AWS Utilization Reporting Automation — Documentation

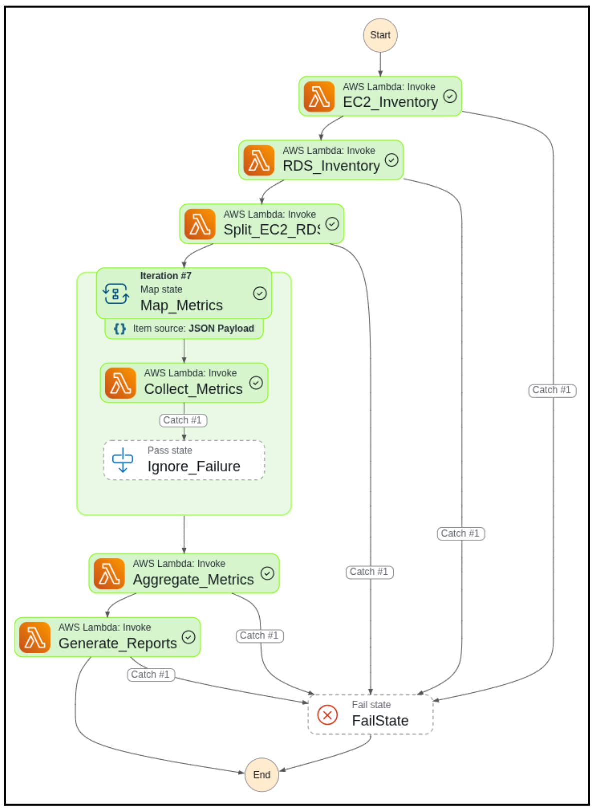
Overview

This automation collects **EC2 and RDS inventory**, fetches **CloudWatch utilization metrics**, and generates **Word & Excel reports** on a **daily, weekly, and monthly basis**.
The reports are stored in **Amazon S3**, and the workflow is orchestrated using **AWS Step Functions**.

Key Outputs

- EC2 & RDS inventory reports
- CPU, Memory, Disk, and RDS performance metrics
- Word (.docx) report with graphs
- Excel (.xlsx) report with averages
- Daily / Weekly / Monthly cost estimation

Architecture



Workflow Steps (Step Functions):

1. **EC2_Inventory** — Lambda: ec2-fetch-inventory
   - Fetch EC2 instance details from AWS
   - Output → $.EC2_Inventory
2. **RDS_Inventory** — Lambda: rds-fetch-inventory
   - Fetch RDS instance details from AWS
   - Output → $.RDS_Inventory
3. **Split_EC2_RDS** — Lambda: rds-ec2-spliter
   - Combine EC2 and RDS inventories into a single array
   - Prepares input for Map state
4. **Map_Metrics** — Map state invoking Lambda: parallel-metrics-functions
   - Parallel collection of metrics from CloudWatch
   - Handles CPU, Memory, Disk for EC2; CPU, FreeableMemory, DBConnections, ReadIOPS, WriteIOPS for RDS
   - Skips stopped instances
5. **Aggregate_Metrics** — Lambda: aggregate-metrics-lambda
   - Merges all collected metrics into a single object
   - Handles failed parallel executions gracefully
6. **Generate_Reports** — Lambda: genrate-word-excel-report
   - Generates Word and Excel reports
   - Uploads metric images and report files to S3
7. **FailState** — Step Functions Fail state in case of unrecoverable errors

Prerequisites & Setup

1. AWS Account Requirements

- AWS Account with Admin permissions for initial setup
- Enabled regions: Asia Pacific (Mumbai) - ap-south-1
- AWS CLI configured (optional but recommended)

2. Initial AWS Setup

1. Create S3 Bucket:
    - Name: aws-utilization-reports-prod1
    - Region: ap-south-1
    - Enable versioning (optional)
    - Configure lifecycle policies for old reports
2. Enable CloudWatch Agent on EC2 instances:
    - Install CloudWatch agent on monitored instances
    - Configure memory and disk metrics collection
3. Enable RDS Enhanced Monitoring:
    - Configure IAM role for RDS
    - Enable monitoring in RDS console

S3 Folder Structure

```
s3://aws-utilization-reports-prod1/
│
├── output/
│   └── YYYY-MM-DD/
│       ├── ec2_inventory.json
│       └── rds_inventory.json
│
├── images/
│   └── YYYY-MM-DD/
│       ├── i-xxxx_CPU.png
│       └── db-xxxx_CPU.png
│
└── reports/
    └── YYYY-MM-DD/
        ├── aws_report_YYYY-MM-DD.docx
        └── aws_metrics_YYYY-MM-DD.xlsx
```

Lambda Function Code Summaries -

- **ec2-fetch-inventory.py**

Fetches all EC2 instances from AWS region. Extracts instance ID, type, state, launch time, and tags. Filters running instances for reporting. Outputs structured JSON data to S3 bucket. Includes error handling for API failures. Requires EC2 read permissions. Generates `ec2_inventory.json` file.

