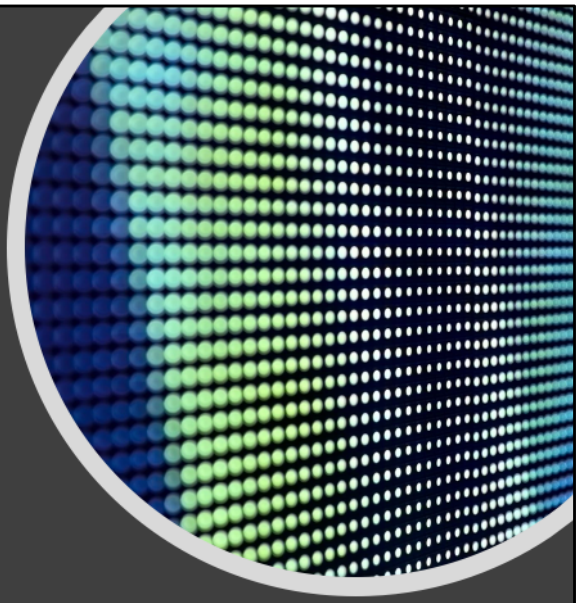
DevOps is a revolution in how technology is approached that gains in adoption every year. Join Ember Crooks as she shares her journey into a fairly mature DevOps shop as a DBA with no previous DevOps experience. In this session, we will:

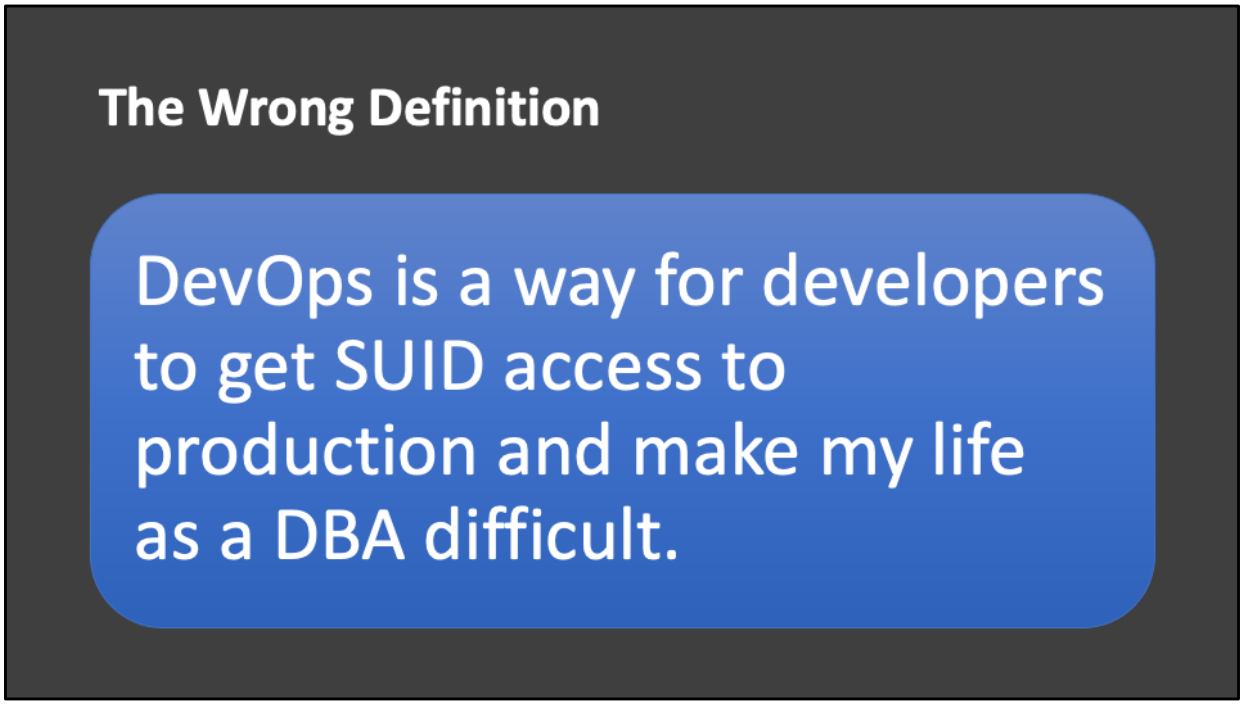
Discuss what DevOps is and what it isn't
Explore what DevOps means to the DBA
Understand at a high-level some of the decision points around running databases in containers
Review a few DevOps tools a DBA should be familiar with

Agenda



01	02	03	04
Discuss what DevOps is and what it isn't	Explore what DevOps Means to the DBA	Understand at a high-level some of the decision points around running databases in containers	Review a few DevOps tools a DBA should be familiar with


Defining
DevOps



When I first started hearing the term DevOps, I thought it was a way for developers to break the then very strict wall between development and operations, and get access to my databases. Generally, developers with access to production was a bad thing that would cause me to get paged. Many years later, I can see that this is not true. Developers still do not have unfettered access to a production database, even in a DevOps environment. There are certainly aspects of my job that are different, and there are things that developers do that make it into production without direct action from me, but this is only through a well defined and thoroughly tested process.

There are a vast array of different definitions of DevOps. To really define DevOps, you probably need a book. There are some pillars to what DevOps means that can help us understand it. A shop that is DevOps may not be all of these things, and may be in a different place in their journey on each of these things.

Automation means eliminating manual processes as much as possible. Our job becomes more the automation of processes than the execution of them.

An agile methodology is common. Agile embraces failing fast, testing thoroughly, and delivering new functionality in smaller chunks instead of huge releases.

