Common Database Deployment Gotchas

sabin.io

Simon D'Morias

SQL Server Consultant @ Sabin.io

Microsoft Certified Master: SQL Server

MCSE: Data Platform & Business Intelligence

simon.dmorias@sabin.io

sabin.io

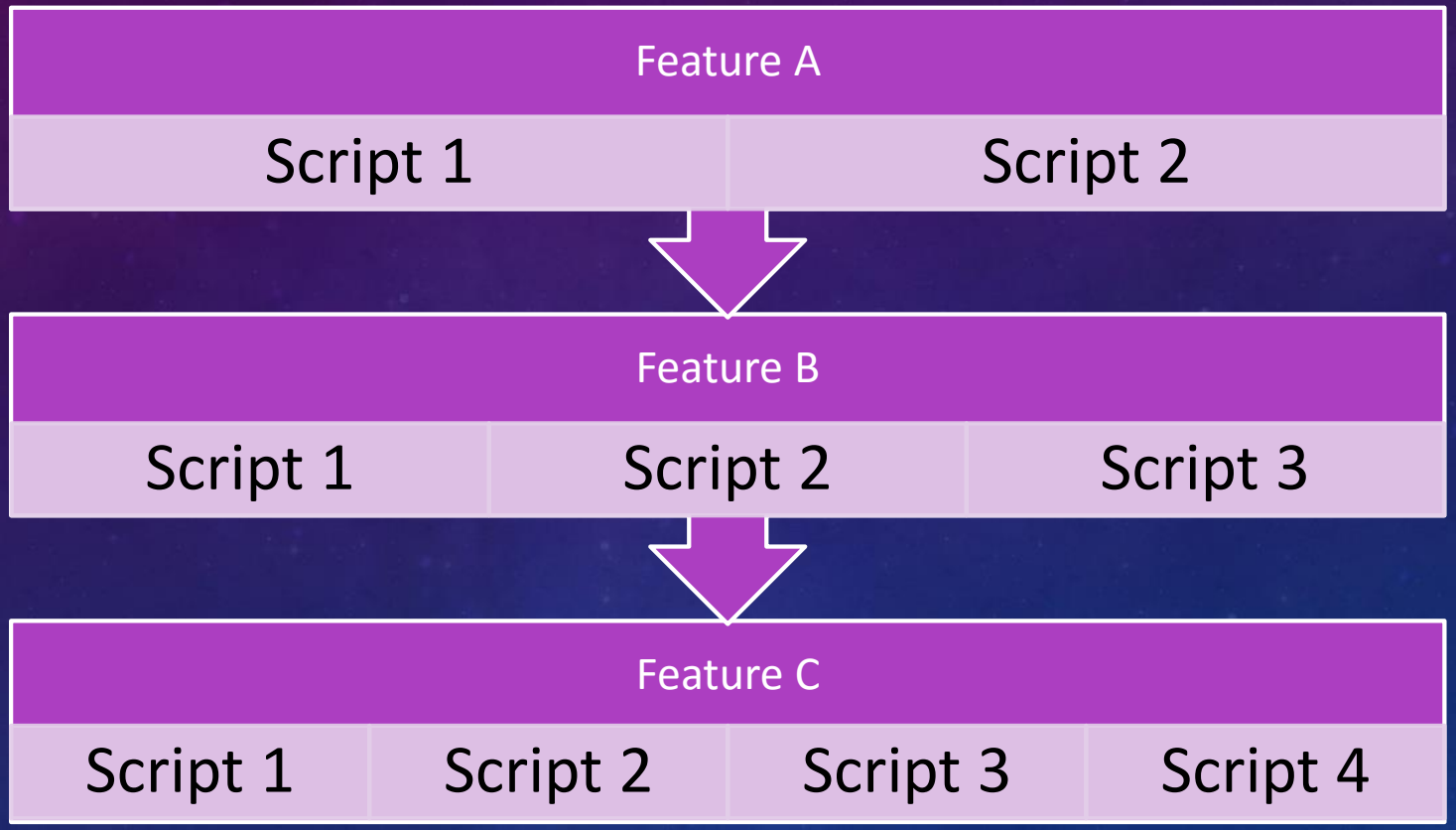
Why database deployments are complicated?

