SQL is not an excuse to avoid DevOps

Santosh Hari

Code PaLOUsa 2019 Sponsors

Santosh Hari

Azure Consultant @ Nebbia Tech

Azure MVP

President, Orlando .NET UG

Organizer, Orlando Codecamp

santoshhari.wordpress.com

santosh.hari@newsignature.com

@_s_hari

/in/santoshhari

Agenda

- Problems database professionals experience during deployments
- Concerns about DevOps
- Why risk avoidance is bad
- How DevOps addresses and minimizes risk
- Basics of database DevOps
- Demo
- Parting thoughts

The factory of the future will have only two employees, a **man** and a **dog**. The **man** will be there to feed the **dog**. The **dog** will be there to keep the **man** from touching the equipment.

Warren Bennis - scholar, organizational consultant and author

The title and some content inspired by

acmqueue

Current Issue Past Issues Topics

The May/June 2019 issue of acmqueue is out now

Subscribers and ACM Professional members login here

Everything Sysadmin

Development

December 12, 2018

Volume 16, issue 5

PDF

SQL is No Excuse to Avoid DevOps

Automation and a little discipline allow better testing, shorter release cycles, and reduced business risk.

Thomas A. Limoncelli

Yes, I got permission

Thomas Limoncelli 6k
@yesthattom

Hey Thomas, loved your ACM article queue.acm.org/detail.cfm?ref... which is very much relevant to some projects I'm doing currently where I'm literally having to force DevOps with databases included down the throats of my clients. I was planning on doing some conference talks and workshops on this same topic. Would you be ok with me using some portions of your post, especially the title? I'll make sure I attribute. Let me know. Either way, great post. The community needed something like this

Dec 18, 2018, 1:19 PM ✓

Hi! It's an academic publisher. Please feel free to quote and cite appropriately! Best, Tom

Apr 18, 2019, 9:02 PM

Thank you for the response and clarification!!

Apr 19, 2019, 1:07 AM ✓

Current state of database deployments

- Deployments take hours if not days
- Weeks of negotiation with moving codebases
- Infrequent aka big bang
- Often performed on weekends
- No control over when to deploy your changes

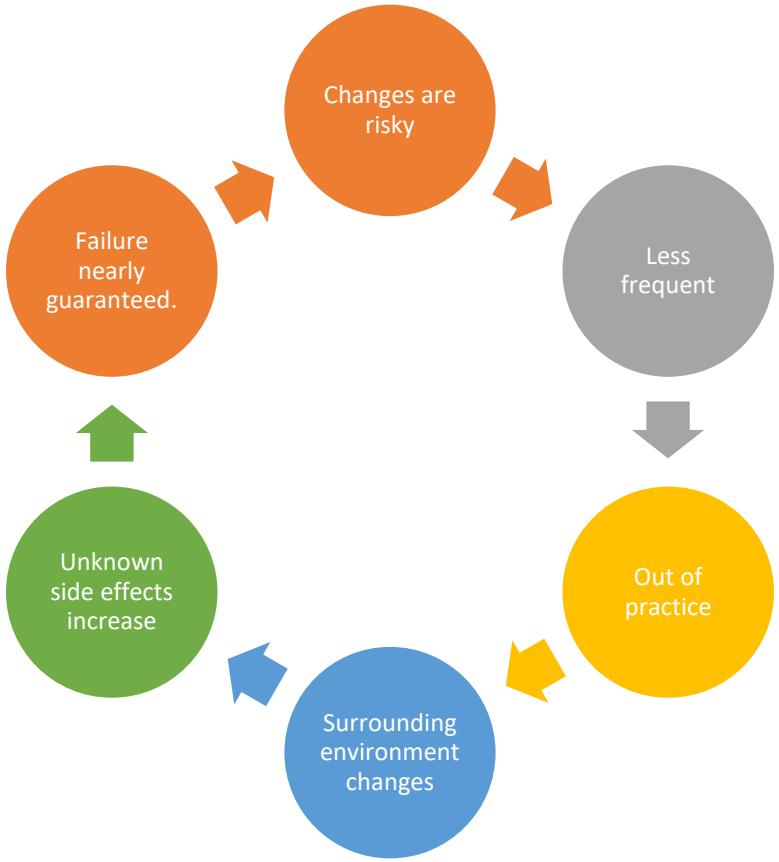
Jumping through hoops for database deployments

- Have to perform gymnastics
- Lock out users/apps
- Take Offline
- Disable data writes
- Run in parallel and sync up data

