

THE AGILE DBA: DATABASE DEVOPS WITH SSDT



Ryan Booz

Senior Software Dev
KCF Technologies, Inc.

@RyanBooz

www.SoftwareAndBooz.com

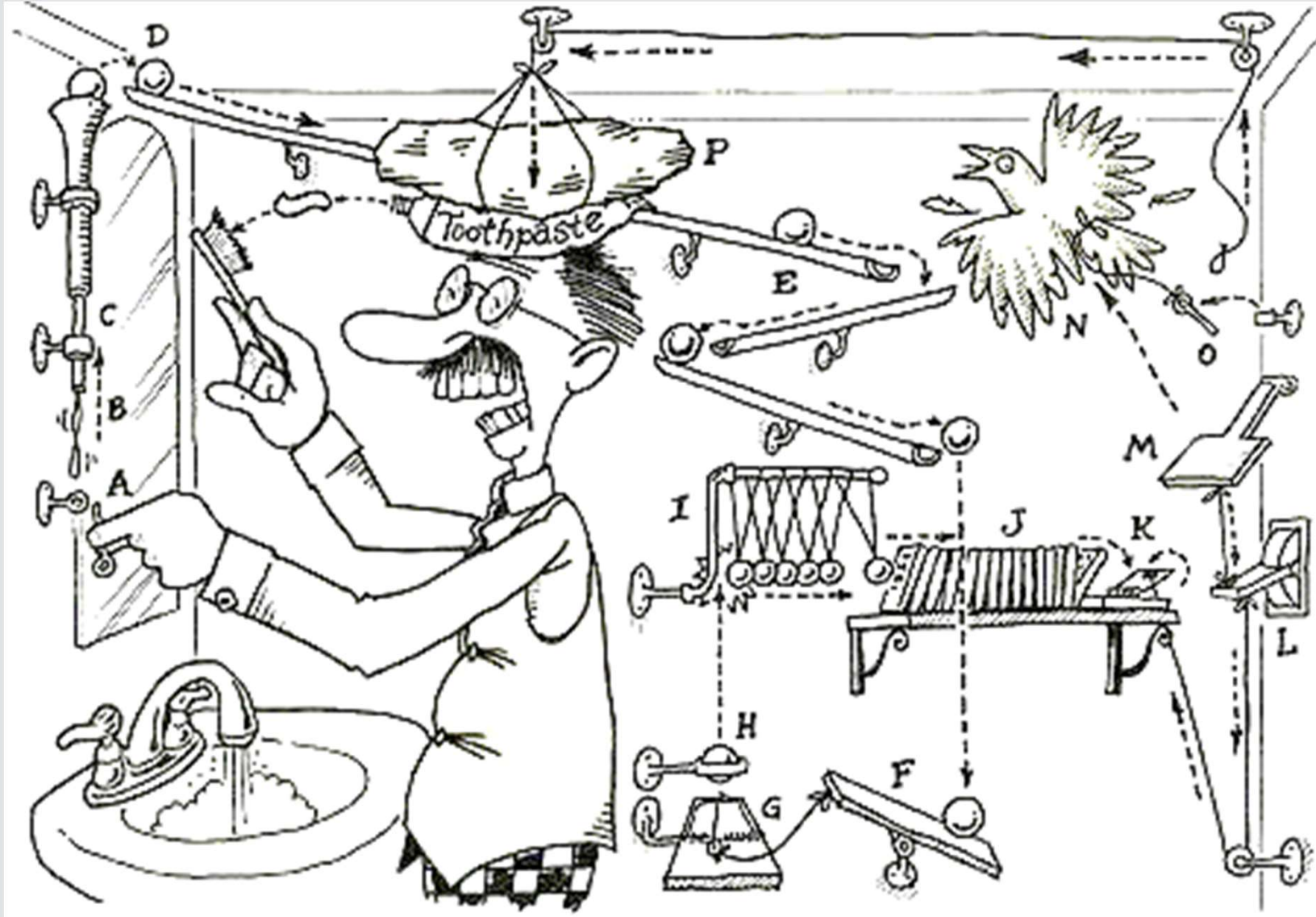
<https://github.com/ryanbooz/presentations>

