Configuring and Deploying SSIS Packages

UNDERSTANDING THE BASICS OF SSIS CONFIGURATION

Chris B. Behrens
SOFTWARE ARCHITECT
@chrisbbehrens

The Fiddly Bits

We're going to look at how packages work a little bit...

But mostly focus on how they're configured

This is the least fun part of development, mostly

The Integration Services Deployment Models

Package and Project

Project is newer, and what you'll usually see recommended

I disagree with the consensus – double-check my reasoning and make up your own mind

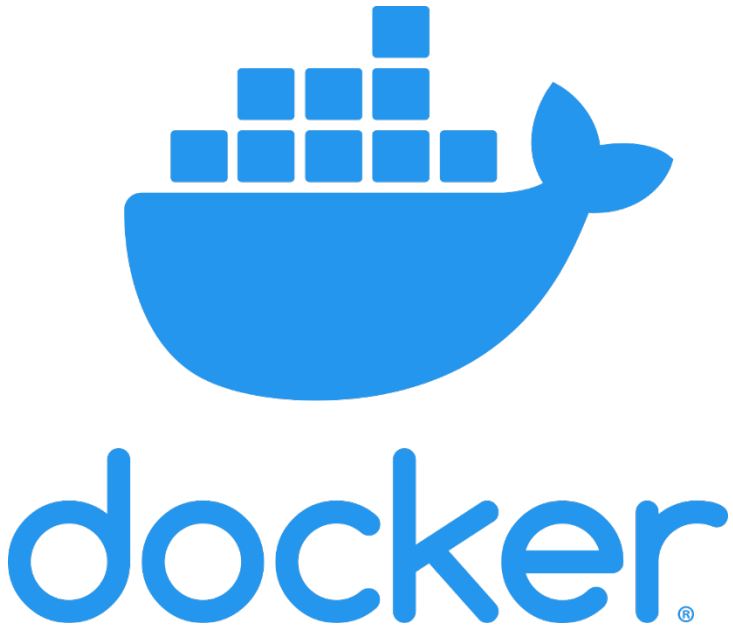
Technology Synthesis

No single tech will do the job

So you have to work with pieces of technology together

Don’t worry (?), I’m not an expert either

Working with Docker



We're going to do some stuff with Docker later on - don't panic

Everything is intro-tutorial level complexity

You don't need to *really* know Docker to work at this level

The Nature of Deployment

The packages are the object

"deploying" and "configuration" are the verbs

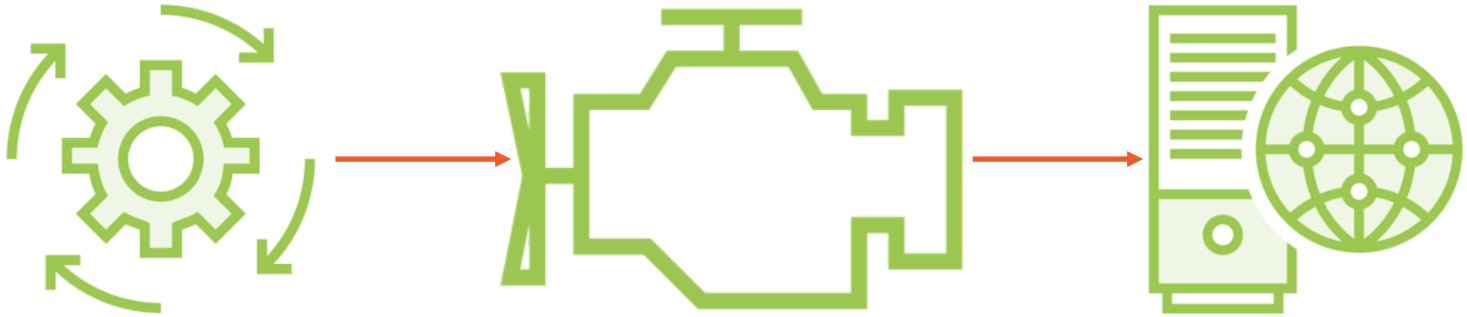
Jenkins will be our primary build server

