Database basics

What is Database DevOps?

Let’s start with the definition of DevOps. DevOps is a set of practices combining software development (dev!) and IT operations (ops!) with the goal of delivering more features, fixes, and updates faster, in alignment with business objectives. Database DevOps applies these same principles, making sure that the database code is included in the same process as development code.

Database DevOps helps teams identify and streamline the application development and release process further by addressing a known bottleneck: database code changes.

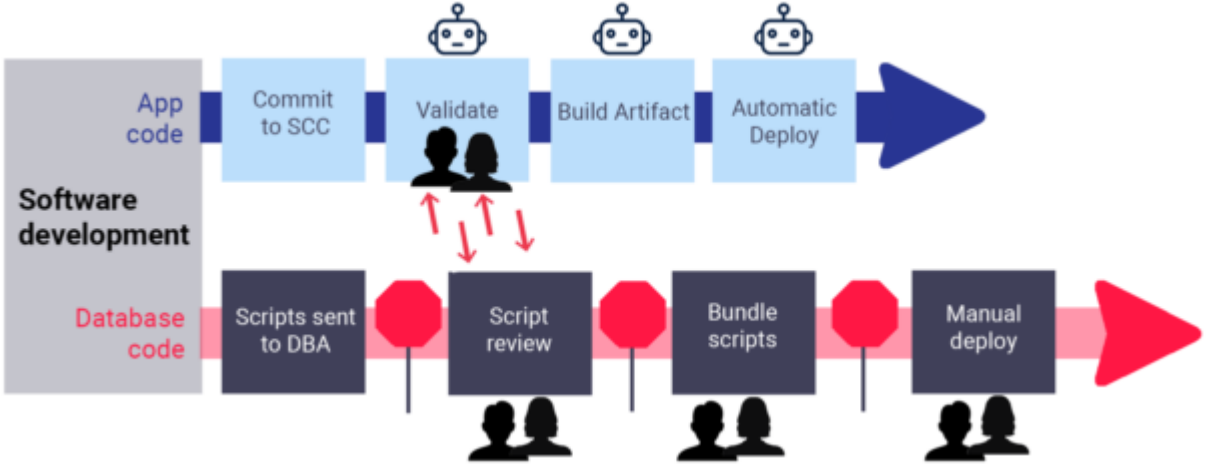
Treating database code like app code

It all starts with developers. Most application code changes that software developers work on also affect the database code. According to the State of Database Deployments in Application Delivery survey, 57% of all application changes require a corresponding database change. Many enterprise developers are tasked with making these changes and write some SQL code. Since SQL is not a developer favorite and because bad database code has the potential to do real damage, all of these changes are typically reviewed by experienced database administrators (DBAs). DBAs review the database code to ensure that the database changes won't affect the app's performance, security, or integration of data.

All of this looks great so far on paper, right? But there are several things that happen in the real world that make this process not-so-great at most companies.

The Database Bottleneck

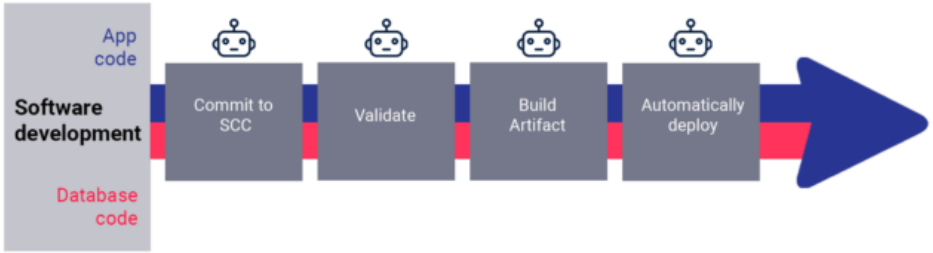
A 2019 State of Database Deployments in Application Delivery report found that for the second year in a row, database deployments are a bottleneck. 92% of respondents reported difficulty in accelerating database deployments.



The Database Bottleneck

Since database changes follow a manual process, requests for database code reviews are often the last thing holding up a release. Developers understandably get frustrated because the code they wrote a few weeks ago is still in review. The whole database change process is just a blocker.