- There is usually just one database
- Active Connections
- Database State
- Release rollbacks are often impossible

Gold Schema vs Migration Scripts

- Point in time representation of our database
- Migration take you from A to Z via B, C, D...
- Gold Schema goes direct from A to Z

Migration Scripts



Pros and Cons of Migration Scripts

PROS:

- Simple to manage (to start with)
- Schema and Data same process

CONS:

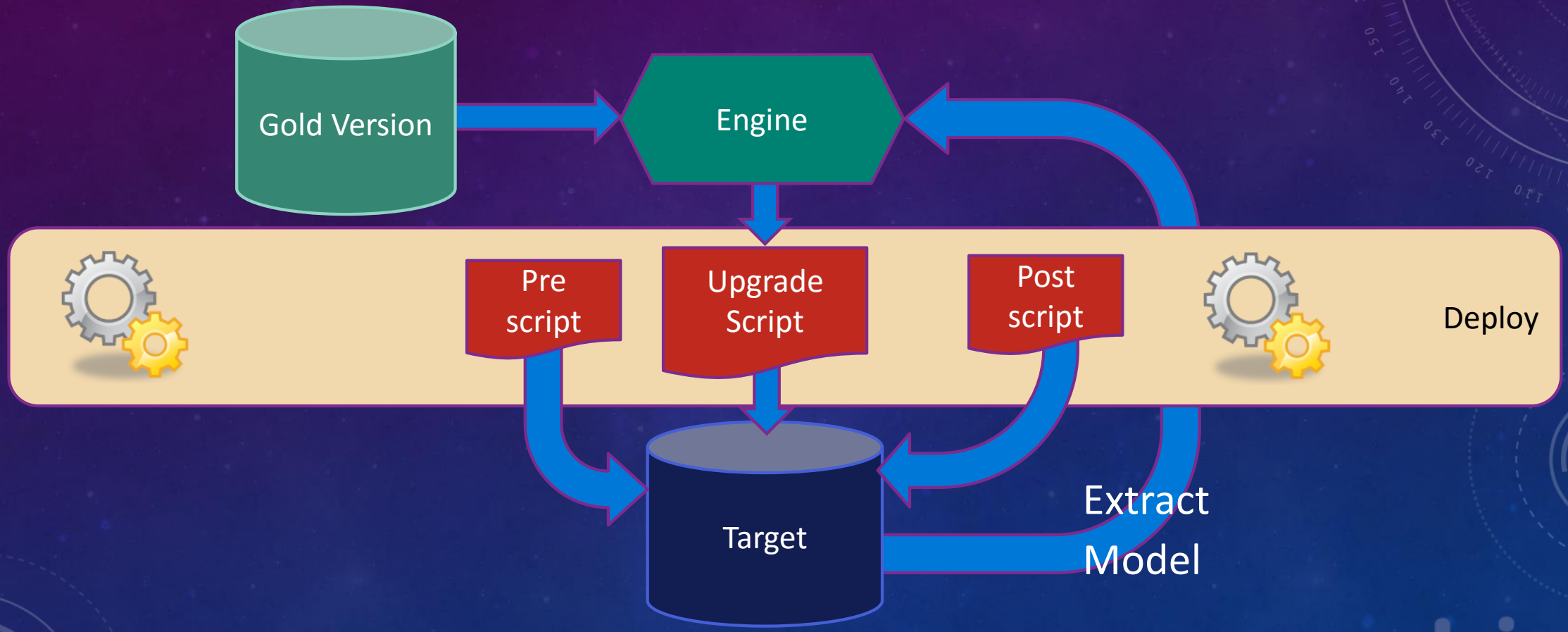
- No proper Source Control
- Transactional deployment not possible

