**Azure Real Time Project - Key Points to note**

| **Project Topic** | **Realtime Project(1 month)** | **Realtime Project (12 months)** |
| --- | --- | --- |
| Team Size | 12 | Ex: 10-20 size +  PM(1),  Scrum master (1),  Arc (1 to 2)  Leads (2 to 3)  Developers (8 to 10) |
| Domain | Telecom | FMCG, Telecom, Hospitality, Banking, Healthcare |
| Project Kickoff: | Project introduction by Naresh  Requirement gathering and clarification. | Initial meetings with stakeholders.  Requirement gathering and clarification. |
| Architecture Design | End to end project workflow, cost analysis, project timelines and POC done by clever studies team | End to end project workflow, cost analysis, project timelines and POC done by PM and enterprise architects |
| Sprints | 9 | 10 + |
| Each sprint duration | 1 day | 2 to 4 weeks |
| Azure Subscription | Free trial and ‘pay as you go’ | Pay as you go |
| environments | DEV, PROD | DEV, TEST,UAT,PROD |
| Resources used | ADF, ADLS GEN2, Databricks, Unity Catalog, RBAC, Logic App, Key Vault, Repo, Devops, CI/CD | ADF, ADLS GEN2, Databricks, Unity Catalog, RBAC, Logic App, Key Vault, Repo, Devops, CI/CD |
| Data Governance | Yes ( RBAC and Databricks Unity Catalog) | RBAC and Databricks Unity Catalog |
| Data Sources | GCP | MySQL, SFTP, Any other cloud, SQL Server, SAP, Salesforce, IOT, Hadoop, GCP etc |
| Type of Project | Batch data processing | Batch and Streaming data processing |
| Apprx.Number of Databases per env | 5 to 6 | 10 to 30 |
| Apprx.tot number of tables per env | 6 to 10 | 300 to 800 (It depends on project requirement) |
| Datasets format | csv | csv , parquet, json, xml,tsv etc. |
| ADF Pipelines | 8 | 100 + |
| Size of datasets | In MB’s | In TB’s , For example  History load : 50 TB (if applicable) -ad-hoc ADF PL to load history.(2021 to 2023 Nov)  Go-Live (After all tests passed-After Prod Deploy)  Batch loads: 5MB to 100MB for each ADF PL  Full loads : 24 ADF PL are full loads -  (10 min - 5 GB  40m min - 40 GB)  Delta Load/Incremental Load:80 ADF PL’s  2 min - 100 MB  6 min - 300 MB  10 min - 4 GB  20 min - 10 GB |
| Approx.databricks notebooks | 15 | 120 (it depends on project scope) |
| Default catalog used | Hive Metastore and Unity catalog | Hive Metastore and Unity catalog. |
| Cluster type used | Without Unity Catalog:  Interactive single node cluster  Policy: unrestricted  Access mode: no isolation shared  With Unity Catalog:  Interactive multi node cluster  Policy: unrestricted  Access mode: shared | **Job clusters** (scalable) - for scheduled jobs  **All purpose clusters**- for developments/for log activity ADF pipeline runs  **Instance Pools** - for processing large load of data |
| Cluster Configuration | Standard\_DS3\_v2 | Job clusters(Standard\_DS3\_v2) (scalable): will get created as & when pipeline is triggered from ADF.  All Purpose Clusters: Standard\_DS3\_v2  Instance Pools(Standard\_f16s\_v2): f8 or f16 for large history loads |
| SQL warehouse | Not Used | Medium to Large size for query data inside Azure Databricks workspace. SQL pro is preferred since it is on upper hand compared to SQL classic , in terms of functionality. For more refer: <https://azure.microsoft.com/en-us/updates/generally-available-azure-databricks-sql-pro/>  Classic - SQL Warehouse  Pro - SQL Warehouse |
| Logging mechanism | Implemented | Will be present |
| Email alerting | Implemented | Will be present |
| Error handling | Implemented | Will be present |
| Data load type | Full load, Delta load | Full load, Delta load,History Loads |
| Authentications for linked service | System Assigned Managed Identity | SAMI, User Assigned Managed Identity, service principal, or Access token is used as per the feasibility. Preferred is SAMI |
| Integration Runtime (IR) | Self hosted IR Dev & Prod env | -Self hosted IR, Managed Vnet IR, or Linked SHIR (rare case/testing)  -Separate SHIR for each env (Dev, Test, UAT & Prod) |
| Azure Active Directory (AAD) | Used | Will be present for maintaining RBAC on each of the Azure services. |
| Language used | SQL, Python | SQL, Python, Scala |
| ADF pipelines monitoring | Azure logic apps | Azure logic apps or Azure monitor or Azure alerts |
| Testing | Unit testing | Unit testing, Integration Testing, User Acceptance Testing |
| Bug Fixes and Refinement | Not used | Bug raised by testing team and resolved by data engineers |
| Deployment Plan(CI/CD process) | Azure DevOps is used for deployments of code/pipelines/notebooks from Dev to Prod | Azure DevOps or Jenkins is used for deployments of code/pipelines/notebooks from Dev to test , UAT & Prod |
| Performance Optimization | Data normalization, De normalization  Multiple pipelines performing similar tasks but with distinct sources or destinations, parameterization enables you to create a single pipeline template that accepts different parameters. This reduces the need for redundant pipeline definitions and optimises resource utilisation.  Design ADF pipeline to execute tasks concurrently (Parallel Processing).  Use the databricks notebooks (Spark) wherever it is required. | Design ADF pipeline to execute tasks concurrently (Parallel Processing).  Utilize compression techniques (like snapy etc) to reduce the size of data being transferred.  Multiple pipelines performing similar tasks but with distinct sources or destinations, parameterization enables you to create a single pipeline template that accepts different parameters. This reduces the need for redundant pipeline definitions and optimises resource utilisation.  Use the databricks notebooks (Spark) wherever it is required.  Data Profiling: Examine your data’s characteristics to determine where optimizations are needed.  Data normalization, De normalization  Implement the partitioning on valid columns.  When dealing with large datasets, opt for incremental loading. This means only new or changed data is transferred, reducing processing time and resources.  Allocate resources based on your pipeline’s needs.  Implement auto-scaling mechanisms to dynamically adjust resource allocation based on workload demands.  Vaccum, Optimize techniques |
| Project Challenges |  | Small files issue  Duplicate records issue  Data mismatch issues  Pipeline run timing issue  Spark memory issue  Standard tables vs Partitioned tables  Source table column datatype Vs Target data mismatch issue |
| Project Responsibilities |  | Creating metadata tables  Inserting data into metadata tables with required info for parameterization  Creating SSHIR  Creating Linked Services  Creating Datasets  Creating ADF PL  Creating Triggers, [Schedule]  Creating Email alerts by using logic app  Used Key vault for secrets  Implemented logging mechanism  Used databricks notebooks for data transformations and developing the business logics  Created tables in different layers such as bronze, silver, gold  Used ADLS GEN2 To store the data for the layers like landing,bronze, silver, gold  Implemented data governance by using AAD, RBAC, UC through databricks |