With or without an agile methodology speed is one of the contentious points between DevOps and non-DevOps teams

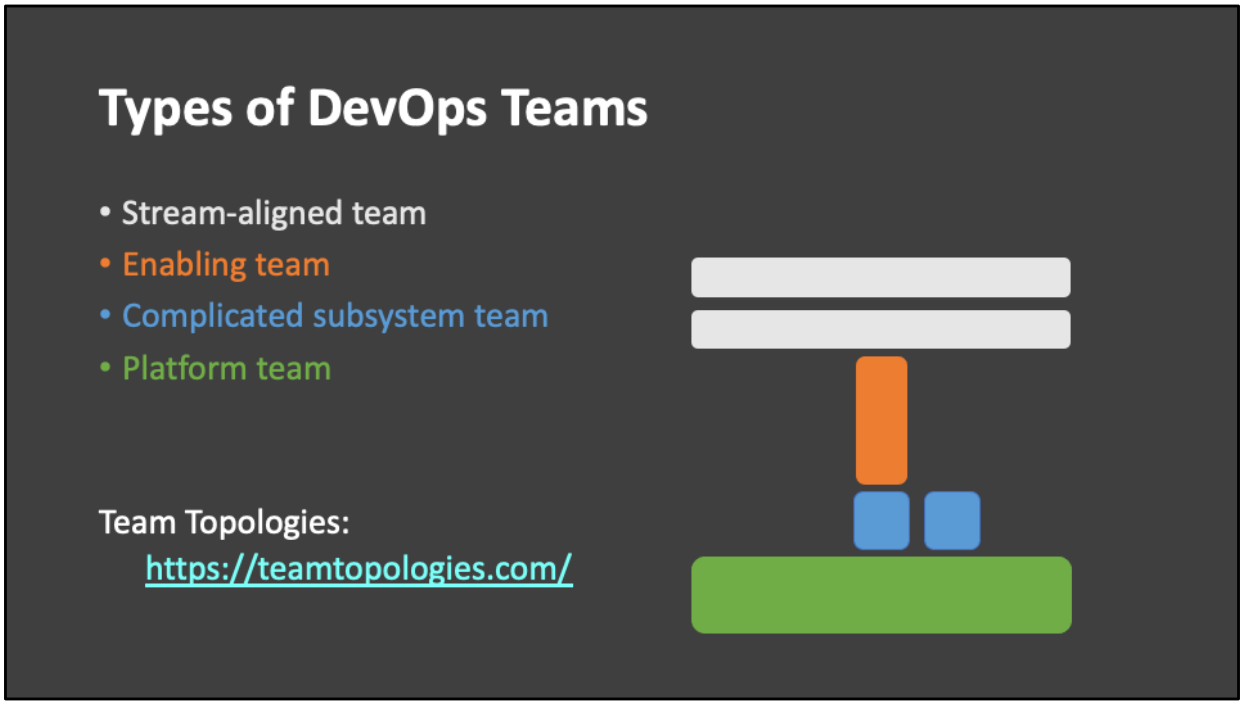
Infrastructure as code means that we manage as much as possible through code and configuration files. We'll talk more later about what that means for Db2.

Monitoring is critical to DevOps so we can identify and heal from failures.

Collaboration is critical to a DevOps team to bridge the vast array of technologies supported in a coherent and reproducible manner.

DevOps is about as far away from a culture of blame as you can get, with the focus on always identifying where we can be better.

DevOps is a culture that must be compatible with the organization's culture to survive.



https://youtu.be/haejb5rzKsM

In a more traditional organization, teams are fully separated by function. At employers I've worked at in the past, we had the Systems Administration group, which was likely separated by operating system, and depending on the size of the company might be segmented much further. I've worked with teams where their only job was access control on in-profile Linux servers. There was a separate team for access control on out of profile Linux servers. Whatever the structure, the answer to the question "whose job is it to do X" was usually fairly straight forward. In a devops organization, the lines might be a bit blurred. I've spent a significant part of my first year in DevOps figuring out whether I should do a thing or if I should ask someone else to do a thing.

Reading the book Team Topologies and even just watching the freely available video has helped me understand a lot in this space.

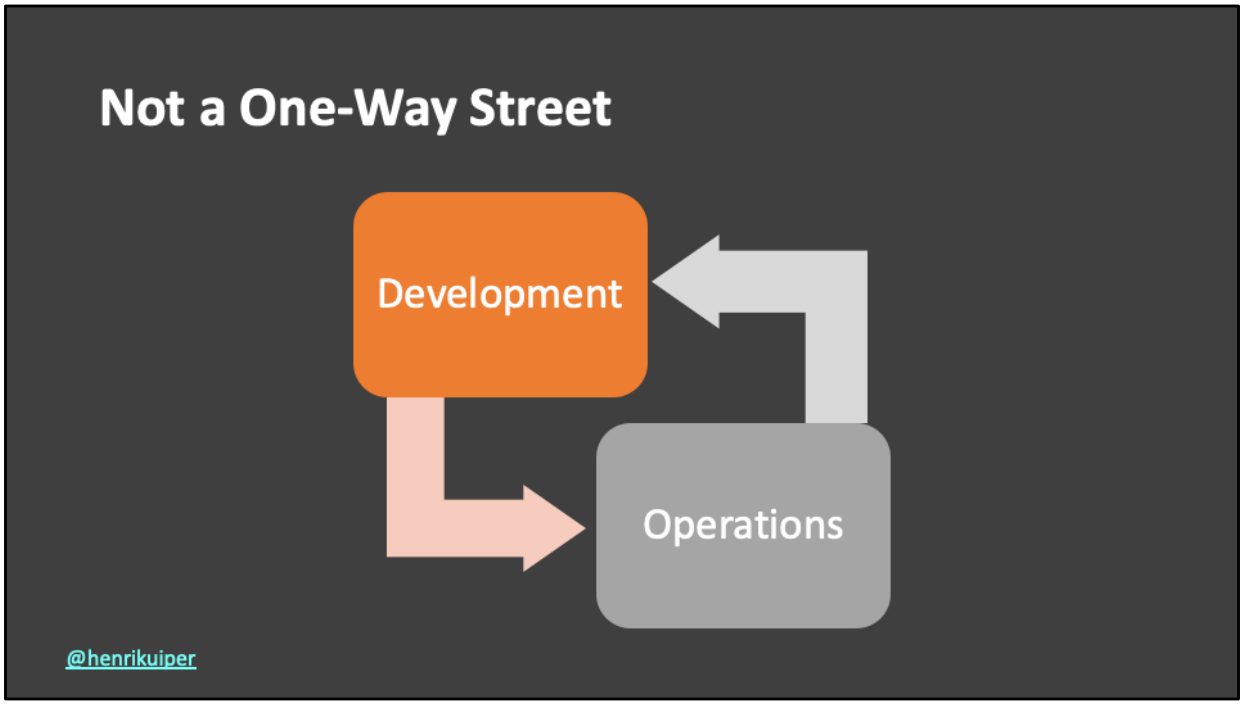
Types of DevOps Teams

- Stream-aligned team
- Enabling team
- Complicated subsystem team
- Platform team