Common Deployment Problems

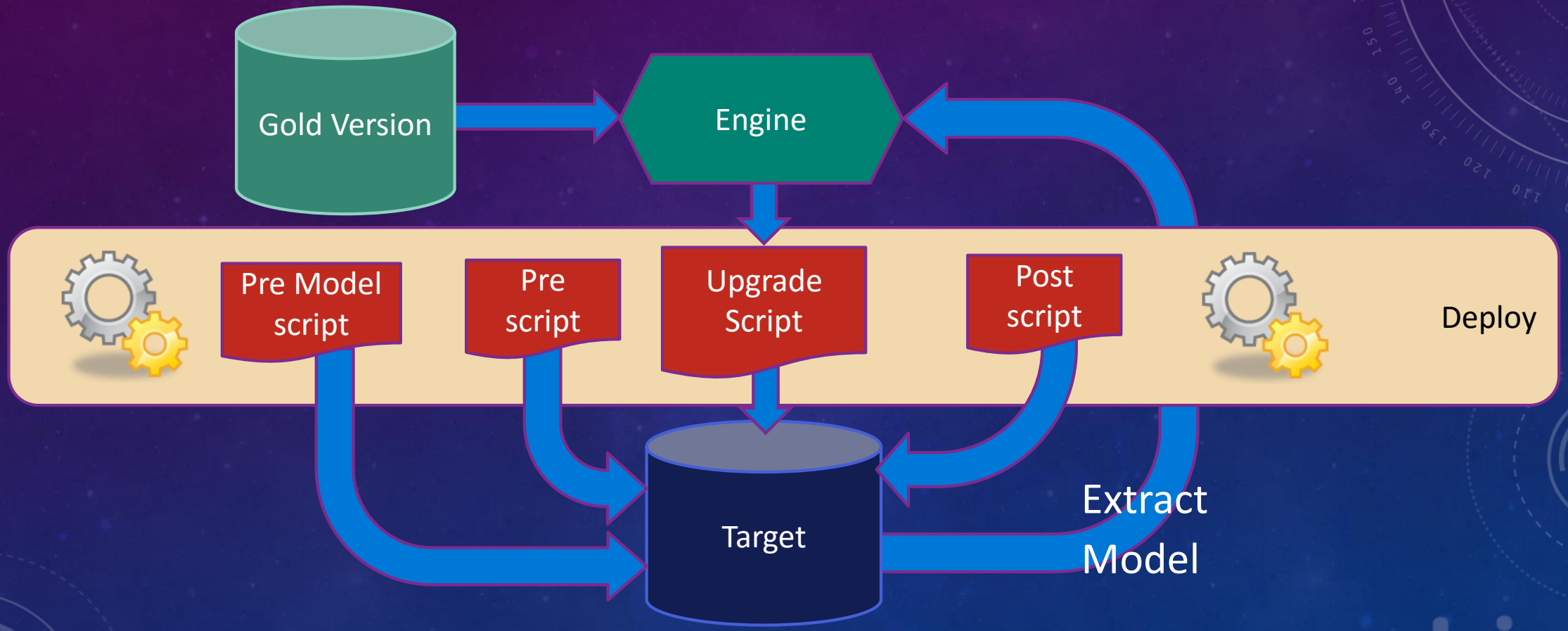
Code vs Provisioning/Operational Changes

- Environments are provisioned to a specification
  - Installed SQL Server
  - Configure
  - File Layout
- Decide what is managed in code and what is not
  - Partitions
  - Files
  - Compression
  - Security

Gold Schema Order of Execution



Reality of Gold Schema



Logins & Security

- Logins are optional
- Consider using Roles for Application Permissions
  - Apply permissions to the roles only
  - Post Deploy create Logins/Users/Role Mappings for environment
- Do include permissions in your project
  - For audit purposes if nothing else

Scripting per Environment

```
IF '$(Environment)' = 'DEV'
BEGIN
    CREATE LOGIN blah FROM WINDOWS 'DOMAIN\user';
    CREATE USER blah FOR LOGIN = blah;
    ALTER ROLE rl_blah ADD MEMBER blah
END
IF '$(Environment)' = 'PROD'
BEGIN
    CREATE LOGIN blah2 FROM WINDOWS 'DOMAIN\user2';
    CREATE USER blah2 FOR LOGIN = blah2;
    ALTER ROLE rl_blah ADD MEMBER blah2
END
```

Releasing Online

- Small Releases
- Large releases in a transaction will cause problems
- Watch out for lock escalation
  - Use batches
- NOT NULL with a Default is slow depending on version
  - 2012+ ok

Replication

- Always requires custom scripts
- Publish Profiles have options to prevent deployment
- Subscribers:
  - Ensure columns are added Pre SSDT Comparison runs
  - Possible Pre-Model Script to add to the publisher
- Publishers
  - Either disable the replication of schema changes (large tables)
  - Or just allow them through to the subscriber
- Add procs to project to drop/create your replication
  - Execute in pre/post deployment when needed

Change Data Capture

- Allow 2 instances of monitoring
- Only an issue when adding new columns that must be tracked
- Write a wrapper function to the CDC functions
- Use a pre-Model script to:
  - Add new column(s) – or use 2 releases just adding the column in the first
  - Add new instance
  - Process through instance 1
  - Remove Instance 1
- Change wrapper function in project/release

