Terraform CloudWatch Deployment Guide

Terraform CloudWatch Deployment Guide

Introduction
This document provides a step-by-step guide on deploying a **unified CloudWatch dashboard** using **Terraform**. It enables cross-account **EC2 instance monitoring** by assuming IAM roles in target AWS accounts.

Overview
We will:
- Deploy a **CloudWatch dashboard** from a **master AWS account**.
- Fetch **EC2 metrics** from multiple AWS accounts.
- Use **Terraform modules** for automation.
- **Assume cross-account IAM roles** for secure access.

Prerequisites
- AWS CLI installed and configured.
- Terraform installed (`>= 1.0` recommended).
- IAM permissions to assume roles in target accounts.
- Access to AWS CloudWatch.

Step 1: Set Up IAM Roles

1.1 Create IAM Role in Target Accounts
1. In each **target AWS account**, navigate to **IAM**.
2. Create a new **IAM Role** with:
   - **Trust Policy:** Allow the master account to assume this role.
   - **Policy:** Attach **CloudWatch Read-Only Access** and **EC2 Read-Only Access**.
3. Example Trust Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "arn:aws:iam::<MASTER_ACCOUNT_ID>:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Copy the **role ARN** for use in Terraform.

Step 2: Configure Terraform Providers

Define providers in `main.tf`:

```
provider "aws" {
  region = var.aws_region
}

provider "aws" {
  alias  = "account1"
  region = var.aws_region
  assume_role {
    role_arn     = "arn:aws:iam::<TARGET_ACCOUNT_1>:role/TerraformDeploymentRole"
    session_name = "TerraformDeploymentSession"
  }
}

provider "aws" {
  alias  = "account2"
  region = var.aws_region
  assume_role {
    role_arn     = "arn:aws:iam::<TARGET_ACCOUNT_2>:role/TerraformDeploymentRole"
    session_name = "TerraformDeploymentSession"
  }
}
```

Step 3: Define Data Sources

Retrieve EC2 instance details for monitoring:

```
data "aws_instances" "instances_account1" {
  provider = aws.account1
  for_each = toset(var.environments)
  filter {
    name   = "tag:Environment"
    values = [each.key]
  }
}
```

Step 4: Build CloudWatch Dashboard
Generate dynamic widgets and deploy the dashboard.

Step 5: Define Outputs
Outputs the dashboard URL.

Step 6: Deploy with Terraform
1. **Initialize Terraform:**
   ```
   terraform init
   ```
2. **Validate Configuration:**
   ```
   terraform validate
   ```
3. **Plan Deployment:**
   ```
   terraform plan -var-file="terraform.tfvars"
   ```
4. **Apply Configuration:**
   ```
   terraform apply -var-file="terraform.tfvars" -auto-approve
   ```
5. **Check Dashboard URL:**
   ```
   terraform output dashboard_url
   ```

Troubleshooting
- **IAM Role Assume Issues:** Ensure correct IAM trust policy.
- **CloudWatch JSON Errors:** Verify correct formatting of `metrics` in JSON.
- **Terraform Errors:** Validate configuration using `terraform validate`.

Conclusion
This guide enables teams to automate **multi-account EC2 monitoring** with Terraform and CloudWatch.