AWS re:Invent

WIN310

Hands-On: Building a Migration Strategy for SQL Server on AWS

Brian Beach
Solutions Architect
Amazon Web Services

AWS re:Invent

aws

Agenda

Common migration strategies (10 minutes)

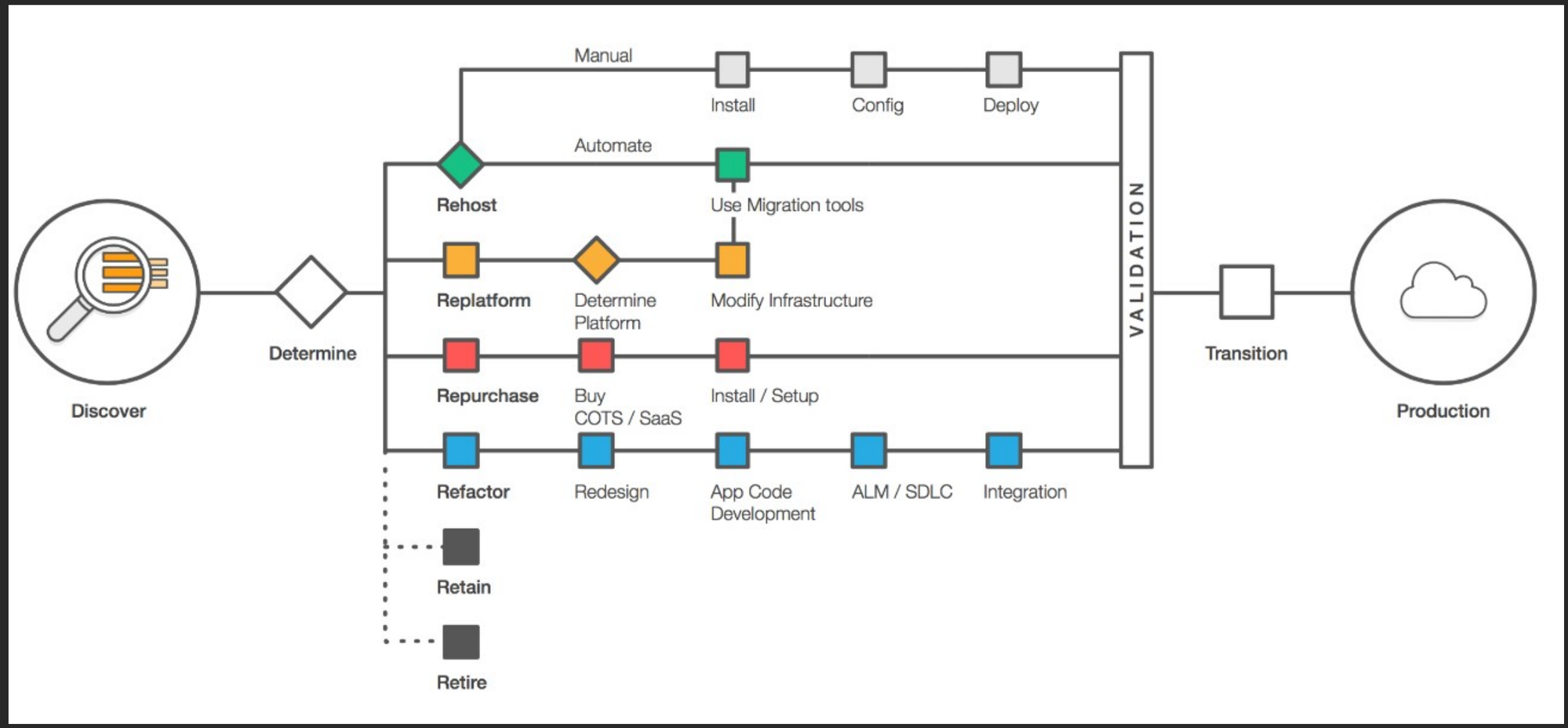
Data migration options (5 minutes)

Workshop scenario (5 minutes)

Work in groups (90 minutes)

Debrief (10 minutes)

Common migration strategies (the Six Rs)



Which option is right for you?

