



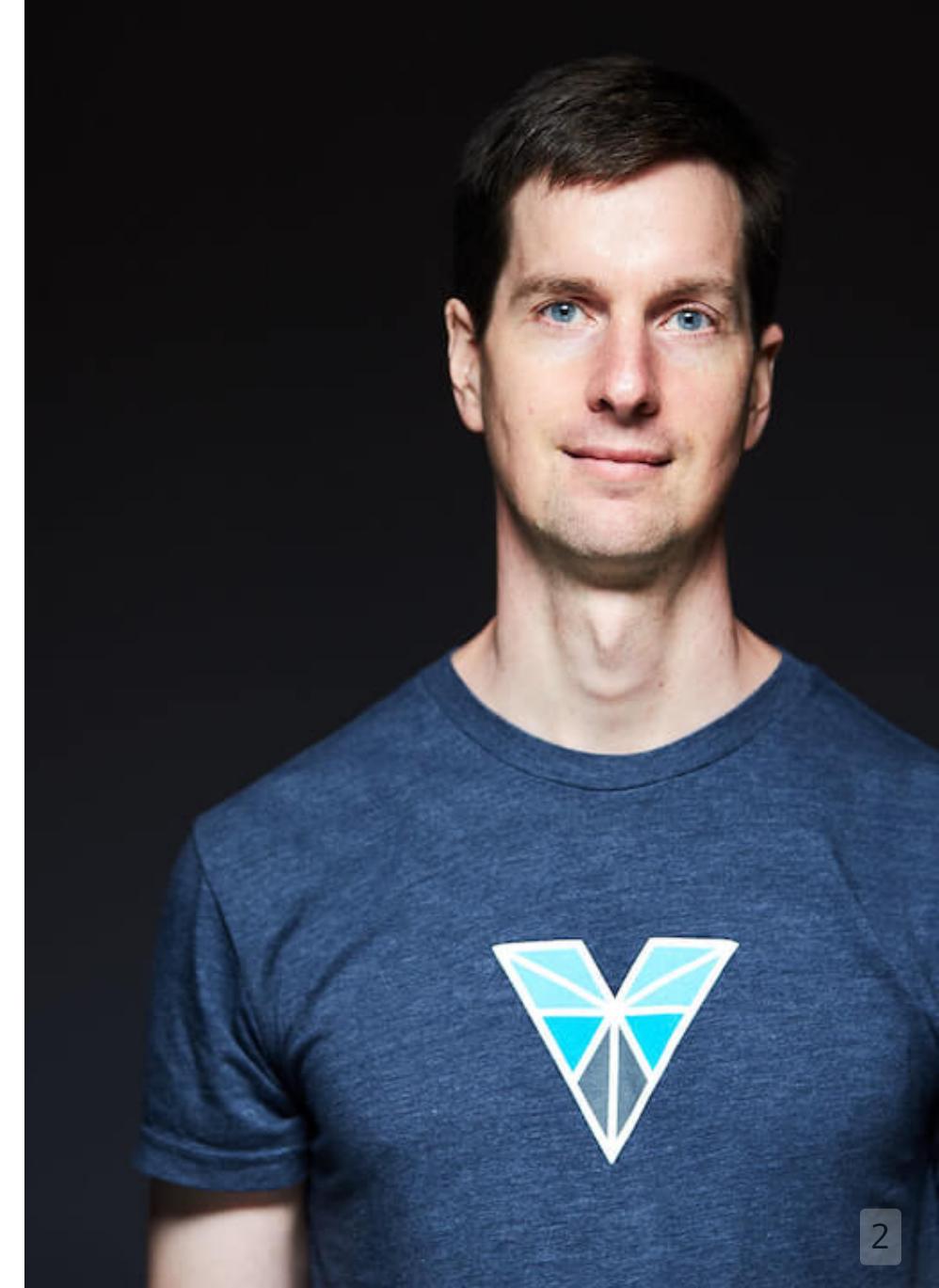
DevOps for the Database

Baron Schwartz · QConSF 2018

Introduction

I've been focused on databases for about two decades, first as a developer, then a consultant, and now a startup founder.

I've written several books (High Performance MySQL)... and created a lot of open source software, mostly focused around database monitoring, database operations, and database performance: innnotop, Percona Toolkit, Percona Monitoring Plugins among others.



Agenda

- Overview of database DevOps
- How companies succeed
 - Tooling, culture, process, people
- How companies fail
- Challenges to database DevOps
- Resources, etc

Slides are online at xaprb.com.