Concerns with DevOps

DEPLOYMENTS TAKE TIME

DEPLOYMENTS ARE RISKY



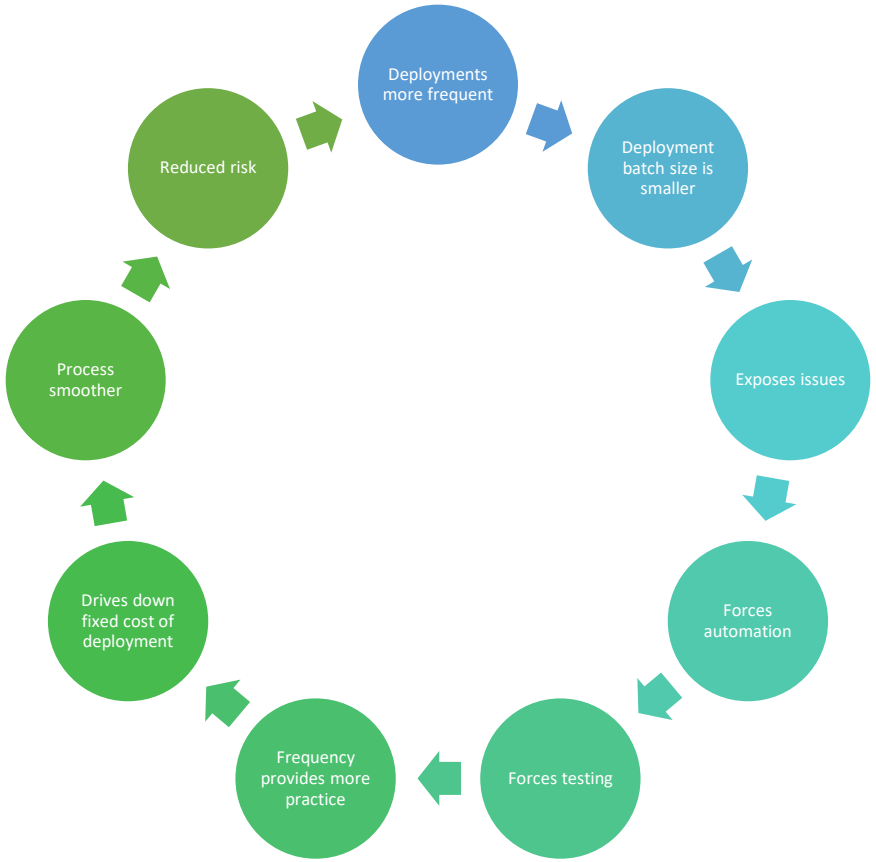
Risk avoidance causes fragility

DevOps is the union of people, process, and products to enable continuous delivery of value to our end users

Donovan Brown, Cloud Advocate Manager of the Methods and Practices Org, Microsoft

How to DevOps

- Continuous Integration (CI)
- Continuous Delivery (CD)
- Version Control (VC), mostly Git
- Agile planning and lean project management
- Monitoring and Logging
- Public and Hybrid Clouds
- Infrastructure-as-Code (IaC)
- Microservices
- Containers
- Shift Left philosophy



DevOps is anti-fragile

DevOps for databases

- Everything is automated
- Schema change management - State-based or migration-based?
- Unit tests
- Schema validation
- Data (Lookup tables, for instance)
- Security – firewall, usernames/passwords
- Infrastructure and config
- Logging and monitoring
- Feature Flags
- Above all, work closely with the rest of the team

- Deploy database from scratch
- Manage subsequent schema changes
- Infrastructure create/change