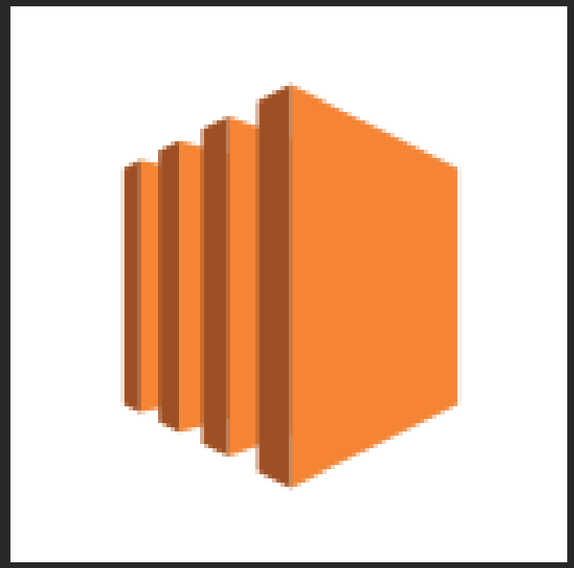
Rehost: Run SQL Server on Amazon Elastic Compute Cloud (Amazon EC2)

Replatform: Migrate to Amazon Relational Database Service (Amazon RDS)

Refactor: Migrate to Amazon Aurora, Amazon Redshift, Amazon DynamoDB, Amazon Neptune, and others

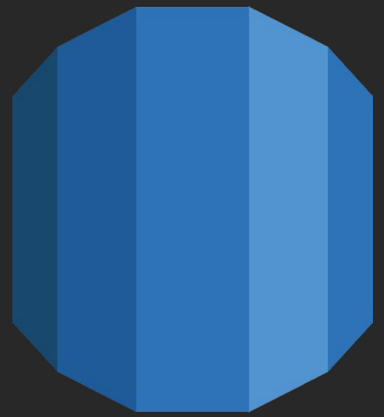
Rehost: Run SQL Server on Amazon EC2

- Familiar administration experience
- Full control over the environment
- All SQL Server features available
- All SQL Server versions supported



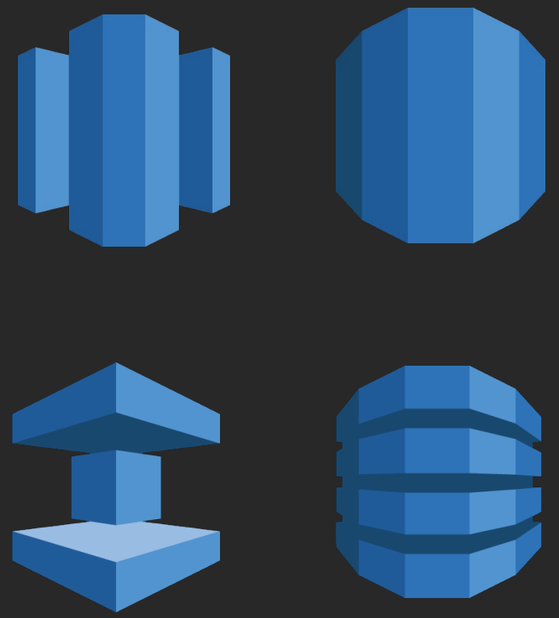
Replatform: Run SQL Server on Amazon RDS

- Optimized architecture
- Automated patching
- Automated backups
- Proven high availability



Refactor: Adopt cloud-native services

- Aurora: SQL/OLTP
- Amazon Redshift: SQL/OLAP
- DynamoDB: NoSQL
- Neptune: Graph



Staying on SQL Server?

Amazon RDS SQL Server

- Managed physical infrastructure
- Managed DB install and backups
- Managed OS and patching
- Managed high availability/scaling

Your responsibility

- App optimization and tuning
- Deployment orchestration

Cloud-native solution focusing on

- Business value tasks
- High-level tuning tasks
- Schema optimization

No in-house database expertise

SQL Server on Amazon EC2

- Managed physical infrastructure
- Managed OS installation
- Managed scaling
- OS-level control

Your responsibility

- App optimization and tuning
- Deployment orchestration
- Monitoring and recovery
- High availability
- Backups
- DB and OS patching

Need control over

- DB instance and OS
- Backups, replication, and clustering
- Sysadmin role

Need capabilities not available in Amazon RDS

Which migration strategy is right for you?

Rehost
SQL Server on Amazon EC2

- Familiar administration experience
- Full control over the environment
- All SQL Server features available
- All SQL Server versions supported

Replatform
SQL Server on Amazon RDS

- Optimized architecture
- Automated patching
- Automated backups
- Proven high availability

Refactor
Adopt cloud-native services

- Aurora: SQL/OLTP
- Amazon Redshift: SQL/OLAP
- DynamoDB: NoSQL
- Neptune: Graph
- Eliminate SQL Server licensing costs

Data migration options

AWS re:Invent

aws

Data migration: SQL Server on Amazon EC2

- Backup/restore
- SQL Server always on
- SQL Server replication
- Third-party tools

