DevOps for DB2 DBAs

Julian Stuhler
Triton Consulting

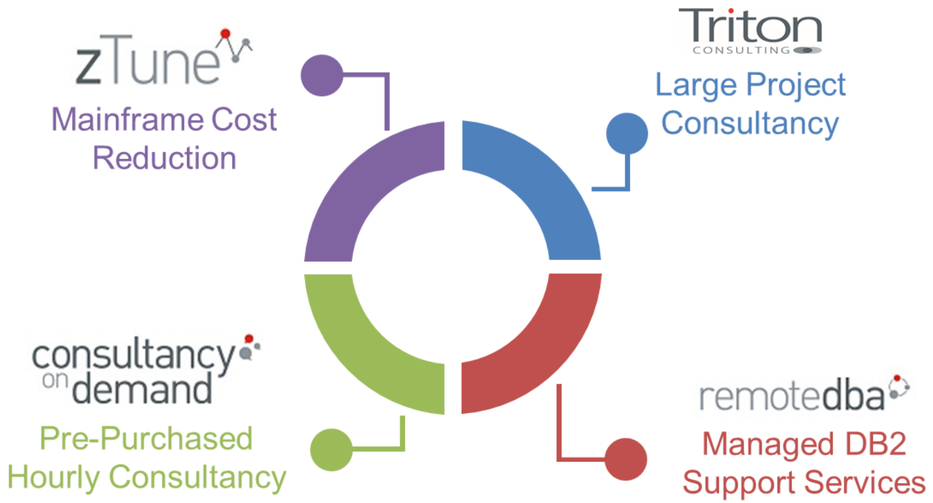
DB2Night Show
20th March 2020

Agenda

- Introduction
- DevOps – A Definition
- DevOps – Why Should I Care?
- DevOps for Data
- Case Study – Automating DB2 Schema Change Deployment
- Summary

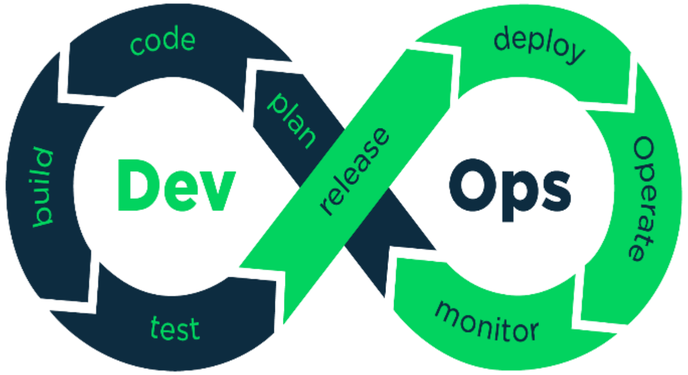
Introduction

- DB2 consultant with Triton Consulting, based in the UK
- 33 years DB2 experience
  - Database Administration
  - Systems Programming
  - Application Development
  - DevOps
- IBM Gold Consultant
- IBM Champion
- IBM White Papers, Redbooks, Flashbooks, etc
- IDUG Best Speaker and Past President



DevOps – A Definition

- A software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops)
- Strongly advocates automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management
- DevOps promises shorter development cycles, increased deployment frequency, and more dependable releases, in close alignment with business objectives
- Emerged as a concept around 2007, but many of the underlying principles have been with us for much longer
- Enterprises adopting DevOps practices for both distributed and mainframe
- Close relationship with agile and cloud computing



DevOps – People, Process, Tools (in that order!)

