ARC414-S

Accelerated analytics: Building the next-gen data platform for Hertz

Lucy Meyo
VP Architecture and
Integration
Hertz

Rohit Sinha
Senior Manager
Analytics and Cognitive
Deloitte Consulting LLP

Gowtham Ramu
Specialist Leader
Cloud Engineering
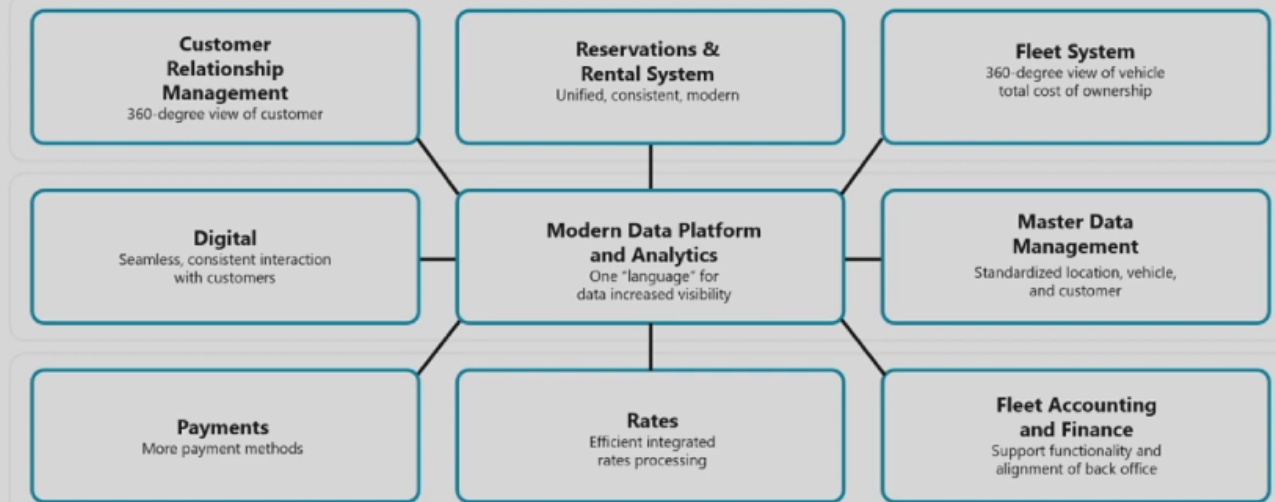Deloitte Consulting LLP

re:Invent

aws

Hertz is undertaking a massive digital transformation to evolve its technology landscape. This move provides an opportunity to extract valuable insights from a large amount of data produced by the new systems. In this session, learn how Deloitte collaborated with Hertz to build a next-generation data platform, which includes an integration hub, a unified reporting layer, and an ecosystem of tools used to build advanced analytics models. Discover how the solution leveraged all native AWS services, such as Amazon S3, Amazon Kinesis, Amazon EMR, and AWS Lambda, to enable cross-functional insights and accelerate the cloud journey in under 12 months. This presentation is brought to you by Deloitte, an APN Partner.

Agenda

1 The opportunity

2 Accelerated journey to the cloud

3 Role of next-generation data platform

4 Cloud-native architecture

5 Key takeaways

Digital Transformation

The goal of the Digital Transformation program at Hertz is to deploy an integrated global solution that provides a better view of our customer and greater visibility into our assets

Customer Relationship Management 360-degree view of customer	**Reservations & Rental System** Unified, consistent, modern	**Fleet System** 360-degree view of vehicle total cost of ownership
Digital Seamless, consistent interaction with customers	**Modern Data Platform and Analytics** One "language" for data increased visibility	**Master Data Management** Standardized location, vehicle, and customer
Payments More payment methods	**Rates** Efficient integrated rates processing	**Fleet Accounting and Finance** Support functionality and alignment of back office

We have over 1.4 million cars, 19 million global customers…we need to provide a seamless new digital journey using our data powered modern data platform and analytics.

The opportunity

A **modern data and analytics platform** was critical to the success of the overall **digital transformation at Hertz** to extract value and insights from large amount of data being produced from new systems and create a common platform for cross-functional insights and answering new business questions using the advanced analytics and data science models.

- Sharply reduce TCO of data management and analytics with the move to AWS

- Leverage scalable elastic compute of cloud infrastructure for modern analytics and data science use cases

- Centralize data assets in AWS data lake

- Improve regulatory and security controls with the process of centralizing data assets in AWS data lake

- Use cloud infrastructure scalability and elasticity as well as spot instances to optimize cost of compute-intensive operations and answer new business questions

- With pattern-based ingestion and component-based architecture, enable quicker time to implement new use cases

- Enable digitization through new business models around data and analytics

Hertz journey of how Deloitte helped build the MVP for the data platform in less than 10 months

Data and reporting requirements	**Data source discovery**	**Architecture alignment and infrastructure standup**	**Infrastructure automation**
Defined data and reporting requirements across all functions at Hertz and programs within digital transformation to support unified reporting, self-service BI, and analytics	Leveraged data and reporting requirements to do source system discovery; mapped new source system integrations as well as legacy systems and catalog integration gaps	Achieve alignment across a scalable, forward-looking data platform architecture, and stand up infrastructure in AWS to support integration, reporting, analytics, and data science use cases	Created automated infrastructure provisioning scripts to standardize infrastructure components across the data and analytics stack