Now, teams no longer have to wait for DBAs to review the changes until the final phase. It’s not only possible but necessary to do this earlier in the process and package all code together.



Top Database DevOps Challenges

Data shows that most application teams are not only thinking about DevOps, but actively adopting a DevOps workflow into parts of their business.1 Companies get it. They want to invest in DevOps practices. The latest industry survey shows that 52% of organizations have already adopted a DevOps approach. An additional 30% plan to adopt the approach to at least some projects. That's 82% of organizations surveyed!1

Then you look at the database team:

- 92% report it's difficult to accelerate the database deployment process2
- 91% have to rework database changes multiple times before they are ready for production deployment2
- 91% face challenges accelerating database deployment2
- 80% agree it takes longer to deploy database code changes than other changes2

Top Database DevOps Challenges

It’s easy to assume that money is the main hurdle. This doesn’t seem to be the case. According to the 2018 State of DevOps survey, only 10% of respondents cited lack of budget as a barrier to DevOps adoption.1 Here are the main challenges preventing end-to-end DevOps at companies of all sizes:

1. Provisioning data for application testing

You’ve got a new feature to developed and you’re ready to go… except you need data. It doesn’t make a lot of sense to use just any old data. You’ve got to get as close the real thing as possible.

Using production data ensures that your code and data work together as intended. But most developers can’t (and shouldn’t) use production data. At least, not before it’s encrypted or masked. Regulatory and compliance requirements (and basic respect for data privacy) demand devs do better. It’s vitally important that all sensitive data is secured in all environments.

Top Database DevOps Challenges

1. Provisioning data for application testing

For many developers, this is how it goes:

- The developer files a ticket with the IT department, listing the resources needed (such as database access, a virtual machine with a certain operating system and system tools installed).
- The developer meets with IT to justify the resources.
- The developer waits for someone in IT to implement the configuration.
- The developer troubleshoots with IT because someone did something wrong. (For example, the wrong specs were provided).
- Finally, days or even weeks later, the developer has an environment ready for development testing.
- A month later, during the release, the developer discovers that a database change happened and their code doesn't work.

It's no wonder why so many developers resort to providing their own resources from the Internet or in-house systems, which is exactly what IT was trying to avoid in the first place. There are also many challenges that DBAs face.

Top Database DevOps Challenges

2. Data persistence

In many companies, application code is in a CI/CD pipeline and is maintained through version control. When a developer submits new code, bam! New code can replace the old code. Then there's the database. Continuous deployment isn't as simple for databases. For a data store to be considered persistent, it must write to non-volatile storage. If the new feature in your application requires a change in schema, for example, you've got to migrate the data to a new structure.

3. So. Many. Tools.

There are amazing tools available to help make the integration of a DevOps process easier. Unfortunately, it's gotten to the point where there are so many tools, that it gets a bit difficult to keep them straight, make sure that they play well with each other and are future-proof. Adding a new tool over here can break something else over there. Or, maybe something wasn't integrated properly? This can lead to a lot of churn and not a lot of work getting done.

Top Database DevOps Challenges

4. People

The biggest challenge? In a word: people. Dev teams and operations teams have to overcome completely different approaches to development within these multi-functional teams. Synchronizing application and database changes is easier said than done. Every company already has a process for merging the two together. It's usually not pretty, but you get used to it. It becomes part of your job. The thought of adding new tools and processes to an already difficult situation feels wrong. It feels like it's adding complexity.

And that's just it; people in this process are already stretched to the breaking point and are barely making their release cadence work as it is. It's tough to imagine making changes. Feelings, in this instance, are overriding the facts.

How to Overcome Database DevOps Challenges

Implementing a Database DevOps process is proven to speed release cycles and reduce mistakes. So, what can your team start doing to overcome these very real challenges?

Start small (but think BIG impact)

In order to get over the fear factor of change, it's best to start small. (Bonus points if you can find an application that makes a big impact.) Find or create a small multi-functional team and put them in charge of a small release. Have the team do it the old way first. Track everything.

