Migration or state based?

How to put your database under source control

Joyful Craftsmen

Business Intelligence Crafted to Joy

Agenda

- Motivation
- State or migration – main gotchas
- Demo
- Summary

About me

- SQL Server Support Engineer @Joyful Craftsmen
- Former SQL Server Lead DBA from ČS, a.s.
- Czech PASS Leader & SQL Saturday co-organizer

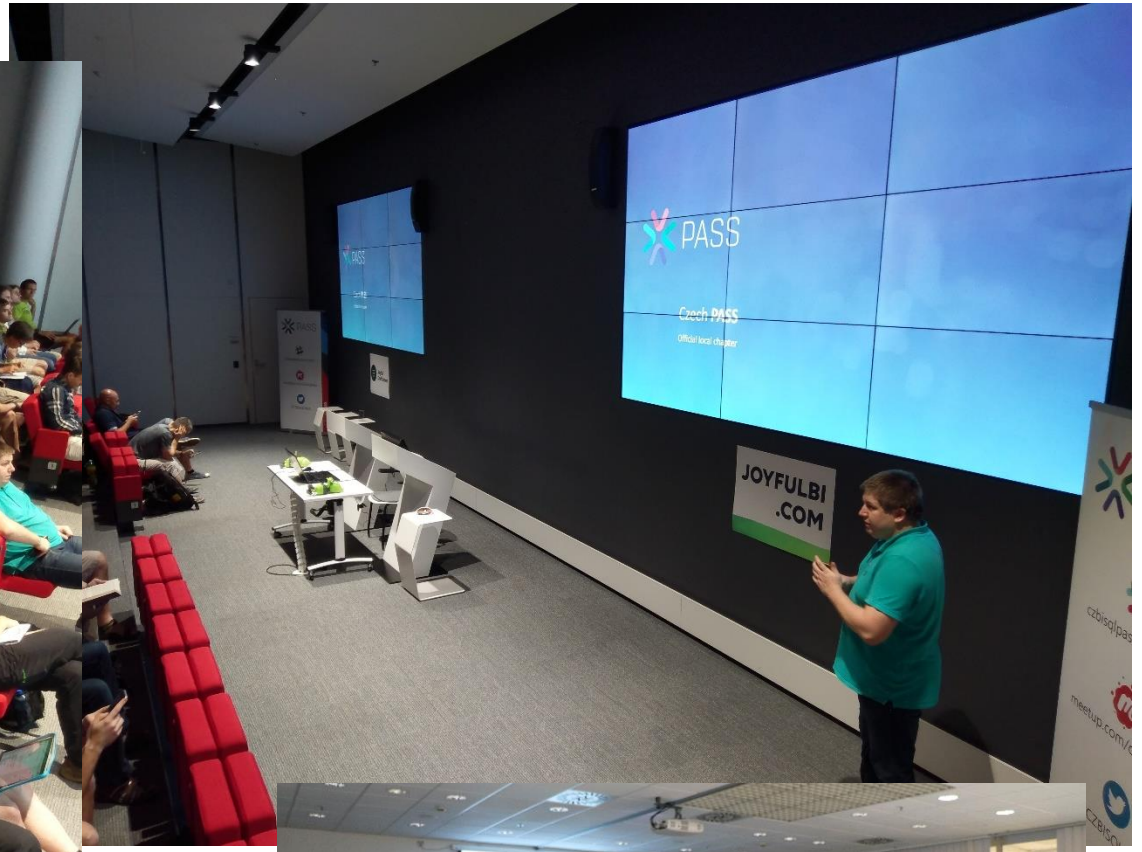
@malekpav

pavel.malek@joyfulcraftsmen.com

https://joyfulcraftsmen.com/

Czech PASS

SQL Saturday Prague #889



PASS

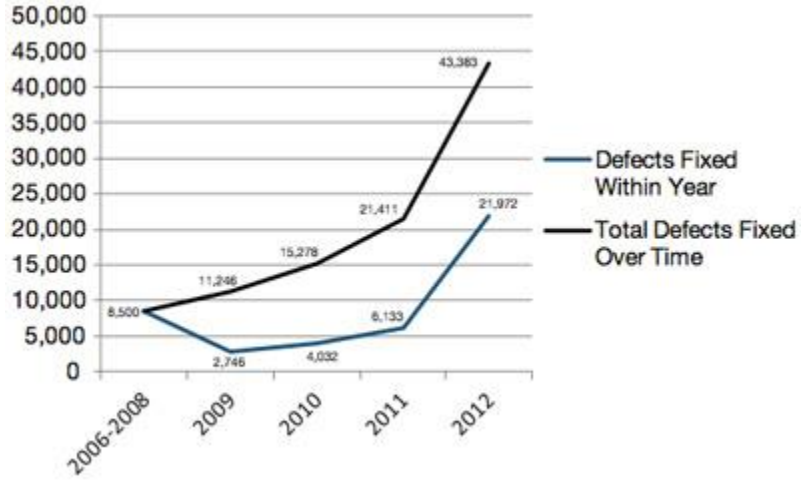
DLM!

What problems we are trying to solve?

Bug density

"Over the past seven years, the Coverity Scan service has analyzed nearly 850 million lines of code from more than 300 open source projects including Linux, PHP and Apache. ... The analysis found an average defect density of .69“

Source:
https://www.helpnetsecurity.com/2013/05/07/analyzing-450-million-lines-of-software-code/



The c

Quality Cost - finding the sweet spot

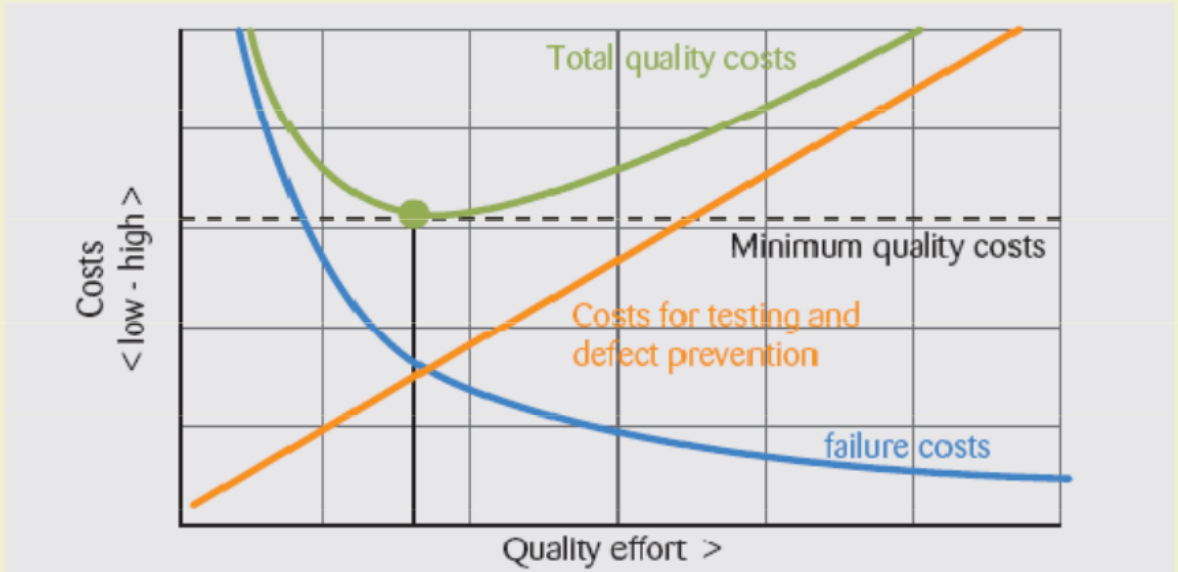


Diagram from Frühauf, K., Ludewig, J., Sandmayr, H. (1990), Software-Projektmanagement und -Qualitätssicherung

