AWS Assume Role + Terraform

Amit Saha
http://echorand.me

About me

Software Engineer - Interests in Infrastructure, Monitoring and Tooling

Leads the API engineering team at Freelancer.com, Past: Red Hat

Author of “Doing Math with Python”

Fedora Scientific creator/maintainer

Blog: http://echorand.me

Twitter: @echorand

Demo code

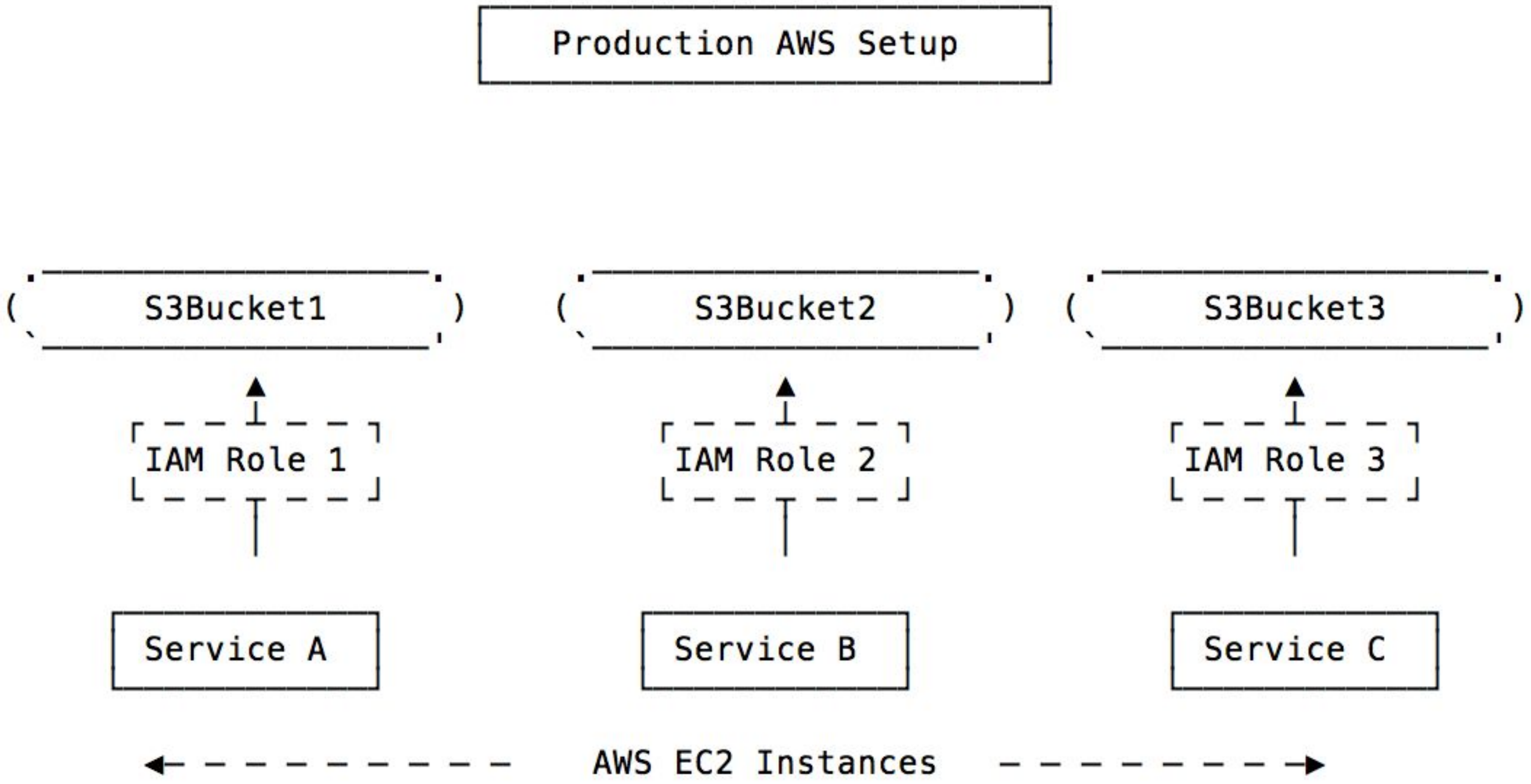
http://bit.ly/aws-assume-role-demo

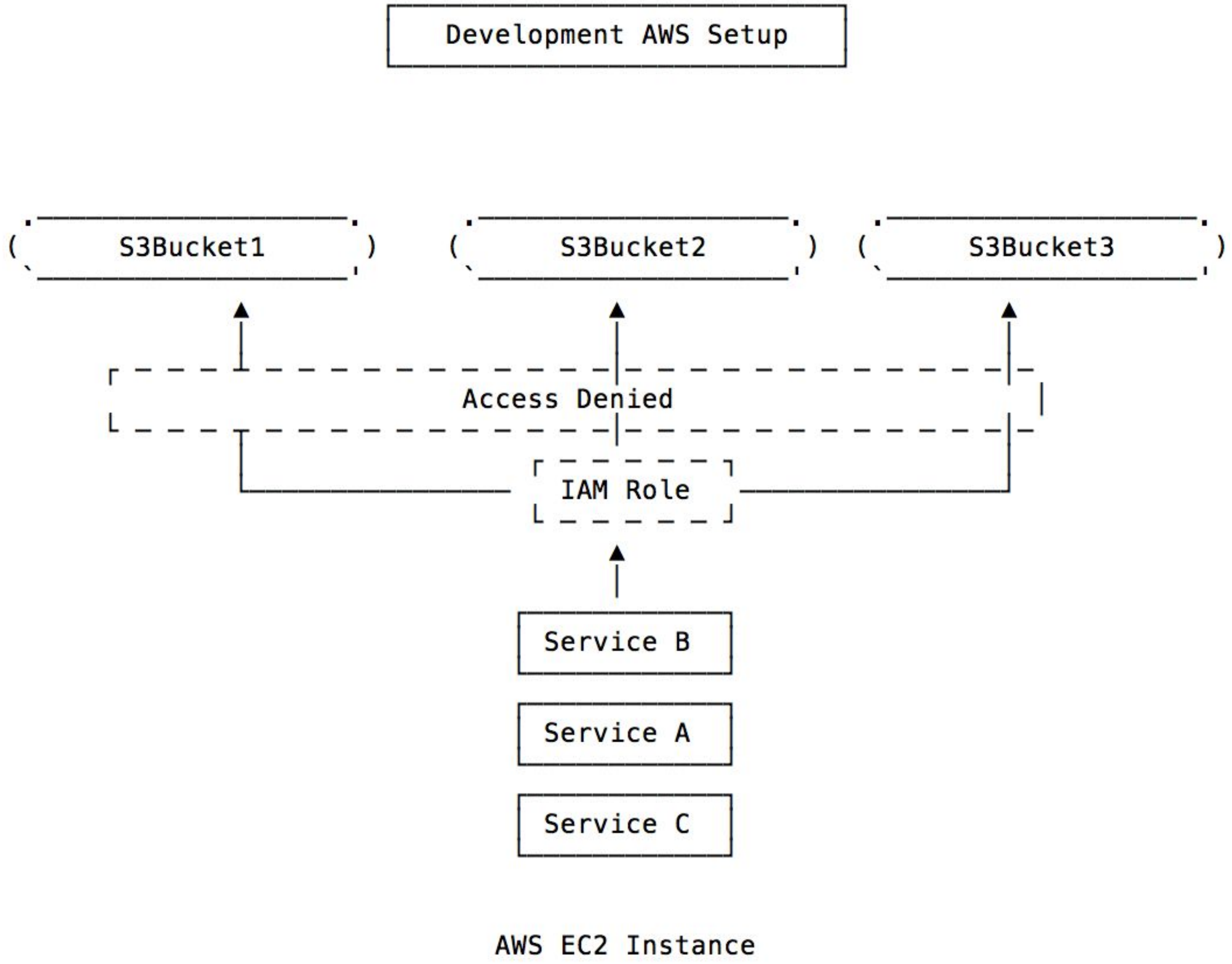
Problem Statement

An IAM role 1 needs to access a resource 2 which can only be accessed by IAM role 2.