Then, task the same team with tackling each of the challenges above head-on, starting with the suggestions below. Have the team try the new method for a couple of releases. Track the same metrics. Now, you'll be armed with data about the speed and agility of this new way of implementing changes that will drop some jaws.

How to Overcome Database DevOps Challenges

Smart, self-serve data provisioning

Self-service data provisioning tools can help IT admins avoid data breaches and noncompliance issues because it prevents developers from accessing systems they shouldn't or taking data somewhere that isn't permitted. And, ideally, the process is so much faster and easier that developers aren't motivated to look at shadow IT options.

We recommend using a tool like Delphix for data provisioning. Imagine a world where your developers can provision data that's based on real data, but continuously masked for security, in minutes to any environment—whether on premises or in the cloud. It's a game-changer.

How to Overcome Database DevOps Challenges

Smart database automation

Speaking of game changers, let's talk about integrating database changes into your DevOps process and leveraging database continuous delivery. When some people hear "treat the database code like app code", the knee-jerk reaction is "no! It's different!" It's true. They aren't the same. But it is possible to put the database into source control and treat database changes in a similar way to application changes. There are a few additional considerations, like adding a migration script to enable existing data to fit in any changed schema, for example. But the idea is to automate these changes and put the database code into source control where continuous delivery is possible. When continuous integration of database changes happen along with the application code, you speed up your application delivery by orders of magnitude. (And it's more accurate. And you've just built in an audit trail.)

How to Overcome Database DevOps Challenges

New roles to support the new process

It seems like a no-brainer that implementing a completely new way of approaching application releases requires a new way of defining roles, but many teams overlook this step. There is no one-size-fits-all in figuring out which roles are needed for your specific application and industry. It requires a lot of trial and error to figure out what works and what doesn’t. Sit down with your newly formed team and talk about which tasks are needed in order to get this particular application released. Then, sit down and figure out where their skills and knowledge fit the bill and define their role.

How to Overcome Database DevOps Challenges

Practice empathy

Change isn't easy. It may seem counterintuitive when you're trying to deliver results faster, but it's important to take time to listen and empathize with everyone on the team.

Make sure you put a system in place that allows everyone on the team to voice concerns. Everyone on the team needs to understand how the system will work so they understand how to bring up issues they encounter and what will happen next so that there's a path forward to solving problems.

Some teams do health checks at the end of each sprint, where team members can rate their feelings about various factors including teamwork and support. Consistent negative ratings can tell you that there's a systemic issue that needs attention. Managers can also specifically ask in 1:1s about how communication is or is not working to identify patterns.

6 Problems Database DevOps Helps Solve

In a recent article on DevOps.com, Helen Beal discusses six common sources of conflict within IT that DevOps can help to solve. These sources of conflict have arisen as IT tries to form new behaviors and processes to meet the needs of the business in delivering faster.

Problem #1: Unplanned Infrastructure Requests

“I need a server… now”

Operations teams have a full plate with work that has been planned and prepared for. When development requests new environments, it adds unplanned work to the mix and catches operations off guard. Development sees this as willful neglect in trying to meet the needs of the business to go faster, which just isn’t the case. Remember that in addition to this type of unplanned work, Operations also deals with “unexpected system failures and defects.”

6 Problems Database DevOps Helps Solve

Problem #1: Unplanned Infrastructure Requests

Beal's biggest piece of DevOps advice here is to invite operations to collaborate sooner in the dev cycle. This way, Operations has a better understanding of the work that's coming down the pipeline and can plan it into their workflows.

The other area where DevOps can reduce the strain is through automation. This makes it easier to deliver new infrastructure by automating it, creating or using cloud and virtualization infrastructure.

6 Problems Database DevOps Helps Solve

Problem #2 – Excessive Access to Production Elements

“I want access to the production systems”

This happens when a development organization is frustrated by what they see as a lack of responsiveness from operations. While it seems like a quick fix, it increases the risk to production systems as more access points are introduced. Increased risk in production leads to more unplanned work for operations. More work means decreasing operations’ ability to support development.

