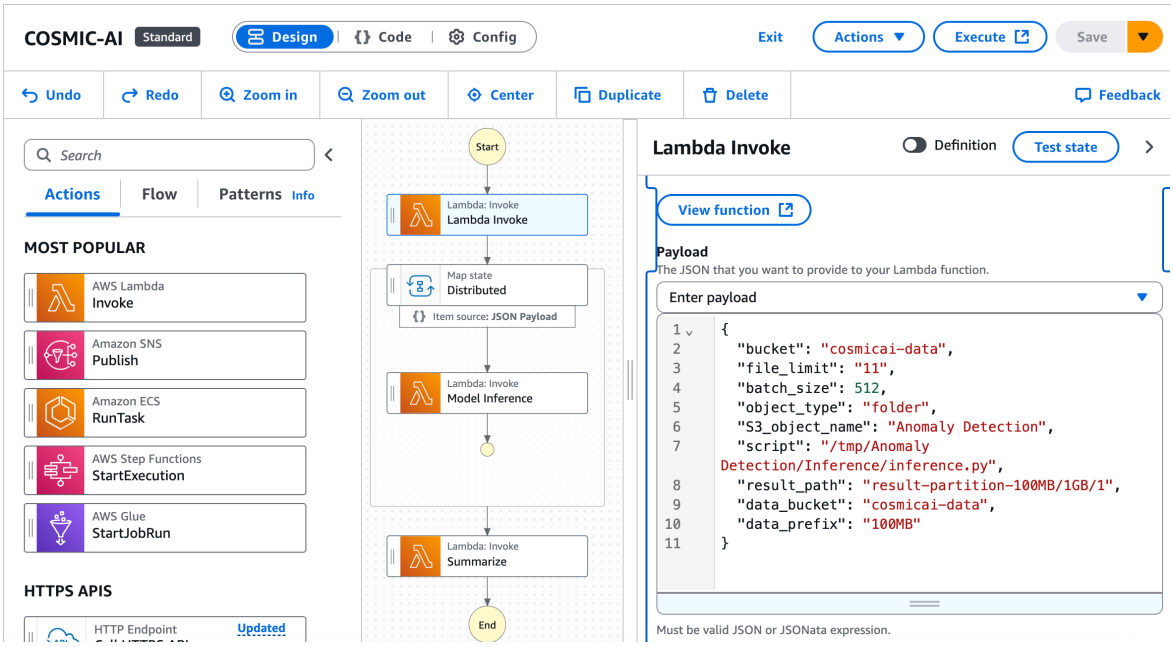
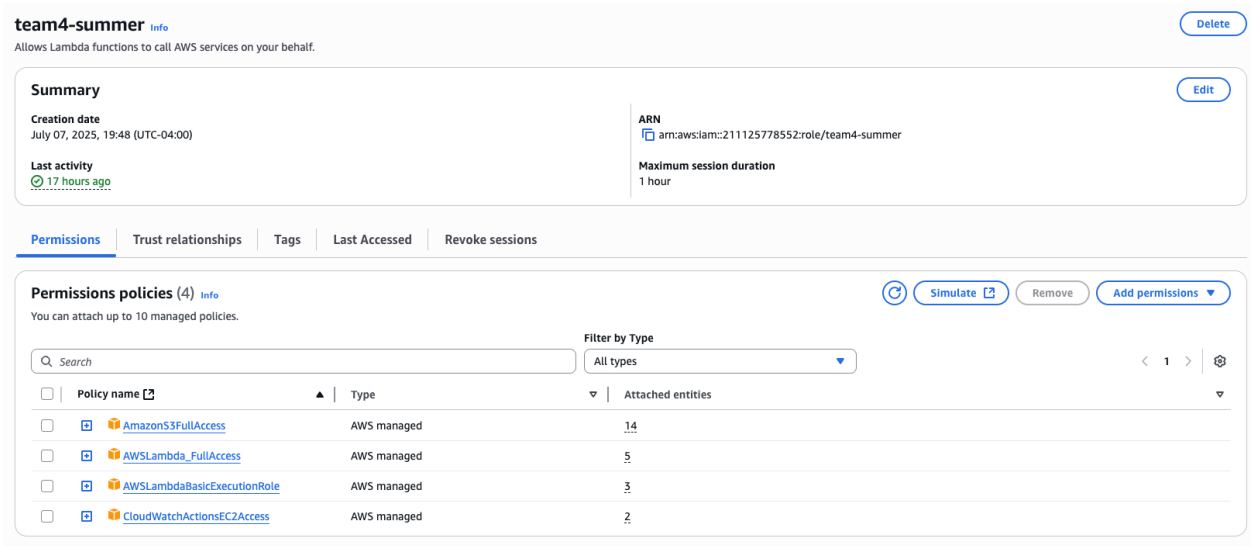
Project Step 1 Assignment: AWS Lambda Step Function