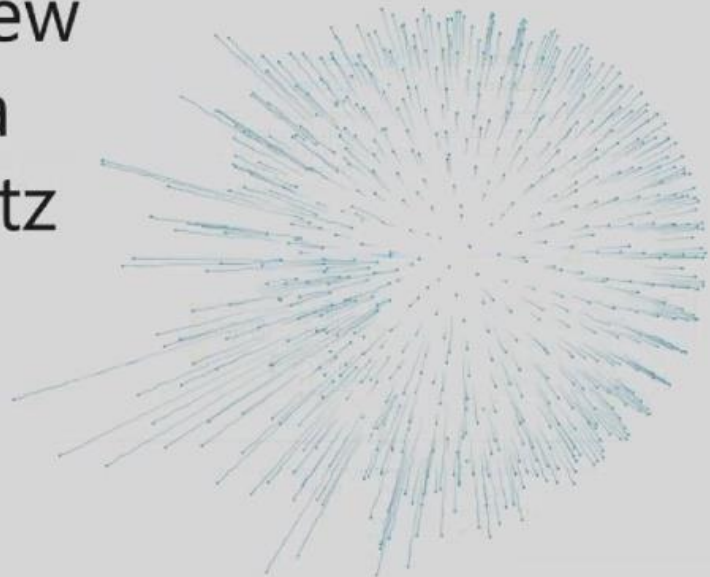**Data platform journey so far**

1	2	3	4
• Conducted workshops with 30+ functional groups/sub-groups to finalize Day1/Day2 reporting and analytics requirements • Rationalized ~1,800+ reports down to ~280 prioritized Day1 reports/dashboards	• Identified 30+ data sources outside of new systems to identify integration gaps based on Day 1/Day 2 critical report requirements and associated data needs; catalogued total of 200+ integrations coming in and out of data platform from different systems	• Finalized data platform architecture blueprint and technical bill of materials • Successfully established the data platform architecture on AWS to support in-time integration with Hertz digital transformation program	• Built all environments (Dev, QA, UAT, and prod) using CFTs and Ansible scripts for infrastructure automation

---

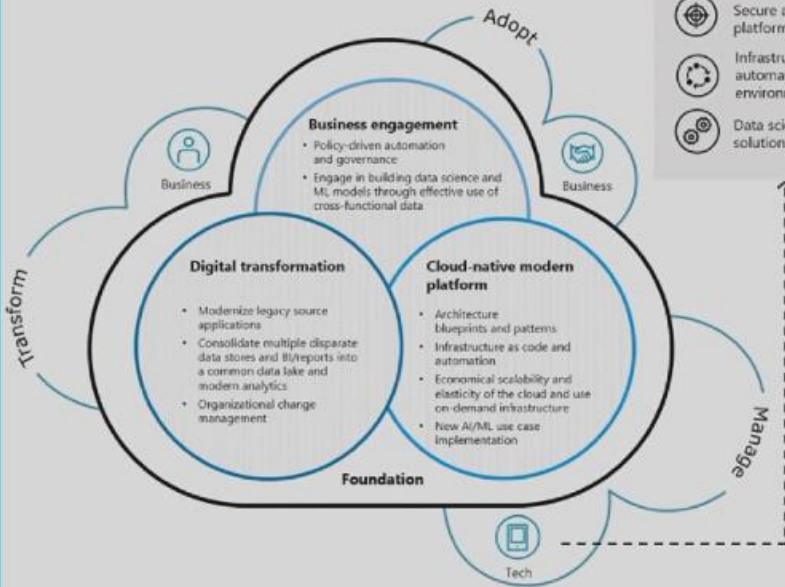# Solution overview and role of data platform at Hertz

Key purpose and underlying architecture

Solution overview

Cloud-native data platform

Enabling digital transformation journey and providing consolidated data across multiple systems for accelerated insights generation
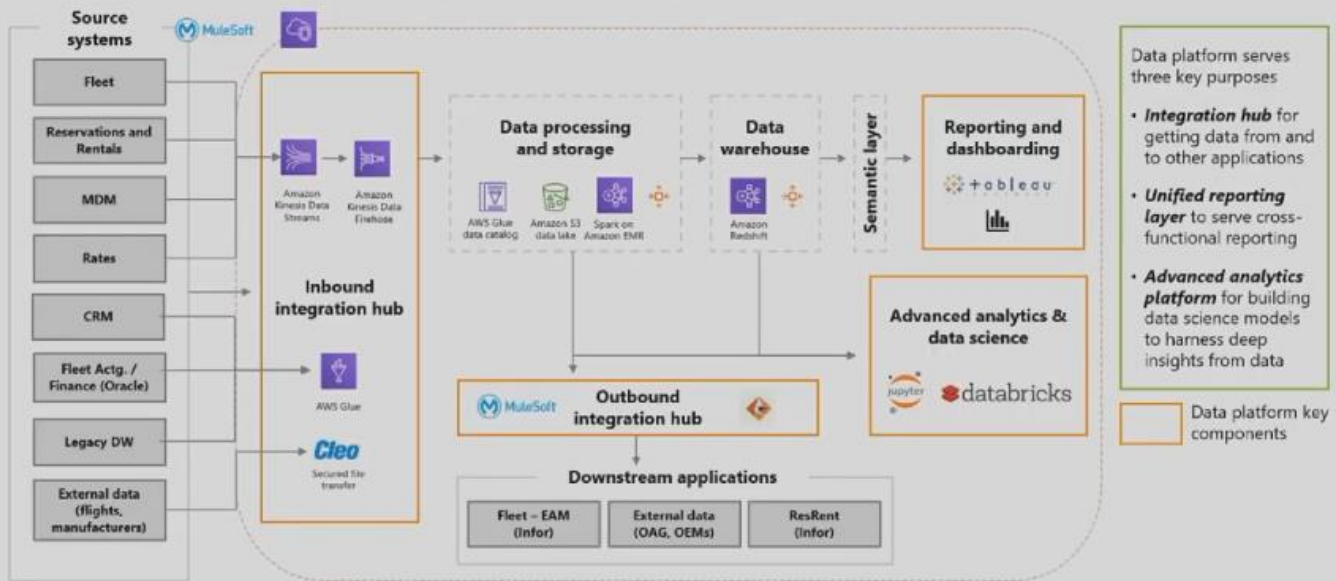