“DevOps doesn’t mean throwing all process out of the window and putting live environments at risk,” writes Beal. “It’s by no means mutually exclusive with methodologies like ITIL but it is about individuals collaborating better to achieve a shared desired outcome.” To foster that collaboration, Beal recommends that operations involve development in “the definition of your change control processes and set up of any systems.” Work to establish some SLAs together as a team.

6 Problems Database DevOps Helps Solve

Problem #3 – Lack of Retrospective Failure Analysis

“Something is wrong… whose fault is it?”DevOps, like Agile, is about removing blame and focusing on developing solutions as a team. “That means no finger-pointing and no swearing at each other,” writes Beal. Two areas where DevOps can help is in:

1. Setting up and running blameless retrospectives

2. Figure out ways through tooling to either “pre-empt failure or fail smartly.”

6 Problems Database DevOps Helps Solve

Problem #4 – Too Many DevOps Monitoring Tools

“So many alerts they are meaningless”

DevOps advocates for establishing feedback loops from production to as far back as development. But, in working with clients, Beal has found that this critical feedback isn’t missing due to a lack of available database DevOps tools.

“One of the challenges we often find,” writes Beal, “is that enterprises already have monitoring systems. Lots of them. Monitoring lots of different things. All producing a multitude of alerts.”

The problem, according to Beal, is that this plethora of monitoring leads to “alert fatigue.” There is data, but there isn’t the kind of feedback necessary to make improvements, or that critical feedback is getting lost in a sea of information.

Beal’s advice in this regard is to combine and rationalize your existing systems. “Take some time to really understand what alerts matter and adjust your policies accordingly,” she writes. “Consider this an ongoing task to tweak your systems as your situation changes.”

6 Problems Database DevOps Helps Solve

Problem #5 – Not Enough DevOps Training Time

“I don’t have time to save time”

This one is the silent killer. The teams which need to use DevOps the most don’t have the time to think about how they can do things better, much less stop to catch their breath. The increase in expectations to deliver innovation, coupled with “the fragility of today’s evolution of IT infrastructure,” result in an increased workload in IT, and a great deal of unplanned work.

These 2 factors are often business drivers behind the adoption of widespread DevOps metrics for the database. According to Beal, “DevOps is designed to streamline the throughput of work at a cultural, interaction and application-based level.” This can help to diffuse “the pressure building up in many IT departments today.”

6 Problems Database DevOps Helps Solve

Problem #5 – Not Enough DevOps Training Time

Her advice on this front is to bring in executive sponsorship early, to help force a change for the better. Two things executives can do to help is:

Launch a DevOps pilot somewhere in the organization to jumpstart change.

Carve out some room in the budget to bring in an external consultancy to look at your “DevOps state.” It’s likely that the consultants won’t tell you anything you don’t already know. It still helps in starting to build a business case to have outside observation and documentation.

6 Problems Database DevOps Helps Solve

Problem #6 – The DBA is a Bottleneck or Roadblock to DevOps

“Our hero is a bottleneck…”

Sometimes, one person becomes the bottleneck in trying to deliver faster, but not for the reasons that you might assume. “What tends to happen is this: one day an engineer decides that building servers is a tedious, manual process and that they could write a quick script to make that bit a bit quicker. The next day, they write another one. And another one.”

This behavior is a good thing. It’s an individual taking the initiative to improve workflow by automating different aspects of the process. But, according to Beal, “The problem is that even though the script monster works pretty well most of the time when it fails it’s virtually impossible for anyone other than its creator to troubleshoot.” This, as you can probably assume, doesn’t scale well. It presents a skewed picture of the level of automation within the organization. It also introduces an individual as the single point of failure for the whole system.

6 Problems Database DevOps Helps Solve

Problem #6 – The DBA is a Bottleneck or Roadblock to DevOps

These heroic individuals began automating tasks to keep their own sanity. Instead of making it a team problem, or a part of the process, these individuals take it on themselves to solve the problem, probably to avoid bureaucracy. If the culture in the organization is more responsive or open to exploring new techniques and methods, it might offer these individuals other alternatives to explore besides fixing it on their own.