Team Topologies:
https://teamtopologies.com/

https://youtu.be/haejb5rzKsM

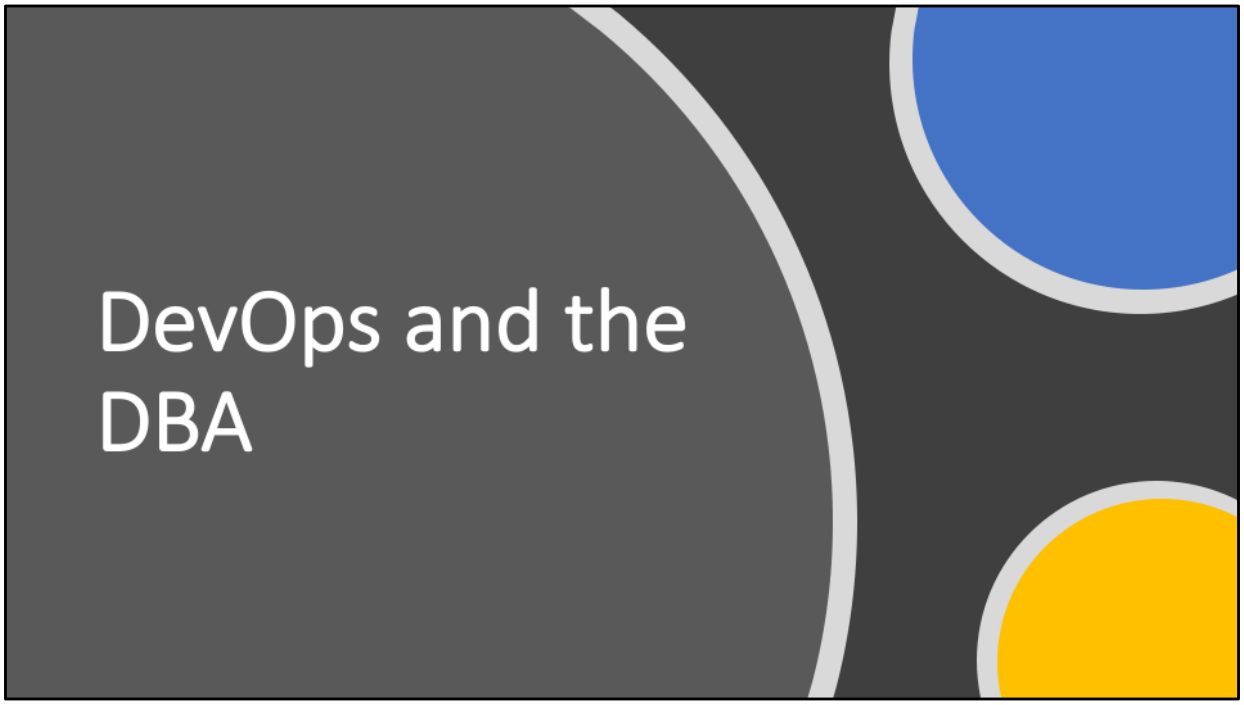
While I've found this structure enlightening, I've also have seen that for smaller teams, the lines are often blurred.