ABOUT ME

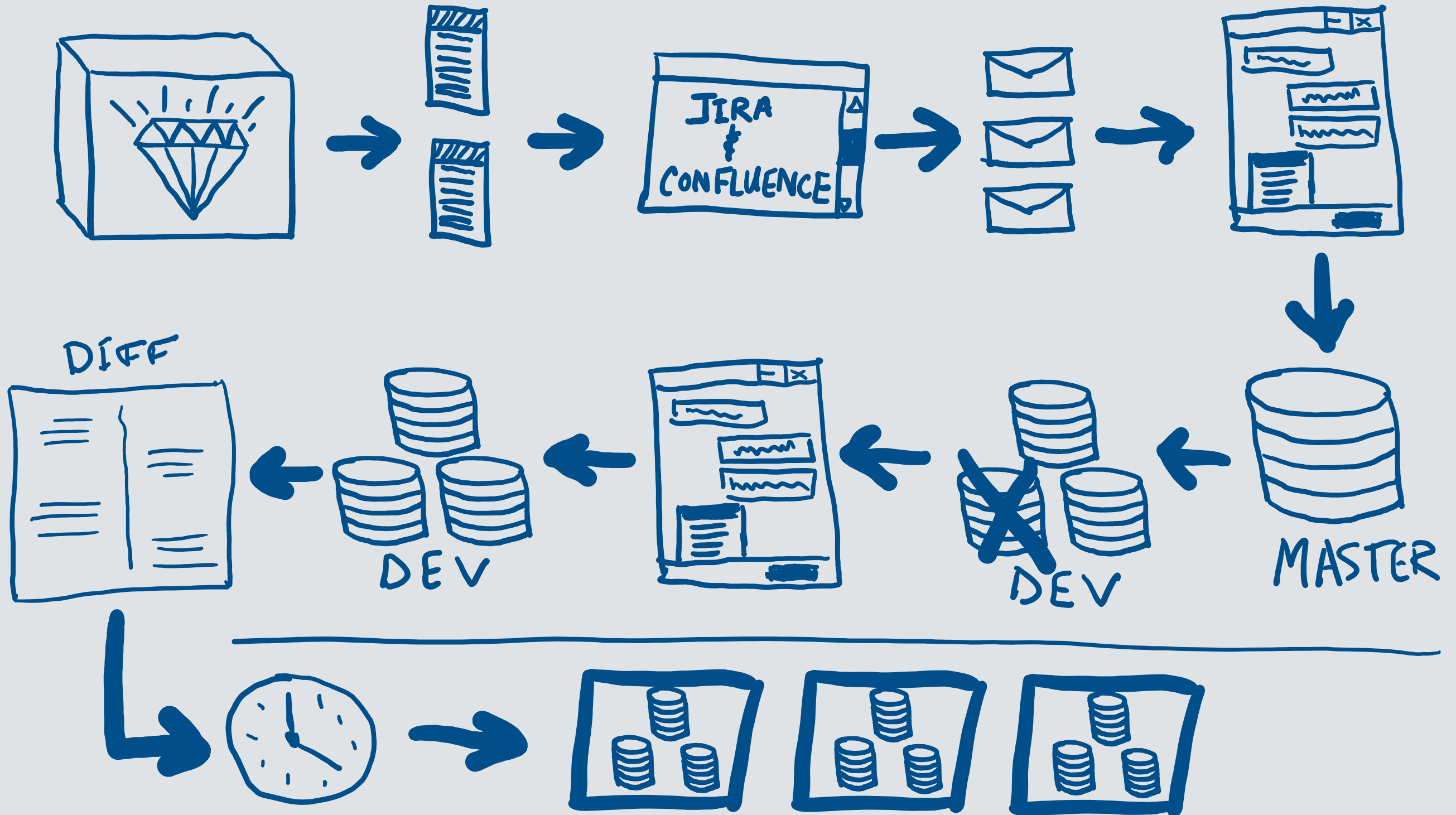
- Husband and Father of 6
- Own 33 acres, 109 chickens & 2 beehives
- Indoor Rower
- 19 years of Relational DB experience – even DB2!!
- Learned to program on a DEC 5000 in high school with the Pittsburgh Supercomputing Center
- Data Science Wannabe!