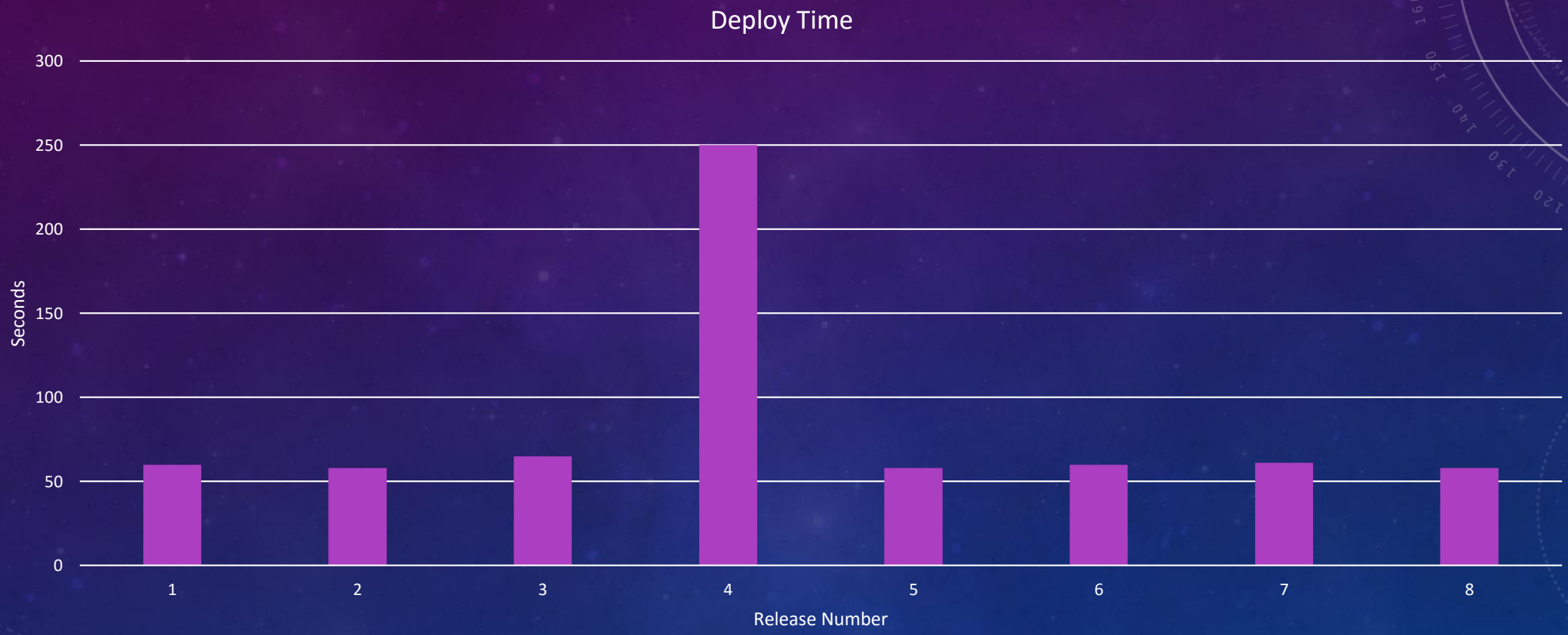
Custom Scripts

- Run using PowerShell and SQLCMD
- Inject variables as needed (use the same names as SSDT)
- Useful for jobs/Replication/SSIS deployments etc

Deploy Contributors

- Available on GitHub as DACExtentions
- Allow you modify/inject code into the SSDT output scripts
- Contribute your own

Monitor Deployment Time in Test/Staging



sabin.io

Rolling Back (or not)

- Restore is the only true rollback
- Can you redeploy a previous build?
  - It depends
    - If data has changed - No
    - If procs/views only then “maybe”
- Management often fail to understand this
- Forward Only
- Rollback is the last resort

Takeaway

- Keep everything in SSDT
  - Even Pre-Model scripts – copy to output folder
- If your project will not build today:
  - Mark items as “not in build” or delete them
  - Move cross referencing objects to post deploy script
  - Make at least part of the database build (even if just tables)
- Every scenario is different
  - Solve with custom scripts
- Work to include the complexities in your test environments

Recommendations

- Release Often
- Release Small
- Only release when you are confident
- Feature Flags are a great way to avoid rollbacks