100x

Cost to Fix Defect

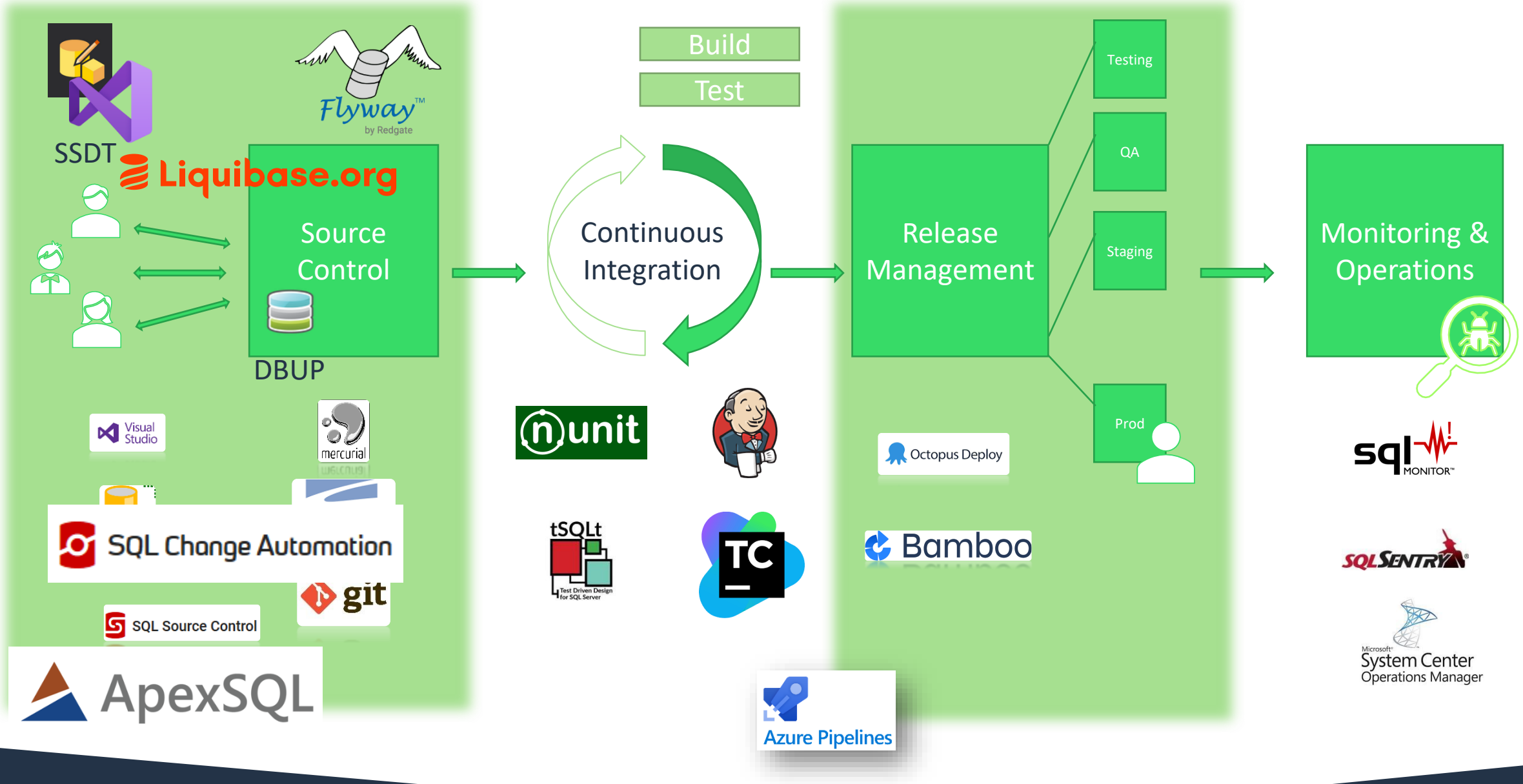
When Detected

14
47
180 1
249
99 4
396

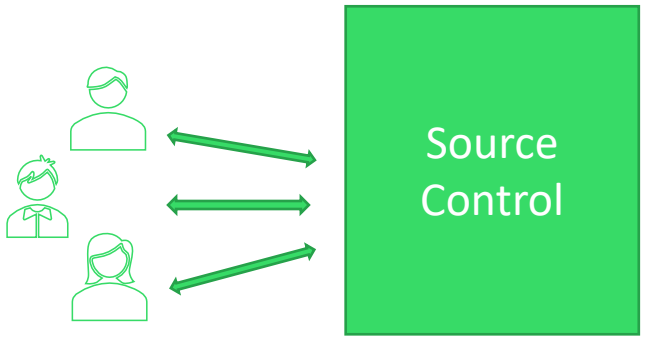
25 1
180 1
25 1
14
47
173 8
249
103
396

47
24 2
14
45 2
159 6

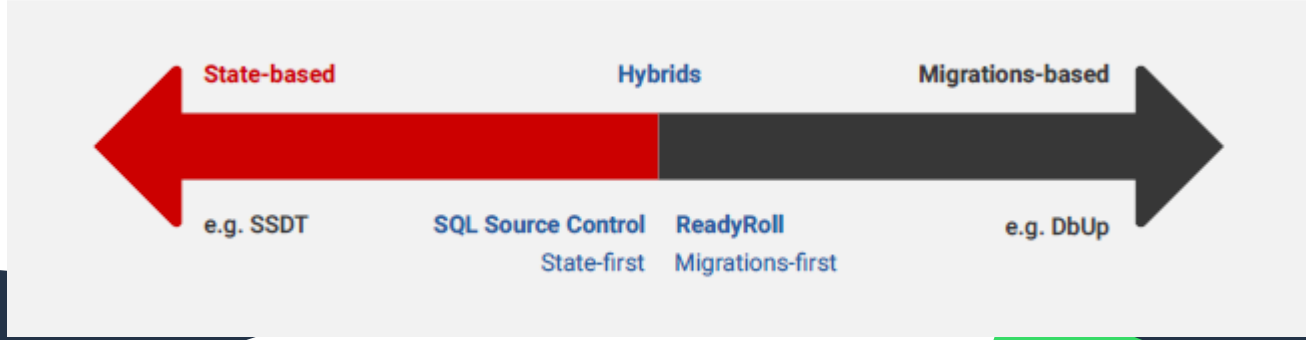
#JOYFULBI


SSDT
Flyway by Redgate
Liquibase.org
Source Control
DBUP
Visual Studio
mercurial
SQL Change Automation
git
SQL Source Control
ApexSQL
Build
Test
Continuous Integration
nunit
tSQLt
Test Driven Design for SQL Server
TC
Azure Pipelines
Release Management
Testing
QA
Staging
Prod
Octopus Deploy
Bamboo
Monitoring & Operations
sql MONITOR
SQL SENTRY
Microsoft System Center Operations Manager

Versioning of databases



- Database is state based – it can't be easily changed
- Two basic approaches
  - Migration Based
  - State Based



State or migrations?

State

- Your source of truth is how the database should be

Migrations