Data migration: SQL Server on Amazon RDS

- Backup/restore: Requires an outage
- AWS Database Migration Service (AWS DMS)
- Third-party tools

Data migration: Heterogeneous

- Database Migration Service
- Third-party tools

Data migration process

Data migration *to SQL Server on Amazon EC2*

- Backup/restore
- SQL Server always on
- SQL Server replication
- Third-party tools

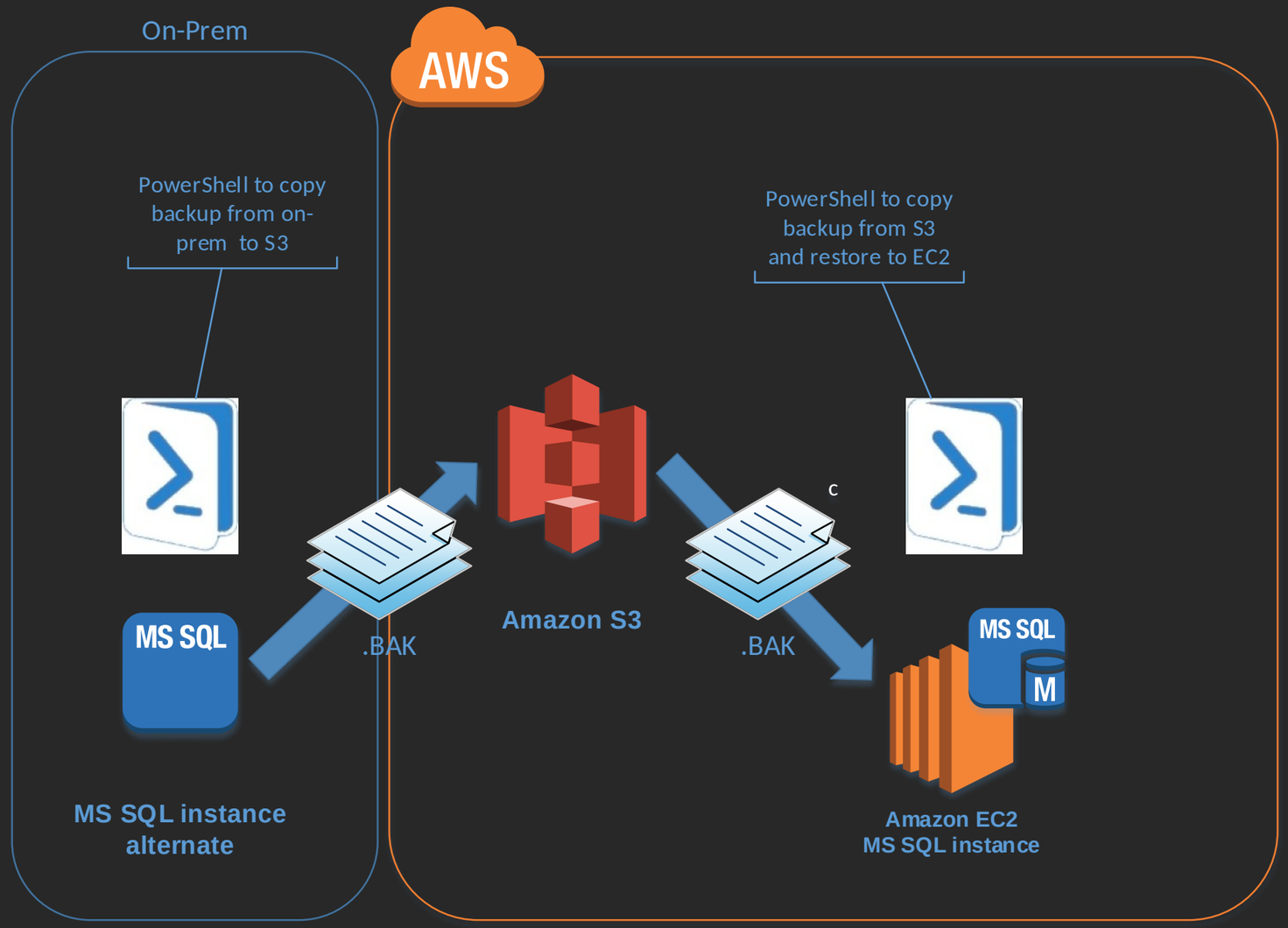
Data migration *to SQL Server on Amazon RDS*

- Backup/restore *(Requires outage)*
- Database Migration Service
- Third-party tools