But we'll also talk a little about Azure DevOps

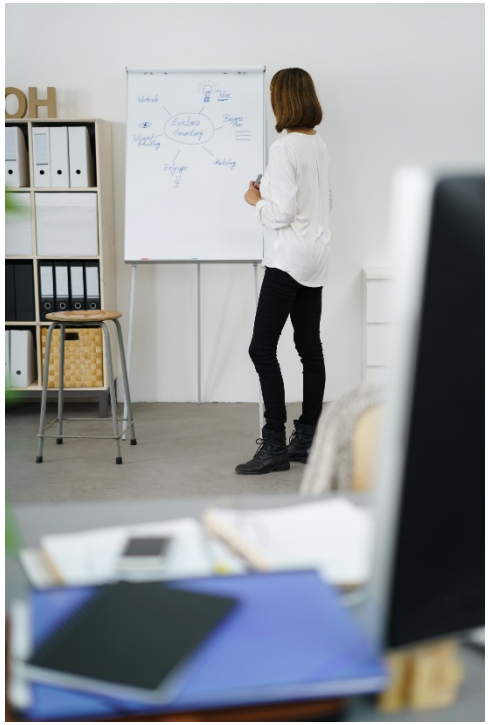
Package vs Project Models

“deployment model”

“deployment model”



Understanding the Project Model



What the project model is:

- Wrap your content in an .ispac file
- Deploy it to a repository inside the target db
- Once inside, you can schedule execution

How Deployment Works Today

1. Store your non-binary asset in version control.
2. Restore that asset from version control as the first step of a build.
3. Apply the meaning of that asset to your build output in whatever asset-specific way that applies.

Database Deployment

Databases are the trickiest thing to deploy

Because their states are irreversible and changing

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

The Problem With Project Deployment

Turn the target database into a repository..

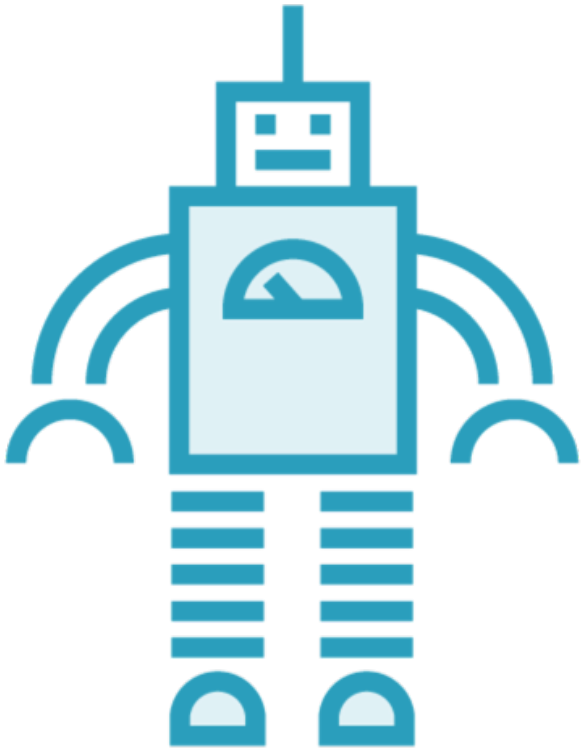
And a build server...

And a deployment engine

Sql Server stored the code in addition to version control

Creating a second source of truth – a very bad thing

Why You Should Use Package Deployment



You *can* automate deployment of the project model

I used SSISUp to do this for a long time

But eventually, you realize that it's easier to restore from version control and execute – like you do with everything else in deployment

"The more modern method of project deployment" – check out SSISUp

In any case, we're going to use this method for integration testing

Introducing Our Simple Integration Services Package

Our simplified package

With our basic understanding, we'll move to configuring it

The Globomantics Demo Scenario

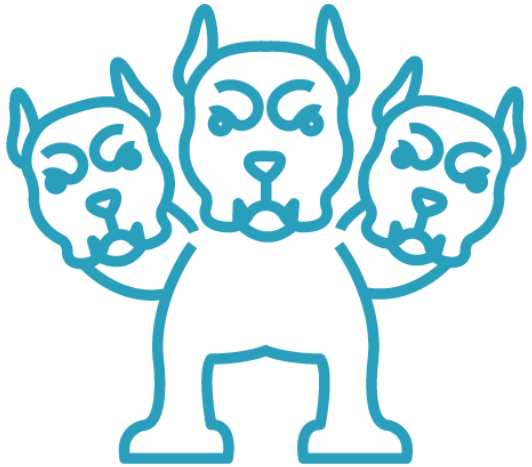
Globomantics runs a wellness program for another company