Example scenarios:

- *Multiple services running in a development setup on a single EC2 instance*
- Containers


Production AWS Setup
S3Bucket1
S3Bucket2
S3Bucket3
IAM Role 1
IAM Role 2
IAM Role 3
Service A
Service B
Service C
AWS EC2 Instances


Development AWS Setup
S3Bucket1
S3Bucket2
S3Bucket3
Access Denied
IAM Role
Service B
Service A
Service C
AWS EC2 Instance

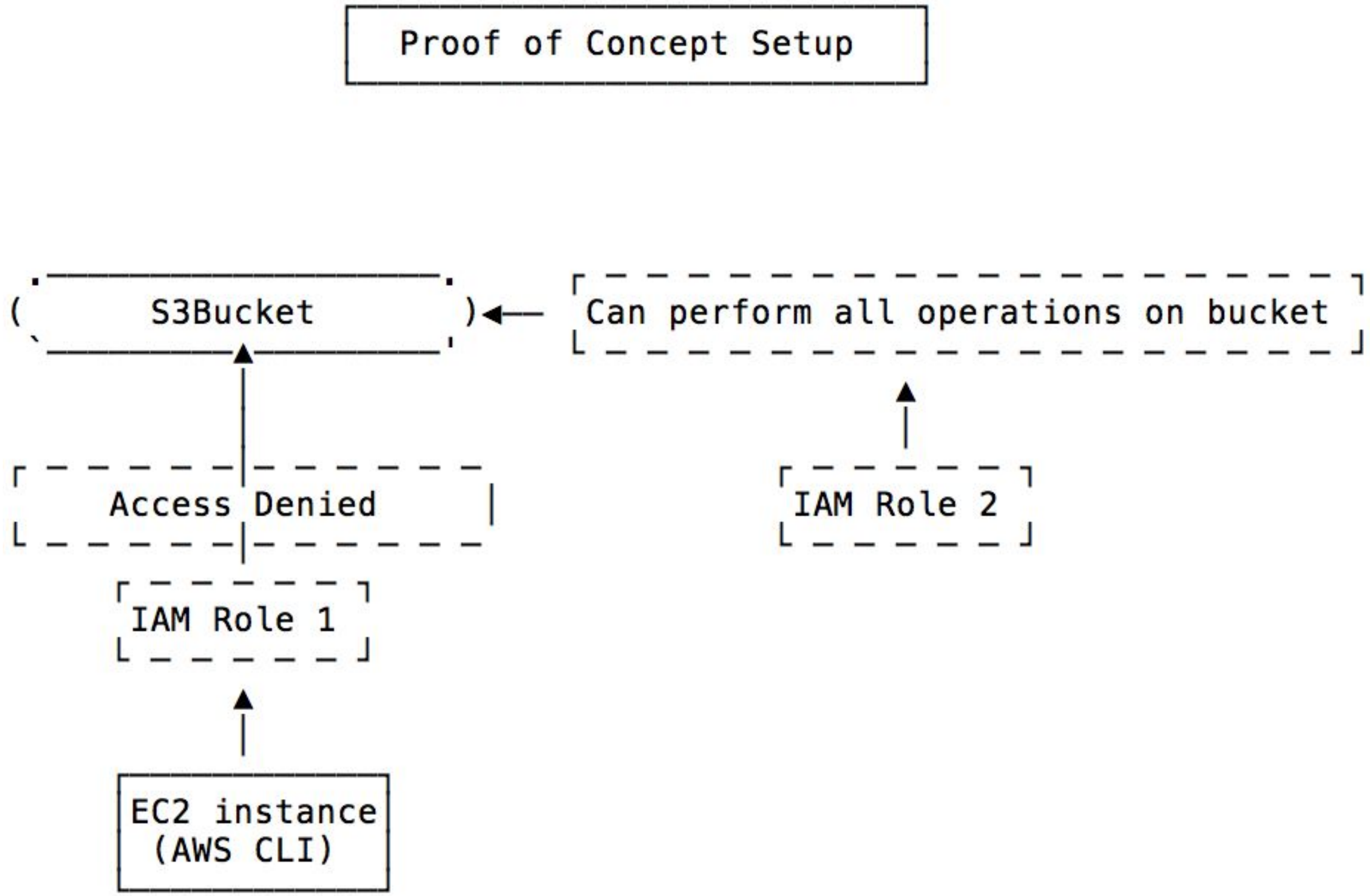
What do we do?