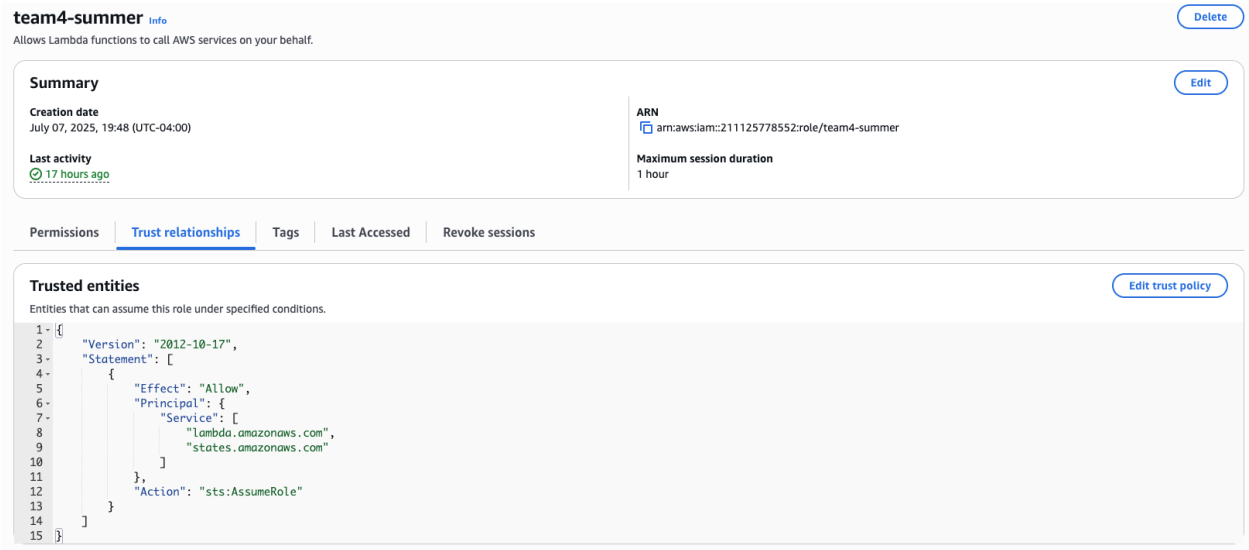
Sam Knisely, Darreion Bailey

1. Screenshots of your designed state machine in AWS Step Functions



2. Copy of IAM role configurations used for S3 access





3. Sample JSON payloads used for parameter passing

We tested 3 different JSON payloads with varying world sizes (11, 50, and 100).

{"bucket": "cosmicai-data",
"file_limit": "11",
"batch_size": 512,
"object_type": "folder",
"S3_object_name": "Anomaly Detection",
"script": "/tmp/Anomaly Detection/Inference/inference.py",
"result_path": "result-partition-100MB/1GB/1",
"data_bucket": "cosmicai-data",
"data_prefix": "100MB"}

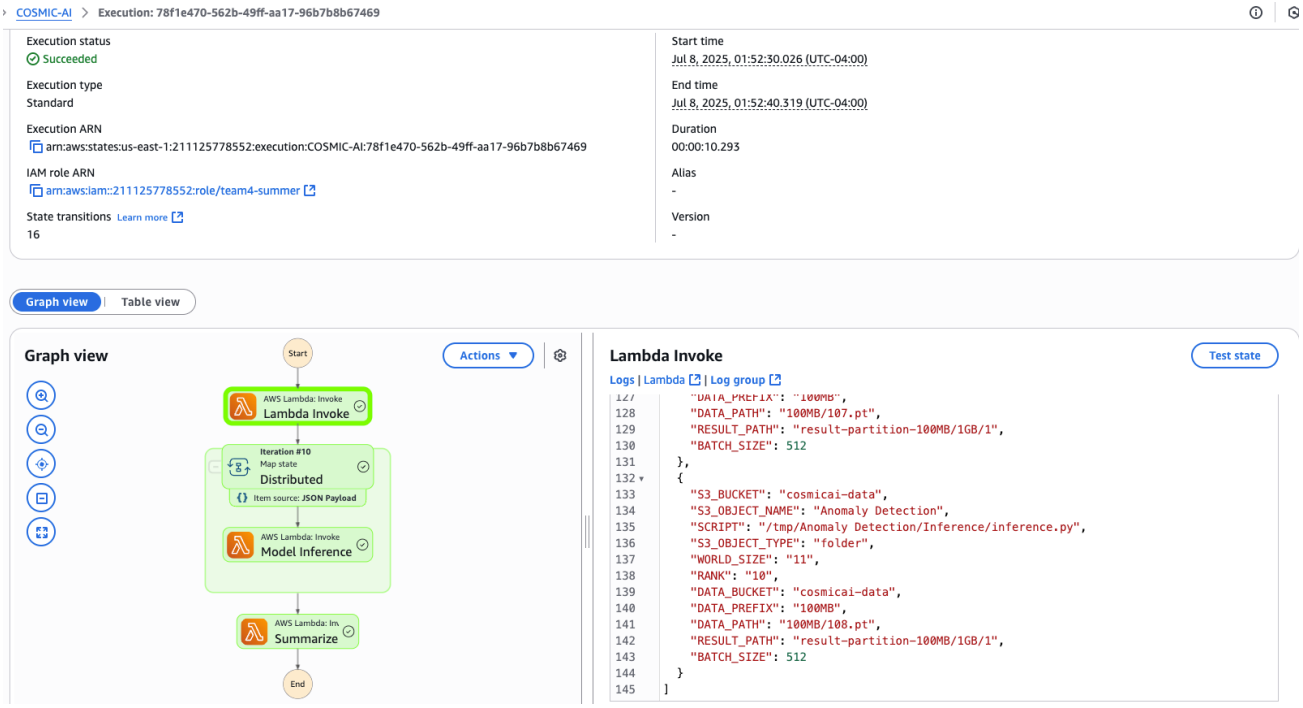
{"bucket": "cosmicai-data",
"file_limit": "50",
"batch_size": 512,
"object_type": "folder",
"S3_object_name": "Anomaly Detection",
"script": "/tmp/Anomaly Detection/Inference/inference.py",
"result_path": "result-partition-100MB/1GB/2",
"data_bucket": "cosmicai-data",