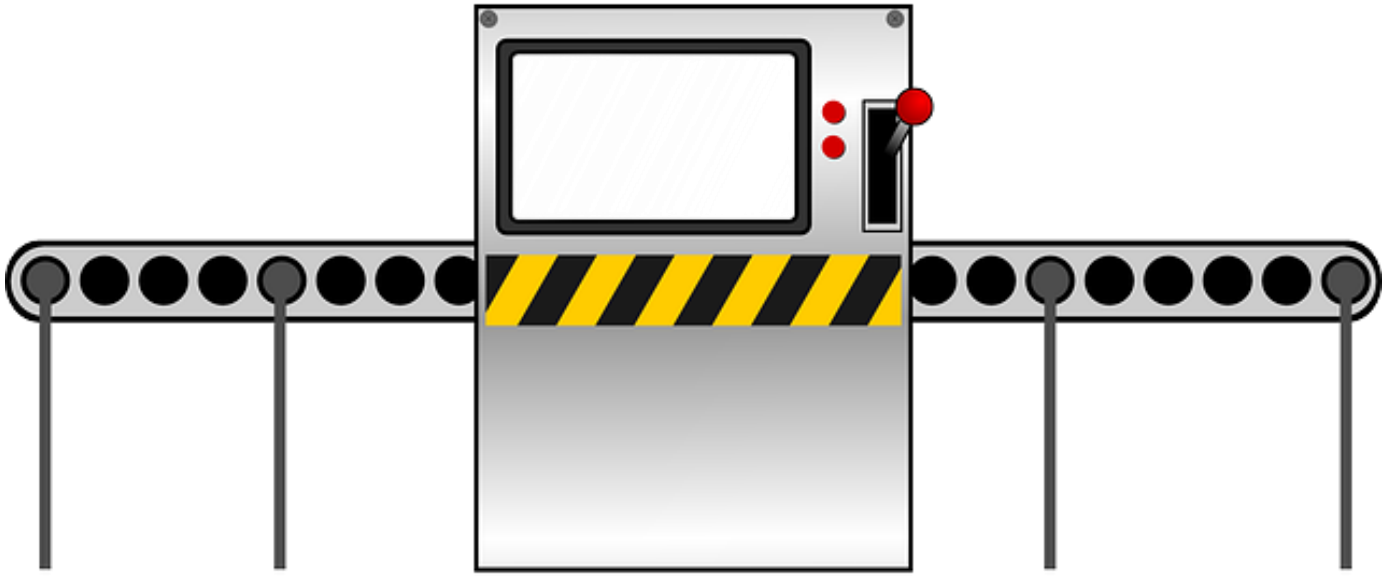
Demo

Parting thoughts

Don’t throw stuff over the wall

Don’t attempt to boil the ocean

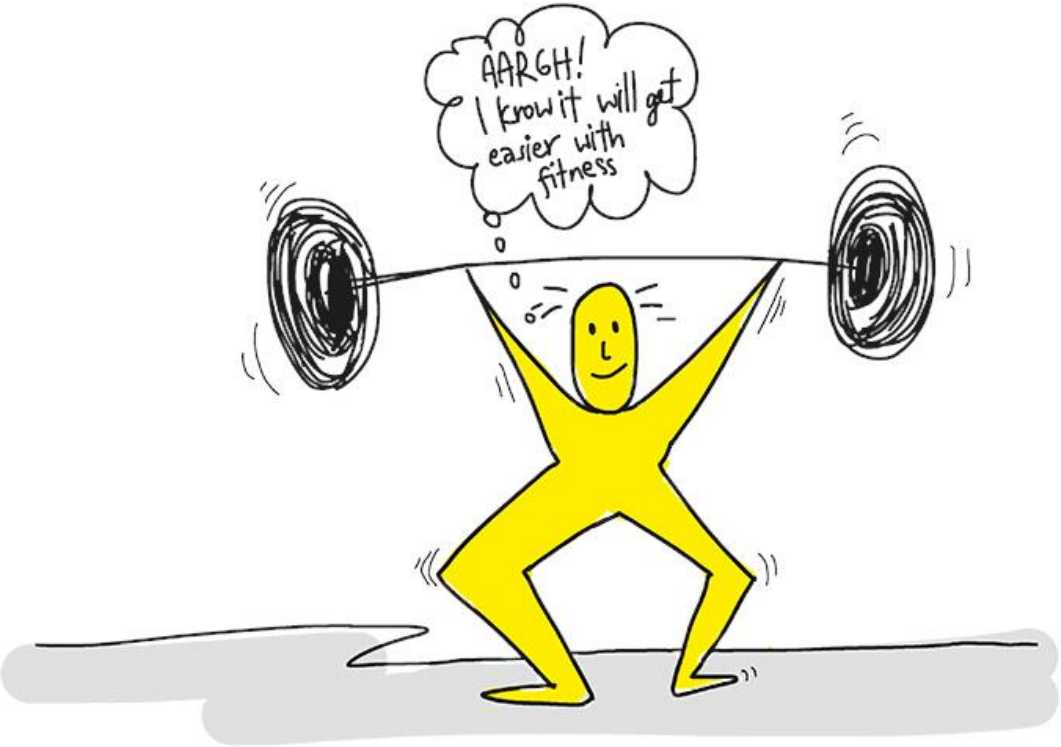
Pipeline and automate everything



Reduce drift between environments

Log and monitor everything

DevOps *will* hurt at first



DevOps is a journey

Resources

SQL is no excuse to avoid DevOps – ACM – Thomas Limoncelli
https://queue.acm.org/detail.cfm?ref=rss&id=3300018

Official Microsoft SQL Server DevOps https://www.microsoft.com/en-us/sql-server/developer-get-started/sql-devops/

What is DevOps? https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops

Redgate – State based or migration based https://www.red-gate.com/library/state-or-migrations-based-database-development

The Phoenix Project - https://www.amazon.com/Phoenix-Project-DevOps-Helping-Business/dp/1942788290

Accelerate - https://www.amazon.com/Accelerate-Software-Performing-Technology-Organizations/dp/1942788339