AGENDA

- What Problem are we solving?
- Overview of SSDT
- Creating a Project from an existing database
- Develop and Deploy Locally
- Continuous Integration & Deployment in Dev
- What about Production?
- A handful of gotchas
- Resources

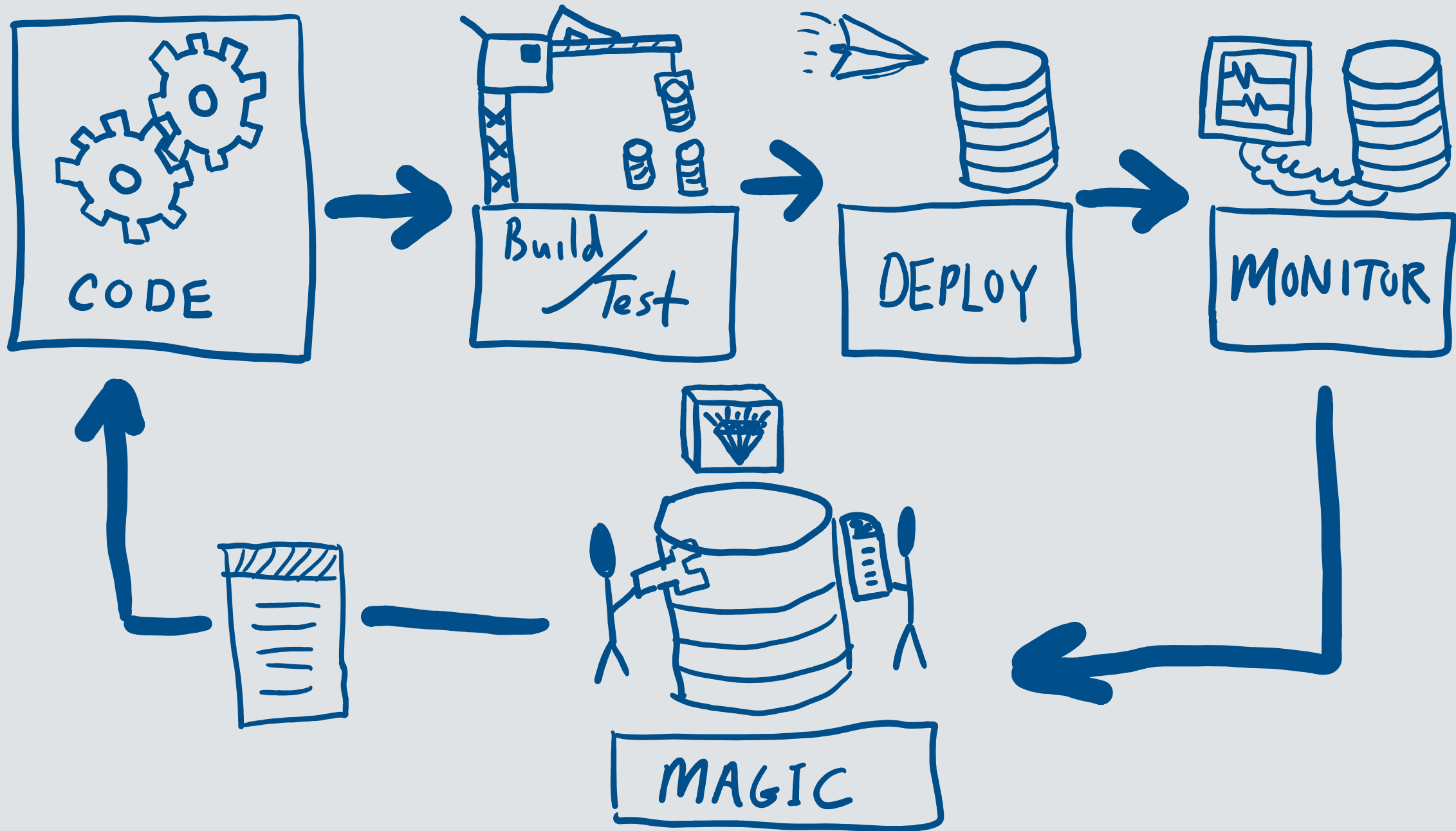


THE OLD
WAY



THE DEVOPS WAY





GITFLOW



MORE DEVOPS INFORMATION

Check out these other resources:

- <https://groupby.org/conference-session-abstracts/bringing-devops-to-the-database/>
- <https://groupby.org/conference-session-abstracts/devops-101-for-data-professionals-how-your-jobs-will-change/>
- The Phoenix Project (book)
- The DevOps Handbook (book)

SSDT OVERVIEW

WHY SSDT?

- Declarative, not Migration (Redgate ReadyRoll)
 - DACPAC is the key here
- Rich exploration of database as code
- Validation of Schema and Objects
- Code Analysis
- Refactoring
- Database Schema Comparison
- Consistent, fine-grained deployment control

PREREQUISITS

- Visual Studio 2017 (any edition)
- A Build environment (Visual Studio Team Services)
- Local version of SQL Server (SQL 2017 Developer)
- Source control (again, VSTS can provide this)
 - Gitflow or similar branching definition

<https://docs.microsoft.com/en-us/sql/ssdt/download-sql-server-data-tools-ssdt>


Workloads

Individual components

Language packs


Installation locations

Web & Cloud (7)




ASP.NET and web development
Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and container development tools.

☒




Python development
Editing, debugging, interactive development and source control for Python.

☐




Data storage and processing