Cloud-native solution architectures

- Batch and streaming ingestion patterns
- Secure and scalable platform
- Infrastructure automation for all environments
- Data science/ML solutions

The role of the next-generation data platform at Hertz

High-level architecture illustration of the data platform at Hertz and the three key purposes of the modern platform

Data platform serves three key purposes

- **Integration hub** for getting data from and to other applications

- **Unified reporting layer** to serve cross-functional reporting

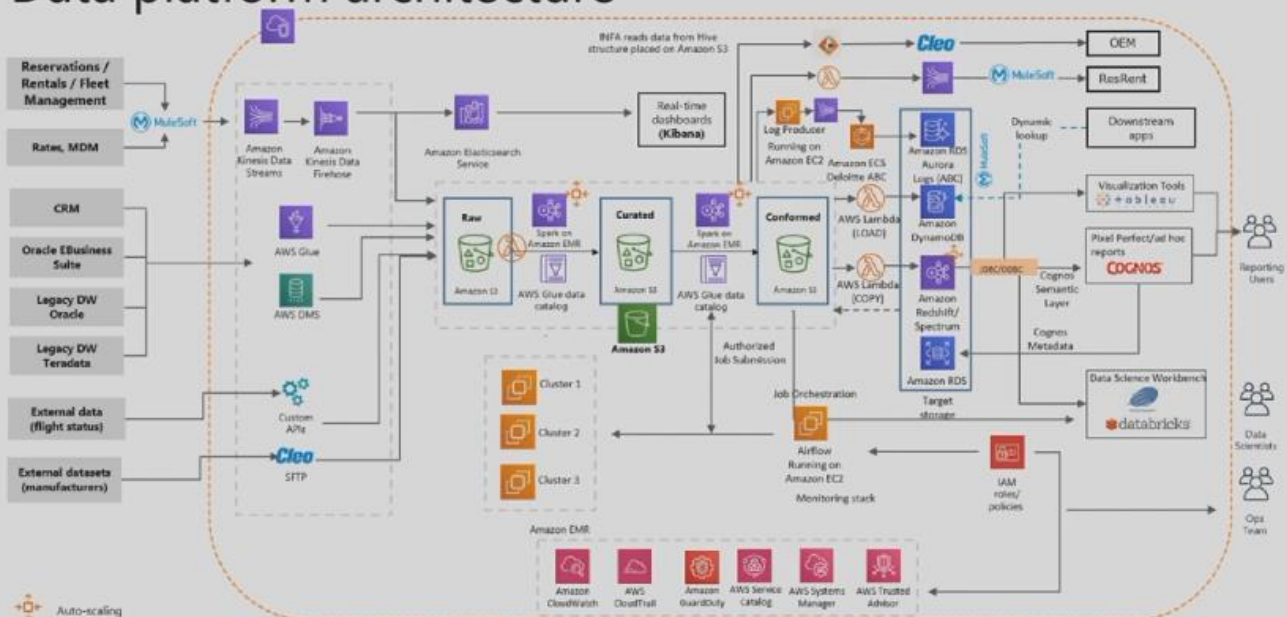- **Advanced analytics platform** for building data science models to harness deep insights from data

Cloud-native architecture

Built with all native AWS services to stand up a modern data platform


Data platform architecture

Repeatable patterns in architecture

The patterns below illustrate high-level reusable components within the data platform at Hertz that was deployed using native AWS services

Real-time stream ingestion in Amazon S3 with Amazon Kinesis Data Streams, Amazon Kinesis Data Firehose, and Amazon S3 for all event-driven ingestion in the data platform

Real time to batch use cases where data-generating system can only produce events and consuming systems can only take batches (orchestrated using Kinesis Data Streams, Kinesis Data Firehose, and AWS Glue or Informatica reading from Hive)

Real-time API call with external system (Flight Aware) to tie reservation data for optimizing fleet management

Low-latency fast lookups by transactional applications to look up in real time performed using lookup tables hosted in Amazon DynamoDB with defined read and write capacities
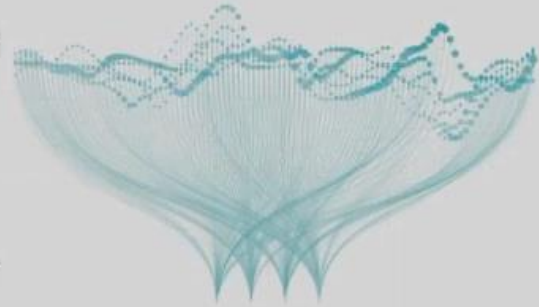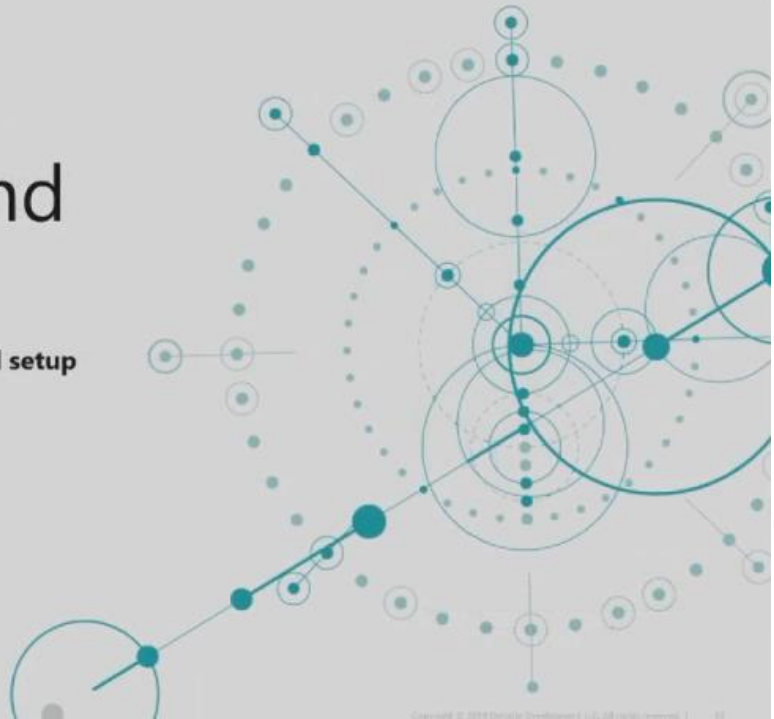
Impact