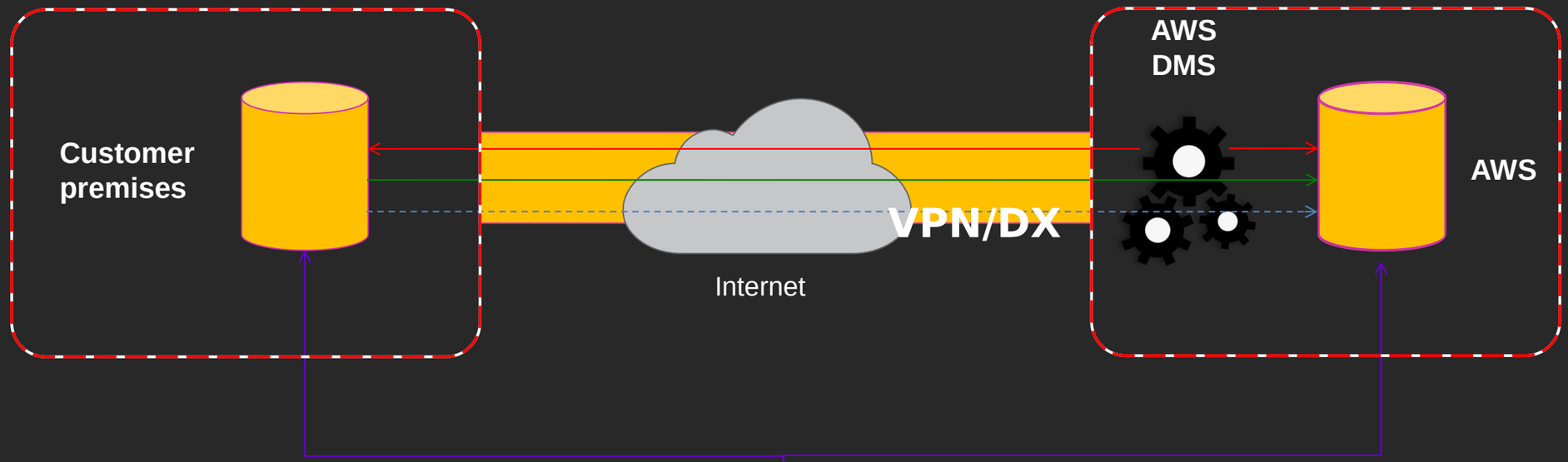
Data migration *to cloud-native services*

- Database Migration Service
- Third-party tools

Backup/restore to Amazon RDS



AWS Database Migration Service (AWS DMS)



Start a replication instance

Connect to source and target databases

Select tables, schemas, or databases

- Let AWS DMS create tables, load data, and keep them in sync
- Switch applications over to the target at your convenience

AWS Schema Conversion Tool (AWS SCT)

The AWS Schema Conversion Tool helps automate many database schema and code conversion tasks when migrating between database engines or data warehouse engines

Features

Oracle and Microsoft SQL Server schema conversion to MySQL, Amazon Aurora, MariaDB, and PostgreSQL
Or convert your schema between PostgreSQL and any MySQL engine
Database Migration Assessment report for choosing the best target engine
Code browser that highlights places where manual edits are required
Secure connections to your databases with SSL
Cloud-native code optimization

Workshop scenario

AWS re:Invent

aws

Workshop scenario: Overview

- You are a contractor for an online ticket broker that sells tickets to sporting events, concerts, and so on
- The company stores data in an SQL Server 2008 R2
- SQL Server 2008 R2 is end-of-life and the company is currently paying for extended support
- The company wants to upgrade to SQL Server 2017
- They see this upgrade as an opportunity to

Workshop scenario: Architecture

- On Prem, SQL Server is running Enterprise Edition
- HA is achieved using a failover cluster with shared storage on a Fibre Channel attached SAN
- SQL Server is running on physical servers with dual socket cores CPUs and 16GB RAM in each server
- The application uses SQL Mail to send order confirmations to customers
- The database schema is available here

Workshop scenario: Humans

- You have the full support of the CIO. He is excited about moving to AWS, and is open to your recommendations. He wants to minimize cost.
- The CISO is open to AWS, but wants to ensure you can achieve end-to-end encryption
- The DBA team has some experience with MySQL but are most comfortable in SQL Server
- The DEV team is comfortable in multiple languages and operating systems

Workshop scenario: Checklist

- What target environment will you recommend?
- How will you migrate data to AWS?
- Is there an outage required during the move?
- What changes need to be made to the application?
- What security controls will you employ?

Debrief

AWS re:Invent

aws

Debrief

- Placeholder

ProServe

- Placeholder for ProServe migration services

Thank you!

Brian Beach

AWS re:Invent

aws

Please complete the session survey in the mobile app.

AWS re:Invent

aws