Connect, develop and test data solutions using SQL Server, Azure Data Lake, Hadoop or Azure ML.

☒




Office/SharePoint development
Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

☒




Azure development
Azure SDK, tools, and projects for developing cloud apps and creating resources.

☒



Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

☐



Data science and analytical applications
Languages and tooling for creating data science applications, including Python, R and F#.

☐

Summary

> Visual Studio core editor

> Universal Windows Platform development

> .NET desktop development

> Mobile development with .NET

> Office/SharePoint development *

> Azure development

> ASP.NET and web development

> Game development with Unity

✓ Data storage and processing

Optional

☒ SQL Server Data Tools

☒ Azure Data Lake and Stream Analytics Tools

☒ .NET Framework 4 – 4.6 development tools

☒ Redgate ReadyRoll Core

☒ Redgate SQL Prompt Core

☒ Redgate SQL Search

☒ F# language support

DATA PROJECTS COMPONENTS

DACPAC	<ul style="list-style-type: none">• Glorified ZIP file with model XML and other meta data
MSBUILD	<ul style="list-style-type: none">• Same build tooling as other development projects• Creates DACPAC from Database Project declared model• Not yet Cross-platform for database projects
SQLPackage	<ul style="list-style-type: none">• Compares two DACPAC files to generate differences• Create change scripts, apply changes, generate deployment reports• Configure through publish profiles and command-line• Deployment contributors provide endless opportunities• Cross-Platform

MISSION: FIND A NEW BABY NAME!