People

Significant culture change, removing “the wall”
Needs input / buy-in from Dev **&** Ops
Engineers not Developers, DBAs, etc

Process

Aligns with Agile methodologies
Emphasis on frequent, small changes (MVP)
Same process to deploy in all environments
Fail fast / shift left
Robust feedback loops – continuous improvement

Tools

Major emphasis on automated testing & deployment
Open source toolchains available for almost everything
Rocket a major enabler on Z

DevOps – Toolchains

- Vast array of proprietary and open-source tooling available
- High degree of interoperability
- Each environment is different, clients often free to mix and match what works best for them

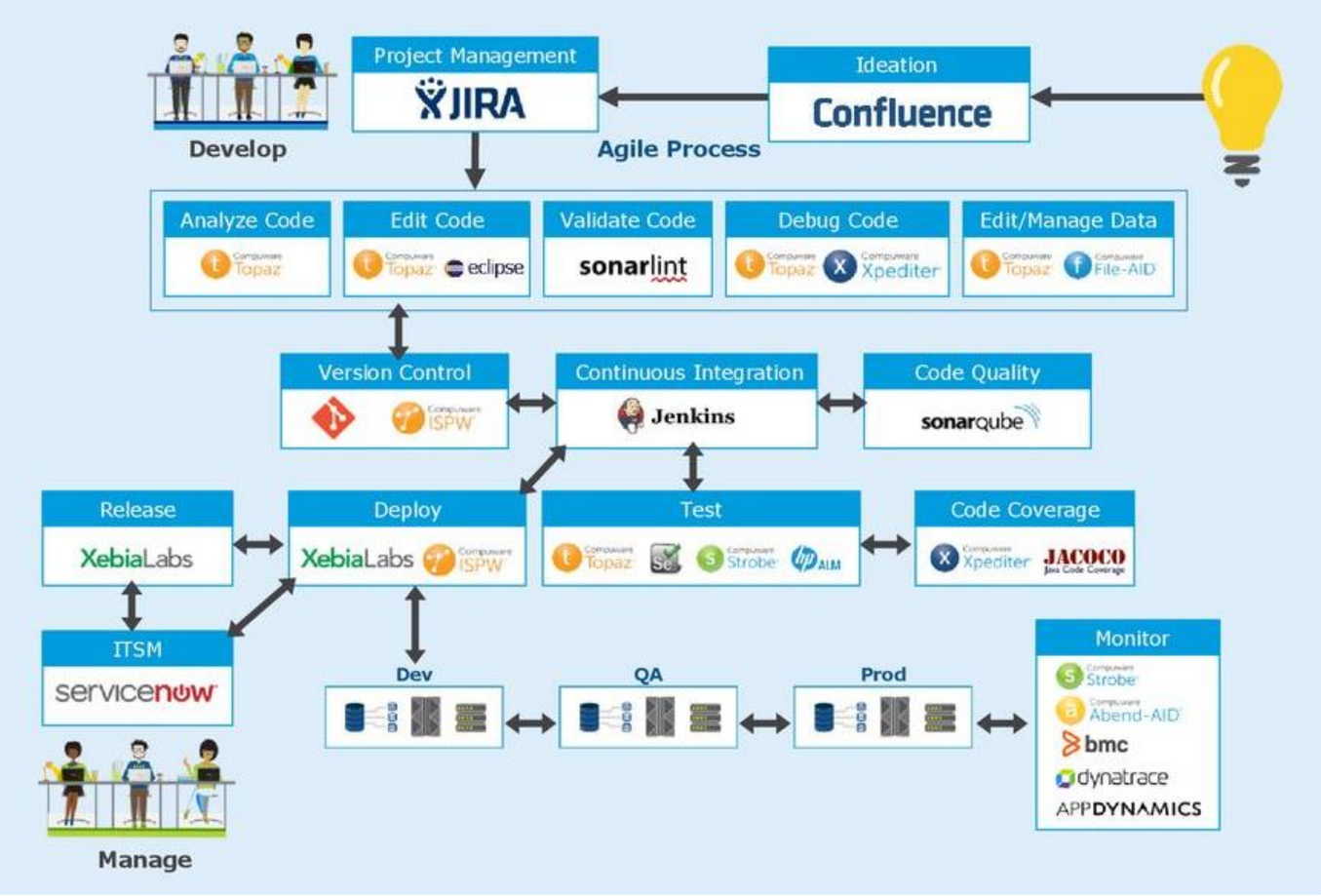


Image: courtesy of Compuware

DevOps – Why Should I Care?