- **rds-fetch-inventory.py**

Collects RDS instance details including DB identifier and engine. Retrieves status, storage, endpoint, and creation time. Filters available databases for metrics collection. Outputs JSON data to S3 storage location. Handles multi-AZ and read replica configurations. Requires RDS read-only permissions. Creates `rds_inventory.json` file.

- **rds-ec2-spliter.py**

Combines EC2 and RDS inventories into single array. Prepares input data for parallel Map state processing. Adds resource type identifiers (EC2/RDS). Filters out stopped instances to optimize execution. Structures data for metric collection functions. Requires S3 read permissions for inventory files. Outputs formatted array for parallel processing.

- **parallel-metrics-functions.py**

Collects CloudWatch metrics for each instance in parallel. Fetches CPU, memory, and disk metrics for EC2. Retrieves RDS performance metrics including connections. Generates metric graphs as PNG images. Uploads images to S3 for report inclusion. Skips stopped instances to save execution time. Handles metric collection failures gracefully.

- **aggregate-metrics-lambda.py**

Consolidates all parallel metric collections into single object. Calculates averages and summary statistics. Merges successful metric data from all instances. Handles failed executions without stopping workflow. Structures data for report generation. Requires memory for data aggregation. Outputs organized metrics object.

- **genrate-word-excel-report.py**

Creates Word document with executive summary and graphs. Generates Excel workbook with detailed metric data. Embeds S3 images into Word report automatically. Calculates daily/weekly/monthly cost estimations. Uploads final reports to S3 bucket. Uses python-docx and openpyxl libraries. Requires matplotlib for additional chart generation.

- **Step Function Code**

Defines workflow state machine for entire automation. Orchestrates Lambda function execution sequence. Implements Map state for parallel metric collection. Includes error handling and retry logic. Configures input/output data passing between states. Defines execution path and transitions. Uses JSON-based state machine definition language.

Practical Implementation Steps

Phase 1: Preparation (15 minutes)

Step 1: Create S3 Bucket

- Go to AWS Console → S3 → Create bucket
- Name: `aws-utilization-reports-prod1`
- Region: Asia Pacific (Mumbai) - `ap-south-1`
- Uncheck "Block all public access"
- Enable versioning (optional)
- Create folders: `output/`, `images/`, `reports/`

Step 2: Create IAM Roles

Create Lambda_Service_Role_Permissions

- Navigate to IAM Console → Roles → Create Role
- Select "AWS Lambda" as trusted entity
- Attach managed policies:

  AmazonEC2ReadOnlyAccess

  AmazonRDSReadOnlyAccess

  AmazonS3FullAccess

  AWSLambdaBasicExecutionRole

  CloudWatchReadOnlyAccess

- Add inline policy Include :

```
 "ec2:DescribeInstances",
"rds:DescribeDBInstances",
"cloudwatch:GetMetricWidgetImage",
"cloudwatch:GetMetricStatistics",
"cloudwatch:GetMetricData",
"cloudwatch:ListMetrics"
  "s3:PutObject",
"s3:GetObject",
"s3:ListBucket"
  "logs:CreateLogGroup",
"logs:CreateLogStream",
"logs:PutLogEvents"
```

Create Step Functions Role

- Create role with "Step Functions" as trusted entity
- Attach AWSStepFunctionsFullAccess policy
- Add Lambda invocation permissions

EC2 Instance Profile

- Create role for EC2 instances
- Attach CloudWatchAgentServerPolicy
- Assign to monitored EC2 instances

RDS IAM Role

- Create role for RDS Enhanced Monitoring
- Attach AmazonRDSEnhancedMonitoringRole
- Assign to RDS instances

Phase 2: Create Python Layer (10 minutes)

Step 3: Package Dependencies

```bash
# Create directory
mkdir lambda-layer && cd lambda-layer
mkdir python && cd python

# Install dependencies
pip install boto3 python-docx openpyxl matplotlib pandas pillow -t .

# Create zip
cd ..

zip -r aws-utilization-layer.zip python
```

Step 4: Upload Layer

- AWS Console → Lambda → Layers → Create layer
- Name: aws-utilization-report-layer
- Upload the zip file
- Compatible runtimes: Python 3.12
- Click "Create"

Phase 3: Create Lambda Functions (30 minutes)

