Cloud Cost Leakage Detection SaaS (AWS + Docker + Jenkins)

📌 Project Overview

This project implements a SaaS-style cloud cost optimization scanner that detects unused or ic
resources across customer accounts and emails monthly cost-saving reports automatically.

The system uses a central SaaS AWS account running Jenkins to execute a Dockerized scan
assumes cross-account roles into customer AWS accounts.

🧭 Problem Statement

Many companies unknowingly waste cloud spend on:

- Idle EC 2 instances
- Underutilized RDS databases
- Unused S 3 storage
- Over-provisioned Kubernetes resources

Cloud bills are complex, and small teams often lack visibility into waste.

🎯 Solution

A centralized SaaS scanner that:

1. Connects to customer AWS accounts via cross-account IAM role
2. Scans for cost leakage resources
3. Generates a summary report
4. Emails results to customers
5. Runs automatically via Jenkins schedule

🏗️ Architecture

```
Customer AWS Accounts
        ↑ (STS AssumeRole)
Docker Cost Scanner
        ↑
Jenkins (SaaS Control Plane)
```

```
         ↑
Scheduled Trigger (Monthly)
```

⚙️ Technology Stack

- AWS IAM (cross-account roles)
- AWS STS AssumeRole
- Python (boto 3)
- Docker
- Jenkins Pipeline
- Gmail SMTP

🔐 Cross-Account Access Model

SaaS Account

IAM user: `saas-scanner`

Permissions: - sts:AssumeRole → customer roles

Customer Account

Role: `CloudCostReadOnlyRole`

Trust policy allows SaaS account to assume role.

🐳 Docker Scanner

The scanner container performs:

- EC 2 idle detection
- RDS idle detection
- S 3 unused detection
- Kubernetes over-provision detection
- Email report sending

Environment variables passed at runtime:

- AWS_ACCESS_KEY_ID
- AWS_SECRET_ACCESS_KEY
- ROLE_ARN
- SENDER_EMAIL

- SENDER_PASS
- RECEIVER_EMAIL

🔄 Jenkins Pipeline

Stages:

1. Pull Docker image
2. Prepare environment file
3. Run scanner container

Pipeline also supports scheduled execution using cron.

⏱️ Scheduling

Monthly automatic scan via Jenkins cron trigger:

```
0 2 1 * *
```

Runs at 2:00 AM on the 1st of every month.

📧 Reporting

Scanner compiles counts of unused resources and emails report to customer.

Example output:

```
EC2 idle: 0
RDS idle: 1
S3 unused: 2
K8s over-provisioned: 0
```

🚀 Deployment Steps

1 . Build Docker Image

```
docker build -t <dockerhub>/cloud-cost-scanner:1.0 .
docker push <dockerhub>/cloud-cost-scanner:1.0
```

2 . Configure Jenkins Credentials

- aws-access-key
- aws-secret-key
- customer-role-arn
- sender-email
- sender-pass
- receiver-email

3 . Configure Pipeline

Add Docker pull and run stages.

4 . Setup IAM Cross-Account Role

Customer role trusts SaaS account. SaaS IAM user allowed to assume role.

5 . Enable Schedule

Add cron trigger to Jenkins pipeline.

📊 Example Execution Flow

```
Jenkins trigger → Docker run → AssumeRole → Scan → Email
```

🌍 Multi-Customer SaaS (Future)

Customer metadata list:

```
[
  {"name":"Acme","role_arn":"...","email":"ops@acme.com"},
  {"name":"Beta","role_arn":"...","email":"cloud@beta.com"}
]
```

Scanner loops through customers and sends individual reports.

✅ Key Features

- Cross-account AWS scanning
- Dockerized execution
- Jenkins automation
- Email reporting
- Monthly scheduling
- SaaS-ready architecture

🧪 Sample Result

```
Cloud Cost Scan Started
EC2 idle: 0
RDS idle: 1
S3 unused: 2
K8s over-provisioned: 0
Report emailed successfully
```

📈 Value

Helps organizations:

- Reduce cloud waste
- Improve cost visibility
- Automate reporting
- Centralize governance

🔮 Future Enhancements

- Multi-customer onboarding
- Web dashboard
- Savings trend analytics
- Cost estimation engine
- SaaS UI portal

👩‍💻 Author

Yamuna Cloud / DevOps Engineer Project

📄 License

Educational / Portfolio Use