- Your source of truth is how the database should change

Neverending fights

"There's nothing more reliable than keeping track of exactly the scripts you intend to run, and running them, without trying to compare state and guess."
Paul Stovell, built Octopus Deploy

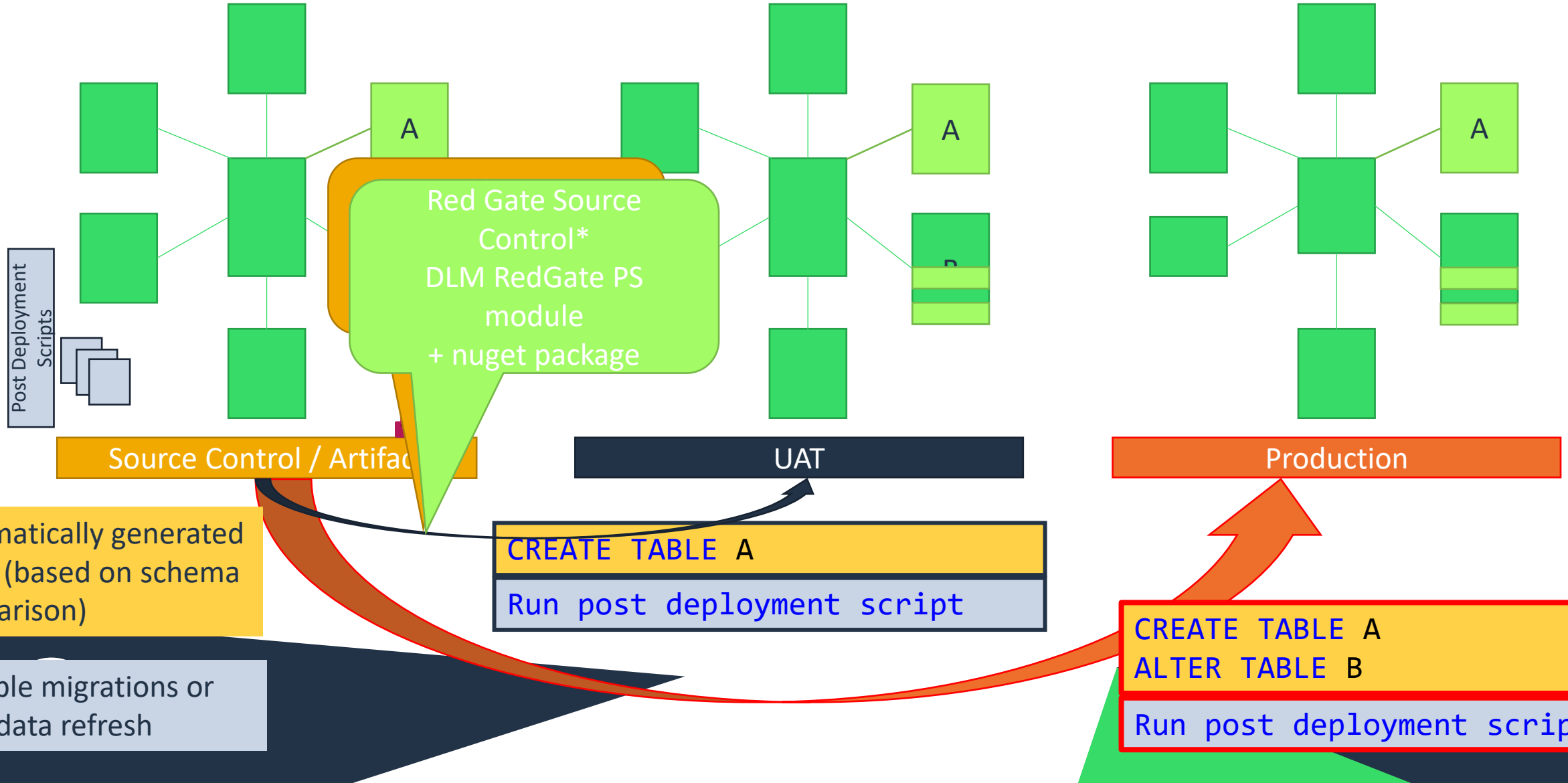
"As soon as you have multiple changes on a single aspect of an object, ordering and the ability to detect which change needs to be made gets very complicated."
Gert Drapers, built DataDude

State based more explained

```
Schema-Model
|-- dbo
|       |-- Tables
|   |   |-- dbo.Customers.sql
|   |   |-- dbo.Orders.sql
|       |-- Types
|   |   |-- dbo.OrderNo.sql
```

- Source control contains
  - All* the objects in the database
- Deployment
  - A magic wand which compares the state in source control and in a target database and creates a change script, which includes all the changes, regardless what the previous state was