Option 1: Duplicate IAM policies in development to mirror those in production

Option 2: AWS Assume Role

AWS Assume Role

Proof of concept scenario


Proof of Concept Setup
S3Bucket
Can perform all operations on bucket
Access Denied
IAM Role 1
IAM Role 2
EC2 instance (AWS CLI)

Create the PoC infrastructure

AWS Console

AWS CLI

Cloudformation

Terraform

Terraform

Setup our PoC Infrastructure

Create a S3 bucket (*github-amitsaha-bucket*)

Create two IAM roles, *role1* and role2

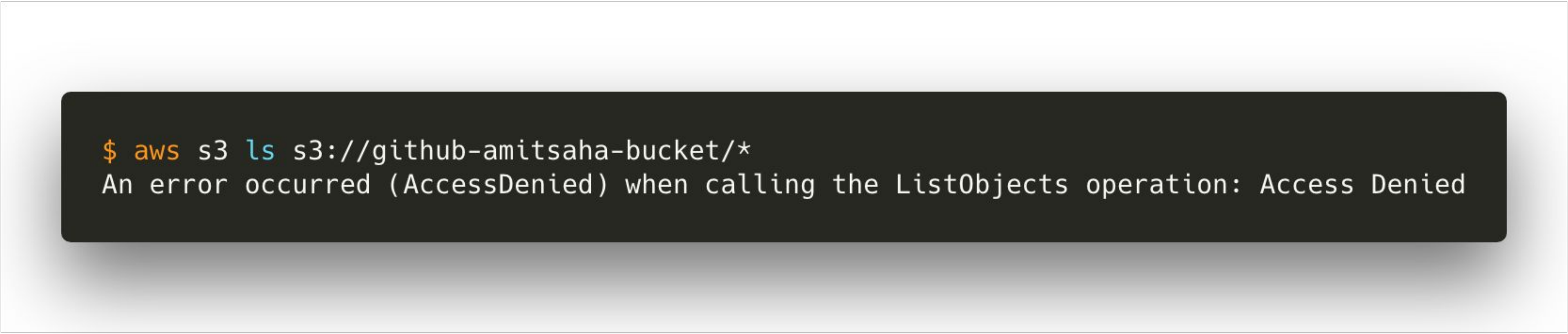
Add a policy to *role2* to be able to perform all operations on the S3 bucket

Spin up an EC2 instance using role1

(Terraform configuration here)

PoC Problem Demo

ssh into the ec2 instance



```
$ aws s3 ls s3://github-amitsaha-bucket/*
An error occurred (AccessDenied) when calling the ListObjects operation: Access Denied
```

