**Decomposition Challenge**

Based on the guidelines given in the challenge specifications, I decided to go for the microservice architecture residing on top of AWS. The AWS services that meet the requirements are:

1. Ability for Document Providers to submit up to 1000 documents per hour, across all data providers. Documents are heavy payload, averaging around 50 Gigabytes each- AWS S3 with S3 transfer acceleration
2. Ability for Extraction / Enrichment team to upgrade extraction / enrichment methods on a weekly basis. AWS Code Deploy
3. Ability for Document Consumers to search for provided documents using extracted / enriched metadata. There are around 100,000 Document Consumers distributed globally. who may undertake up to 5,000 concurrent searches-AWS Cloud search and AWS Textract
4. Ability to deliver Documents requested by Document Consumers. Document consumers request delivery of around 20,000 documents per day in aggregate-AWS CloudFront.

**Architecture Diagram**

**MS**

**MS**

**MS**

**Source Control**

**AWS Code Deploy**

**Parsing**

**AWS Cloudsearch/Textract**

**Uploader**

**S3 AWS transfer acceleration**

**System Storage**

**AWS S3**

**Note: ‘MS’ stands for Micro Service**

**Description of the system**

The system works by the customer company’s microservices utilizing the publicly available AWS APIs for the services shown in the diagram above. Special configuration for extra functionality is provided by API gateway for developers to use.

The description of the services as used in the architecture is as follows:

**AWS S3**: AWS S3 will serve as our document repository service . As an object store, S3 is designed for enriched documents and with transfer acceleration, can allow globally distributed users to upload documents to regions using an optimized network. In addition, S3 has very high levels of durability and availability

**AWS Code deploy**: Code deploy is a service that will enable the document enrichment team to make frequent upgrades to their code, executables and packages without having to wait for formal release schedules. With AWS S3 and Github compatibility, the number of new tools that the team has to upskill on is reduced.

**AWS Cloud search/AWS Textract::**AWS cloud search is a critical component of this architecture. It enables the document consumers to easily search indexed documents and is very easy to setup by the development team. In addition, AWS Textract is a tool that allows specific searches by the document consumers, thereby increasing search efficiency and granularity

**AWS CloudFront**: For fast document delivery to document consumers, AWS Cloudfront is our distribution solution. The fact that it utilizes edge locations that reduce latency to the consumers as well as in built security features like AWS shield that prevent denial of service attacks is a bonus. Lastly AWS cloudfront has 7 different routing policies to utilize gives a lot of flexibility when managing quality of service for document consumers.

**User Stories for The First Sprint**

1. As a developer, I want to create Microservices that will interact with the AWS services in my architecture that will be able to perform the 3 functions specified in the system use case
   1. Create an AWS account or use an existing enterprise one if possible
   2. Build the microservices using the principles of the 12factor app to ensure stability, statelessness, and concurrency, etc
   3. Build the architecture using the 5 pillars of the well architected framework:
      1. Operational excellence
      2. Security
      3. Reliability
      4. Performance efficiency
      5. Cost Optimization
   4. Use AWS API gateway to create RESTful APIS for the microservices to interact with AWS services
2. As a developer, I wan to ensure redundancy and failover for my apps to ensure quality of service
   1. Use AWS services like Roue53 to provide efficient routing of upload and download requests in the system
   2. Use AWS Direct connect as a backup to Route 53
3. As a developer, I want to provision suitable storage for my system with high levels of availability and durability
   1. Setup S3 service as a system object storage and also an origin for objects served by AWS Cloudfront
   2. Enable S3 transfer acceleration to ensure fast upload and download speeds by users