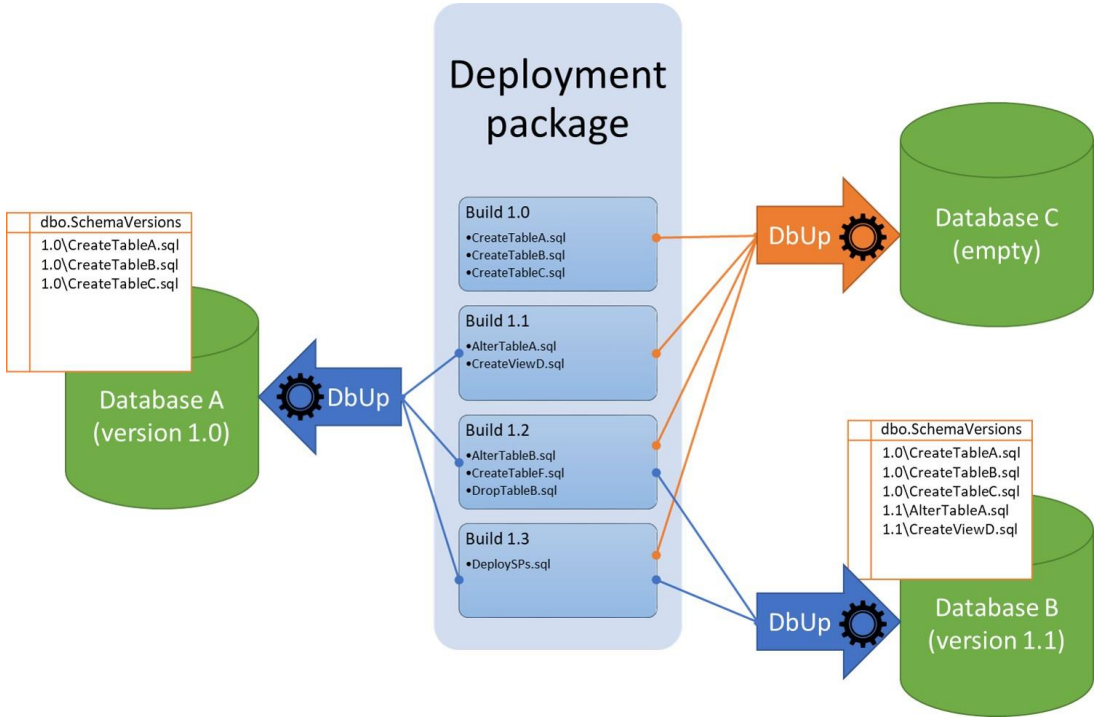
State Based Automation



Migration based explained

- Source control contains
  - All the scripts we want to run against our database
- Deployment
  - Simple run all the scripts against the target which haven’t run yet

Migration based automation



Src: https://github.com/sqlcollaborative/dbops

Major problems with each approach by example

1. Include MiddleName into dbo.GetAccountInfo procedure output
2. Rename the table dbo.Usr to dbo.User

Two developers start to work on their taks simultaneously…

```
ALTER PROC dbo.GetAccountInfo AS
BEGIN
  SELECT Name, MiddleName FROM dbo.Usr
END
```

Rev 112

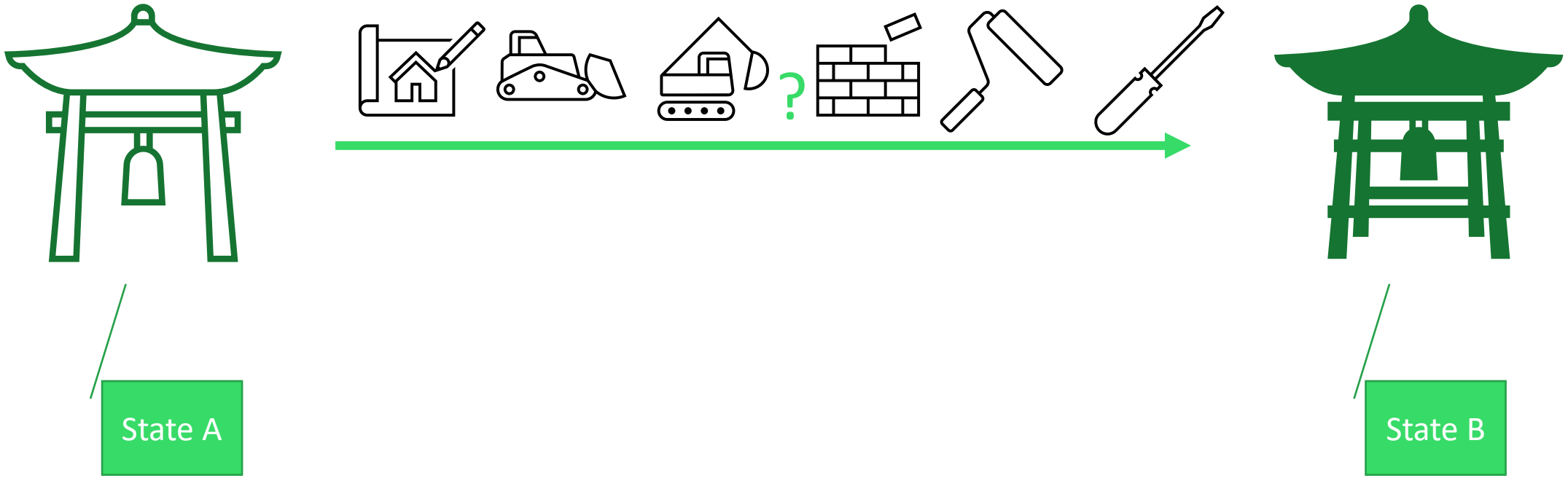

```
EXEC sp_rename 'Usr', 'User'

ALTER PROC dbo.GetAccountInfo AS
BEGIN
  SELECT Name FROM dbo.User
END
```

Rev 113

Migration based

One developer “refactor” the database



State based

One developer “refactor” the database

```
DROP TABLE dbo.Usr

CREATE TABLE dbo.User (id int, Name varchar(50), MiddleName varchar(50))


ALTER PROC dbo.GetAccountInfo AS
BEGIN
SELECT Name, MiddleName FROM dbo.User
END
```