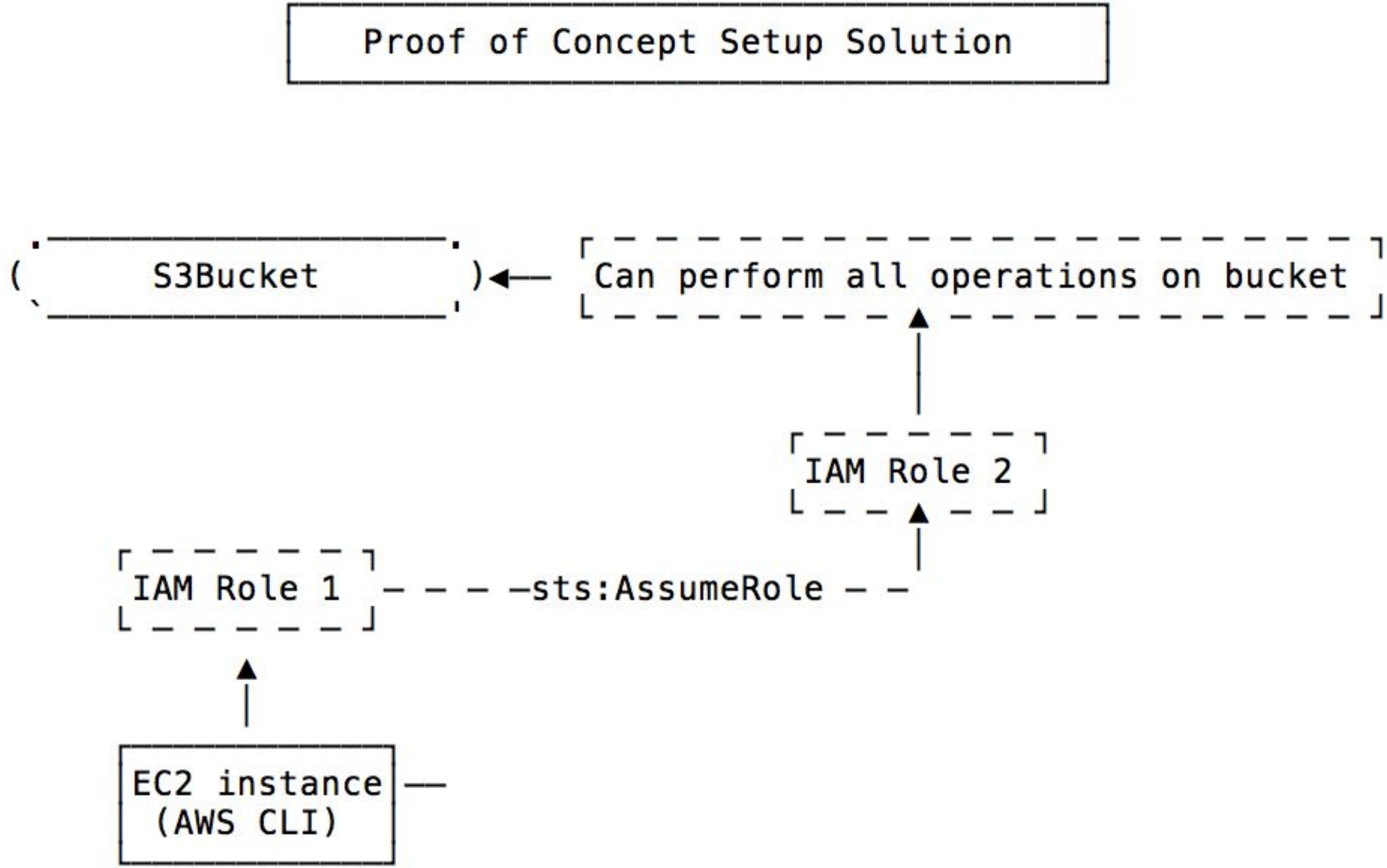
PoC Solution: Infrastructure Update

Update terraform configuration to allow role1 to assume role2

1. Role1 should be allowed to perform a *sts:AssumeRole* operation
2. Role2 should allow it’s policy to be *assumed* by Role1

Apply the changes

(Terraform configuration here)