Another option is to remove these talented individuals from the daily process. You've recognized that these are some of the most talented individuals in the organization. Removing them from the daily work can focus their talent and innovation on optimizing the whole process or developing new ones. Their general approach can help to elevate the entire system.

4 Reasons Why DevOps Ignores Data Professionals

Avoid the data group at your peril. Too often we have seen database DevOps being adopted at companies without engaging with the data group. Those efforts will fail to deliver the benefits initially sought by DevOps.

Simply put, a pack can only move as fast as its slowest member. If you look at how wolves travel, you'll notice that the slowest members are at the front, the fastest, strongest at the back

But, time and time again we see DevOps teams completely ignore data professionals. From tool evaluation, to process improvement, those calling for DevOps adoption routinely ignore data professionals and fail to embrace them. The irony is not lost on us as we see a movement built around inclusiveness ignoring a key component to a companies' success. Here are the 4 reasons why DevOps ignores data professionals.

4 Reasons Why DevOps Ignores Data Professionals

1. **Easy to ignore them.**
   Most companies are able to deploy database changes to their dev and test databases. There was a time when that responsibility was owned by the Data Group. But, with the adopting of Agile, companies have ceded that responsibility of deploying database changes to the development and test team. Thus, dev and test have no need to interact with the Data Group. This has led to the Data Group being completely cut out of data architectural decisions as they must simply accept the changes as requested by the development team. If the Data Group does make issue of bad database schema changes (“What! Is that an index with eight columns?!?!?”), the release teams use deadline pressure to force the Data Group to make the change anyway. Therefore, there is no need to do anything but ignore the Data Group. The similarity to Douglas Adams “Somebody Else’s Problem Field” is uncanny.

4 Reasons Why DevOps Ignores Data Professionals

2. **Data is hard.**
   The concept of state is very difficult for development teams to deal with. One of the 12 factors for applications is stateless applications. Thus, to introduce the idea of handling state for the persistence layer is often dismissed out of hand. As described before, it's somebody else's problem so why even bother addressing it. If somebody else is responsible and the problem is hard, there is no reason to address it. After all, "that's an operations problem".

4 Reasons Why DevOps Ignores Data Professionals

3. **Data professionals have different goals than developers.**
   The development group has an incentive to change things. The operations group is incentivized to maintain stability. Change is the enemy of stability. Thus, we have adopted DevOps to mitigate the conflict and find a way to have both teams reach their goals. The Data Group has slightly different goals. They too seek stability, but the integrity and security of the data is also paramount. Remember that the most valuable asset a company has in the tech stack is the data. Look at photo-sharing apps as an example: it doesn't matter that you can put dog ears on your photos if the app continues to forget who you are connected too and you can't share the photo. Features are important, but only if the data is there to support the features. Of course, the people that create those features might argue against that. Thus, we have a difference in values that leads to avoidance.

4 Reasons Why DevOps Ignores Data Professionals

4. **Data Professionals say “No” too often.**
   Finally, Data Professionals do have a well-deserved reputation as Dr. No. In the past, DBAs have been guilty of using data as a Billy club to get their way. As a result, they have taken all responsibility for production database changes. This includes all of the glory and all of the headaches; today it is all headaches, pain, and hand wringing. Because of their long history of mandating they be the final arbiters of production database deployments, and often being dissatisfied with development created schema and logic changes, they have caused the rest of the company to voice their frustration with a terse, “Fine. Do it yourself then.”Because of such a long time of demanding to be the single point through which all change flows, the Data Group has not only caused their workload to increase with the level of application releases (exponentially!), now that they are suffering with high demands, the rest of the wolf pack is not very eager to help.

We Can Fix This

These challenges are not insurmountable. We can resolve these issues by addressing the root cause: the data group is not a part of our New Development teams. We have cross-functional teams in development now; no longer do we have separate UX and Business Logic teams. As such, we need to embrace distributed data management and move some of our Data Professionals into these Development Groups.