The program promotes employee wellness through healthy activities

Completing these activities accrues points, which can be traded in for rewards

Globomantics' ETL

All this is reported on via a data-driven website



Employees SSO into the site

Data transfer is effected via nightly FTP feed

Demo

Lay out the business demo scenario for the course

Review our simplified test database

Look at a sample data file

Look at our simple import package

Execute it

Review the results

Configuring Your Package

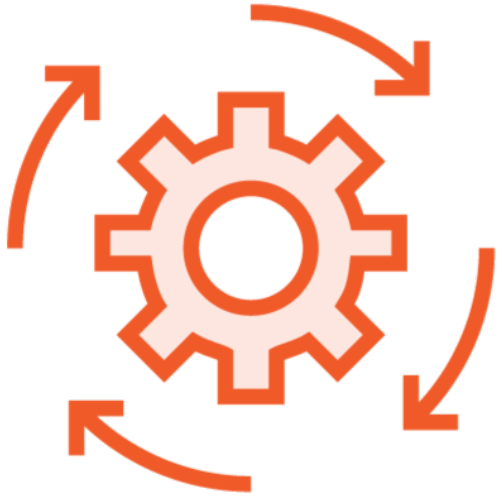
Configuration is necessary when configuration values differ in different environments for the application.

“I think something might change.”

Why This Is Wrong

Of course, something is going to change

Configuration is just about the worst way to mitigate change



Continuous delivery is the right way to mitigate change

Configuration is necessary when configuration values **differ in different environments** for the application.

Ensure Your Versions Match

In all my years of working with SSIS and Sql Server...

My tools version was current with or ahead of the target db

But when I was running a bleeding edge Sql Server version for this course...

My tool version was behind

This resulted in some cryptic error messages

```
SELECT @@VERSION
```

How to Check This

It's pretty simple:

- The tools will identify their version at the top
- The Sql Server version can be determined with the above query

Configuration with XML

The XML Content of Our Package File

```
<DTS:Property
    DTS:Name="PackageFormatVersion">8</DTS:Property>
  <DTS:ConnectionManagers>
    <DTS:ConnectionManager
      DTS:refId="Package.ConnectionManagers[Employees File]"
      DTS:CreationName="FLATFILE"
      DTS:DTSID="{8647D455-B5D2-4F9A-ACEC-E63346AE1052}"
      DTS:ObjectName="Employees File">
      <DTS:ObjectData>
        <DTS:ConnectionManager
          DTS:Format="Delimited"
          DTS:LocaleID="1033"
          DTS:HeaderRowDelimiter="_x000D__x000A_"
          DTS:RowDelimiter=""
          DTS:TextQualifier="_x003C_none_x003E_"
          DTS:CodePage="1252"
          DTS:ConnectionString="C:\Users\chris\Google Drive\PluralSight\Configuring and Deploying SSIS
Packages\Code\Globomantics\Incoming\employees.txt">
```