- Use Kendra Little's BabbyNames database
 - <https://github.com/LitKnd/BabbyNames>
- Apply Trigram search ability based on POC from Paul White
 - <https://sqlperformance.com/2017/09/sql-performance/sql-server-trigram-wildcard-search>
- Find names similar to the one we are searching for, most popular first

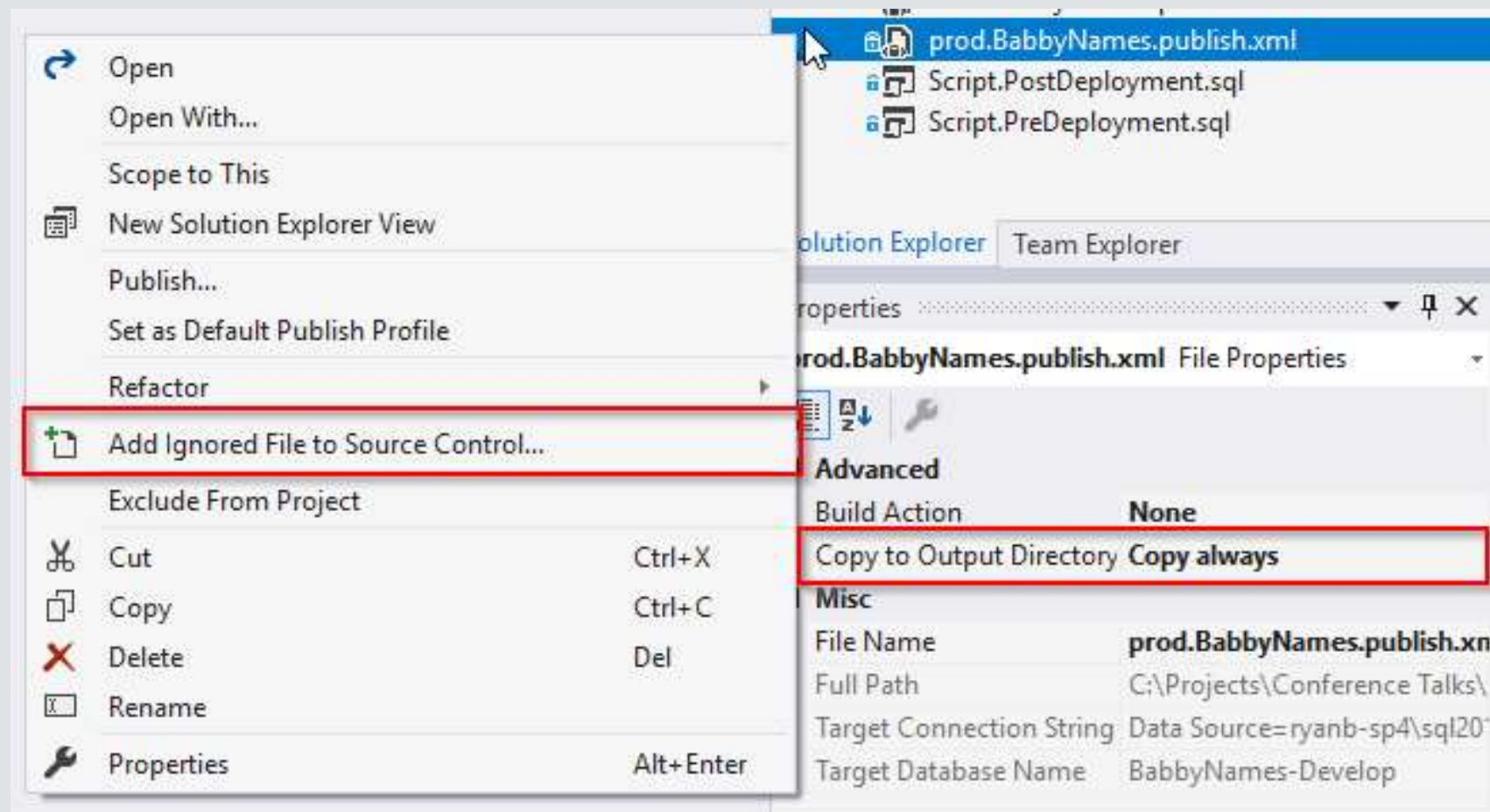
SSDT DEMO

MINIMUM PROFILE SETTINGS

- At least consider these for each profile:
 - Generate smart defaults
 - Ignore column order
 - Automatically take a backup
 - Include Transactional Scripts
 - Allow Incompatible Platform – between version deployments
- Never select “Always Re-create Database”
 - Just sayin’