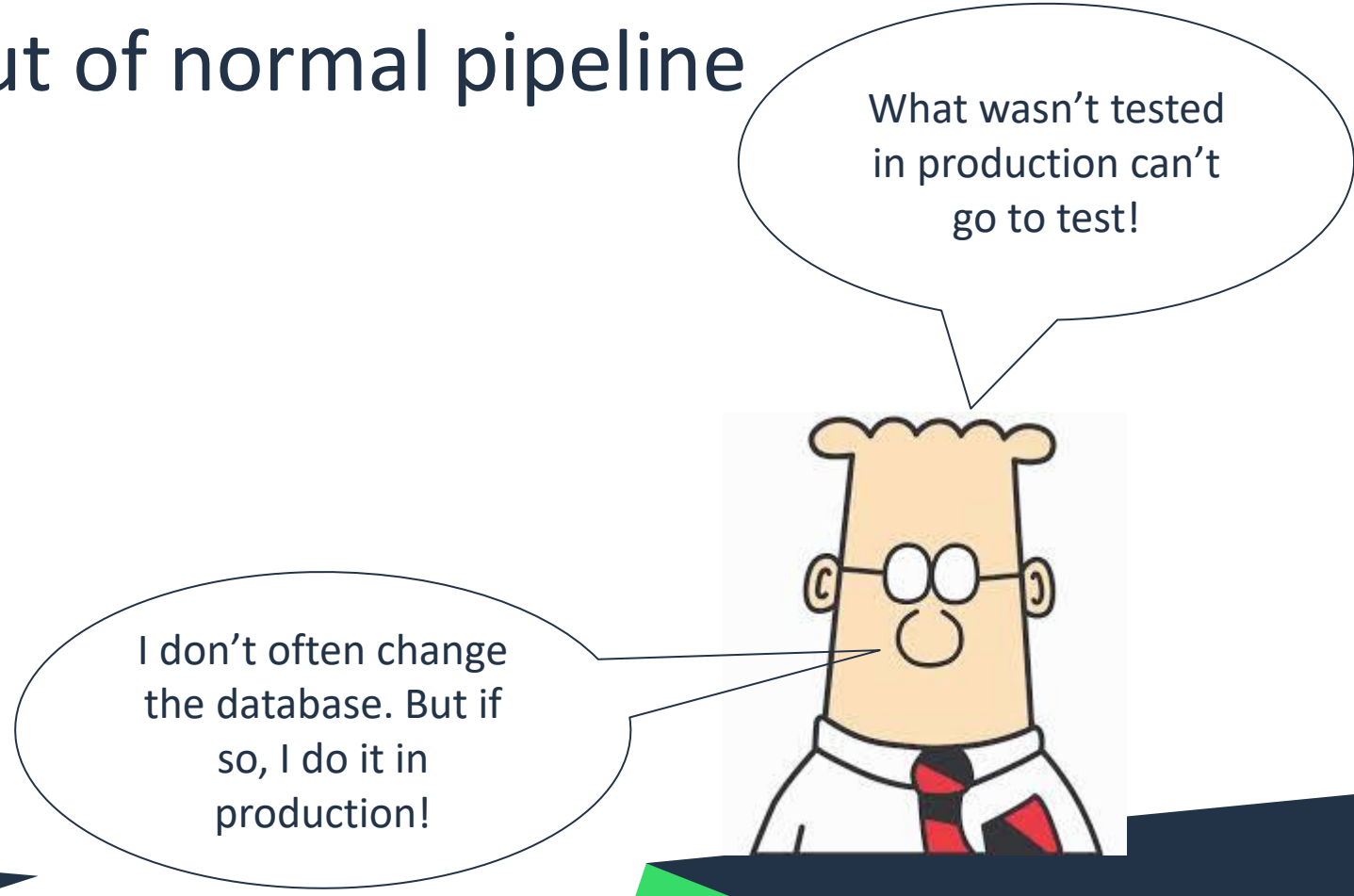
State based

Trouble with both approaches

- Migrations
  - Changes are overwritten and conflicts are missed
  - Last revision always wins
  - Harder to review or manage pull requests
  - How we get the change?
- State based
  - Need to understand and trust your tool
  - Must have a clear rollback plan
  - Must have tests for the data loss
  - Usually longer deployment times

One more thing - drifts explanation

- Changed performed out of normal pipeline
- Samples:
  - New index
  - Nullability of columns
  - Computed columns
  - Partitions



Some of the other key factors

- How easy we work with tool
- Refactoring
- Loss of changes
- Multiplatform
- Customization
- Price
- IDE
- Rollback scenarios
- Suitable for selected development model