Demo

Execute our package from the command line

- With no configuration
- Using the native configuration we've already specified

Check the target database to make sure that works

Create a package configuration file

- That reflects a different target db

Execute it again with that configuration

Review the results

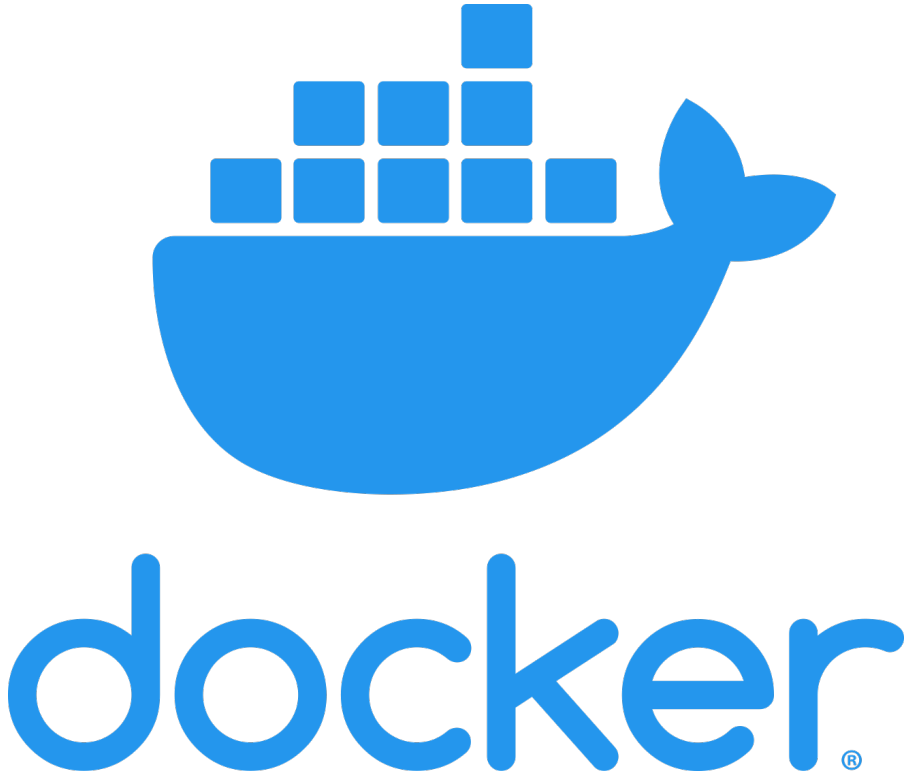
Environment Variables

You can specify properties via environment variables as well

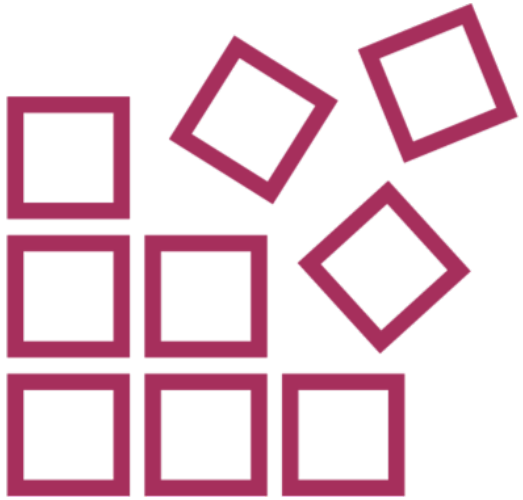
But this means another environment aspect to maintain, when you could just get it from version control

Setting environment variables is easy in Docker

And comes directly from version control (in the form of a Dockerfile)



Registry Entries



Storing properties in the registry...

Of the target machine that the package executes on

So the staging box has staging settings, and the production box has production settings

Don’t ever use the registry
for anything ever anytime.

Will the Registry Survive?

Windows is moving ever further in the direction of Linux

Linux architecture is fundamentally opposed to a registry structure

Look for Windows to abandon the registry in the future

https://www.tutorialgateway.org/ssis-package-configuration-using-registry-entry

A Sql Server Table

```
CREATE TABLE [dbo].[SSIS Configurations]
(
    ConfigurationFilter NVARCHAR(255) NOT NULL,
    ConfiguredValue NVARCHAR(255) NULL,
    PackagePath NVARCHAR(255) NOT NULL,
    ConfiguredValueType NVARCHAR(20) NOT NULL
)
```

The Problem with Sql Server Table Config

If you store the config in the table...

Now you have to maintain that server and keep it as part of your delivery pipeline

You've created another database to manage your database, which you have to manage

A Parent Package Variable

I'm mostly down on these other methods

But you can use parent variables to help modularize your packages

For example, creating a generic "populate membership" child package

How It Works

Your child package manages user passwords

With control flows for encrypting or hashing

Which path is executed is driven by a parent *variable*

Configuring this is just specifying the name of the variable

Configuring Via the Command Line

A journalism expression, “burying the lede”

“A minor house fire on 7th Avenue...”

“That was caused by an alien invasion”

We can simply drive these properties from the command line

Summary

The nature of deployment

Where integration services fits with that

A minimal useful package for our demos

Five different ways to configure it:

- An Xml file
- Environment variables
- Registry entries
- A parent package variable
- Sql Server table

The most useful ways

- Xml
- Environment variables
- How to make them work