DBdeployer

About the Speaker:

CJ Estel, Operations Database Engineer
CoverMyMeds
cj@cjestel.net

Github:
https://github.com/covermymeds/dbdeployer
https://github.com/cjestel

Company Tech Blog:
https://www.scriptscribe.org/

Some Quick Definitions

CRUD: Acronym for Create, Read, Update, and Delete

DDL: Data Definition Language

DML: Data Manipulation Language

SLA: Service Level Agreement

Some Common Database Administration Problems

- Lots of environments and they all have different SLA's and deployment schedules
- DBA's need early visibility into changes to audit them for normalization, architecture, and impact

Problems Continued

- Application users should not have more permissions than they require by the application (CRUD)
- Developers would like to be able to stand up a database that has minimal seed data allowing them to work and test locally

Problems Continued

- Different environments could have different seed data, different schemas or different table structures
- Some replication strategies do not easily support DDL changes and require a coordinated deployment effort
- Deployments must meet company change control requirements

Production Change Requirements

- Production changes shouldn't be able to be made to the database outside of:
    - the application
    - an administrative interface
    - a peer reviewed change deployed through a known and tested process
    - directly through change control

Some things We Considered

- Ruby Rails Migrations
- dbmaintain
- deltasql

Problems with these tools

- Some tools deploy via application user requiring elevated permissions
- These tools rely on a driver layer to interact with the database instead of using native tools
- Tools are great for schema but do not handle data maintenance well
- Tools require remote access with an administrative account
- Some tools do not handle intended differences in environments, seed data, or schemas

Ruby Migrations in Particular

- Intentionally built to use as generic of sql as possible which may not take advantage of features available to your system
- Can reduce visibility on what sql will actually be deployed to servers (i.e, Has no regard for how long tables may be locked during a migration against a large table)
- Migrations can be difficult to stage if they have downstream dependencies
- If more than one app uses the same database, which app manages database setup

What is DBdeployer

- A command line utility written in-house that:
  - Executes sql via the native database binary and is compatible with MSSQL, Postgres, and is extensible to other RDBMS's
  - Tracks deployments to each database
  - Meets change control requirements
  - Has a one touch button to bring a database to current state
  - Allows for one off deployments to fix application data
  - Ensures we always have a "tiny db" for each database managed with it
  - Supports differences in a database based on server or environment

Advantages of DBdeployer

- Changes are reviewed as the sql that will be executed. This can help decide if an ORM generated the best sql to apply in prod
- Changes can be audited in advance so that ETL jobs or replication that needs staged is less likely to be affected
- Database changes are scripted and can be deployed like a regular application
- Database changes are managed separately from the application

Advantages of DBdeployer (cont)

- Can be used to facilitate staged rollouts of database changes where trigger based replication is utilized (update idle/reporting node first)
- Can tie into a monitoring system to alert if an environment is out of date
- Environment specific data can be handled with DBdeployer

How Does DBdeployer Work?

- There are folder locations you can store sql files:
  - schema
  - seed
  - changes
  - rollback
  - archive

How Does DBdeployer Work (continued)?

- Written in bash which ensures it calls the database binaries the same way you would without relying on a driver layer

- Tracks deployments in its own database

- Deployment ordering is based on the default alpha-numeric ordering of the file system. It is recommended to use a date-based naming convention for files

Our Workflow

Dev → Feature Branch, PR, add review tag → GitHub

DBA → Test, audit for safety to the system → Approve or provide feedback → GitHub

DEV → Modify or request testing deploy → Add tag ready for test → GitHub

Our Workflow (Continued)

INF

Deploy test server with dbdeployer



INF

Comment Repository, tag in test



DEV

Test application

Tag branch ready for Production



INF

Merge to master

Deploy all servers



Will the demo Gods cooperate?

- A quick demo...