Patterns simplify the ingestion and consumption out of the data platform and avoid point-to-point integration with different components and designs, making it modular code

Infrastructure provisioning and automation

Under the hood infrastructure design and setup

Infrastructure blueprint
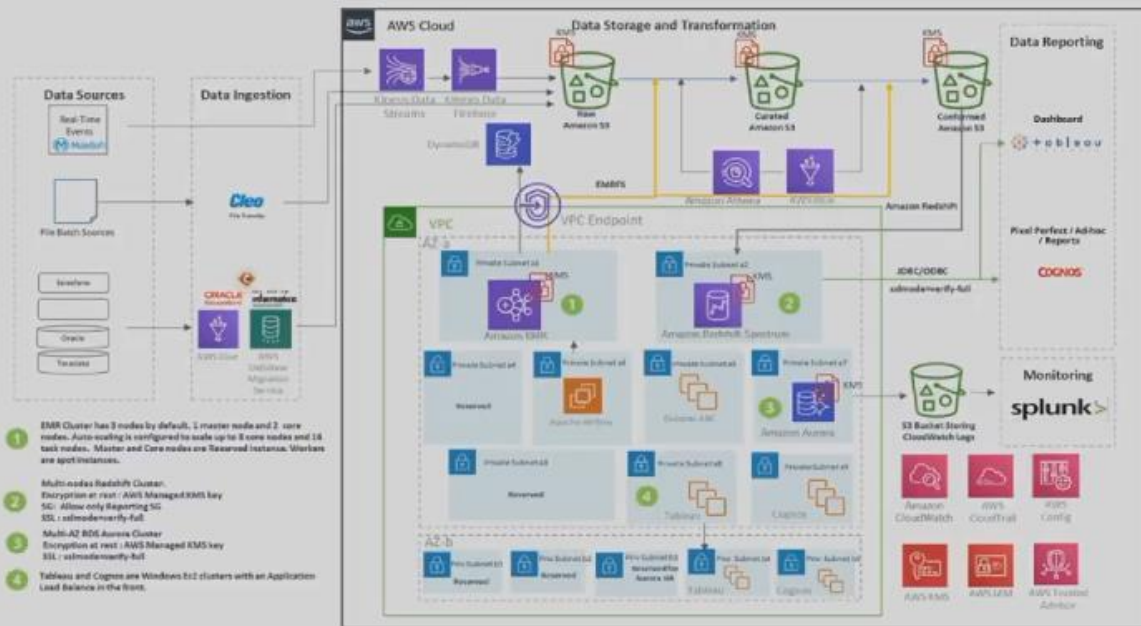
Amazon EMR stack diagram
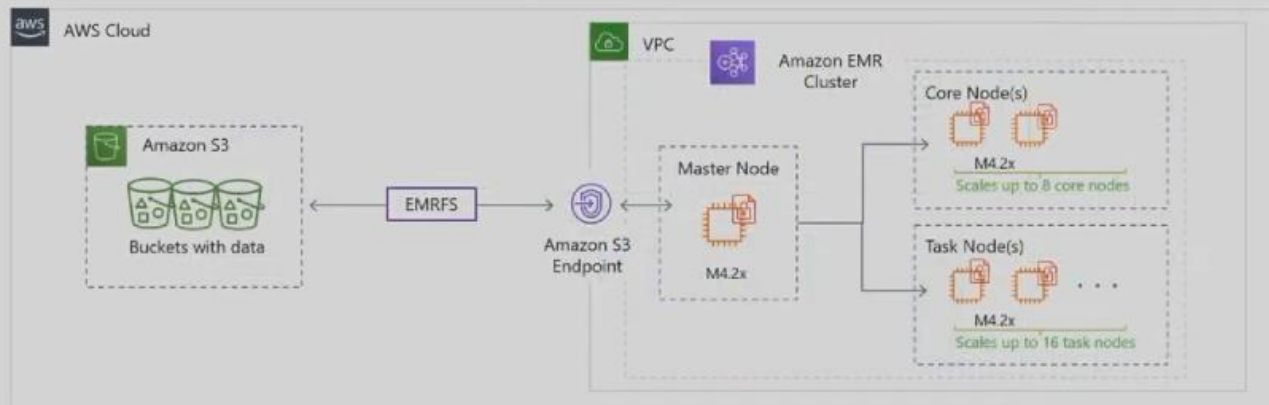
Auto-scaling is configured to scale up to 8 core nodes and 16 task nodes

- Scale-Out "Add 1 Instance if **ContainerPendingRatio** is greater than **0.7** for **1** five-minute period with a cooldown of **300** seconds"
- Scale-In "Terminate 1 Instance if **ContainerPendingRatio** is less than **0.2** for **3** five-minute periods with a cooldown of **300** seconds"
- Scale-Out "Add 1 Instance if **HDFSUtilization** is greater than **90** for **1** five-minute period with a cooldown of **300** seconds"
- Scale-In "Terminate 1 Instance if **HDFSUtilization** is less than **70** for **3** five-minute periods with a cooldown of **300** seconds"

The setup includes YARN Resource Manager, Spark History Server, and JupyterHub notebooks