Santosh Hari

Azure Consultant @ Nebbia Tech

Azure MVP

President, Orlando .NET UG

Organizer, Orlando Codecamp

santoshhari.wordpress.com

santosh.hari@newsignature.com

@_s_hari

/in/santoshhari

State-based database change management

- SSDT or Redgate SQL Source Control (supports hybrid)
- Single step between current and desired state
- Mostly CREATE statements
- Easy workflow for large teams and complicated databases
- Scripts generated at deployment time using comparison
- Developer involvement minimal
- UI based and automated options available
- Bad for scenarios like column-split or datatype/constraint changes because context is lost

Migration-based database change management

- Migration-first: EF, DbUp, Redgate SQL Change Migration (supports hybrid)
- Captures individual change scripts during dev to effect larger iterative migrations
- Numerically ordered migrations
- Mostly ALTER commands
- Works well with simple data stores for table refactoring or data migration
- Dev and DBA collaborate early in dev cycle to define deployment steps
- Can be executed as code/scripts both manual and through automation
- Last check-in would dominate

Automated Schema Updates

- Automated Schema Updates

Coding for Multiple Schemas

- Coding for Multiple Schemas

Infrastructure and environments

Generate Test Data

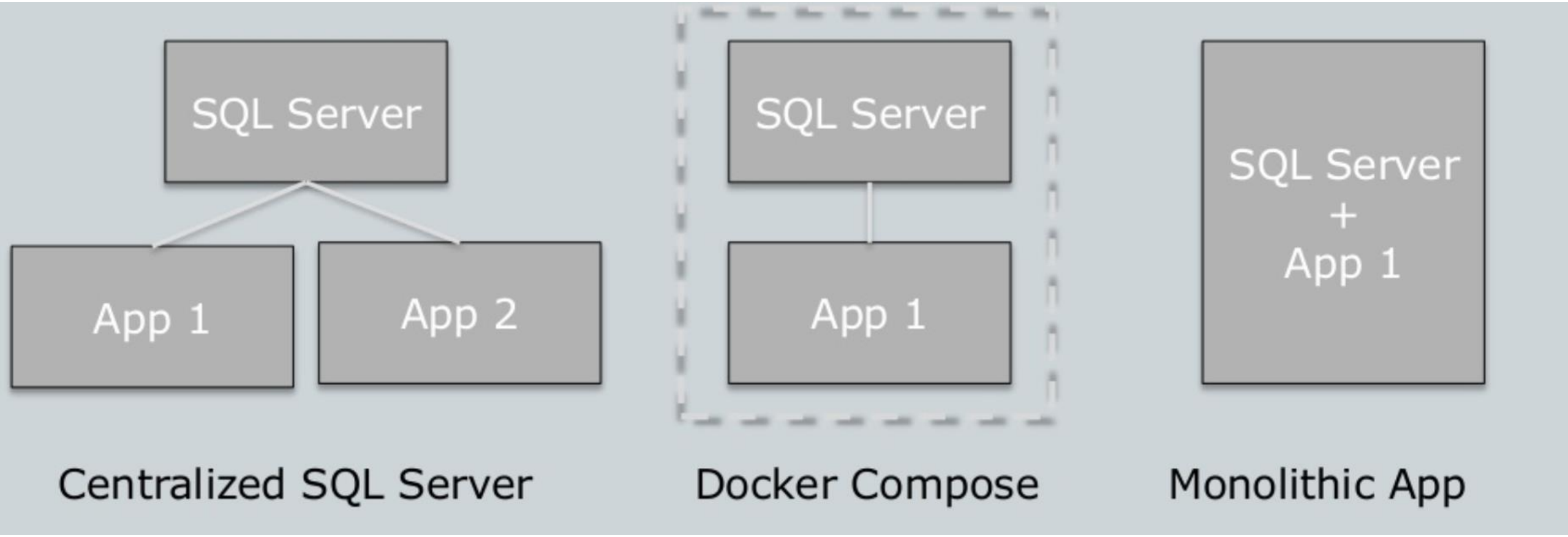
- Generate Test Data

Log and monitor everything

Multi-phase schema commit

- Propose New Schema: checkin new schema, contains both old (primary) and new (backup) fields, no field deletes/renames, DAO abstracts column type changes, make app new schema, write to both old and new columns, migrate old data as needed, easy to rollback
- Flip source of truth: new fields should be primary, old fields will still be valid. Previous release will still work
- Remove old references: Once you're certain you're not going to roll back, stop reading/writing the old data. This is your hard flip.
- requires at least two periods of schema change and one period that involves multiple software rollouts
- requires that your application is well formed. Your application needs to be properly modularized, use reasonable DAOs or otherwise have complete understanding of how the data is accessed

App Deployments with Containers



Santosh Hari

Azure Consultant @ Nebbia Tech

Azure MVP

President, Orlando .NET UG

Organizer, Orlando Codecamp

santoshhari.wordpress.com

santosh@nebbiatech.com

@_s_hari

/in/santoshhari