Proof of Concept Setup Solution
S3Bucket
Can perform all operations on bucket
IAM Role 2
IAM Role 1
sts:AssumeRole
EC2 instance
(AWS CLI)

Allow role1 to assume role2

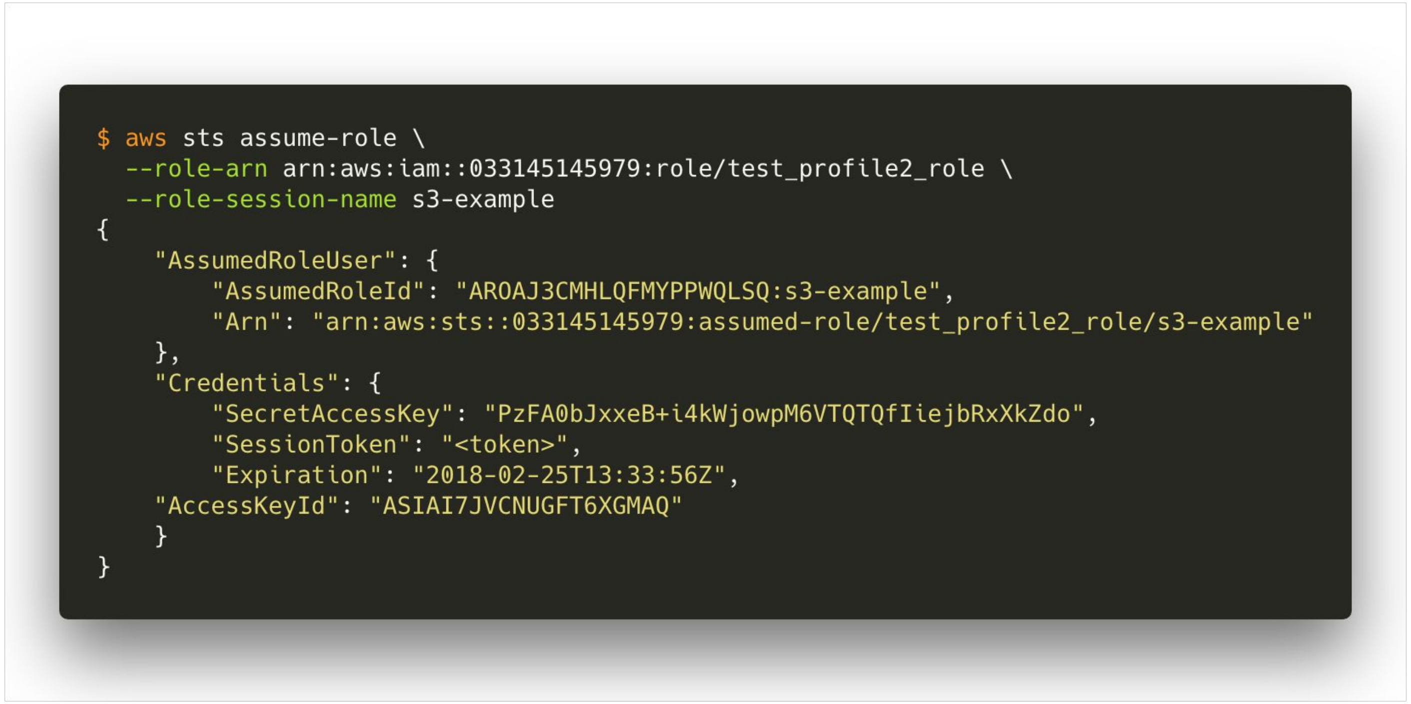
```
data "aws_iam_policy_document" "assume_role2_policy" {
  statement {
    actions = [
      "sts:AssumeRole",
    ]
    resources = [
      "${aws_iam_role.role2.arn}",
    ]
  }
}

resource "aws_iam_role_policy" "role1_assume_role2" {
  name   = "AssumeRole2"
  role = "${aws_iam_role.role1.name}"
  policy = "${data.aws_iam_policy_document.assume_role2_policy.json}"
}
```