- IBM has embraced Agile / DevOps principles, and we are already starting to see the impact
  - Continuous delivery of new function for DB2 for z/OS (from V12 onwards) and DB2 for LUW (from V11.1 onwards)
  - Combination of continuous delivery and APPLCOMPAT features in DB2 for z/OS will have significant potential impact on method and frequency of new function rollout
- Your organisation is probably already adopting DevOps practices for in-house development (even if they are not referred to as such)
  - DevOps is not new, but adoption by larger enterprises is more recent and continues to build momentum

DevOps – Why Should I Care?

- DevOps will have a significant impact on traditional DB2 DBA and Sysprog / Sysadmin roles
  - New development paradigms and tools must be understood
  - Self provisioning of new DB2 environments and schemas
  - Automated delivery of schema change
  - Automated QA checking for DDL, SQL, stored procedures, etc
  - Remove organisational / cultural barriers between traditional dev, DBA, sysprog roles
  - Potential to reduce amount of routine donkey work and free up time for higher-value work such as database design, performance tuning, etc
- DevOps encourages engineer empowerment, and culture is typically to first challenge and then route around any “blockers”
  - Traditional support / operations roles must evolve or die

DevOps for Data

- DevOps as a movement has existed for several years, but is now being taken in new directions
- Enterprise Focus
  - Many “Digital Native” companies such as Google, Facebook and Amazon have successfully used DevOps for many years, but usage has now spread to more traditional enterprises (e.g. banks, building societies)
- Expansion to include automation of database activities as well as code
  - Test data management
  - Reference data management
  - Quality assurance / static analysis for DDL and DML
  - Schema deployment to new environments
  - Automated Schema Change Deployment (ASCD) to existing environments
- Combination of these factors has lead to significantly increased interest in “DevOps for Data”

ASCD – Considerations

1

Industry Maturity

DevOps not new, but many sites have not yet tackled database automation.

2

Tooling

Correct tooling will be critical in determining how robust the automation can be.

3

Approach

Early decision needed on overall approach (Compare v Delta).