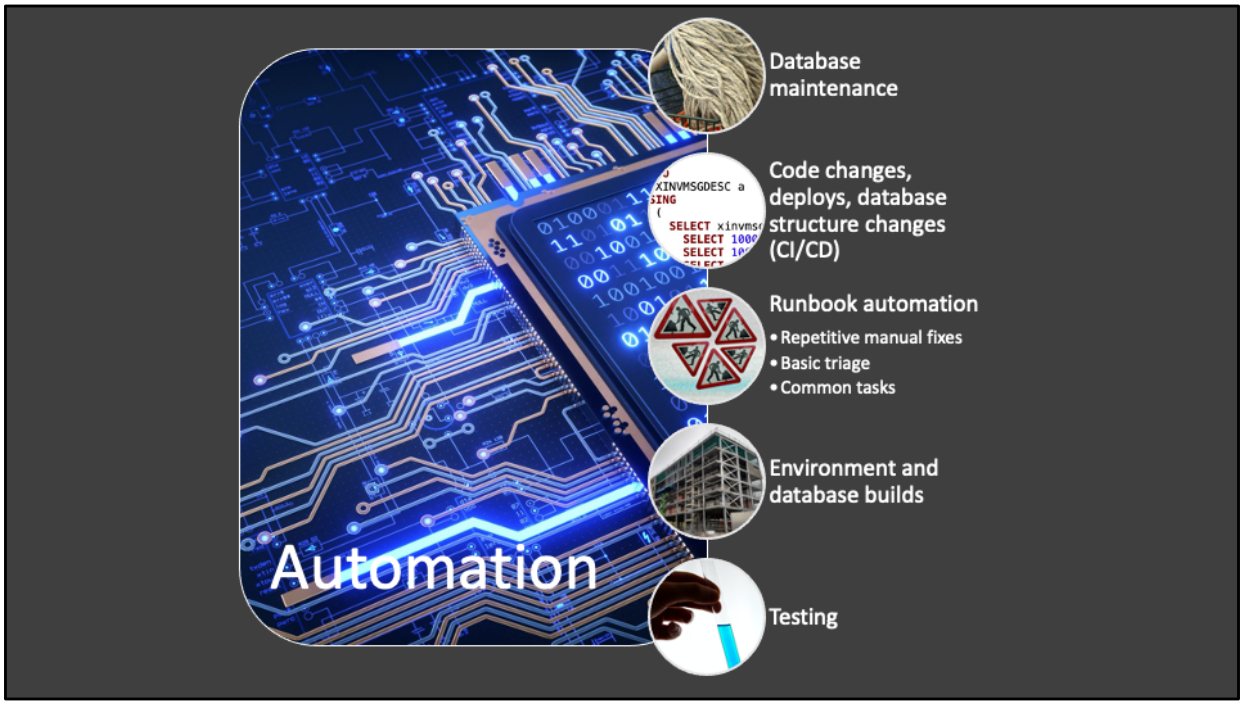


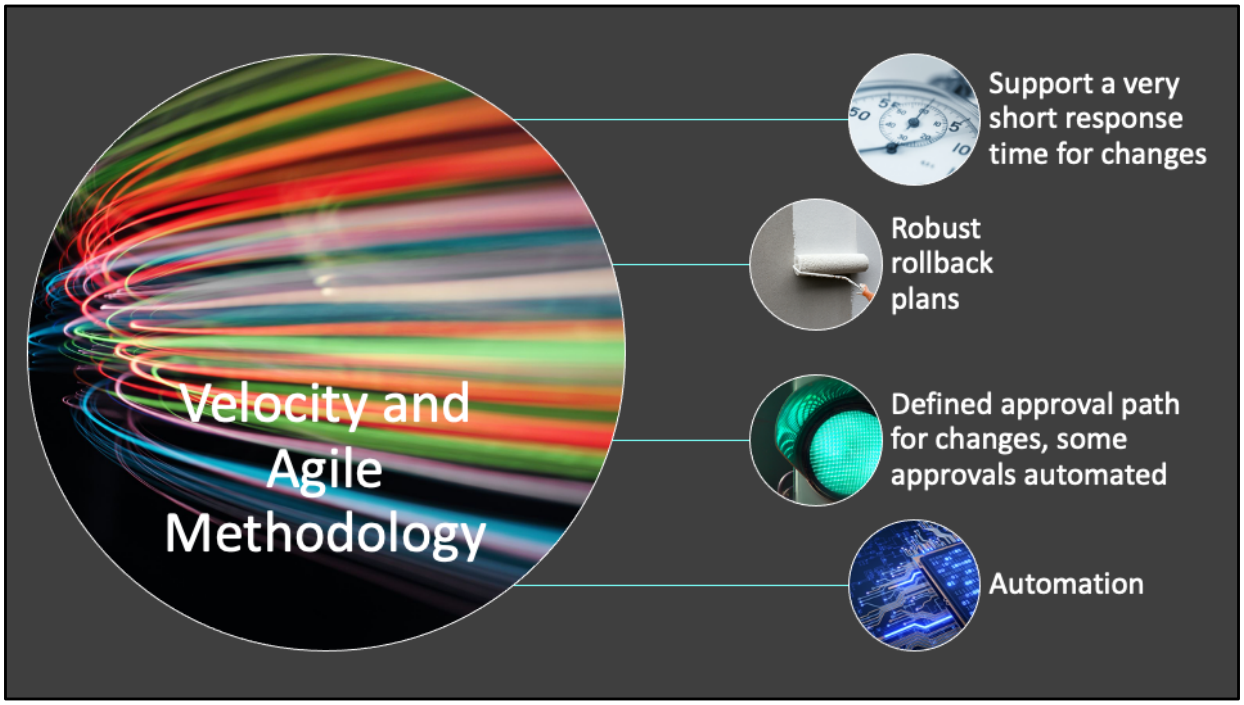
Recently I participated in a discussion about DevOps, and I absolutely loved what Henri Kuiper said in that discussion. DevOps is not just about infusing Dev into Ops, but also about the other direction – infusing Ops back into Dev.

https://www.youtube.com/watch?v=OXczEXXff74

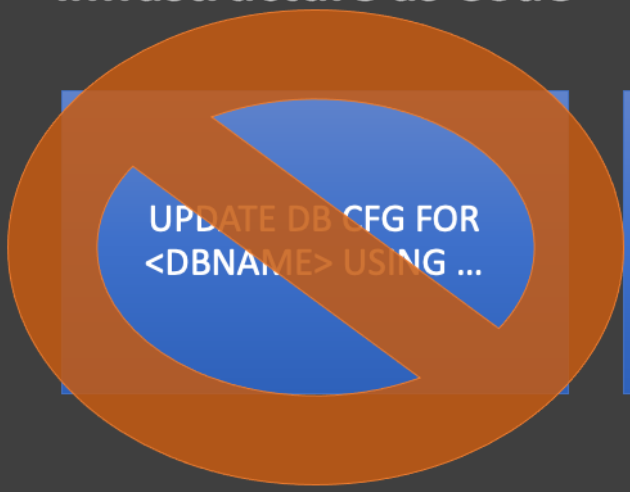
This comes home to roost nearly every day in my interactions with developers.


DevOps and the
DBA


Automation
Database maintenance
Code changes, deploys, database structure changes (CI/CD)
Runbook automation
• Repetitive manual fixes
• Basic triage
• Common tasks
Environment and database builds
Testing


Velocity and Agile Methodology
Support a very short response time for changes
Robust rollback plans
Defined approval path for changes, some approvals automated
Automation

Infrastructure as Code



Change in a text file, change is moved to servers using:

- Container: deploy of the container
- VM/Server: configuration management tool

Monitoring and Continuous Improvement

If it fails, build a new one

DevOps is a journey, not a destination

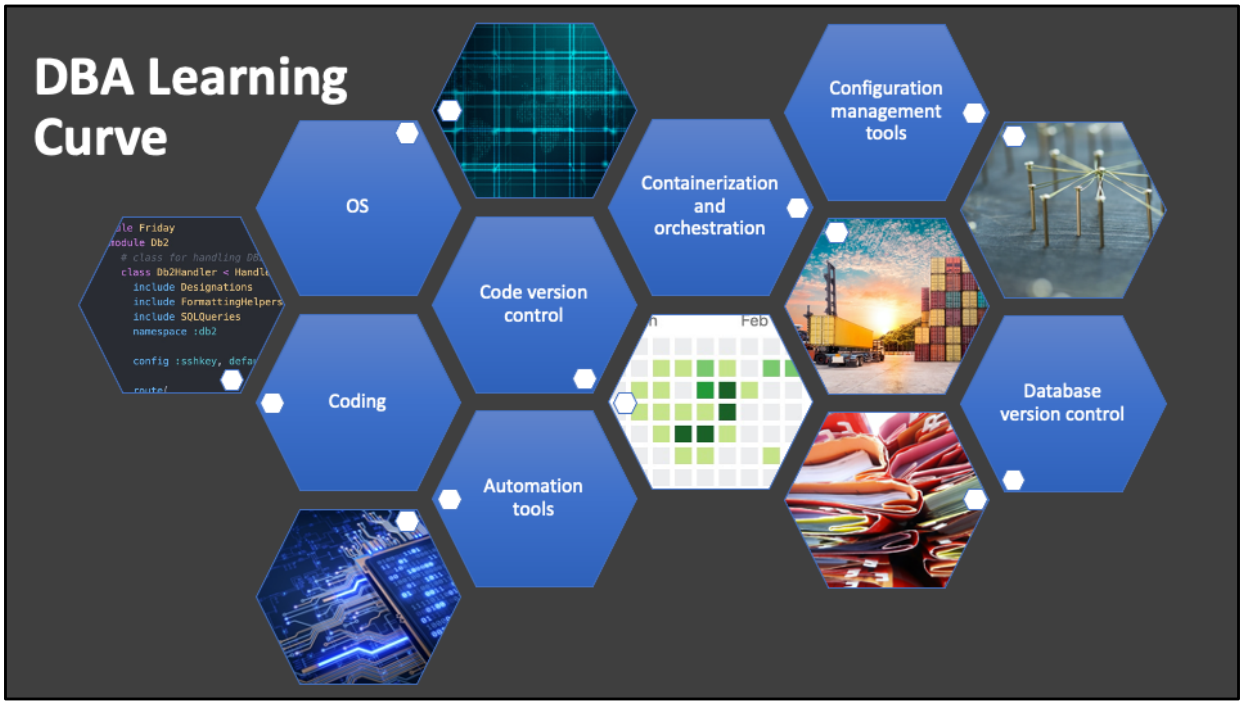
Collaboration and Culture

It's OK to say "I don't know"

Work with other teams to understand how you can help them and how they can help you

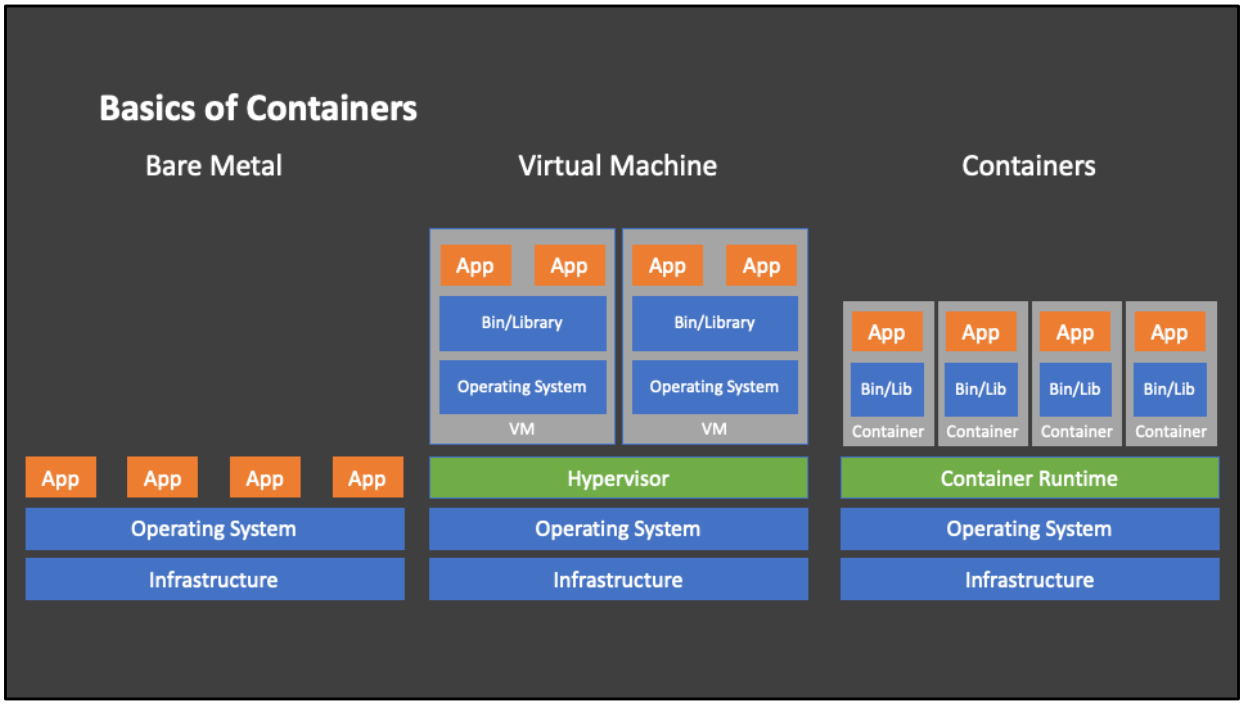
Management can't implement DevOps without team member buy-in

Teams can't implement DevOps without Management buy-in


DBA Learning Curve
OS
Configuration management tools
Containerization and orchestration
Code version control
Coding
Database version control
Automation tools


Containerization
of Databases

Containerization is NOT required for DevOps



https://www.docker.com/resources/what-container

How Do I Create a Container?

- Dockerfile
  - Starting image
- Entrypoint script
- Build image
- Run image

Persistence

- Containers are disposable
  - If it fails, build a new one
- Databases are not disposable
- Databases often have connections between the version and the data on disk

Advantages of Containerization for Databases

- Ease of automated builds
- Ease of managing configuration
- Ease of upgrade
- Supporting multiple development streams