Some tools for DLM

Liquibase.org

ApexSQL

Flyway™ by Redgate

SSDT

SQL Source Control

SQL Change Automation

DBUP

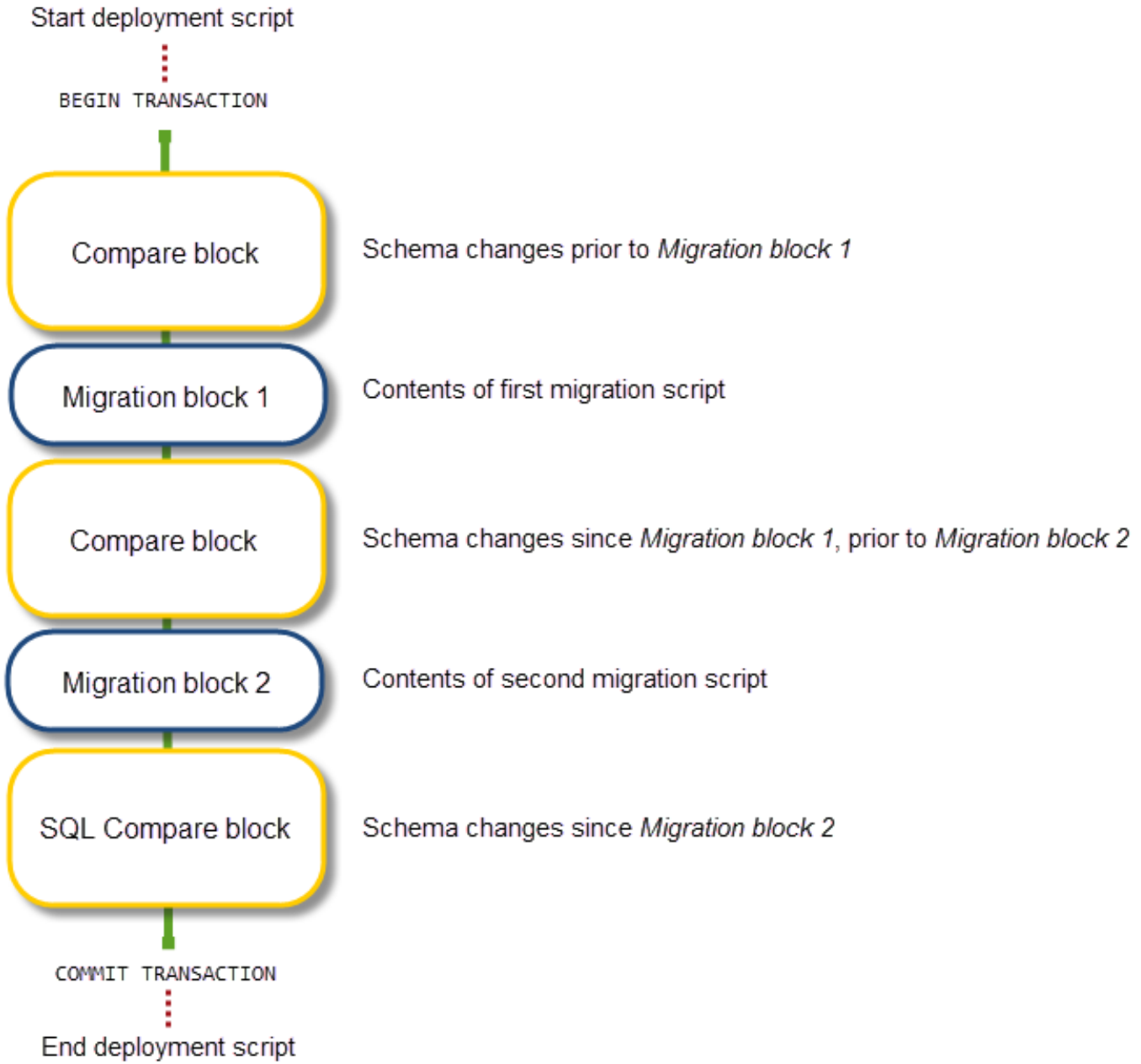
State Based

Migration Based

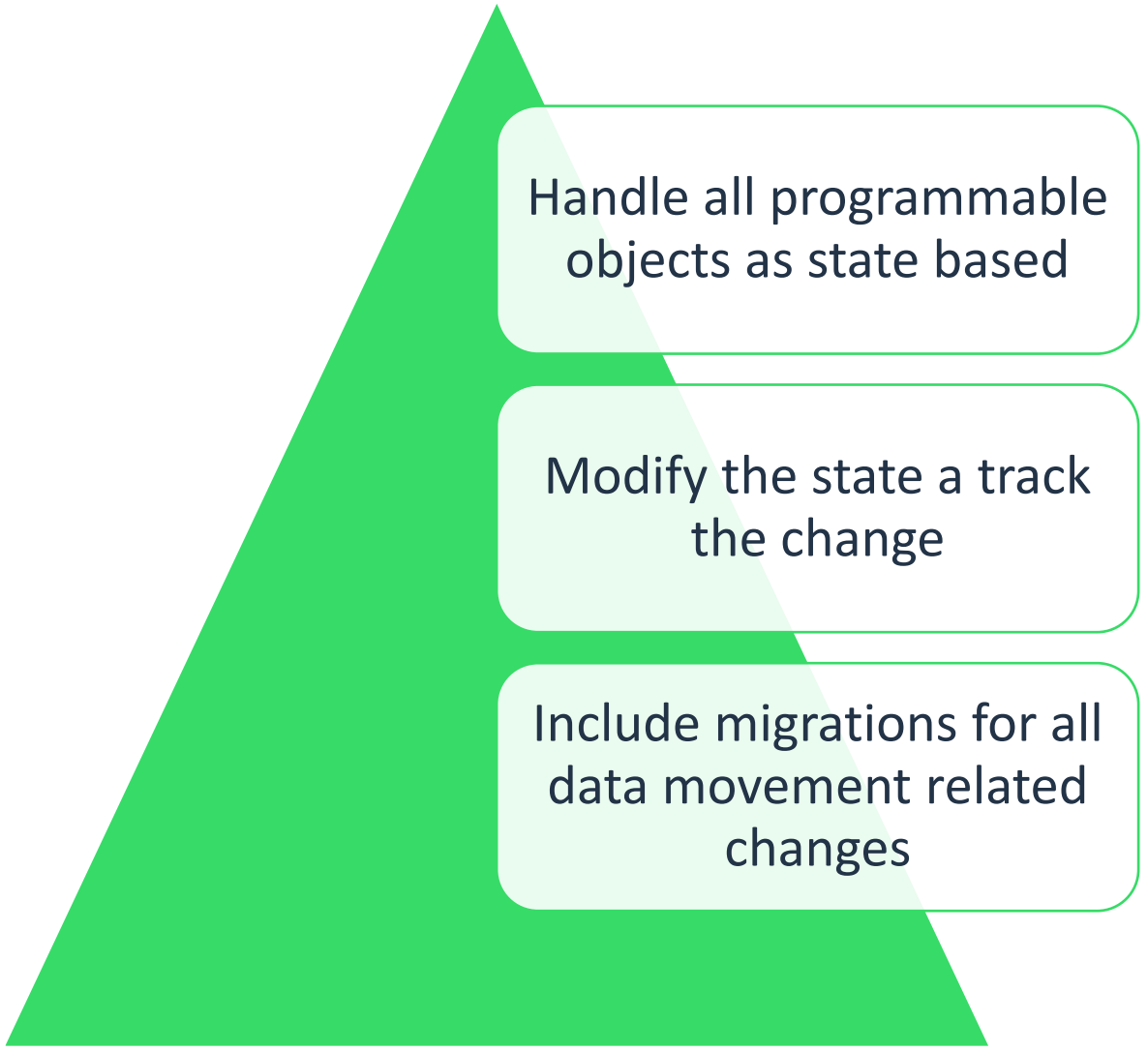
Demo

Let’s see the tools in the action!

