

We utilize various AWS resources such as S3, Cloud Watch, Lambda, and Sagemaker in this architecture.

S3: We will use S3 to store document

Cloud Watch: To keep watch on S3 bucket to see if new document is uploaded to S3

Sagemaker Endpoint: Total we will deploy 4 endpoints.

1. To Check if document is about politics
2. Detect Language of document
3. Search organization name in document headline
4. Perform Sentiment analysis on document

Lambda: Lambda function will have entire code to invoke all endpoints and store information about each document back to S3 along with its metadata(Metadata will have all date which we received from sagemaker endpoints ) so other stakeholders can access that information.

External datasource /Database will upload document to S3 in every minute which will trigger cloud watch to trigger lambda function.

Cloud watch will keep watch on s3 bucket, once data is uploaded to S3 then it will trigger lambda function. Lambda function will have entire login to invoke all sagemaker models and obtain predictions for each task.

The Lambda function will concurrently send requests to the "Check Politics," "Detect Language," and "Headline" models. However, we are only interested in obtaining the sentiment of English documents. Hence, the Sentiment Analysis model will only be invoked if the "Detect Language" model detects that the document is in English. Therefore, the sequence of execution should be "Detect Language" followed by "Sentiment Analysis" models.

Once the Lambda function receives all the responses, it will store the document along with its metadata back to the S3 bucket for use by other stakeholders. The metadata will include the document ID, a summary of the document, the language of the document, the sentiment of the document (applicable only to English documents), and whether the document pertains to Politics.

As here we will use Sagemaker endpoints, we don’t have to take care of scaling. Scaling will be handled by AWS itself.

In second approach, instead of Sagemaker endpoints we can use our own set of resources for inference.

1. We will write flask APIs which will serve inference request. We will use S3 for storing mode artifact.
2. Then we will wrap flask API code in docker image and store it ECR service.
3. Spin up Ec2 instance. Fetch docker image form ECR and model artifact from S3.
4. Infront of flask API we can add NGINX as well which will work as a proxy server and Load Balancer.
5. We can add our own scaling group to this EC2 instances.
6. For security stand point we will create VPC and private subnet to add all of this resources under private subnet and communication will happened through private IPs.