My Twitter is @xaprb, my email is baron@vividcortex.com

Three Database DevOps Stories

1. One DBA, hundreds of developers, growing the DBA team



Three Database DevOps Stories

1. One DBA, hundreds of developers, growing the DBA team
2. One DBA, from 20 to 100 developers



Three Database DevOps Stories

1. One DBA, hundreds of developers, growing the DBA team
2. One DBA, from 20 to 100 developers
3. Two database ops folks, seventeen developers



Benefits of Database DevOps

DevOps brings the same benefits to the database as everywhere else.



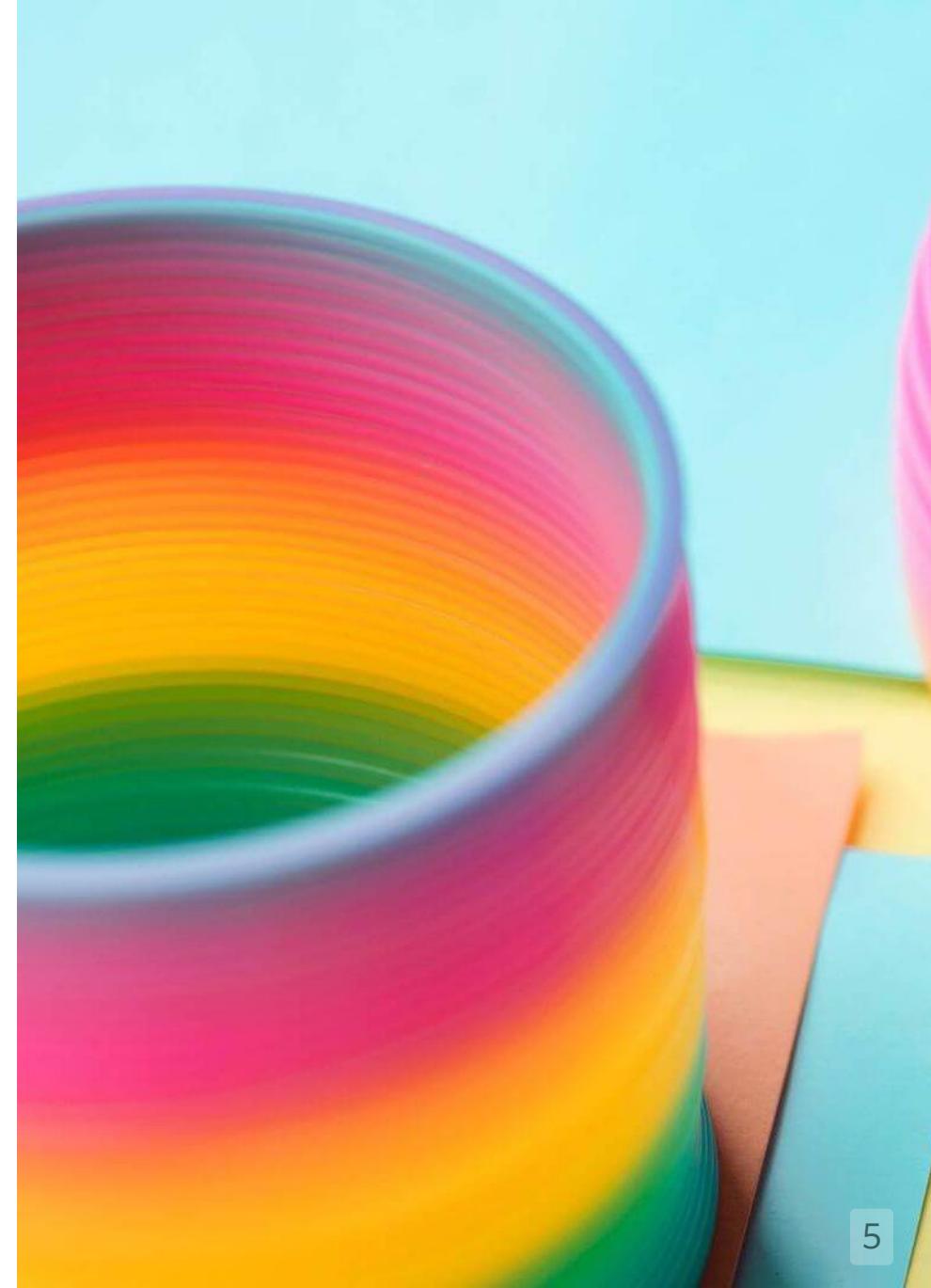
Benefits of Database DevOps

DevOps brings the same benefits to the database as everywhere else.