Fine tuned state based approach

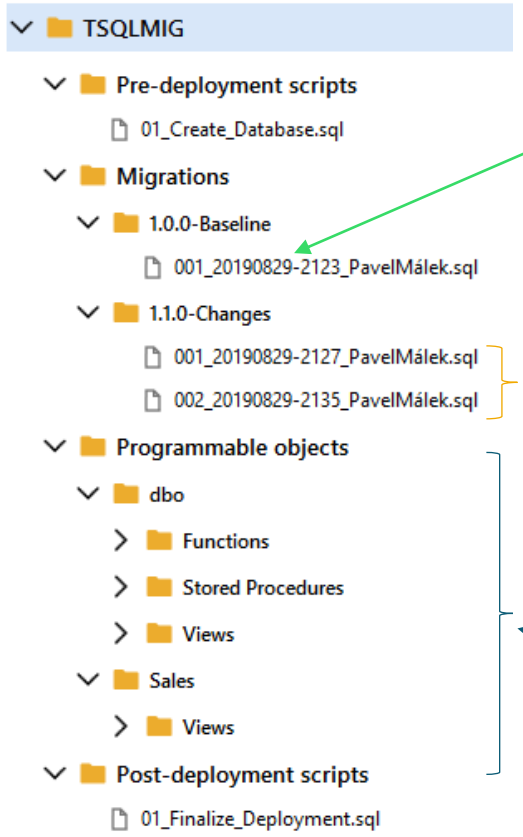


Fine tuned migration based approach



Hybrid approach 1/2

Via SQL Change Automation

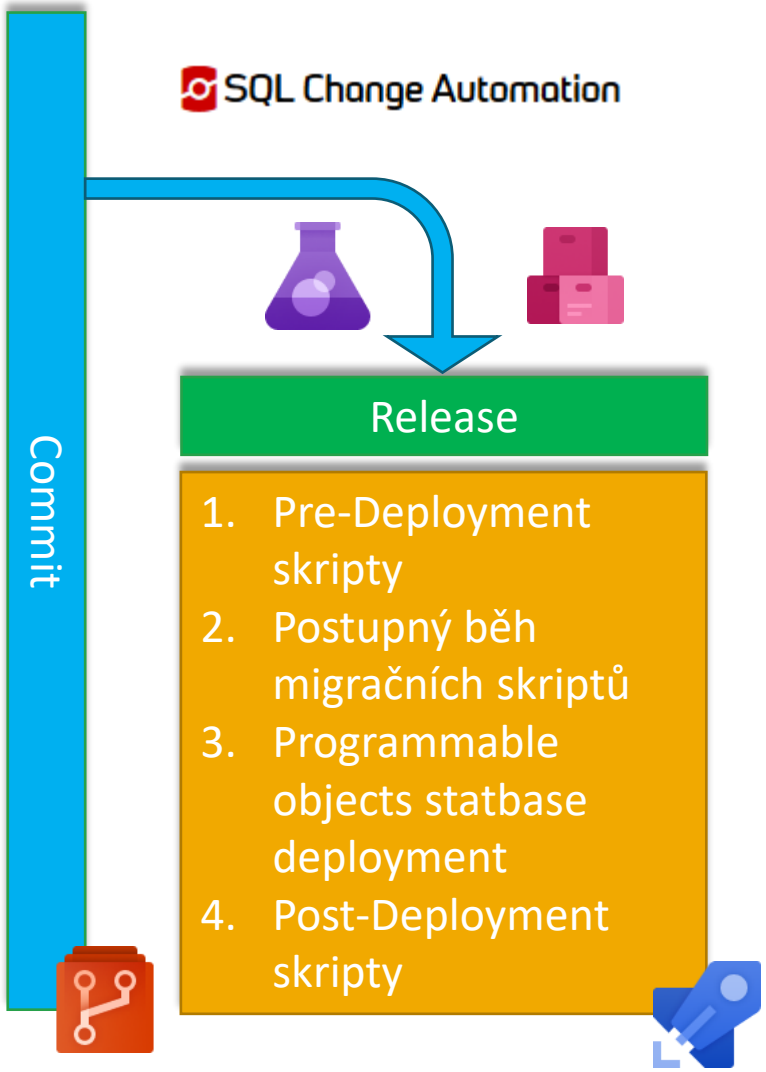


Baseline state z produkce

Feature / Sprint / Verze

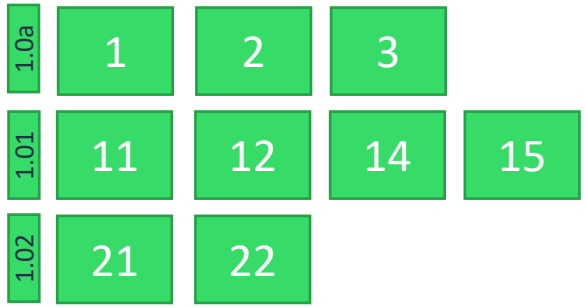
Změnové data releated skripty a případné (meta)datové úpravy