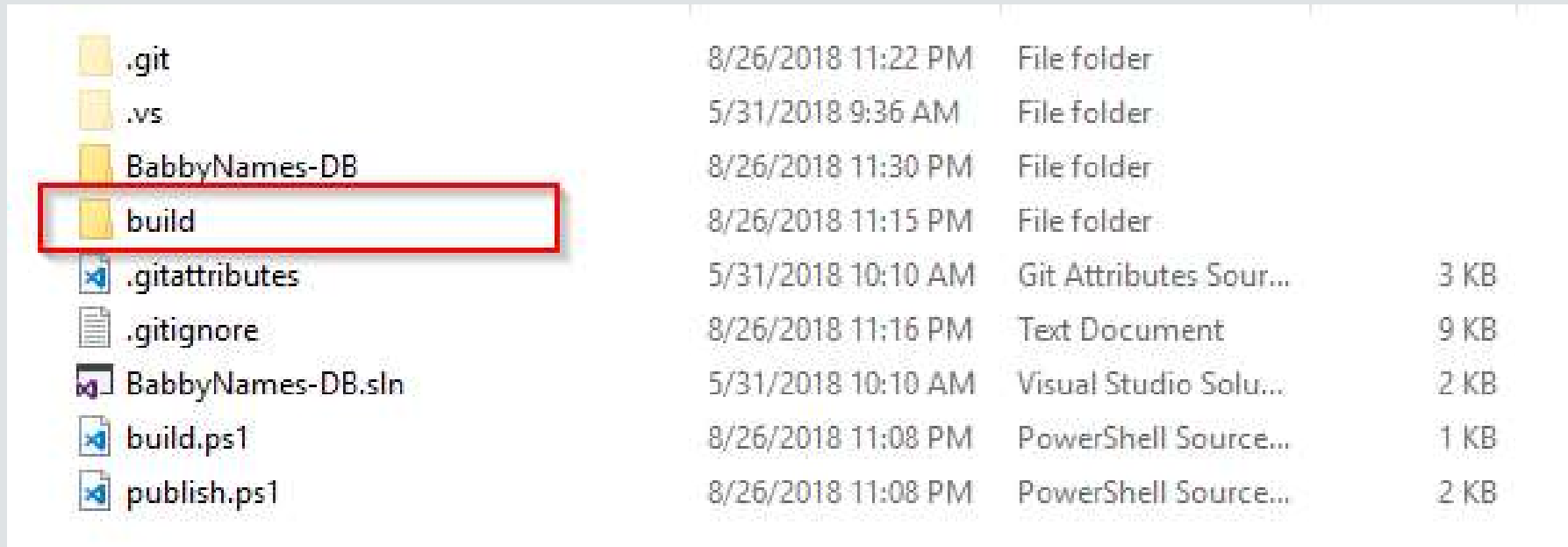
MINIMUM DEPLOYMENT STEPS

1. Publish Profile for each environment/type



MINIMUM DEPLOYMENT STEPS

2. MSBuild Tools installed in Solution Folder (not included in solution definition)












.git	8/26/2018 11:22 PM	File folder	
.vs	5/31/2018 9:36 AM	File folder	
BabbyNames-DB	8/26/2018 11:30 PM	File folder	
build	8/26/2018 11:15 PM	File folder	
.gitattributes	5/31/2018 10:10 AM	Git Attributes Sour...	3 KB
.gitignore	8/26/2018 11:16 PM	Text Document	9 KB
BabbyNames-DB.sln	5/31/2018 10:10 AM	Visual Studio Solu...	2 KB
build.ps1	8/26/2018 11:08 PM	PowerShell Source...	1 KB
publish.ps1	8/26/2018 11:08 PM	PowerShell Source...	2 KB