Amazon Redshift stack diagram

AWS Cloud

Amazon S3 — Buckets with data

VPC

Amazon Redshift Cluster

Leader Node — dc2.large

Queries

Dashboard — tableau

Compute Node0 — dc2.large

Compute Node1 — dc2.large

Pixel Perfect/Ad hoc /Reports — COGNOS

Loading/unloading data
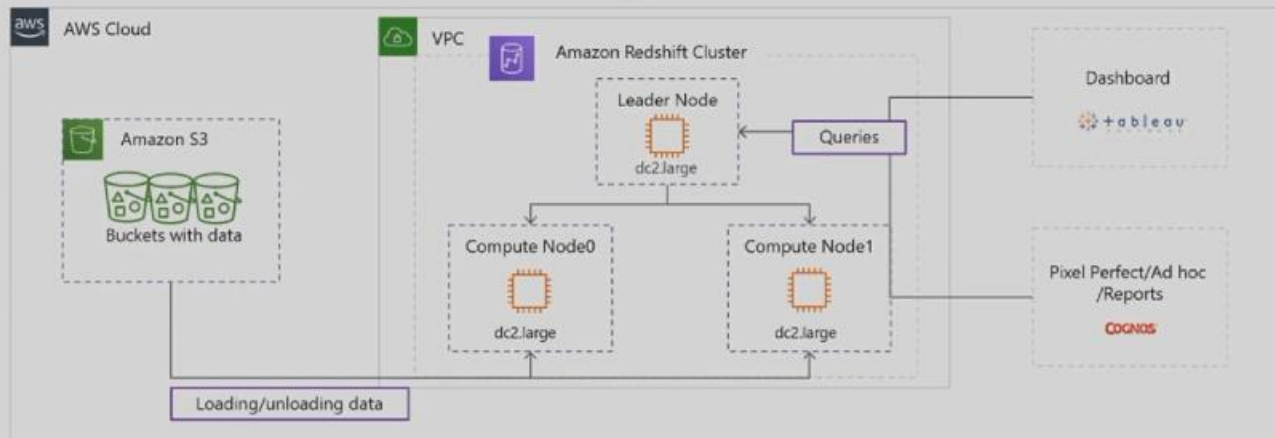
- spectrumNodeType : dc2.large
- spectrumClusterType: multi-node
- Publicly accessible: No

Amazon RDS Amazon Aurora stack diagram

AWS Cloud

VPC

Amazon EC2 — Deloitte ABC

Amazon RDS Amazon Aurora Cluster

Primary Aurora PostgreSQL Instance — db.r4.xlarge

Read/Writes

Data Copies

Data Copies

Data Copies
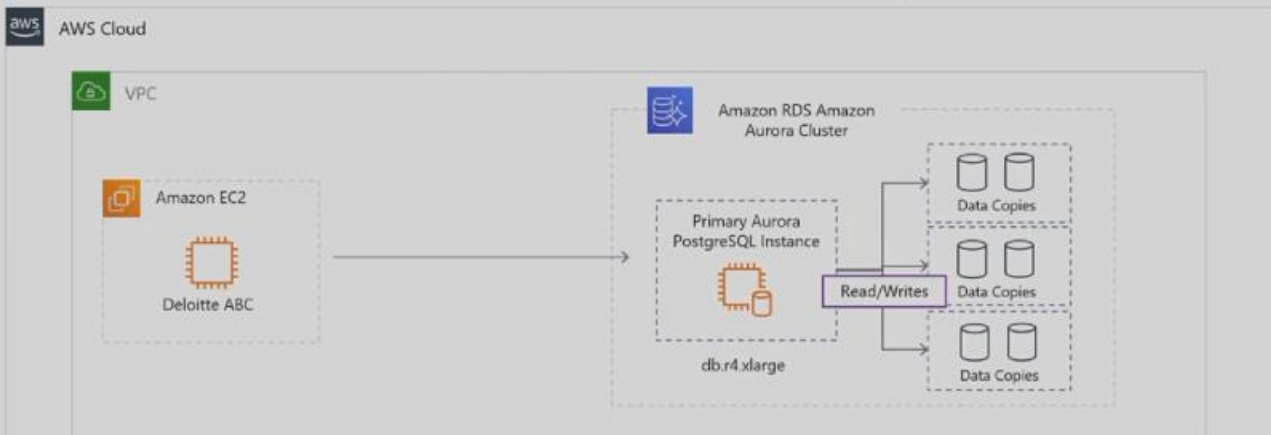
Deloitte's Audit-Balance-Control (ABC) Framework uses Amazon Aurora Postgres to write errors and logs entries for every data movement with the platform in an asynchronous manner

DBInstanceClass: db.r4.xlarge

DbEngineVersion: 9.6.9

MultiAZ: True

RDSDBParameterGroup: Customer CFT

Amazon Kinesis API stack diagram

AWS Cloud

VPC

Private Subnet — Requestor Lambda function

Private Subnet — Processor Lambda function

Primary Amazon Aurora PostgreSQL Instance

GET/POST

Users → Amazon API Gateway → Requestor Lambda function → Processor Lambda function → Amazon Kinesis Table

Kinesis data stream → Kinesis Data Firehose stream → Raw Amazon S3

Every item pushed to S3 is prefixed with format: !(timestamp:yyyyMMdd)/

POST or GET calls to API Gateway invoke Requestor Lambda
Requestor Lambda invokes the Processing Lambda.
This Lambda creates the appropriate Kinesis data stream, delivery stream
The information of the streams being created is updated in the PostgreSQL Amazon RDS DB

kinesis_stream_name	kinesis_firehose_stream_name	s3_bucket_name	status	timestamp
logs_dataplatform_abc	logs_dataplatform_abc			
algo_dataplatform_xyz	algo_dataplatform_xyz			

AWS Glue stack diagram

AWS Cloud

AWS Glue Data Catalog

SQL Queries — Amazon Athena

Hive / Spark metastore — Amazon EMR Cluster

M4.2x Master Node
M4.2x Core Node
M4.2x Core Node

Crawlers → Raw Amazon S3
Crawlers → Curated Amazon S3
Crawlers → Conformed Amazon S3