For software delivery performance in particular:

- Faster, better, cheaper—pick all three
- stability -> speed -> stability -> speed

(See the 2018 State of DevOps Report!)





Detriments of Lacking DevOps

Without DevOps, speed and quality suffer:

- Mismatched responsibility and authority
- Overburdened database operations personnel
- Broken feedback loops from production
- Reduced developer productivity

What Is Database DevOps?

Attributes I've seen in companies that apply DevOps to their database:

- Developers own database schema, workload, and performance
- Developers debug, troubleshoot, and repair their own outages
- Schema and data model as code
- A single fully-automated deployment pipeline
- App deployment includes automated schema migrations
- Automated pre-production refresh from production
- Automation of database operations, to an RDS-like level

From the 2018 State of DevOps Report

Database changes are often a major source of risk and delay when performing deployments... **integrating database work into the software delivery process** positively contributed to continuous delivery... good communication and comprehensive configuration management that includes the database matter. Teams that do well at continuous delivery **store database changes as scripts in version control and manage these changes in the same way as production application changes**... when changes to the application require database changes, **these teams discuss them with the people responsible for the production database**

Emphasis mine. See p. 57

Bringing DevOps to the Database



Core Elements

1. People
2. Culture
3. Structure and Process
4. Tooling



Tooling: Deploy/Release

- You need frequent, automated deploys
- Eliminate manual work (toil)
- Continuous integration and deployment

Tooling: Monitoring and Observability

- Instrumentation, telemetry, analytics, monitoring, observability
- Monitoring: the Seven Golden Signals ([CELT + USE](#))
- Observability is built on these foundations

Tooling: Shared Knowledge and Process

- DBRE processes, mentality, rubrics
- Deploy confidence procedures
- Documentation to share SME experience & skill
- Notifications should link to runbooks

Deploy to Production

51069 → 51069

PROD!!! →

Important links:

- What to watch after a push
- Log Watcher (supergrep)
- Forums

Structure: Team Orientation

Teams work best when they:

- Are service- or product-oriented
- Are loosely coupled, autonomous
- Are highly aligned and trusted
- Own what they build

My personal experience: Conway's Law is true.

Process: First, Do No Harm

- Stabilize the patient, *then* transport
- Protect production (no holes below the waterline)



Process: Plan And Roadmap

- Work is work—maintain a single backlog
- Embrace DevOps in small chunks and build on success
- Lay out the progression in stages
- Emphasize what's not changing, too



Process: Getting Started

Pick a place to start.

- One team
- One app, service, or product
- First new, then established/legacy

Culture: Change

- Culture is emergent; you can't operate on it directly
- Create a new path of least resistance
- Include everyone in consequences and benefits



Culture: Leadership Support

- Exec mandate sometimes works; so does attraction
- Starve the old way
- Leadership isn't just top of org chart

Culture: Communication and Trust

- Align around a North Star: a simple, compelling *why*
- Customer-centric culture, customer empathy
- Psychological safety enables risk-taking



People: You Need Experts

- You do need database expertise
- You do not need a database caretaker
- Engineers can learn database competency



Pathways To Failure

Tooling FAIL

- Fragile or too-ambitious automation
- Lack of automation / accepting manual toil
- Two routes to production—code vs DB

Culture FAIL

- Any friction in the way of change
- Failure to create incentives to change
- Relying on a vendor to bring culture
- Insisting on adherence to One True Way
- Clinging to legacy DBA roles and duties

Aside: Legacy DBA Roles

In a traditional sense, the job of the DBA means she is the only person with access to the servers that host the data, the go-to person to create new database cluster for new features, the person to design new schemas, and the only person to contact when anything database related breaks in a production environment.