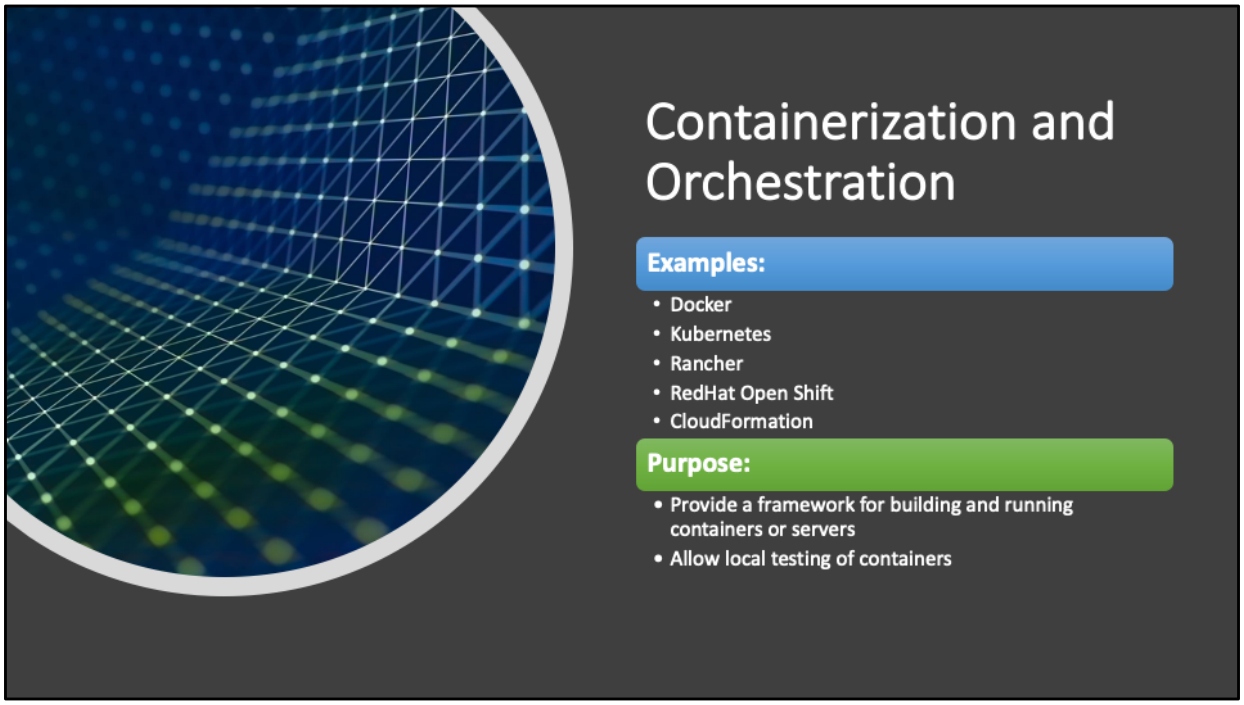
Disadvantages of Containerizing Databases

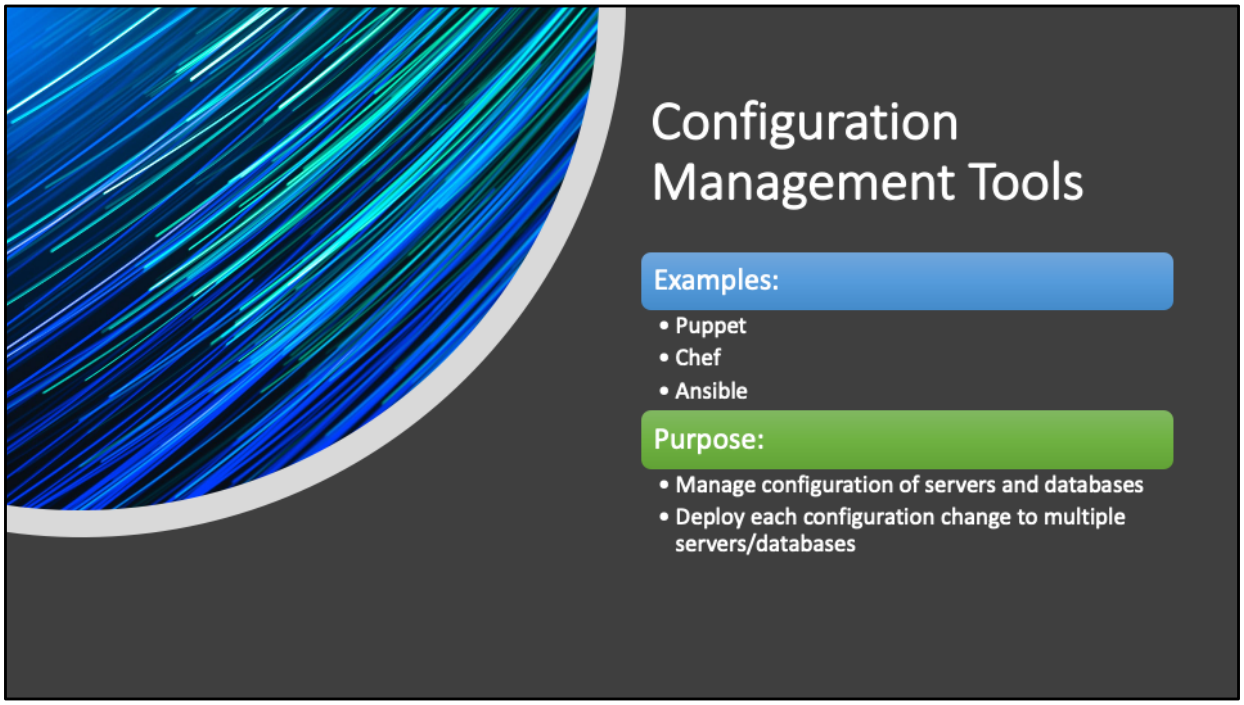
- Persistence
- Learning how to containerize
- Vendor support
  - IBM only provides Db2 production containers on RH Open Shift
- Host size
- Gotchas and edge cases
- DB-level HA vs. container-level HA

Choices Made in My Current Environment

- DB2 in containers for dev and QA
  - Other administrators can spin up new dev environments at will, without a DBA
  - What data goes into a new environment?
- Db2 in EC2 instances for Stage, Load Test, and Production
  - Cloud formation templates
  - Use the same build scripts used for docker containers to build the environments.
  - Puppet manages configuration after build


DevOps Tools
for the DBA


Containerization and Orchestration
Examples:
• Docker
• Kubernetes
• Rancher
• RedHat Open Shift
• CloudFormation
Purpose:
• Provide a framework for building and running containers or servers
• Allow local testing of containers


Configuration Management Tools
Examples:
• Puppet
• Chef
• Ansible
Purpose:
• Manage configuration of servers and databases
• Deploy each configuration change to multiple servers/databases