"data_prefix": "100MB"}

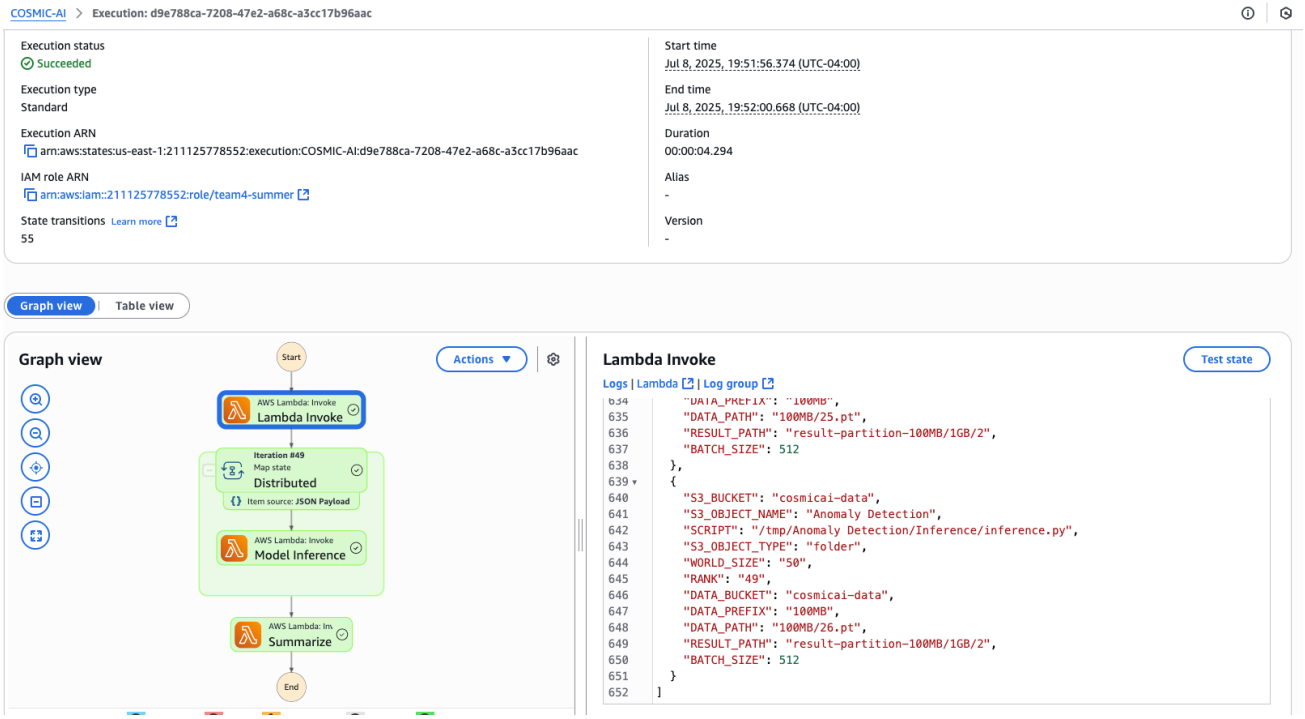
{"bucket": "cosmicai-data",
"file_limit": "100",
"batch_size": 512,
"object_type": "folder",
"S3_object_name": "Anomaly Detection",
"script": "/tmp/Anomaly Detection/Inference/inference.py",
"result_path": "result-partition-100MB/1GB/3",
"data_bucket": "cosmicai-data",
"data_prefix": "100MB"}