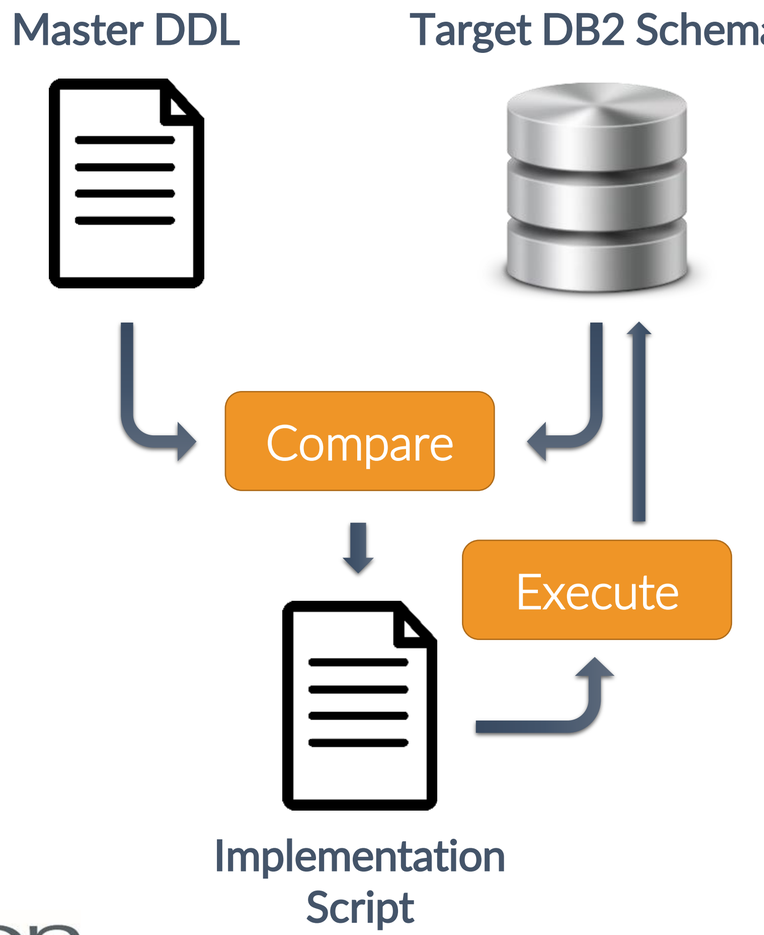
4

Control

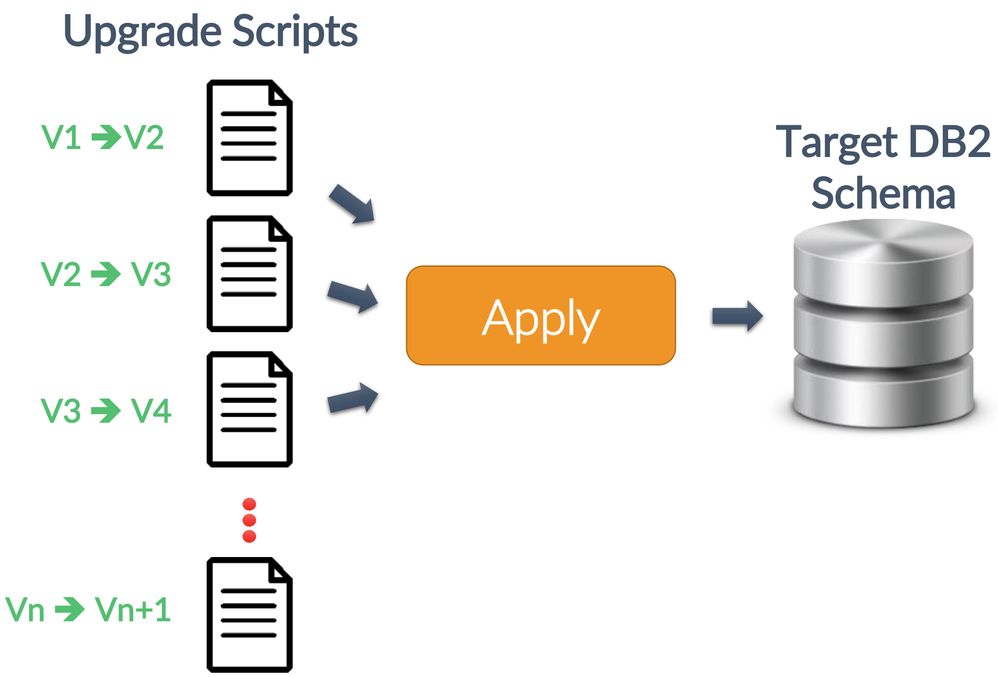
Self-service is the ultimate goal, but essential to keep a DBA approval step in the process

ASCD – Approach Options

Compare Approach



Delta Approach



Triton CONSULTING

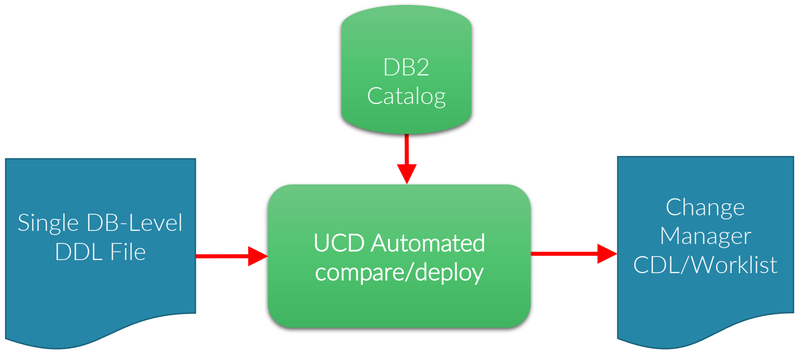
ASCD – Approach Trade-Offs

	Compare Approach	Delta Approach
Pros	• Very robust – can take a given environment from any version to any version with a single operation • Guaranteed consistency – as environment always compared with "master" DDL, all environments remain consistent. Any "ad-hoc" local changes will automatically get removed on next environment refresh • Easy to create new environments – just compare required DDL master with blank environment • Minimal inputs required for deployment once set up – just master DDL and name of target environment • Easy roll-back – just run compare on previous version • Master DDL can be source-managed in Git	• Open source tooling available such as Liquibase • Probably closer to existing deployment processes so may be easier to implement • As a result of the above, more general understanding/experience in the industry • Scripts can be source-managed in Git • Same script used in all environments, can be more predictable

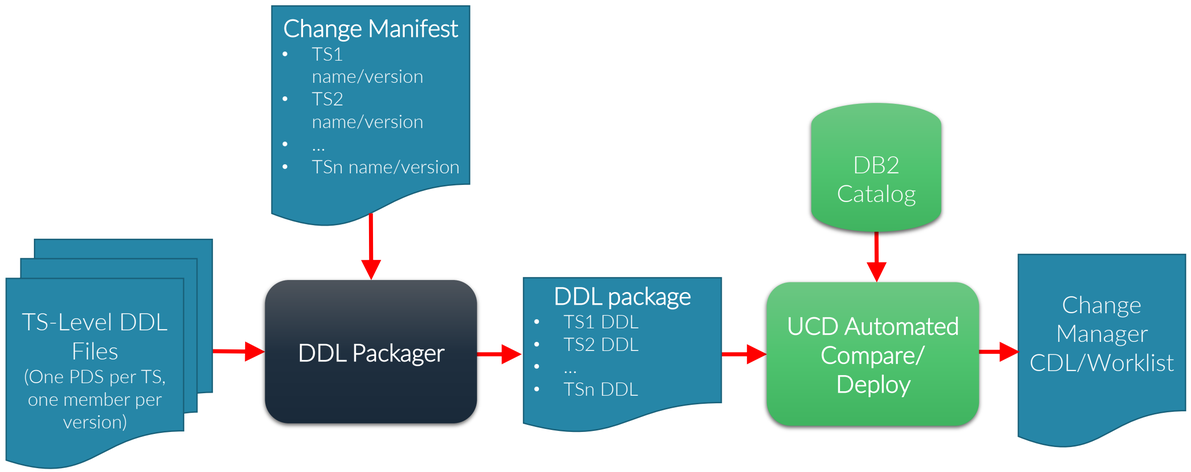
Cons	• Needs a robust schema compare tool for DB2, which will probably be chargeable.
	• Likely some up-front work required to get agreed baseline of existing schema, depending how messy existing environment/process is
		• Any differences between environments (e.g. ad-hoc changes) will remain following upgrade
		• Some environment differences may cause delta script to fail as it expects DB to be in a given state
		• Creating new environments requires many different scripts to be run sequentially (or ongoing effort to rollup scripts)
		• More thought/manual effort required to get inputs
		• Rollback can be painful – need to create manual rollback scripts under some circumstances
		• Some variants/features still cost money
		  ◦ Liquibase Pro
		  ◦ Flyway

ASCD – Compare Granularity

- Flexibility v complexity trade-off for granularity of DDL used for compare
  - Database-level gives simpler process and easier tracking, but can cause issues in agile environments with multiple sprints proceeding on RTL at different speeds
  - Tablespace-level* makes process more complex (may need to glue together several bits of DDL for a given deployment), but gives more flexibility in agile / parallel development environments



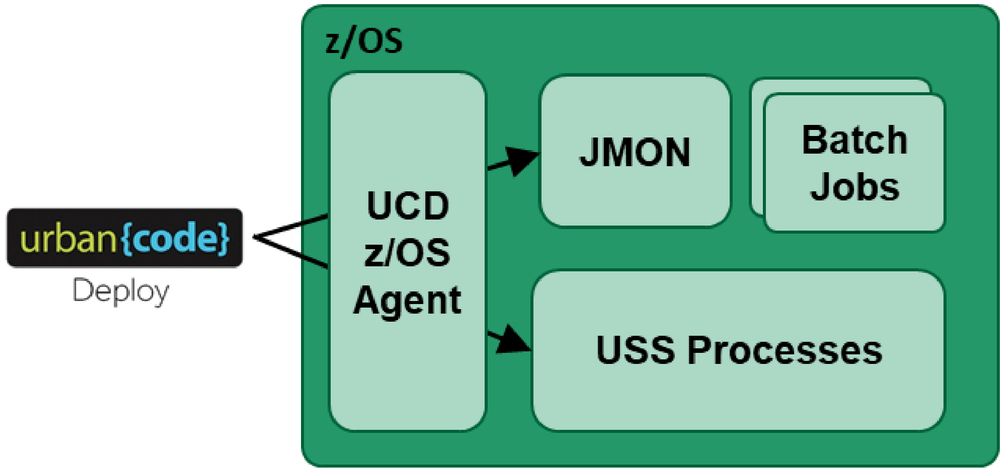
DB-Level Master DDL



TS-Level Master DDL

*TS-Level DDL includes all dependent objects (tables, indexes, aux tables/tablespaces, etc)

ASCD Case Study - Background



- IBM's UrbanCode Deploy (UCD) used for automating the deployment of application changes into all environments (non-prod and prod)
- UCD has a z/OS Agent that is capable of
  - Invoking arbitrary z/OS JCL, and passing return codes back to UCD
  - Invoking Unix System Services (USS) commands

bmc

BMC Change Manager

- Meanwhile, the central DBA Team used traditional, well-proven process to deliver DB2 schema changes
  - BMC Change Manager generates JCL to implement changes into a given environment, but with manual intervention / edits often required

ASCD Case Study – Problem Statement

Problem Statement

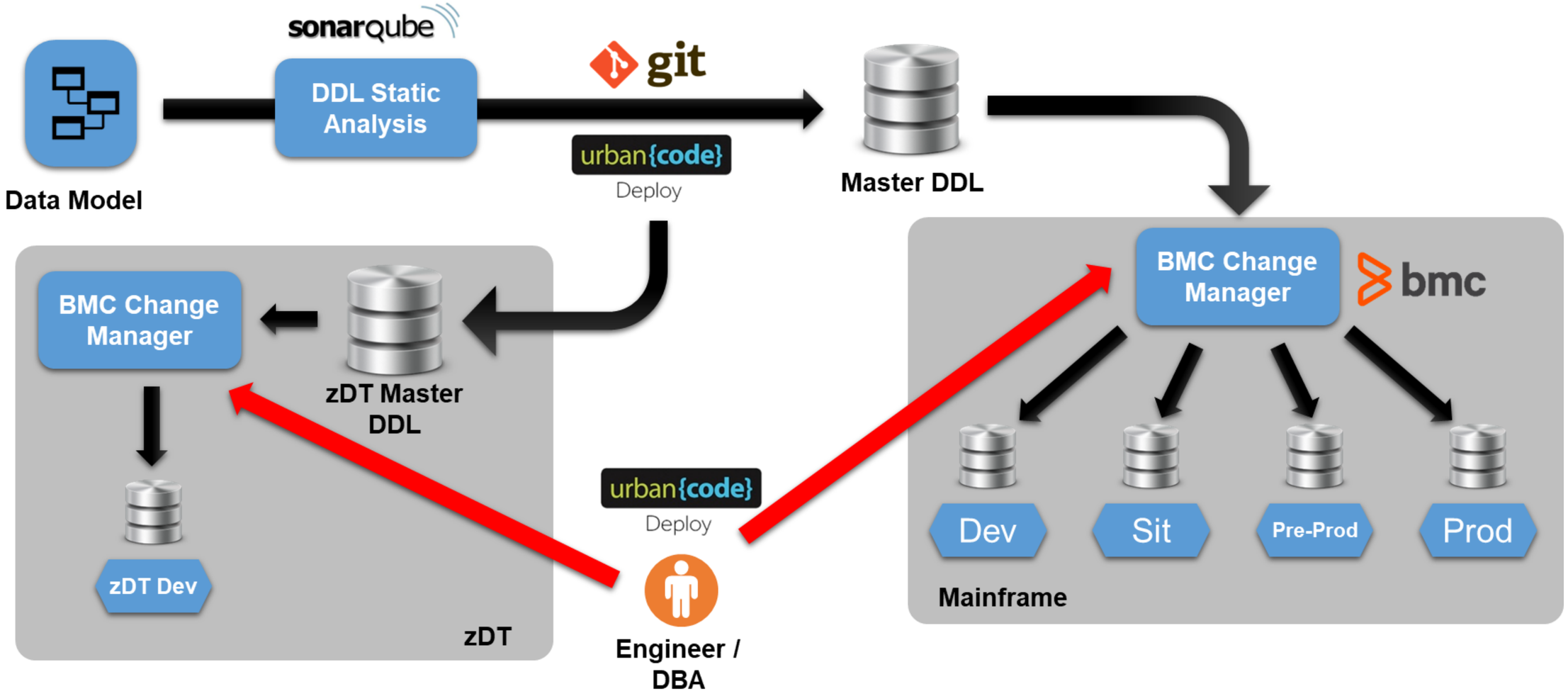
"Velocity mismatch" between DevOps mentality / processes being adopted by development teams and more traditional process used by central DBA team to manage DB2 schema change

Objective

Explore ways in which DB2 schema change can be more closely integrated into DevOps processes to increase automation, decrease implementation effort and reduce risk of human error….

…without an unacceptable increase in the risk inherent in the deployment process

ASCD Case Study - Toolchain



ASCD Case Study – Key Features

- Developers / engineers have self-service provisioning / deployment capability
  - Deployment can be requested via UCD GUI, web page, REST call, Jenkins, etc
- Process can be used to create new test environments (EaaS) as well as upgrade existing ones
  - Compare to blank target schema
  - Additional steps needed to populate with standard test data
- Automated DBA approval emails for non-Dev environments
  - Email includes attachments for execution JCL and worklist, and warning if object drop / re-create is needed
- Option to request backout capability
  - Full CM recovery baseline taken before change
  - Additional job generated to allow recovery back to full baseline
- UCD Templates used to allow multiple application teams to easily use the same processes

ASCD Case Study – Benefits

- Schema changes can be integrated as part of a wider application deployment
- Use of master DDL library ensures all environment remain in lock-step as schema versions are promoted
- Reduction in time/effort required by DBAs (and developers) to rollout schema changes, leaving more time for higher-value activities such as tuning and design
- Allows easy tracking of what's been deployed where, and by whom
- Self-service capability for developers and engineers
- Reduction in potential for human error
- Schema change backout capability minimises elapsed time in the event that change has to be regressed
- Single standardised process for all schema changes, that can be enhanced with additional checks

ASCD Case Study – Lessons Learned

- You MUST use a model management tool of some kind (Data Studio, erwin, etc)
  - Ensure DDL consistency / validity to make automation more robust
- Get z/OS sysprogs and IT security on side early
  - Reassure them that UCD z/OS agent isn't inherently evil
  - Agree responsibilities / RACI
  - Consider use of non-human IDs, with RACF PassTickets and impersonation
- Engage your tool vendor
  - All change/compare tools already have batch capabilities but some activities can be made much easier with relatively minor enhancements
- Continuous Delivery encourages a different mindset for DB schema change
  - Emphasis on avoiding any kind of disruptive change due to delivery frequency
  - Work within limitations of ALTER (e.g. use of "replacement tables", add columns to end of existing table, etc)
  - Process automatically generates warnings if DROP / CREATE strategy is adopted
- Not all developers / engineers like the UCD interface, and freeform entry of target environments etc can generate errors
  - Consider front-ending generic UCD processes with customised request portal and / or invoke from Jenkins

Summary

- If it’s not already prevalent, DevOps is coming to a site near you, and soon
- DevOps brings significant culture and process change, and DBAs are not immune
- Will you embrace the new opportunities to automate the donkey-work and concentrate on higher-value activities, or be one of those that are worked around?

Questions?