Case study

Db2 for z/OS Schema Automation

A large financial institution wished to implement DevOps practices for their mainframe platform to increase the speed and quality of development.

The DBAs made extensive use of BMC's Change Manager tool for implementing Db2 schema changes. This was a tried and tested solution that could generate a script to implement a given change, automatically dealing with any underlying complexities (such as the need to drop and re-create objects that couldn't be changed via a simple ALTER, while retaining data and dependent objects such as views and indexes).

The Challenge

Change Manager already handled much of the heavy lifting when it came to the mechanics of deploying Db2 schema changes. The problem was the highly manual way in which it was used: changes would often be requested via an email or a spreadsheet, which then required the DBA to be allocated to the task and manually enter the details into Change Manager via an ISPF session – a time-consuming and error-prone process.

The resultant "velocity mismatch" between application and database change was negating many of the potential benefits of the institution's DevOps initiative. Triton Consulting were asked to design, develop and deploy a solution that would automate the deployment of Db2 schema changes, without compromising the quality or stability of the production environment.

Solution overview

Primary deliverable

Automated static code analysis of Db2 for z/OS DDL

Technologies used

Db2 for z/OS / BMC Change Manager for Db2 / GitHub Enterprise / SonarQube / IBM UrbanCode Deploy / Visual Studio Code / Jira

Principal benefits

Reduced development cycle times

Quality and reliability of each deployment has improved

The Solution

Empowering Engineers

Given that BMC's Change Manager was already known and trusted by the DBAs, it was decided to base the automated deployment around the proven capabilities of this product. Triton implemented a robust methodology using a versioned set of "primary DDL" datasets that represent a single version of the truth, to be used as a basis for all deployments. The BMC Change Manager deployment process was then encapsulated in a set of JCL procedures and honed to the extent that the entire end-to-end process would run reliably as a set of batch jobs.

As IBM's UrbanCode Deploy (UCD) product was already widely used as the standard way of deploying code, it was a natural choice to also adopt it to orchestrate the Change Manager batch jobs. This allowed Db2 schema changes to be easily deployed alongside code changes, while UCD's GUI interface made the deployment process more accessible to non-mainframe developers.

A single button click would release the script and allow the change to be implemented in minutes, instead of days.

With this basic level of automation in place, developers were able to directly request the deployment of a new set of schema changes for a given environment (either via UCD's web interface or via a Jenkins build job or a REST API call). The automation would then take over to generate a BMC Change Manager script to implement the change within the requested environment. The DBA team would receive an automatic email notification asking them to review the script and either approve or reject it, thereby retaining overall control and preventing poor or ill-advised changes from proceeding.

Assuming the DBA was happy with the change, a single button click would release the script and allow the change to be implemented. A process that used to take days or sometimes weeks could now happen in minutes.

Enhancing the Process

With the basic process automated and delivering key benefits, attention turned to a number of enhancements to further reduce cycle times and improve quality. These included:

- **GUI for viewing/editing Db2 data models**.
  With the initial solution, engineers would create a new version of the "primary DDL" for a given Db2 object and edit it using ISPF. This required mainframe access and good knowledge of Db2's DDL syntax – not always ideal. By introducing a GUI editing tool (Visual Studio Code with the IBM Db2 for z/OS Developer Extension), engineers could more easily view and amend the Db2 data model without these disadvantages. Changes made in Visual Studio Code can be automatically versioned in GitHub Enterprise and immediately copied to the relevant mainframe LPARs ready for deployment.
- **DDL Static Analysis**.
  The automated process to copy the DDL to each mainframe LPAR presented an ideal opportunity to enforce some quality assurance checks on the newly-generated DDL before it became available for use. A custom plugin was written for SonarQube (an open-source tool to perform static analysis) and some rules were provided to allow basic quality checks to be enforced (such as site naming standards, DDL physical design guidelines). Any DDL failing these checks didn't get deployed to the mainframe LPARs and therefore couldn't be deployed until it had been corrected.

A custom plugin was written for SonarQube and some rules were provided to allow basic quality checks to be enforced

- **Copy Back Processing**
  With the ability to more rapidly and safely deploy schema changes through to production, an additional requirement was identified to ensure any parallel environments that were skipped on the way to production could be retro-fitted with the new schema version. The automation already logged each deployment (to a Db2 table) so it was relatively straightforward to construct a "copy back" process to ensure that all prior environments were automatically upgraded to at least the level just implemented in production. This process was triggered automatically following a successful production deployment.
- **Drag and Drop Deployments with Jira**
  Although schema change deployments could be easily triggered via the UCD web interface or Jenkins scripts, many application teams already made extensive use of Jira to document and track the status of schema changes through each environment on the route to live. Using Jira triggers and Jenkins, further automation was developed to allow deployments to be triggered entirely from Jira, by simply dragging a ticket on a Kanban board. The UCD Jira plugin was also used, to allow the Jira ticket to be automatically updated as the deployment progressed.

Using Jira triggers and Jenkins, further automation was developed to allow deployments to be triggered entirely from Jira

The Results

The automation has transformed the client's approach to deploying Db2 schema changes. Developers and engineers can now use a modern GUI to specify changes to their Db2 data model, with quality checks ensuring that only "clean" DDL is available on the mainframe for deployment.

Schema changes can be requested simply by dragging a Jira ticket to a new column, and deployments that require both code and database changes can be easily coordinated within a single request. Skilled DBAs no longer need to manually re-type changes and mindlessly deploy them to environment after environment, and can spend their valuable time on higher-value activities such as tuning and problem diagnosis.

Above all, the classic promise of DevOps has been realised:

development cycle times have been significantly reduced

while the **quality and reliability** of each deployment has improved

The customer is now looking at repeating this approach for several other aspects of their mainframe environment, including Db2 REST Services and CICS application definitions.

Talk to our expert team
about how DevOps could help you get more from your mainframe

Contact

Rob Gould Business Development Lead
+44 (0) 7766 838 904
rob.gould@triton.co.uk

Paul Stoker Sales & Marketing Director
+ 44 (0) 870 241 1550
paul.stoker@triton.co.uk

triton consulting