Of course, we still need our Data Professionals that understand the challenges of storage, performance, high availability for databases. But, a large percentage of our Data Professionals need to look at their peers in Systems Administration and mirror their career paths by adopting Infrastructure as Code.

By adopting this new approach with teaming and distribution of responsibilities, we can move our pack faster to the end goal which is faster software releases and delighted customers.

Why CIOs Need to Pay Attention to DevOps

The number one threat to your organization's Agile or digital transformation efforts is how well your team manages the database release process. When software releases happened only once or twice a year, having manual reviews for database changes made sense. This required DBA resources to also look at the current state of the database and understand the changes in context. "Is the new index going to create too many on a table? Do the columns in this foreign key already have indexes?" are questions to be answered prior to the database change deployment.

Now, with the number of applications and their releases increasing, database DevOps and digital transformation are threatened by an error-prone, tedious, and slow process.

Here are five reasons why CIOs should care about database releases

1. Your company is a software company.

As Marc Andreesen told us in 2011, software is eating the world. Since all companies will engage their customers through compelling applications, the companies with the best software will be the winners. The proven way to improve software is releasing early and often. However, if your organization is mired in old fashioned manual change, your business will never reach its full potential as a software company. When you hear the statement “That’s not the way it’s done here”, you need to immediately question why that is the case and help your team understand that old biases need to shape how they reach their goals. It’s time to break some glass.

Here are five reasons why CIOs should care about database releases

2. Unless you fix the database problem first, you are wasting money on other improvements.

Time and time again, we hear technology leaders say they should have addressed the database deployment problem before tackling the easy problem of application release automation. A herd can only move as fast as its fastest member. No matter how much money and time you invest in DevOps, it will be wasted effort until you speed your database releases. We call that the “Velocity Gap” when the pace of application releases surpasses database releases. I like to think of it as putting wagon wheels on a Tesla; it just doesn’t make sense to use old and new methods of software releases together.

Here are five reasons why CIOs should care about database releases

3. Companies that completely adopt DevOps outperform the S&P 500.

The best part of the State of DevOps report is evidence that companies that have adopted DevOps outperformed the S&P 500 over a three-year period. As detailed above, you must adopt a complete DevOps strategy and include the database to fully realize your company's potential. DevOps companies are more nimble and competitive in the market. They can take advantage of opportunities faster.

Here are five reasons why CIOs should care about database releases

4. Your competitors are addressing this problem and they are beating you.

A survey by Liquibase (Datical) and CIO Magazine detailed that the pace of database releases is the biggest blocker of digital transformation. 91% of DBAs surveyed stated that their current process of database release is slowing application releases. Your greenfield competitors do not have this problem. Also, your older existing competitors are just as aware of this problem as you are. With competition from startups trying to take away your high-margin businesses and older existing competitors fixing this issue, you cannot afford to delay.

Here are five reasons why CIOs should care about database releases

5. You will retain your employees and improve their skills.

The best and brightest DBAs with decades of experience are tired of working weekends. This directly impacts their quality of life and is completely unnecessary. As the US economy flirts with full employment for high-skilled workers, CIOs cannot afford to lose their best employees to competitors that offer better working environments. We're not talking about Aeron chairs and foosball tables; we're talking about piano recitals, little league games, date nights being missed because your team members are waiting on a database release. Unless CIOs resolve this problem, your employees will find another job where this isn't a problem.

Best Practices for Database DevOps

Although the database poses several unique challenges, it should be managed using the same basic DevOps protocols that ensure secure and reliable source code, task, configuration, build, and deployment management. However, many times the database is neglected because the database is trickier. Unlike other software components or compiled code, you can’t just easily copy database changes from development to testing to production. Since the database is a repository of your most valued asset—your data—preserving it accurately is imperative to continuous delivery.

Here are the best practices we’ve found for teams that want to integrate database change and deployment into the overall process to improve release quality and release faster.

Best Practices for Database DevOps

Use the same process for all code delivery

Since you likely already have tools in place to move your application code, it's very simple to use the same tools and processes for database code. Check all code into source control and use CI/CD the same way for both application code and database code. Database code does require an extra layer of attention.

