# AWS Free Tier Auto-Shutdown Project (Zero Billing Goal)

## 📊 Project Goal

Automatically detect AWS usage costs and immediately stop or delete services to **ensure no bill is generated**, staying strictly within the AWS Free Tier.

## 🔧 Free Tier Services Used

| AWS Service | Usage Purpose | Free Tier Eligible? |
| --- | --- | --- |
| AWS Lambda | Execute cleanup logic | ✅ Yes (1M calls/month) |
| Amazon CloudWatch | Schedule and monitor | ✅ Yes (basic use) |
| AWS Budgets | Detect billing threshold | ✅ Yes |
| AWS IAM | Role and permissions | ✅ Yes |
| Amazon S3 | Logging (optional) | ✅ Yes (5GB) |
| EventBridge | Scheduled rule to run Lambda | ✅ Yes |

## ⚖️ Billing Strategy

* **AWS Budget Threshold**: $0.01 (lowest allowed)
* **Trigger Lambda cleanup** immediately when $0.01 is reached
* **No resource runs long enough to generate a bill**

## ✅ Step-by-Step Setup

### ✅ Step 1: IAM Role for Lambda

* Go to IAM > Roles > Create Role
* Use-case: **Lambda**
* Attach policies:
  + AmazonEC2FullAccess
  + AmazonS3FullAccess
  + AmazonDynamoDBFullAccess
  + AmazonAPIGatewayAdministrator
  + AWSLambda\_FullAccess
  + AWSBudgetsActionsWithAWSResourceControlAccess
  + IAMReadOnlyAccess *(to log IAM usage only)*

### ✅ Step 2: Create AWS Budget

* Go to **Billing > Budgets > Create Budget**
* **Type**: Cost Budget
* **Amount**: $0.01
* **Threshold**: 100% of actual cost
* **Action**: Trigger SNS notification

### ✅ Step 3: SNS Topic

* Go to **SNS > Create Topic**
* Name: BudgetAlert
* Subscribe:
  + Your email (optional)
  + Lambda function (see Step 4)

### ✅ Step 4: Lambda Cleanup Function

Deploy the following Lambda using Python:

import boto3  
  
def lambda\_handler(event, context):  
 ec2 = boto3.client('ec2')  
 instances = ec2.describe\_instances(Filters=[{'Name': 'instance-state-name','Values': ['running']}])  
 for r in instances['Reservations']:  
 for i in r['Instances']:  
 ec2.stop\_instances(InstanceIds=[i['InstanceId']])  
  
 dynamodb = boto3.client('dynamodb')  
 tables = dynamodb.list\_tables()['TableNames']  
 for table in tables:  
 dynamodb.delete\_table(TableName=table)  
  
 lambda\_client = boto3.client('lambda')  
 functions = lambda\_client.list\_functions()['Functions']  
 for func in functions:  
 lambda\_client.delete\_function(FunctionName=func['FunctionName'])  
  
 apigateway = boto3.client('apigateway')  
 apis = apigateway.get\_rest\_apis()['items']  
 for api in apis:  
 apigateway.delete\_rest\_api(restApiId=api['id'])  
  
 s3 = boto3.resource('s3')  
 for bucket in s3.buckets.all():  
 try:  
 bucket.objects.all().delete()  
 bucket.delete()  
 except Exception as e:  
 print(f"Bucket error: {bucket.name} - {str(e)}")  
  
 print("All deletions attempted.")

### ✅ Step 5: Schedule Cleanup Rule (Optional)

* Go to **EventBridge > Rules > Create Rule**
* **Type**: Schedule
* **Frequency**: cron(0 \* \* \* ? \*) (every hour)
* **Target**: Lambda

## 🚫 Preventing Hidden Costs

| Service | Caution |
| --- | --- |
| Elastic IPs | Charges when unattached |
| CloudWatch Logs | Never expire by default (set TTL) |
| NAT Gateway | Always billed, avoid entirely |
| Snapshots/Backups | May persist after EC2 termination |

## 📊 Monitoring & Summary

* Monitor costs via **Cost Explorer**
* Run Lambda manually for testing
* Keep CloudWatch logs short-lived

### 📅 Summary Table

| Resource | Action | Triggered By |
| --- | --- | --- |
| EC2 | Stop | Lambda + Budget Alert |
| Lambda | Delete | Lambda |
| API Gateway | Delete | Lambda |
| S3 | Empty/Delete | Lambda |
| DynamoDB | Delete | Lambda |
| IAM | Log Only | Optional |
| Lex | Manual Delete | Advanced Use Case |

## 📁 Output

* PDF available
* GitHub Repository:
  + Lambda script
  + Setup guide (README)
  + IAM policy templates
  + CloudFormation template (optional)

## 📆 Final Notes

This setup ensures **zero billing** from AWS by proactively cleaning up free-tier-exceeding resources and monitoring usage every hour or at first sign of cost.

Would you like the GitHub repo to include: