In this section, you are going to test the RAG pipeline created earlier. To demonstrate a very important feature of multiple files processing, we have introduced error code in a Lambda function. When you test the pipeline, one of the iterations will fail. You will fix the Lambda code and restart the pipeline from the point of failure rather than processing all the files all over again.

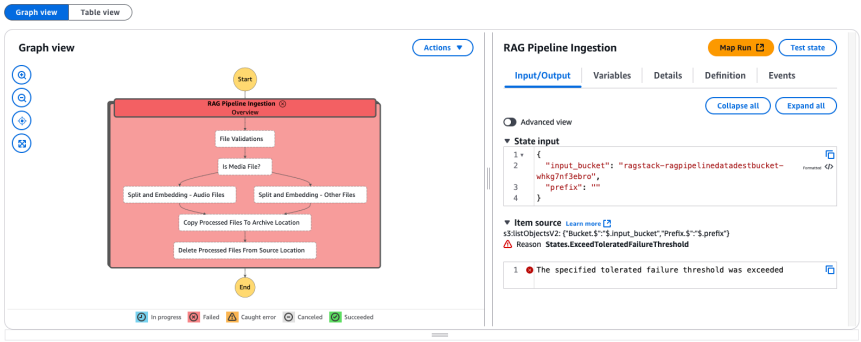
Running the workflow

1. Open the S3 console in a different tab and search for `ragpipelinedatadestbucket` and note down the name.
2. Return to Step Functions console. Select `Execute` button
3. Paste the following input in the dialog that appears. Make sure to substitute the {{INPUT_BUCKET}} with the S3 bucket name in step 1. Select **Start execution**.

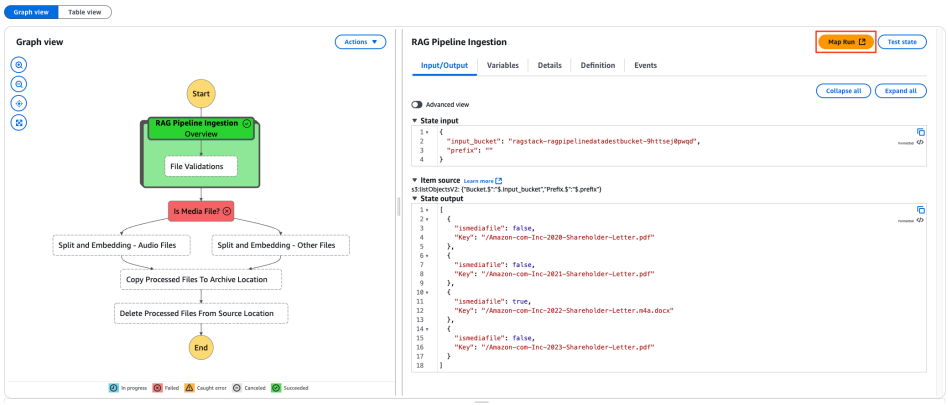
```
{
   "input_bucket":"{{INPUT_BUCKET}}",
   "prefix": ""
}
```

4. Select the map state in the graphical view.

Do not be alarmed by the deliberately introduced "States.ExceedToleratedFailureThreshold" error.

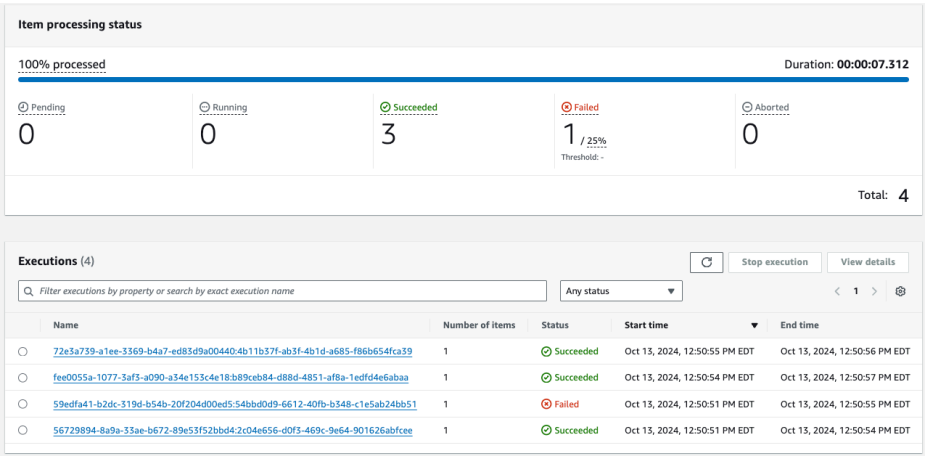


5. Select **Map Run** to view details of the Distributed Map execution.



5. Take sometime to review the details of the map run and the configurations.

6. Notice one of the iterations failing. You might see the results similar to below screenshot



Fixing the issue that caused the failure

The reason for failure is we deliberately introduced error in `rag-pipeline-ingestion-file-validation` lambda function. You are going to fix it and redrive the workflow.

> **Why did we introduce the error?**
> When processing large number of files, processing might fail due to data quality or exceptions in business logic. In such situations, restarting the processing of all the files can incur significant time loss and money. For example, suppose if processing failed at 1002 file when processing 1500 files, you do not want to reprocess all the 1001 files which were successfully processed. In such situations, you can use Step Functions redrive feature to restart from the point of failure.