Make small, incremental database changes

If DevOps has taught us anything, it's that small changes are better. This is especially true with databases. Getting to the deployment phase of a software release and realizing that there are bad database changes that need to be found and reworked by a DBA grinds everything to a halt. That's not even the worst-case scenario. If the bad database change gets deployed it can force your team to bring everything down and then spend the next several days getting everything back in order. Using a tool that allows small, trackable changes is key.

Best Practices for Database DevOps

Make it easy to merge database changes from multiple sources

In modern organizations, data rarely resides in one place. Also, it’s best to be proactive because things like company mergers and acquisitions happen. Planning ahead and ensuring that your process is set up from the beginning for ease of merging changes from different sources is smart.

Best Practices for Database DevOps

Enable fast feedback loops

A critical piece of creating good software is good communication. Feedback loops are mechanisms used to validate and get feedback about the software development process and it's important to include database feedback loops, if you haven't already. The goal is to get positive and negative feedback and to act on it quickly to improve the process.

Datical has automated feedback loops built directly into the system. The database rules engine validates database code. If the code doesn't pass the rules validation, feedback is sent directly back to the developer immediately to fix. In addition, the Datical simulates database changes before deployment. If anything is amiss, developers are alerted immediately to fix before any bad database changes are deployed.

Best Practices for Database DevOps

Use database change versioning so you have granular control of features

Imagine a world where you can see every change that's been made to your databases all on one dashboard. Many companies deploy many database changes at once and that causes issues when something goes wrong. No one knows which change caused the issue. Also, if your company decides that a feature won't be deployed with a particular release, you can easily see which database changes are associated with the feature.

Best Practices for Database DevOps

Test early and often

Continuous testing has emerged as a best practice for integrating testing throughout the development lifecycle. By 2020, Gartner predicts DevOps initiatives will help 50% of enterprises implement continuous testing. Continuous testing relies on tools, like Jenkins, to check in source code, run tests in parallel, and notify developers when the build is rejected. This prioritizes actionable items so teams can focus on tasks that have the greatest impact on business priorities.

Best Practices for Database DevOps

Build once, deploy many (using artifacts)

Most software processes use something like the following:

- Check app code into source control (like Git)
- Build artifact with a CI tool (like Jenkins)
- Store in a repository (like Artifactory)
- Deploy with a CD tool (like Jenkins)

Database DevOps best practice would look something like the following:

- Check app code and database code into source control (like Git)
- Perform database change rules validation (automated with Datical)
- Build artifact with a CI tool (like Jenkins)
- Store in a repository (like Artifactory)
- Simulate changes to ensure ideal state database (automated with Datical)
- Deploy with a CD tool (like Jenkins)

Best Practices for Database DevOps

Provide production-like environments for developing changes

In order to make sure both application code and database code changes play nicely when deployed to the production platform, the best thing to do is to develop and test against an environment that is as close to your production environment as possible.

The more production-like your testing and staging environments are, the more confident you can be, up to and including the point of fire-and-forget releases.

That being said, your development environment does not need to be the same as production. For example, you should definitely not be using real production data. It makes much more sense to use a tool like Delphix.

Best Practices for Database DevOps

Instrument the process for visibility, consistency, and reliability

It’s important to set up your whole process so you can easily get data to find issues and leads to an objective, blameless path toward improvement. Data should be transparent and accessible to everyone. It should be meaningful and easily visualized to spot problems and trends.

The ultimate goal is to build applications that collect data on business, application, database, and infrastructure. For database DevOps, Liquibase collects detailed information about every database deployment it performs and automatically logs it to a centralized database, eliminating the human error inherent in manual entry.

Liquibase Business and Enterprise customers get easy on-demand access to database deployment information with a database deployment management console, making it easy to keep tabs on the deployment results. Stakeholders across the enterprise are able to quickly see the status of any database changes, errors, warnings, and rules violations.

Summary

You can only ship code as fast as you can deploy it. Automating database change and deployments is critical. It removes human error and speeds up the process. If, after you deploy, you need to quickly hotfix a bug, being able to check it in and do a fast, new deployment is important.

Q&A