Install into folder from: <https://www.nuget.org/packages/Microsoft.Data.Tools.Msbuild/>










MINIMUM DEPLOYMENT STEPS

3. Build Script (shown at root of Solution)

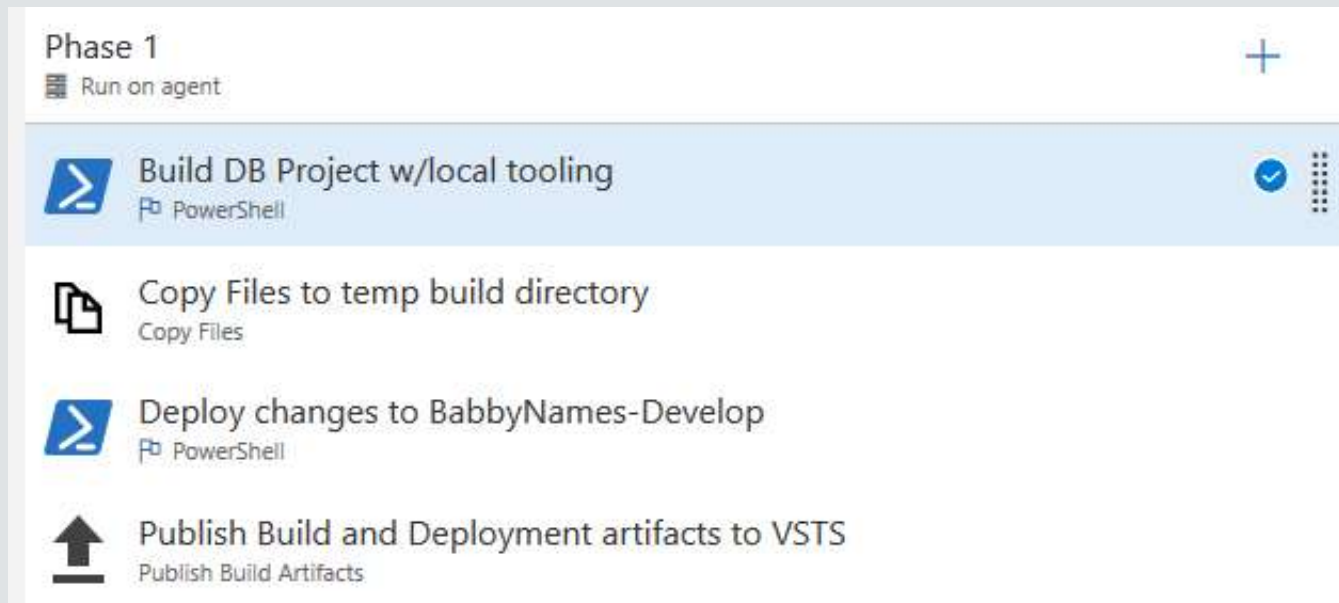
 .git	8/26/2018 11:22 PM	File folder	
 .vs	5/31/2018 9:36 AM	File folder	
 BabbyNames-DB	8/26/2018 11:30 PM	File folder	
 build	8/26/2018 11:15 PM	File folder	
 .gitattributes	5/31/2018 10:10 AM	Git Attributes Sour...	3 KB
 .gitignore	8/26/2018 11:16 PM	Text Document	9 KB
 BabbyNames-DB.sln	5/31/2018 10:10 AM	Visual Studio Solu...	2 KB
 build.ps1	8/26/2018 11:08 PM	PowerShell Source...	1 KB
 publish.ps1	8/26/2018 11:08 PM	PowerShell Source...	2 KB