— Silvia Botros, SendGrid



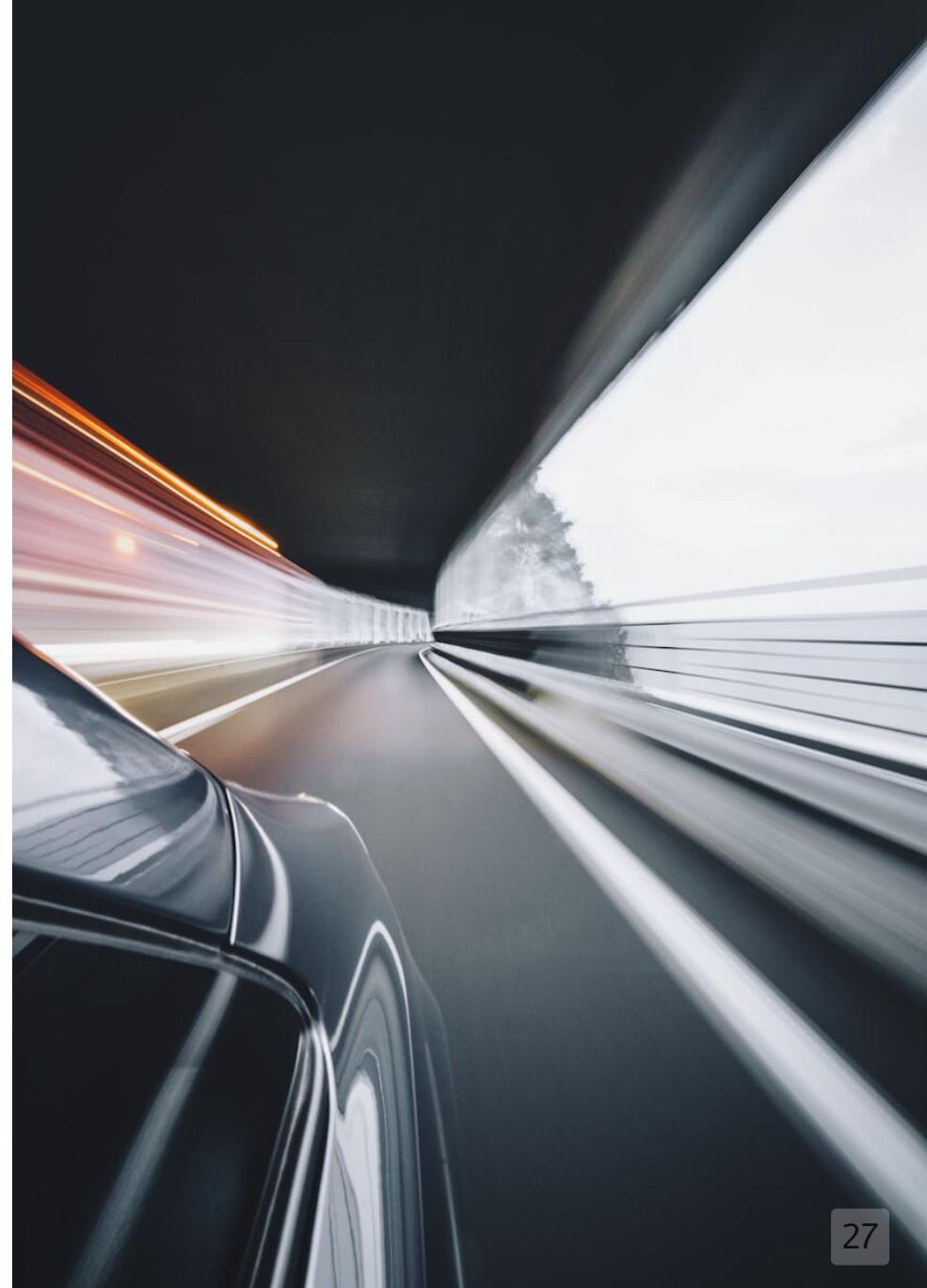
Leadership FAIL

- Underinvesting in experience and skill
- Lack of management support
- Micromanagement, failure to manage up

See also my [Kafka Summit talk](#) on advocating technical decisions.

Planning FAIL

- All-or-nothing
- Too much too fast
- Velocity over resilience



A large, complex stone maze with a circular entrance on the right. The path is winding and filled with many dead ends, symbolizing a difficult challenge.

What's The Hardest Part?

Challenge: Politics

- Selling DevOps to the DBA
- Selling DevOps to leadership
- Creating culture change

Challenge: Tooling

- Legacy databases aren't cloud-native
- Data tier ops tooling isn't as mature
- Schema changes
- Integration with e.g. canary deploys, feature flags

Challenge: HA, Scale, Performance

- Failover and recovery
- Locking/blocking
- Nonblocking schema changes at scale

Whither The DBA?