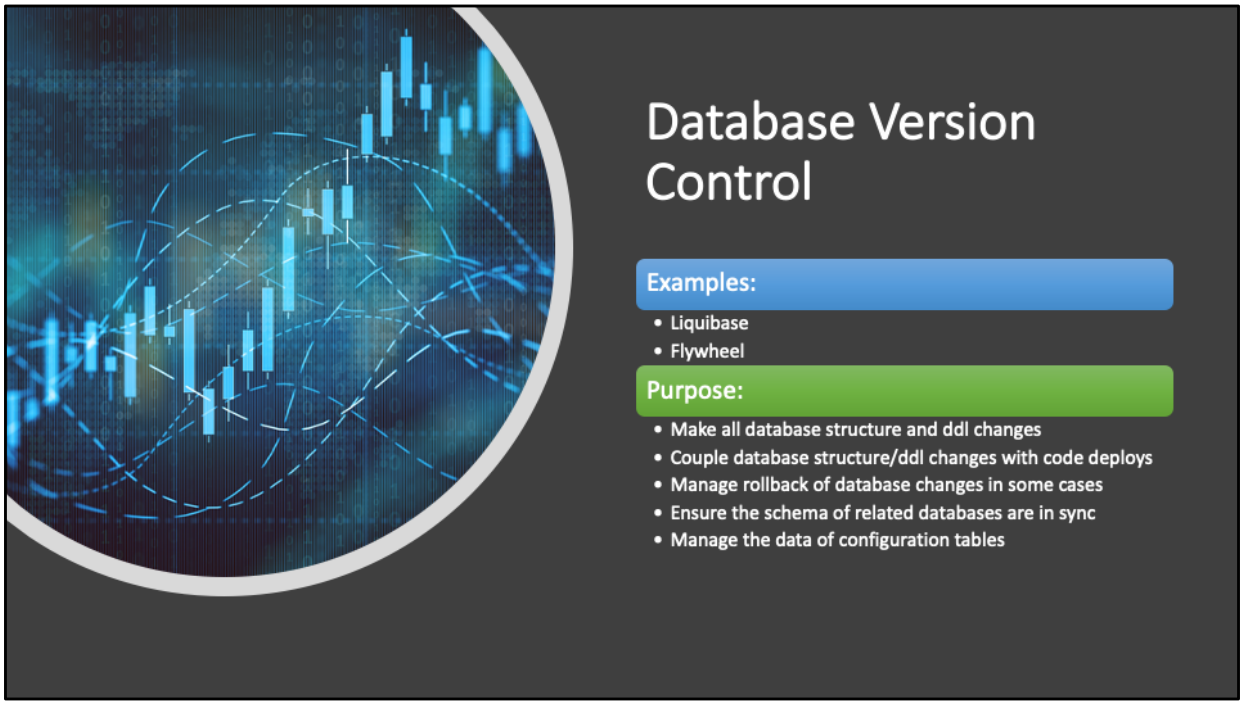
Code Version Control

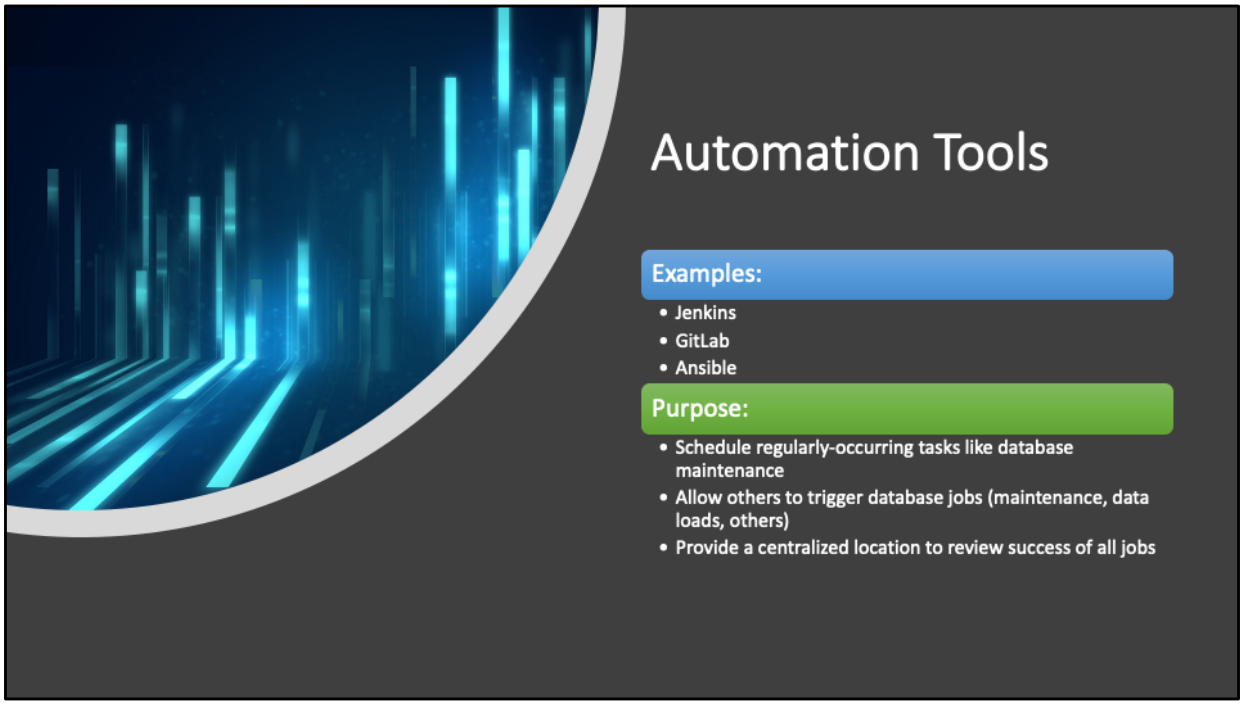
Examples:

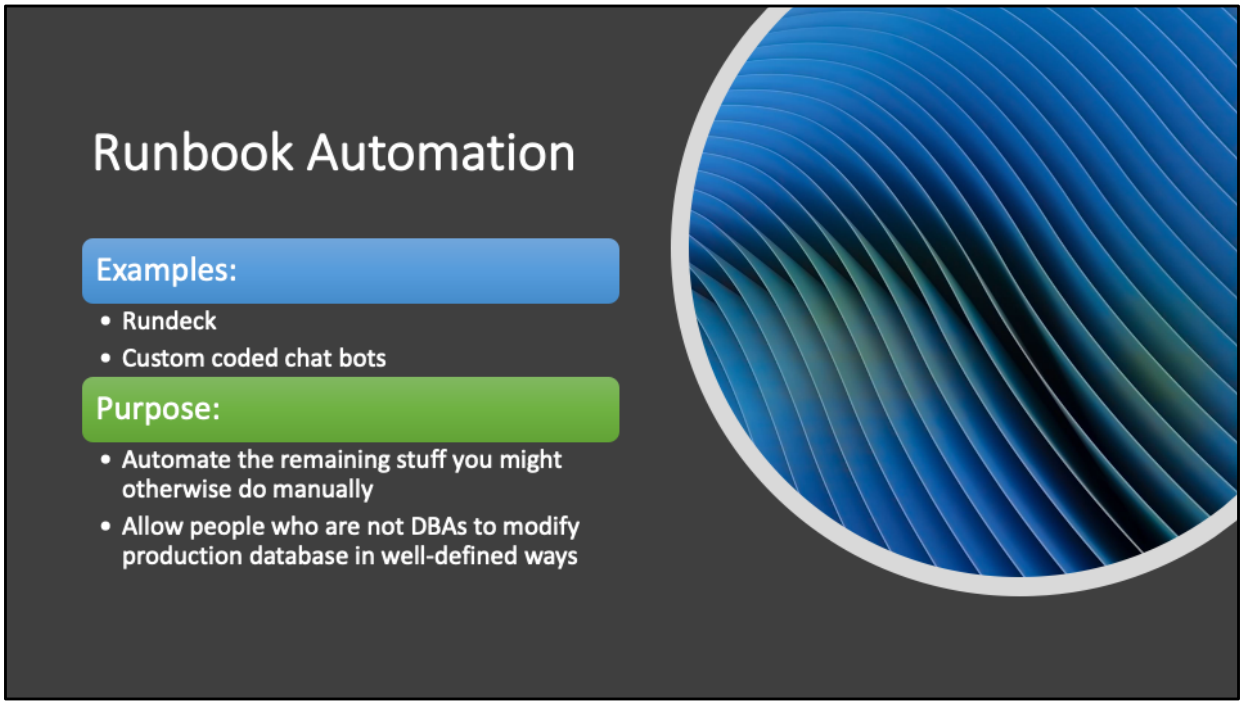
- GitHub
- Git
- SVN

Purpose:

- Manage changes to:
  - Infrastructure code
  - Automation code
  - Monitoring code

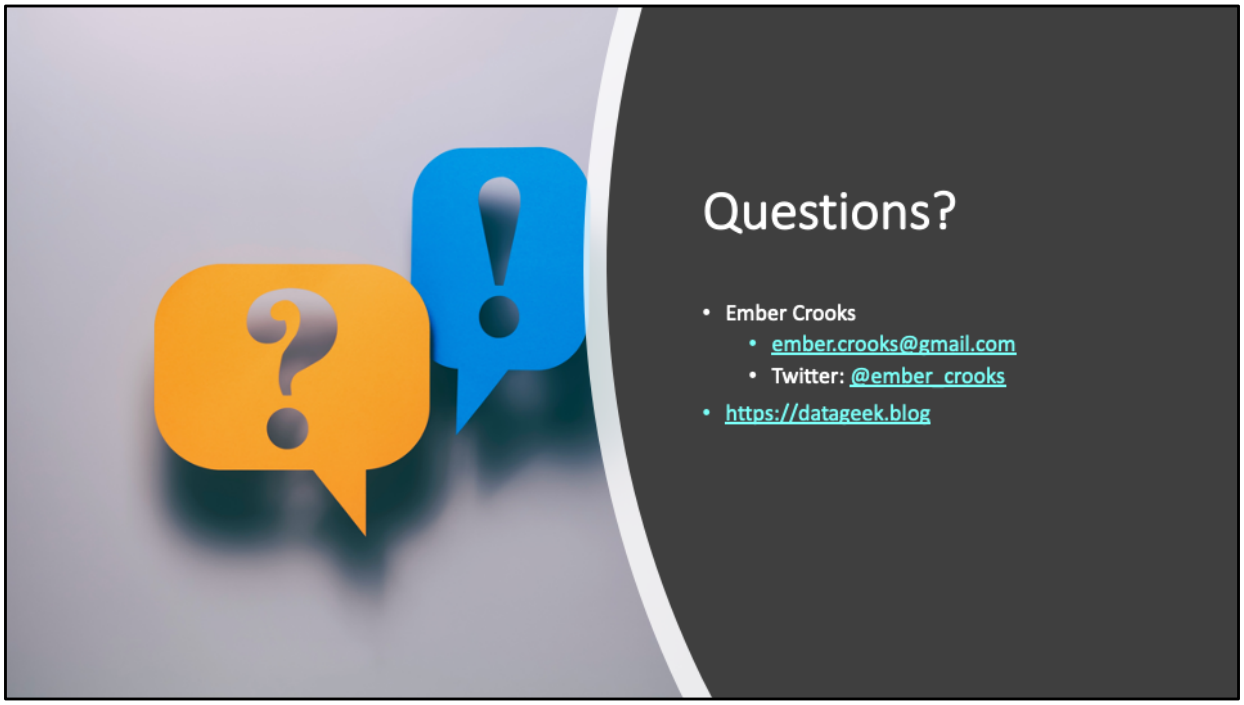

Database Version Control
Examples:
• Liquibase
• Flywheel
Purpose:
• Make all database structure and ddl changes
• Couple database structure/ddl changes with code deploys
• Manage rollback of database changes in some cases
• Ensure the schema of related databases are in sync
• Manage the data of configuration tables


Automation Tools
Examples:
• Jenkins
• GitLab
• Ansible
Purpose:
• Schedule regularly-occurring tasks like database maintenance
• Allow others to trigger database jobs (maintenance, data loads, others)
• Provide a centralized location to review success of all jobs



Examples to share

- User status update
- Enable/Disable store online ordering
- Enable/Disable website features


Questions?
• Ember Crooks
• ember.crooks@gmail.com
• Twitter: @ember_crooks
• https://datageek.blog