4. **Screenshots of execution results showing successful workflow completion**

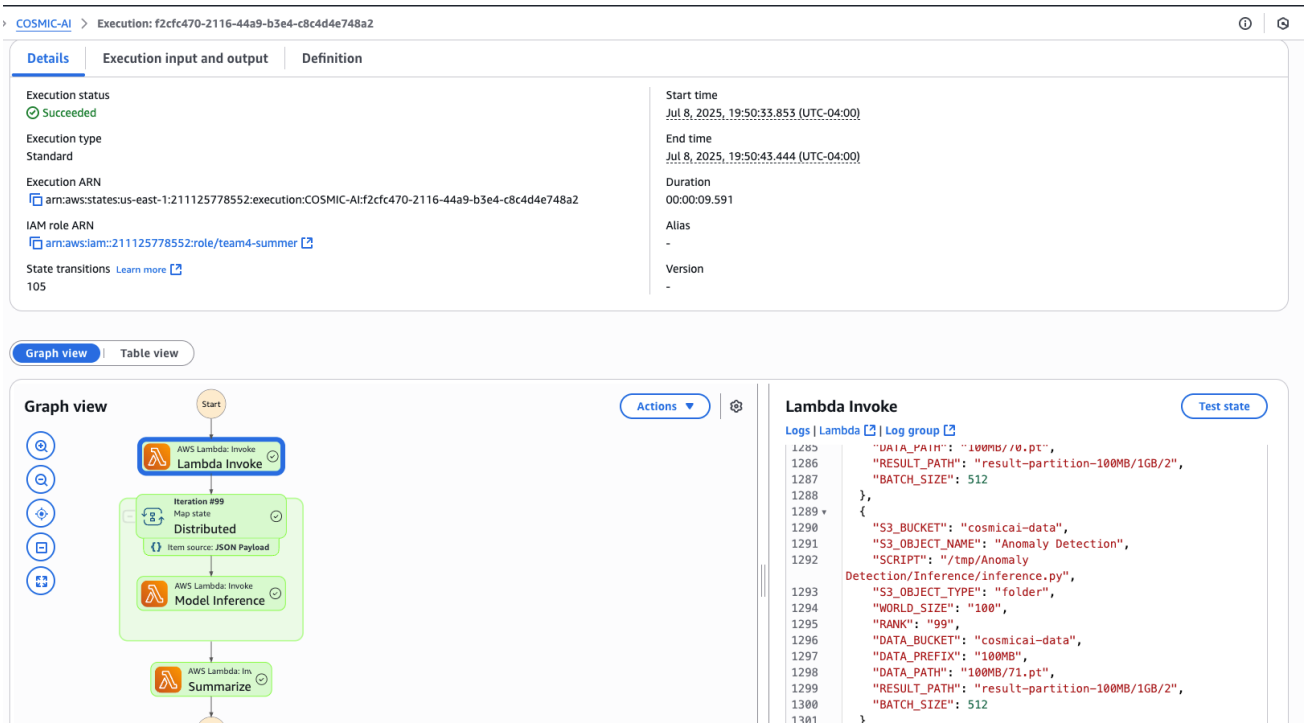
World size = 11



World size = 50



World size = 100



5. **Performance measurement table showing memory usage, duration, and cost metrics**

World Size	Partition	Requests	Duration (s)	Memory	Cost ($)
100	100MB	10018	2.64	1.4	0.62
50	100MB	5008	2.86	1.8	0.43
11	100MB	3327	1.87	1.9	0.2

6. **Brief explanation (1-2 paragraphs) of your implementation approach and any challenges encountered**

We implemented an AWS step function and tested the various JSON payloads mentioned above. This involved testing different world sizes (11, 50, and 100) of the 100 MB partition sized data. The state machine starts with an initialization using a lambda function, then a distributed map is created using a lambda container and a script is run to fetch the environment variables, JSON payload, and run inference and save JSON outputs for each rank. Lastly, a final lambda function is used to combine the results from the previous inference JSON and save a final output.

Our team created our own state machine before the class walkthrough on July 8th. We followed the steps on the GitHub and created our lambda functions using the python codes provided there. We also created our own IAM role with the necessary permissions. We called out state machine “COSMIC-AI”, our IAM role “team4-summer”, and our lambda functions “data-parallel-init2”, “inference”, and “summarize”. We utilized the comsmicai-data S3 bucket that was already created. We ran into some challenges with the IAM role creation because we gave it the necessary permissions but also had to go in and update the trusted relationships. Overall, it took us some time but we were able to figure out how to set up the state machine ourselves.