S3 bucket naming structure:
environment-platform-appID-type-clientname-ID-region
S3 encryption: AES-256 (SSE-S3)
Amazon EMR cluster encryption: KMS key
AWS Glue database naming structure: environment_platform_appID_SeqNum
Amazon Athena will use the AWS Glue catalog for querying

Infrastructure automation

Demo
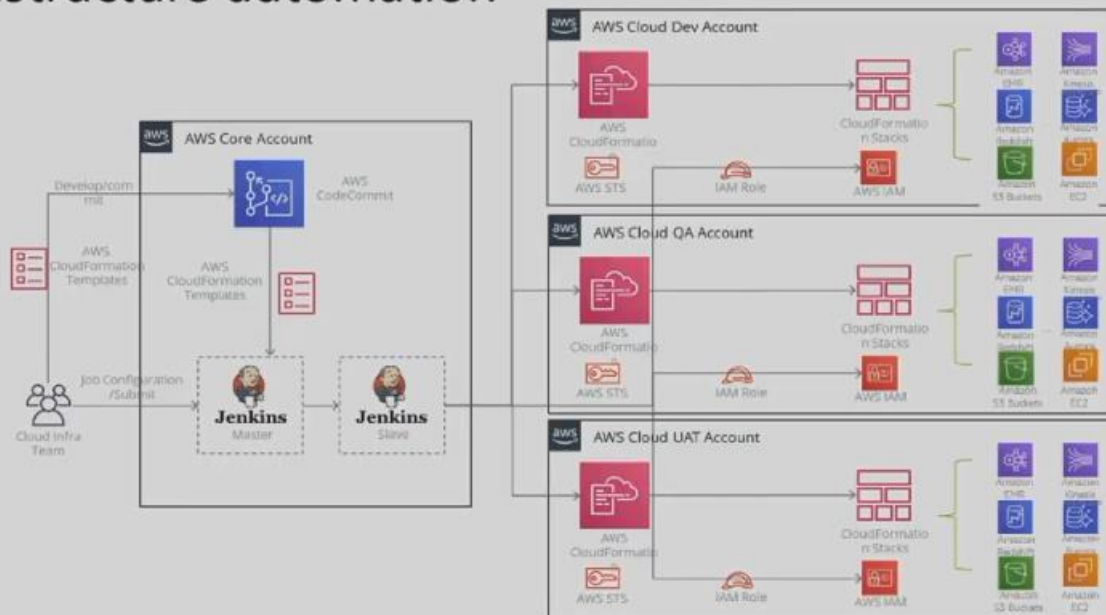

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS CloudFormation Template for S3 bucket creation.


Parameters:
  S3BucketName1:
    Type: String
    Description: Provide a unique bucket name. Bucket names must follow naming standard.
    AllowedPattern: ".*"
    MinLength: 6
    MaxLength: 63


  S3BucketName2:
    Type: String
    Description: Provide a unique bucket name. Bucket names must follow naming standard.
    AllowedPattern: ".*"
    MinLength: 6
    MaxLength: 63