Step 5: Create Each Lambda Function For each function (6 total):

- Lambda → Create function → Author from scratch
- Function name: [function-name]
- Runtime: Python 3.12
- Architecture: x86_64
- Permissions: Use existing role → lambda-utilization-report-role

- Advanced settings: Memory (128MB-1024MB), Timeout (3-10 min)
- Add layer: `aws-utilization-report-layer`
- Environment variables: `S3_BUCKET=aws-utilization-reports-prod1`, `REGION=ap-south-1`

Function Configurations:

- `ec2-fetch-inventory`: Memory 128MB, Timeout 3min
- `rds-fetch-inventory`: Memory 128MB, Timeout 3min
- `rds-ec2-spliter`: Memory 128MB, Timeout 3min
- `parallel-metrics-functions`: Memory 1024MB, Timeout 5min
- `aggregate-metrics-lambda`: Memory 512MB, Timeout 3min
- `generate-word-excel-report`: Memory 1024MB, Timeout 10min

Step 6: Upload Code For each function:

- Copy code from documentation
- Lambda → Function → Code tab
- Paste code in `lambda_function.py`
- Click "Deploy"

Phase 4: Create Step Functions (15 minutes)

Step 7: Create State Machine

- Step Functions → Create state machine → Write your workflow in code
- Paste Step Functions JSON code from documentation
- Replace `${ACCOUNT_ID}` with your actual AWS Account ID
- Permissions: Use existing role → `stepfunctions-utilization-role`
- Name: `AWS-Utilization-Reporting-Workflow`
- Click "Create state machine"

Phase 5: Testing

Step 8: Test Workflow

- Step Functions → Select state machine → Start execution
- Input: `{}` (empty JSON)
- Monitor execution progress
- Check each Lambda's CloudWatch logs for errors
- Verify S3 bucket has generated files

Step 9: Validate Outputs

- Check S3 bucket:
  - output/[date]/ → JSON inventory files
  - images/[date]/ → PNG graphs
  - reports/[date]/ → Word and Excel files
- Download and open reports to verify formatting

Costing Of This Automation :

https://calculator.aws/#/estimate?id=16c678ce9805501275e57349ff6b1dde1a608e04

AWS Service	Component / Function	Region	Monthly Cost (USD)	Notes
AWS Lambda	ec2-fetch-inventory-function	Asia Pacific (Mumbai)	0	Memory: 128 MB, Ephemeral storage: 512 MB, Requests: 30/month
AWS Lambda	rds-fetch-inventory-function	Asia Pacific (Mumbai)	0	Memory: 128 MB, Ephemeral storage: 512 MB, Requests: 30/month
AWS Lambda	rds-ec2-spliter-function	Asia Pacific (Mumbai)	0	Memory: 128 MB, Ephemeral storage: 512 MB, Requests: 30/month
AWS Lambda	parallel-metrics-functions	Asia Pacific (Mumbai)	0.06	Memory: 1024 MB, Ephemeral storage: 512 MB, Requests: 30/month
AWS Lambda	aggregate-metrics-lambda	Asia Pacific (Mumbai)	0.01	Memory: 512 MB, Ephemeral storage: 512 MB, Requests: 30/month
AWS Lambda	generate-word-excel-report	Asia Pacific (Mumbai)	0.06	Memory: 1024 MB, Ephemeral storage: 512 MB, Requests: 30/month
Amazon S3	Bucket_Cost (reports & images)	Asia Pacific (Mumbai)	0.05	2 GB storage, 500 PUT/POST/Copy requests, 200 GET requests
Amazon CloudWatch	CloudWatch_Only_Images	Asia Pacific (Mumbai)	0.04	1300 GetMetricWidgetImage requests, 1300 GetMetricData requests, 100 other API requests
AWS Step Functions	Step_function_code	Asia Pacific (Mumbai)	0	30 workflow requests/month, 6 state transitions/workflow

Total Monthly Cost: $0.22 USD
Total 12-Month Cost: $2.64 USD

Notes:

- All Lambda functions assume minimal memory usage (128 MB) and ephemeral storage (512 MB).

- Costs are based on **Asia Pacific (Mumbai) region**.
- Data transfer within the same region is free; external data transfer is **not included**.
- CloudWatch costs are for metrics and graph images only.
- The estimate excludes taxes, discounts, and any free-tier benefits already applied.
- The automation can run on **free-tier eligible accounts for testing**, and will incur the above costs when deployed in production accounts.

Conclusion

This automation provides a **low-cost**, **fully serverless**, and **scalable** solution for AWS infrastructure utilization reporting. It is suitable for both **production** and **free-tier testing** environments.