- Become a DBRE instead of a DBA
- Focus on data platform and architecture
- Be the subject matter expert supporting product teams

The Rewards

- The outcomes
- The process itself
- Individual benefits



Acknowledgments

Many people contributed to this talk. Thank you.

(When I get their OK I will add their names.)

Slides and Contact Information

Slides are at

<https://www.xaprb.com/talks/> or you
can scan the QR code.

Contact: baron@vividcortex.com,
@xaprb

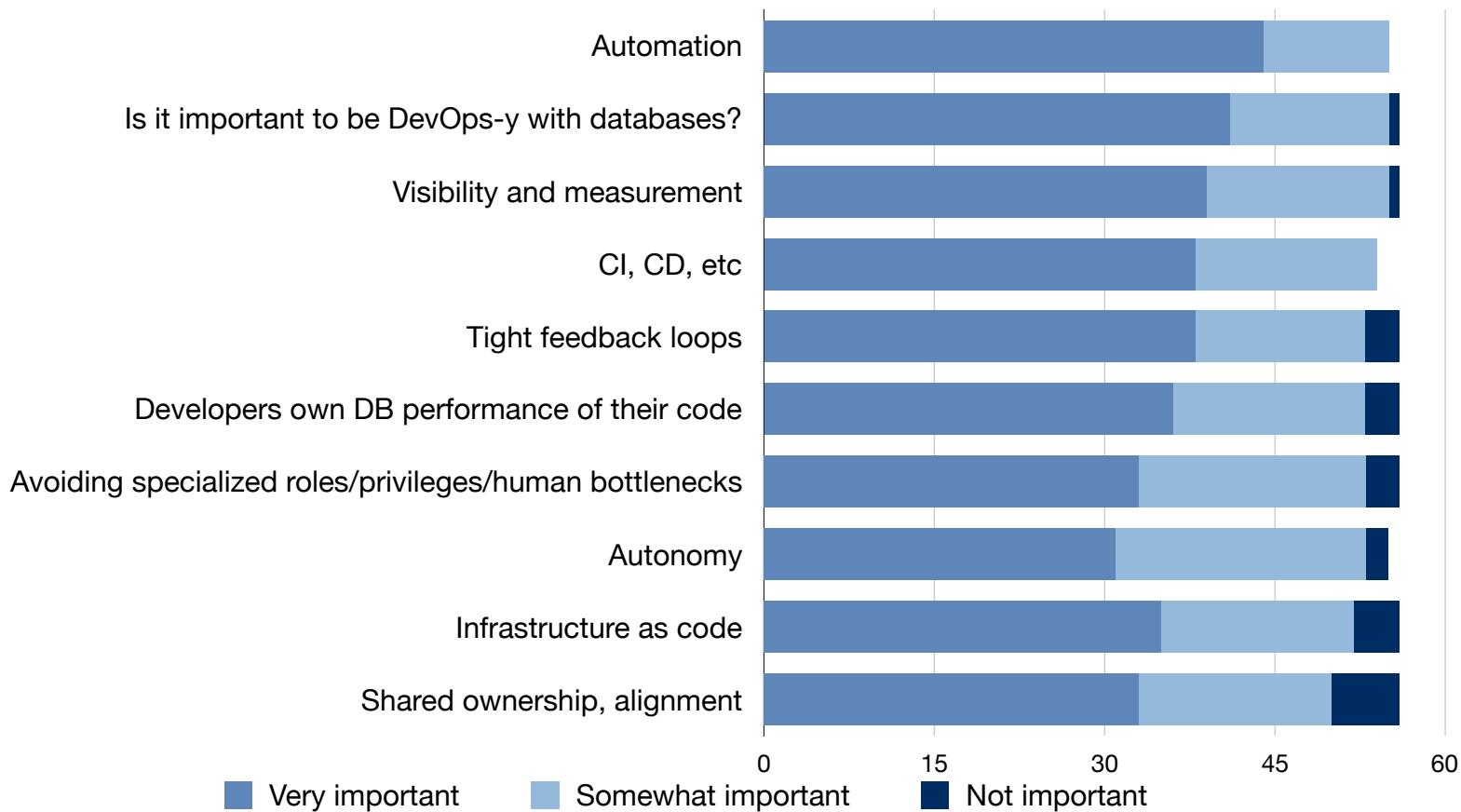


Appendix: Survey

I asked my Twitter followers to respond to an informal, nonscientific survey. The following two slides summarize some of the quantitative feedback.