  S3BucketName3:
    Type: String
    Description: Provide a unique bucket name. Bucket names must follow naming standard.
    AllowedPattern: ".*"
    MinLength: 6
    MaxLength: 63
```
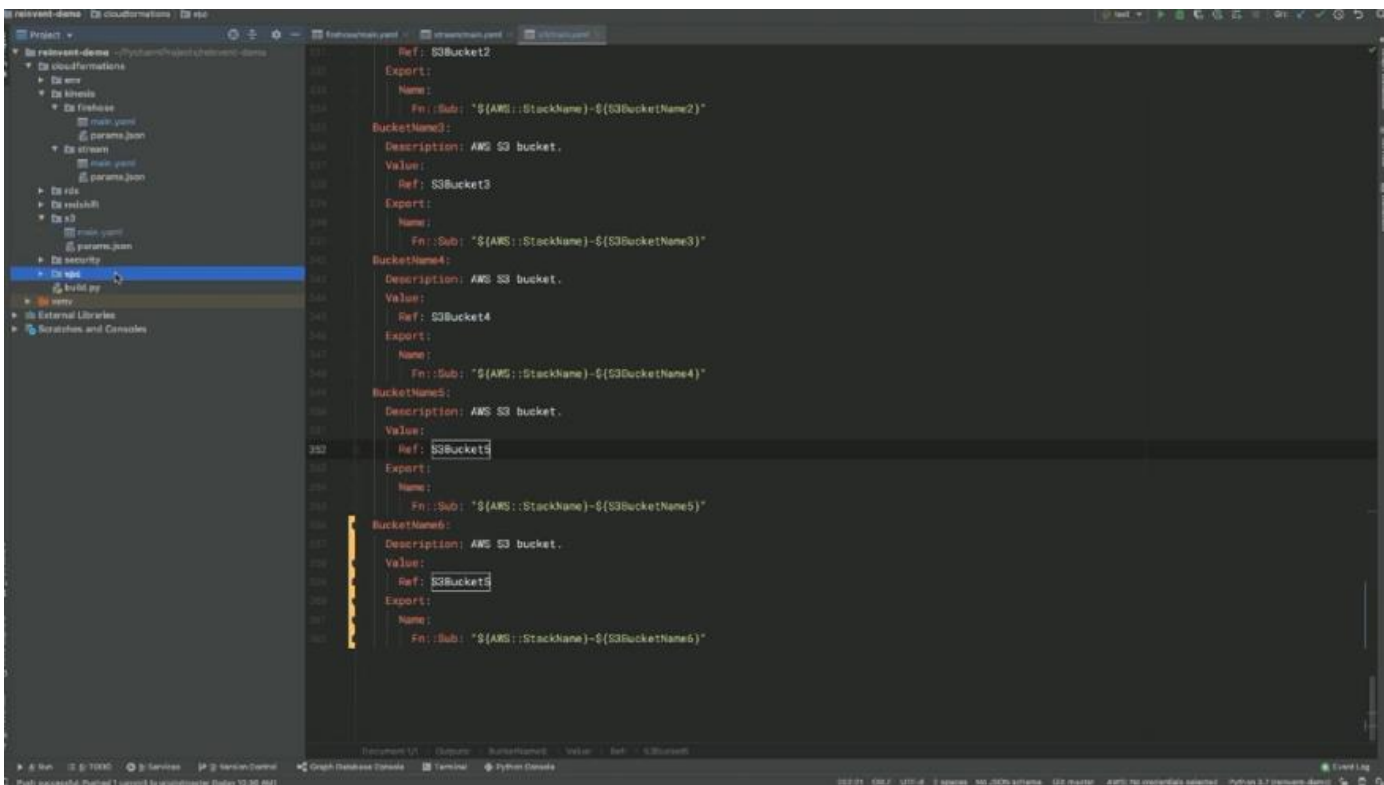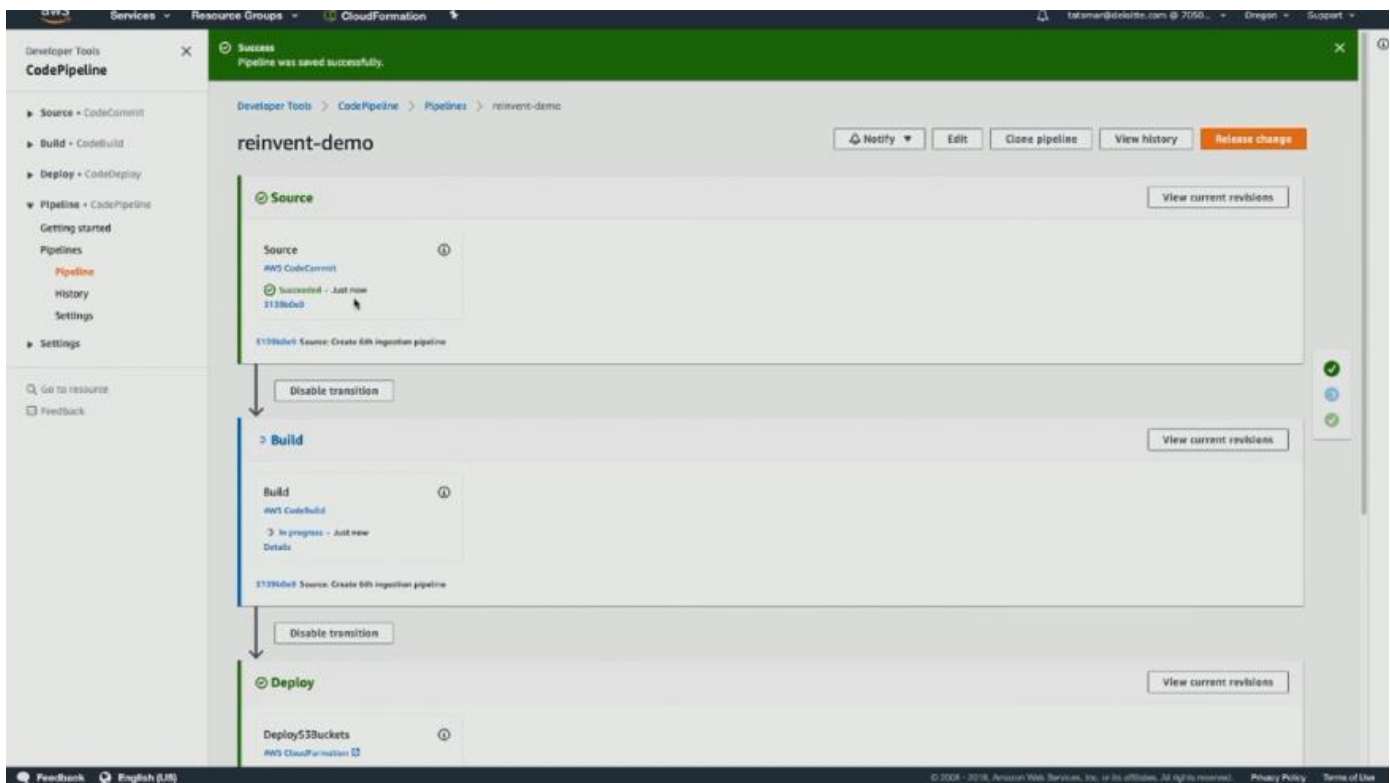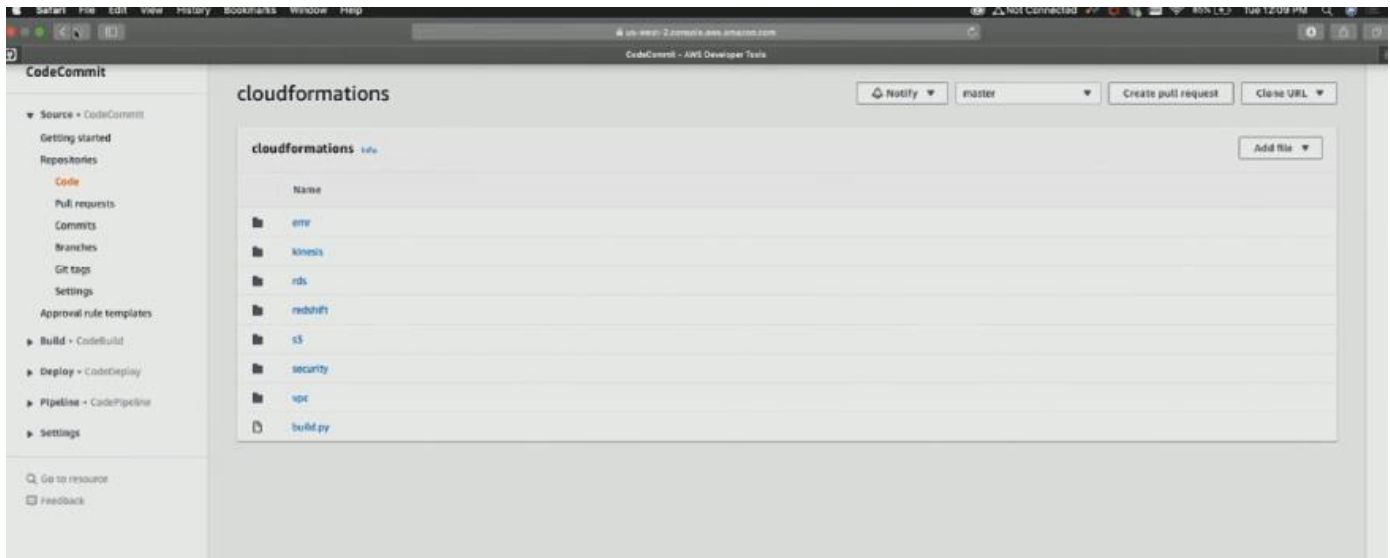
Document 1/1 > Outputs: > BucketName5: > Value: > Ref: > S3Bucket5

