Alternative “Cloud-Native Data Platform Architecture on Amazon Web Services (AWS)”

Here's a comprehensive, step-by-step guide on setting up an **AWS Cloud-Only Data Platform Architecture** using **AWS-native services** along with **Snowflake** and **Databricks**. This guide is based on the architecture provided but has been expanded to cover additional configuration steps and best practices for building a modern, cloud-native data platform.

Step 1: Set Up Core AWS Infrastructure

1. **Create an AWS Account and Organize Resources with AWS Organizations**
    - Sign up for AWS and set up **AWS Organizations** if you manage multiple accounts.
    - Use **Service Control Policies (SCPs)** to enforce security policies across accounts.
2. **Set Up Virtual Private Cloud (VPC)**
    - Create a **VPC** with multiple subnets across different Availability Zones (AZs) for high availability.
    - Define public, private, and isolated subnets to separate internet-facing services, backend services, and sensitive data storage.
    - Configure **VPC Peering** if you need to connect multiple VPCs or with on-premises infrastructure.
3. **Configure VPC Security Groups and Network ACLs**
    - Set up **Security Groups** to control inbound and outbound traffic to resources within the VPC.
    - Use **Network ACLs** for additional traffic filtering at the subnet level.
4. **Implement PrivateLink and VPC Endpoints for Secure Service Access**
    - Enable **AWS PrivateLink** for secure connections to AWS services without internet exposure.
    - Set up **VPC Endpoints** for key services like S3, DynamoDB, and Snowflake, ensuring data transfer remains within AWS.
5. **Enable AWS Identity and Access Management (IAM) for Access Control**
    - Use **IAM Roles** and **Policies** to assign permissions and control access.
    - Enable **AWS Single Sign-On (SSO)** for centralized user management if required.

Step 2: Configure the OLTP Database Layer

1. **Set Up Amazon RDS or Aurora for OLTP Workloads**
    - Use **Amazon RDS** or **Amazon Aurora** to handle OLTP databases. Aurora offers high availability and scalability.
    - Choose database engines (MySQL, PostgreSQL, or Aurora Serverless) depending on your requirements.

- Enable **multi-AZ deployment** for high availability and configure **automated backups** and **point-in-time recovery**.

2. **Enable RDS Performance Insights and Monitoring**
    - Enable **RDS Performance Insights** to monitor query performance.
    - Set up CloudWatch **Alarms** for CPU, memory, and other performance metrics to optimize database health.
3. **Implement Change Data Capture (CDC) with Amazon Database Migration Service (DMS)**
    - Use **AWS DMS** to capture real-time database changes and stream them to Amazon Kinesis or S3 for downstream processing.
    - Configure DMS for **ongoing replication** to keep data updated in real-time.

Step 3: Implement NoSQL Database for Metadata Storage

1. **Set Up Amazon DynamoDB for Metadata Storage**
    - Use **DynamoDB** to store metadata and schema information, which supports fast reads and writes.
    - Enable **DynamoDB Streams** to capture changes to the data in real-time.
2. **Enable DynamoDB Global Tables for Multi-Region Availability**
    - If required, enable **Global Tables** to ensure metadata availability across multiple regions.
3. **Stream DynamoDB Data to Amazon Kinesis (Optional)**
    - Use **Kinesis Data Streams** or **AWS Lambda** to process and transform data from DynamoDB Streams in real-time.

Step 4: Set Up the Staging Layer for Data Ingestion

1. **Set Up Amazon Kinesis or Kafka for Real-Time Data Streaming**
    - Use **Amazon Kinesis Data Streams** or **Amazon Managed Streaming for Apache Kafka (MSK)** for high-throughput data ingestion.
    - Configure Kinesis Data Streams to handle real-time ingestion from various data sources.
2. **Create AWS Lambda Functions for Data Transformation**
    - Use **AWS Lambda** functions to process and transform streaming data from Kinesis in real-time.
    - Configure Lambda to output processed data to Amazon S3 or directly to your data warehouse.
3. **Set Up IAM Roles and Policies for Kinesis and Lambda**
    - Ensure **IAM roles** with the least privilege are configured for Kinesis and Lambda for secure access.

Step 5: Implement Data Lake Storage with Amazon S3 & Delta Lake for Lakehouse Capabilities

1. **Set Up S3 Buckets for Data Lake and Delta Lake Storage**
    - **Create S3 buckets** structured into zones:
        - **Raw Zone** for unprocessed data.
        - **Processed Zone** for data transformed with Delta Lake.
        - **Curated Zone** for data integrated with Snowflake and ready for analytics.
    - Use **S3 Object Versioning** and **Lifecycle Policies** for storage cost management and data history.
2. **Implement Delta Lake on Databricks for ACID-Compliant Data Lakehouse**
    - Use **Delta Lake** on Databricks for managing data with ACID transactions, ensuring data consistency, and enabling schema enforcement and data indexing.
    - Store Delta Lake tables in Amazon S3 to act as a central storage layer, providing **ACID compliance** directly on data in S3.
3. **Configure Delta Lake Metadata Management and Delta Log**
    - **Enable Delta Log** in Delta Lake to track and manage table history and transaction logs, supporting time travel and rollback capabilities for robust data recovery.
4. **Optimize Data Using Delta Lake Features**
    - Implement **Delta Lake Compaction** to merge smaller files and optimize performance.
    - Use **Z-Ordering** for faster query performance on specific data columns.
    - Set up **Data Versioning** and **Time Travel** features to access previous states of data for analysis or auditing.