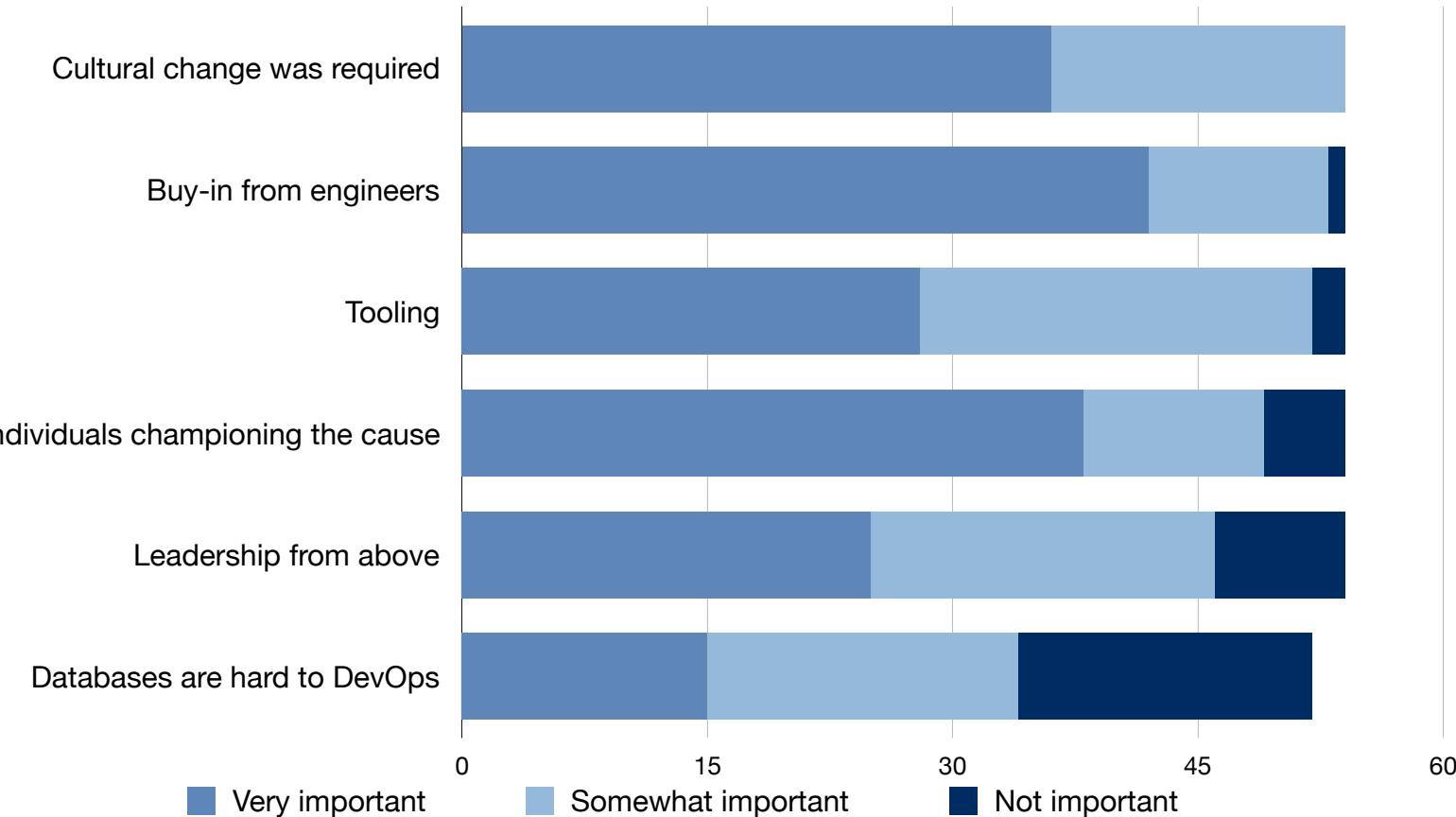
Survey Results

How important are each of the following in your view of “Database DevOps”?



Survey Results Cont'd

How do you rate the importance of these factors in making progress?



Resources

- Slides/Video: [How to Monitor a Database](#) including the Seven Golden Signals (CELT+USE)
- [The Phoenix Project](#)
- [Database Reliability Engineering](#)
- The [2018 DORA State of DevOps Report](#)
- [Three Steps To Psychological Safety](#)
- My [Kafka Summit talk](#) on advocating technical decisions