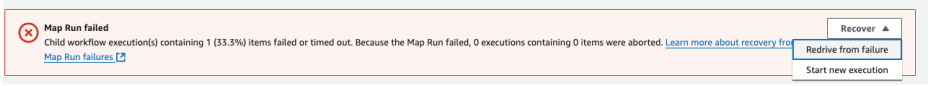
1. Access the Lambda console and edit `rag-pipeline-ingestion-file-validation` function.
2. Search for `#### This change is done to introduce error scenario` .
3. Replace the entire return statement (from **line#15** to **line#21**) that included the searched string with the following code.

```
        return {
            'ismediafile': True,
            'Key': '/'+filename
        }
```

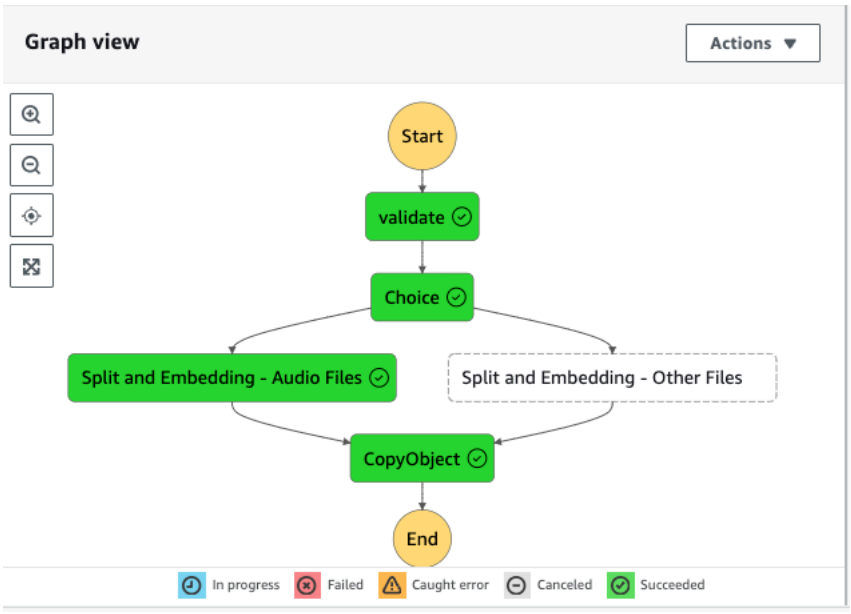
4. Make sure the indentations are correct.
5. Deploy the Lambda function

Restarting from the point of failure (Redrive)

1. Return to the Step Functions console.
2. Access the failed maprun
3. Restart the workflow by selecting the `recover` option from the failure banner. Select `Redrive from failure.`

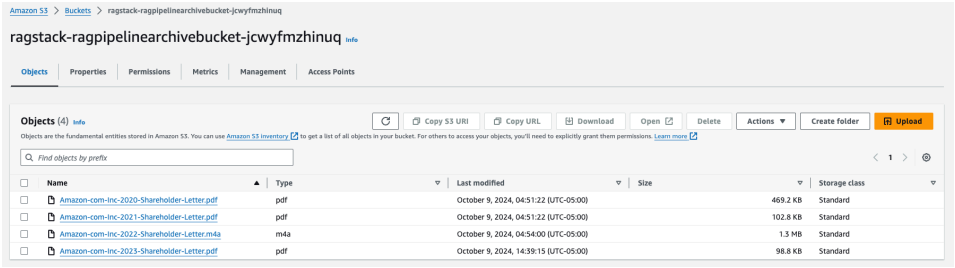


4. In the following popup, select the button `Redrive execution.`
5. Go to **Map Run**. Notice that just the one iteration that failed got restarted. This time the file is processed by the correct audio file processing step



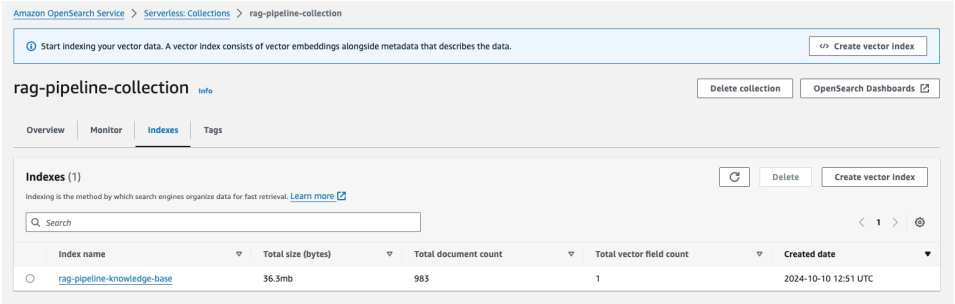
Verifying the processed files

1. Navigate to S3 console. Search for `ragpipelinearchivebucket` bucket.
2. Check the Archive folder. Once all the files are processed successfully those will be moved to Archive folder.



Verifying Vector store

1. Navigate to the Amazon OpenSearch Service Console
2. Select **Collections** from left menu
3. Select **rag-pipeline-collection**
4. Navigate to indexes tab and review the indexes **rag-pipeline-knowledge-base**. You will see the number of indexed documents.



Congratulations! You have successfully built and tested the RAG pipeline. The data is ready in vector format in the vector database. In the next section, you will test if the chatbot is able to use the vector data to fetch you the right results

Previous Next