Step 6: Set Up Snowflake for Centralized Data Warehousing

1. **Deploy Snowflake on AWS**
    - Set up **Snowflake on AWS** for centralized data warehousing, with data integration from S3.
    - Configure **virtual warehouses** for scalable compute, with auto-suspend and auto-resume to save costs.
    - Connect Snowflake with **Amazon S3** to load data directly using Snowflake's `COPY INTO` command.
2. **Optimize Snowflake Storage with Partitioning and Clustering**
    - Implement **clustering keys** and **materialized views** in Snowflake to enhance query performance.
    - Enable **auto-clustering** for frequently accessed tables to maintain optimal performance.
3. **Enable Snowflake Security Features**

- Configure **Role-Based Access Control (RBAC)** and use **multi-factor authentication (MFA)** for secure data access.
- Enable **Data Encryption** and **Time Travel** for data recovery and compliance.

Step 7: Set Up Databricks for Big Data Processing with Delta Lake

1. **Create Databricks Delta Lake Clusters**
    - Deploy clusters in **Databricks** with **Delta Lake integration** on top of Amazon S3.
    - Enable **Auto-Scaling** and **Cluster Policies** to manage resources based on workload demand.
    - Configure **structured streaming** in Databricks to ingest data continuously into Delta Lake tables.
2. **Set Up Data Streaming and Real-Time Processing**
    - Configure **Apache Spark Structured Streaming** in Databricks to stream data from Amazon Kinesis or S3, transforming data in real-time before storing it in Delta Lake tables on S3.
3. **Implement Databricks Delta Sharing for Data Access**
    - Use **Delta Sharing** for secure sharing of Delta Lake data between Databricks and Snowflake, allowing seamless integration with downstream analytics tools.

Step 8: Orchestrate Data Processing with AWS Glue for Delta Lake Integration

1. **Catalog Delta Lake Tables with AWS Glue Data Catalog**
    - **Integrate AWS Glue** Data Catalog to automatically catalog Delta Lake tables, making them accessible to both Snowflake and AWS services like Athena and Redshift Spectrum.
    - Use Glue Crawlers to periodically update Delta Lake table metadata, ensuring that schema changes are reflected in the Glue Data Catalog.
2. **Configure Glue ETL Jobs for Data Movement into Snowflake**
    - Set up **ETL jobs in AWS Glue** to transform and load data from Delta Lake tables into Snowflake, ensuring cleaned and curated data is available in Snowflake for reporting and analytics.

Step 9: Business Intelligence and Analytics with Amazon QuickSight

1. **Connect Amazon QuickSight to Snowflake and S3**
    - Use **Amazon QuickSight** to create dashboards and visualizations based on data in Snowflake and S3.
    - Set up **SPICE** (Super-fast, Parallel, In-memory Calculation Engine) in QuickSight to enhance performance on frequently accessed data.
2. **Configure QuickSight User Access and Sharing Options**

- Set up IAM-based access control in QuickSight to restrict report and dashboard access.
- Configure **email subscriptions** and **scheduled refreshes** for updated analytics.

Step 10: Set Up Machine Learning Pipelines with Amazon SageMaker & Databricks Delta Lake

1. **Integrate SageMaker with Delta Lake for Model Training**
    - Use **Amazon SageMaker** to train models on data stored in Delta Lake tables, leveraging Delta Lake's time travel capabilities to experiment with historical datasets.
2. **Deploy Real-Time Inference Using Delta Lake as Input**
    - Set up **real-time inference pipelines** in SageMaker that pull data directly from Delta Lake tables for continuous predictions.

Step 11: Monitoring, Security, and Optimization

1. **Set Up Monitoring with Amazon CloudWatch and AWS CloudTrail**
    - Use **CloudWatch** to monitor resource utilization, application performance, and set up custom dashboards.
    - Configure **CloudTrail** to log API activity for auditing and compliance.
2. **Enable AWS GuardDuty and AWS Config for Security and Compliance**
    - Use **GuardDuty** for threat detection and **AWS Config** for resource compliance monitoring.
    - Regularly review and update IAM policies, VPC security settings, and resource configurations.
3. **Optimize Costs and Performance with AWS Cost Explorer and Trusted Advisor**
    - Use **Cost Explorer** to analyze spending patterns and optimize resource usage.
    - Use **AWS Trusted Advisor** for best practices on cost optimization, fault tolerance, and security.

Final Note on the Architecture with Databricks Delta Lake Integration

This updated architecture uses **Delta Lake** as the core of the Lakehouse, providing a robust, ACID-compliant data storage layer on Amazon S3. The integration with **Snowflake** allows for seamless access to curated data for analytics and reporting, while **AWS Glue** orchestrates data processing across Delta Lake and Snowflake. This architecture supports real-time, scalable data processing and unifies data for machine learning, analytics, and BI tools. Regularly review AWS and Databricks best practices to optimize performance, security, and cost.