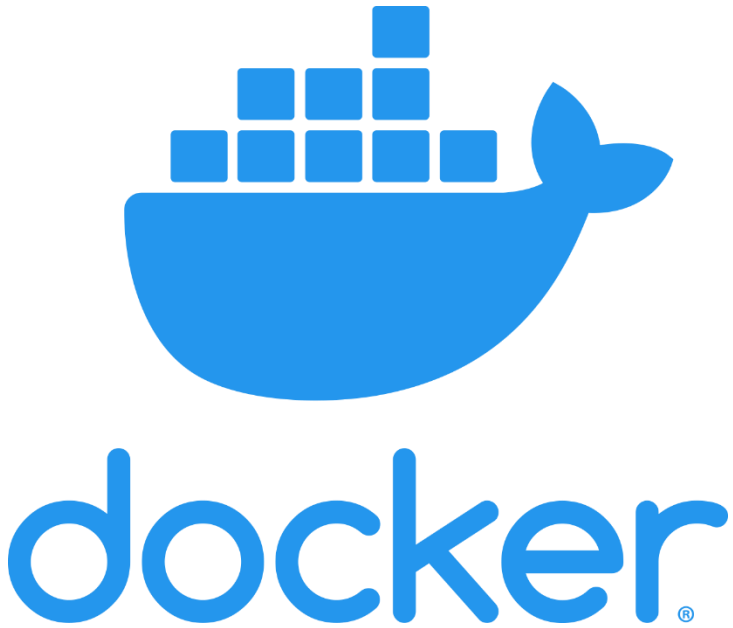
Going Deeper with this Pattern

Our assertion is trivial

The content could be entirely wrong, and just have the right row count

But this is a jumping off point for deeper testing

How This Would Really Work



Building everything from scratch every time is slow

- We've only done this to keep things simpler

If we cooked all this into a Dockerfile and built an image off of it, it would be much, much faster

https://bit.ly/2zcasjB

This pattern is not Sql Server specific

Other databases have been in Docker a lot longer

Summary

Develop a unified testing methodology for our package

- Using Docker images
- A scripted assertion
- Jenkins builds