```yaml
          Value: !Ref Lifecycle

  S3Bucket6:
    Type: AWS::S3::Bucket
    Properties:
      BucketName:
        Ref: S3BucketName6
      VersioningConfiguration:
        Fn::If:
        - HasVersioning
        - Status: Enabled
        - Ref: AWS::NoValue
      BucketEncryption:
        ServerSideEncryptionConfiguration:
        - ServerSideEncryptionByDefault:
            SSEAlgorithm: AES256
      PublicAccessBlockConfiguration:
        BlockPublicAcls: true
        BlockPublicPolicy: true
        IgnorePublicAcls: true
        RestrictPublicBuckets: true
      LifecycleConfiguration:
        Rules:
        - Id: GlacierRule
          Prefix: !Ref LifeCyclePrefix
```



```yaml
          Ref: S3Bucket2
        Export:
          Name:
            Fn::Sub: "${AWS::StackName}-${S3BucketName2}"
    BucketName3:
      Description: AWS S3 bucket.
      Value:
        Ref: S3Bucket3
      Export:
        Name:
          Fn::Sub: "${AWS::StackName}-${S3BucketName3}"
    BucketName4:
      Description: AWS S3 bucket.
      Value:
        Ref: S3Bucket4
      Export:
        Name:
          Fn::Sub: "${AWS::StackName}-${S3BucketName4}"
    BucketName5:
      Description: AWS S3 bucket.
      Value:
        Ref: S3Bucket5
      Export:
        Name:
          Fn::Sub: "${AWS::StackName}-${S3BucketName5}"
    BucketName6:
      Description: AWS S3 bucket.
      Value:
        Ref: S3Bucket5
      Export:
        Name:
          Fn::Sub: "${AWS::StackName}-${S3BucketName6}"
```

CodeCommit

- ▼ Source • CodeCommit
  - Getting started
  - Repositories
    - Code
    - Pull requests
    - Commits
    - Branches
    - Git tags
    - Settings
  - Approval rule templates
- ▶ Build • CodeBuild
- ▶ Deploy • CodeDeploy
- ▶ Pipeline • CodePipeline
- ▶ Settings

🔍 Go to resource
☐ Feedback

cloudformations

△ Notify ▼ | master ▼ | Create pull request | Clone URL ▼

cloudformations Info

Add file ▼

	Name
📁	emr
📁	kinesis
📁	rds
📁	redshift
📁	s3
📁	security
📁	vpc
📄	build.py

---

aws   Services ▼   Resource Groups ▼   🔶 CloudFormation   ↘   △   tatsmar@deloitte.com @ 7050... ▼   Oregon ▼   Support ▼

Developer Tools
CodePipeline

- ▶ Source • CodeCommit
- ▶ Build • CodeBuild
- ▶ Deploy • CodeDeploy
- ▼ Pipeline • CodePipeline
  - Getting started
  - Pipelines
    - Pipeline
    - History
    - Settings
- ▶ Settings

🔍 Go to resource
☐ Feedback

⊘ **Success**
Pipeline was saved successfully.   ✕

Developer Tools > CodePipeline > Pipelines > reinvent-demo

reinvent-demo

△ Notify ▼ | Edit | Clone pipeline | View history | **Release change**

⊘ Source

View current revisions

Source   ⓘ
AWS CodeCommit
⊘ Succeeded - Just now
3139b0x0

3139b0x0 Source: Create 6th ingestion pipeline

[Disable transition]

⊗ Build

View current revisions

Build   ⓘ
AWS CodeBuild
⊗ In progress - Just now
Details

3139b0x0 Source: Create 6th ingestion pipeline

[Disable transition]

⊘ Deploy

View current revisions

DeployS3Buckets   ⓘ
AWS CloudFormation ☑

Key learnings

Leverage AWS native services to enable a more seamless deployment (AWS pipeline and deploy)

Amazon RDS and Amazon Redshift don't enforce encryption in transit by default, and effort for encryption needs to be factored in

Streamline instance type to leverage reserved instance (RI) discount for better cost optimization

Don't hesitate to deep dive and ask questions or seek help; e.g., Customize AWS Secrets Manager password spec

Key takeaways

Benefits realized

Improvements realized

Stood up a modern data and analytics platform on AWS to create a consolidated data lake for value and insights generation from a large amount of data being produced from Hertz digital transformation and new systems

Enabled enhanced reporting and visualization of cross-functional data to support multiple business functions (sales, marketing, ops, customer care, etc.)

Created a platform for cross-application integration and cross-functional insights, and delivered on Hertz's analytical needs

Onboarded selected high-value data science use cases (e.g., fleet optimization, capacity substitution model, net price, etc.) on the data platform that needed scalable compute

Thank you!

aws