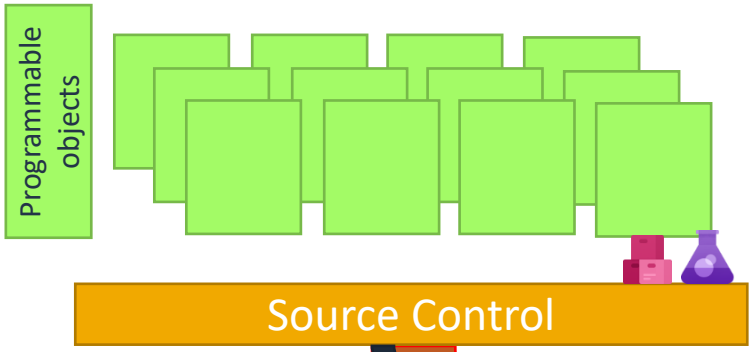
State všech programmable objects

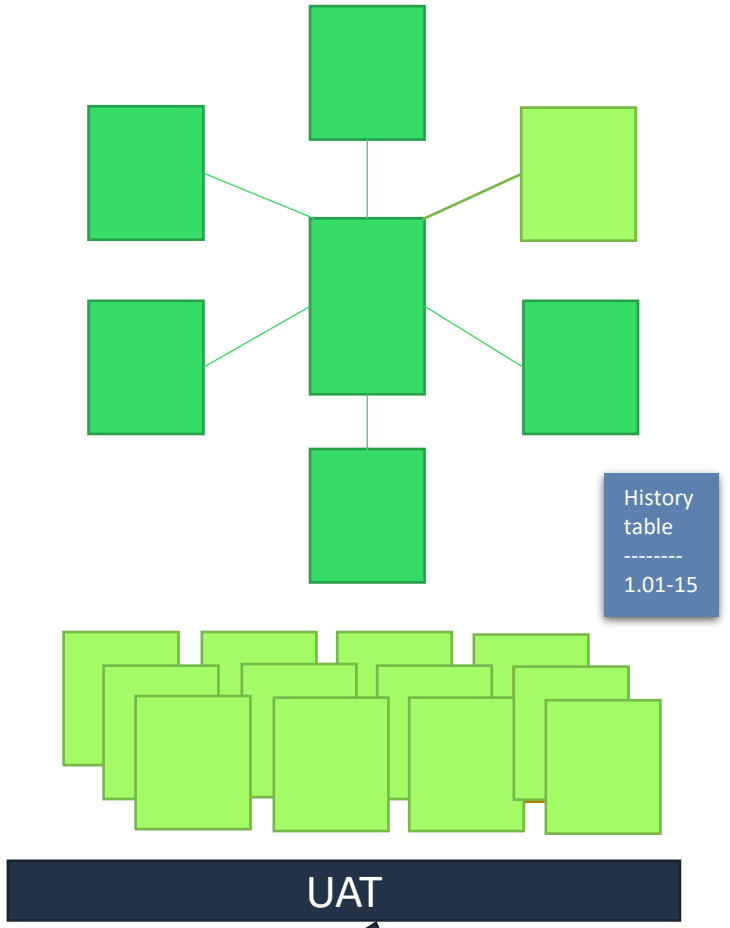


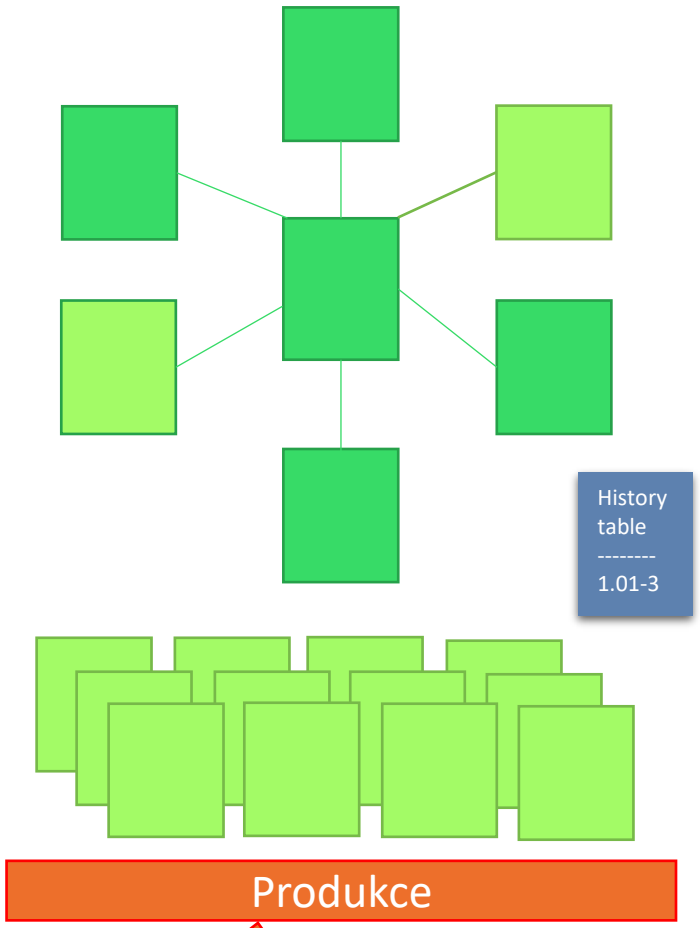
Hybrid approach 2/2

Via SQL Change Automation










```
Run migration scripts from 1.01-15
Compare programmable objects
```

```
Run migration scripts from 1.01-3
Compare programmable objects
```

One more thing!

Dedicated vs shared development model

Dedicated

- Everyone has his own copy of the database
- When work is done, commit goes to a shared repo

Shared

- Multiple developers on one database
- When work is done, selected changes goes to a repo

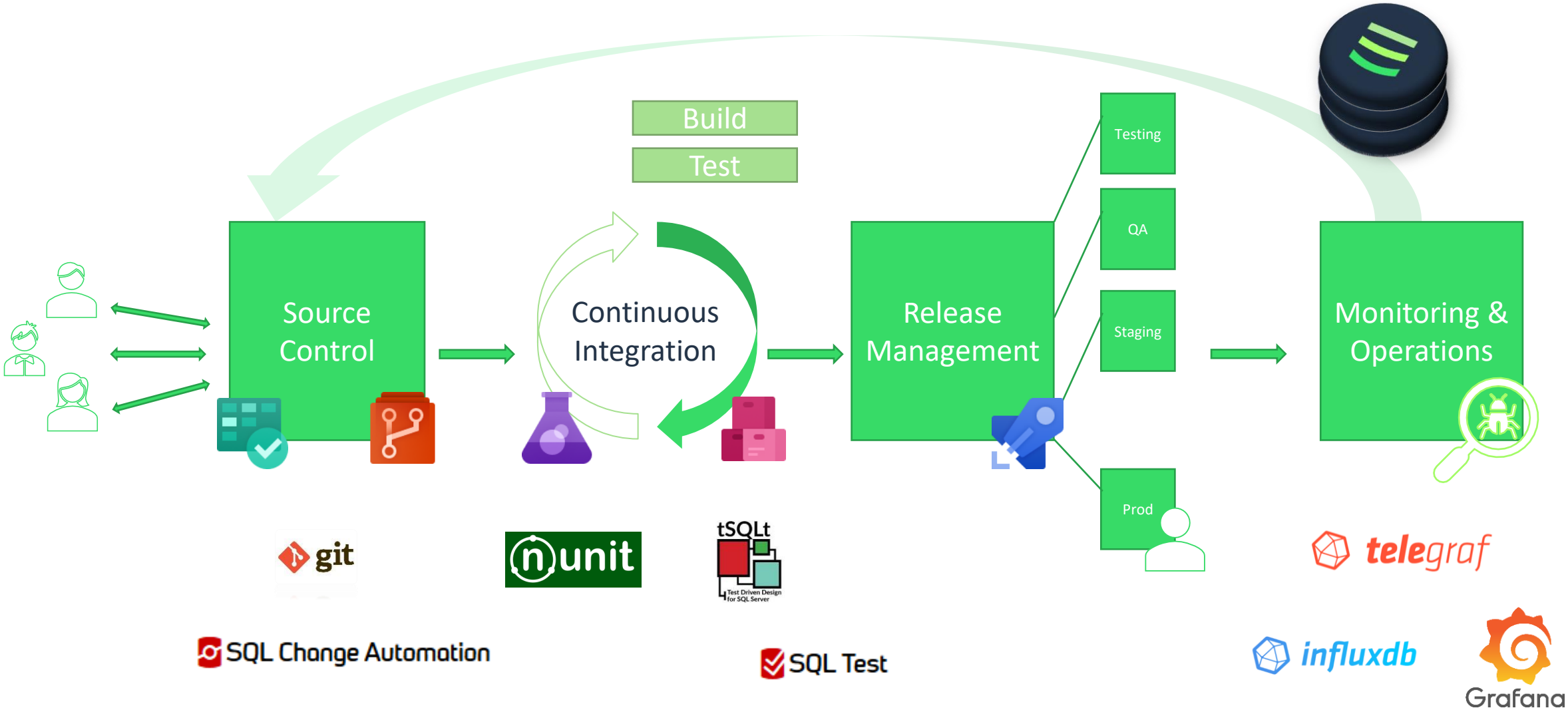
Dedicated vs shared development model

Dedicated +-

- Fast change and commit
- Tests “consistency”
- Branches can be easily used
- Allows experiments
- My house, my castle

Shared +-

- Demanding for proper communication
- Slow for changes detection
- Cannot run test until without risking a break


Build
Test
Source Control
Continuous Integration
Release Management
Testing
QA
Staging
Prod
Monitoring & Operations
git
nunit
tSQLt
Test Driven Design for SQL Server
telegraf
SQL Change Automation
SQL Test
influxdb
Grafana

Summary

- State
  - Easier
  - Way better for programmable objects
  - Works well in large/distributed teams
  - Great for frequent changes
  - Can handle large dependencies
  - DRIFT: rolled back
- Migrations
  - More control (harder to manage)
  - Better for changes which change data schema (migrations)
  - More suitable for small teams
  - Preferred if nr of changes is relatively small
  - No that good for many dependent objects
  - DRIFT: remains

What to choose

- When you have a fortune
  - Choose SQL Change Automation or SQL Source Control
- When you are more app developer
  - Use DbUp or FlyWay
- When you develop a warehouse or DB centric app
  - Use SSDT bud think about migration based approach (DbUp)

Joyful Craftsmen

EVERYBODY MAKING DATA-DRIVEN DECESIONS SMARTLY.

Need support? Let us know!

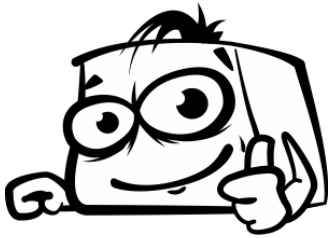
Contact: pavel.malek@joyfulcraftsmen.com

Thank you!

Questions?

Joyful Craftsmen

Business Intelligence Crafted to Joy



Joyful Craftsmen

Business Intelligence Crafted to Joy