MINIMUM DEPLOYMENT STEPS

4. Publish Script (shown at root of Solution)

 .git	8/26/2018 11:22 PM	File folder	
 .vs	5/31/2018 9:36 AM	File folder	
 BabbyNames-DB	8/26/2018 11:30 PM	File folder	
 build	8/26/2018 11:15 PM	File folder	
 .gitattributes	5/31/2018 10:10 AM	Git Attributes Sour...	3 KB
 .gitignore	8/26/2018 11:16 PM	Text Document	9 KB
 BabbyNames-DB.sln	5/31/2018 10:10 AM	Visual Studio Solu...	2 KB
 build.ps1	8/26/2018 11:08 PM	PowerShell Source...	1 KB
 publish.ps1	8/26/2018 11:08 PM	PowerShell Source...	2 KB

BUILD AND PUBLISH – ON-PREM



The screenshot displays a build system interface for 'Phase 1'. At the top, it says 'Phase 1' with a plus icon and 'Run on agent' with a server icon. Below this is a list of four tasks:

- Build DB Project w/local tooling** (PowerShell icon) - This task is highlighted in blue and has a blue checkmark icon to its right.
- Copy Files to temp build directory** (Copy Files icon)
- Deploy changes to BabbyNames-Develop** (PowerShell icon)
- Publish Build and Deployment artifacts to VSTS** (Publish Build Artifacts icon)

- Install and configure local build agent
- Install Microsoft.Data.Tools.Msbuild NuGet package locally
- Create Powershell script to run the build
- Create Powershell script to run SqlPackage to Publish the database

BUILD AND PUBLISH - AZURE

Phase 1

 Run on agent



Build BabbyNames sqlproj

MSBuild



Copy Files to temp build directory

Copy Files



Execute Azure SQL : DacpacTask

Azure SQL Database Deployment



Publish Build and Deployment artifacts to VSTS

Publish Build Artifacts

BUILD & DEPLOY DEMO

DEPLOYMENT CONTRIBUTORS

GAME CHANGING FEATURE

- Examine and modify DB Model
- Deployment-time decisions with access to script, source and target model
- Provides significant flexibility for well understood needs
- Steep learning curve
- But come on... it's fun!

DEPLOYMENT CONTRIBUTOR DEMO

WHAT ABOUT PRODUCTION?

- Depends how automated you want to be and risk tolerance
 - Lots of DBs might not scale based on maintenance windows
 - Can be transactional, which should minimize some risk aside from wasted time
- Deployment Contributors
- “Generate Script” against previous version of database
 - Allows final review of scripts
 - Any additional edits that might not be feasible in SSDT
 - Provide to clients
 - Use familiar tooling - SSMS

GOTCHAS

- MSBuild for Database Projects is not Cross-Platform yet (Windows only)
- Inconsistencies between your code and how SQL stores it, causing the same upgrade every time:
 - Data Motion
 - Check constraint (IN vs OR)
 - Fill Factor isn't always ignored
 - Permissions out of Sync
- Using References is not trivial
- Order of upgrade script is determined by SQLPackage, which might not always make preferable decisions
- No concept of “Rollback”, “Roll-forward” instead

FURTHER READING

- SSDT Docs: [https://msdn.microsoft.com/en-us/library/hh272686\(v=vs.103\).aspx](https://msdn.microsoft.com/en-us/library/hh272686(v=vs.103).aspx)
- SqlPackage Docs: [https://msdn.microsoft.com/library/hh550080\(vs.103\).aspx](https://msdn.microsoft.com/library/hh550080(vs.103).aspx)
- Deployment Contributors: [https://msdn.microsoft.com/en-us/library/dn306642\(v=vs.103\).aspx](https://msdn.microsoft.com/en-us/library/dn306642(v=vs.103).aspx)
- Azure CI/CD Docs: <https://blogs.msdn.microsoft.com/ssdt/2016/04/06/sqlldb-cicd-intro/>
- Ed Elliot: <https://the.agilesql.club/>
 - <https://github.com/GoEddie/DeploymentContributorFilterer>

WHAT QUESTIONS DO YOU HAVE?

THANK YOU!