Allow role2 to be assumed by role1

```
resource "aws_iam_role" "role2" {
  name = "test_profile2_role"
  path = "/"
  assume_role_policy = <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Principal": {
               "Service": "ec2.amazonaws.com",
               "AWS": "${aws_iam_role.role1.arn}"
            },
            "Effect": "Allow",
            "Sid": ""
        }
    ]
}
EOF
```

PoC Solution: Application Demo

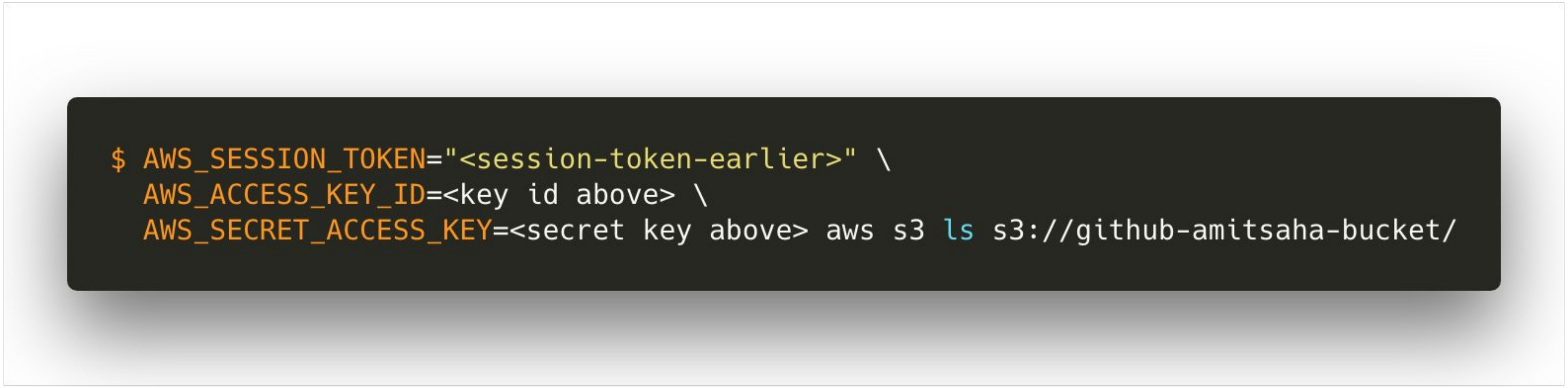
Perform assume role operation



```
$ aws sts assume-role \
  --role-arn arn:aws:iam::033145145979:role/test_profile2_role \
  --role-session-name s3-example
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAJ3CMHLQFMYPPWQLSQ:s3-example",
        "Arn": "arn:aws:sts::033145145979:assumed-role/test_profile2_role/s3-example"
    },
    "Credentials": {
        "SecretAccessKey": "PzFA0bJxxeB+i4kWjowpM6VTQTQfIiejbRxXkZdo",
        "SessionToken": "<token>",
        "Expiration": "2018-02-25T13:33:56Z",
    "AccessKeyId": "ASIAI7JVCNUGFT6XGMAQ"
    }
}
```

PoC Solution: Application Demo

Use the temporary credentials to access the resource



```
$ AWS_SESSION_TOKEN="<session-token-earlier>" \
  AWS_ACCESS_KEY_ID=<key id above> \
  AWS_SECRET_ACCESS_KEY=<secret key above> aws s3 ls s3://github-amitsaha-bucket/
```

“Scaling” the solution

When you have more than a few IAM roles, changing each IAM role’s policy to be assumed may not scale well or introduce unnecessary dependency:

```
Principal": {
                "Service": "ec2.amazonaws.com",
                "AWS": "${aws_iam_role.role1.arn}"
            },
```

We can change it to:

```
60c60
<                     "AWS": "${aws_iam_role.role1.arn}"
---
>                     "AWS": "arn:aws::iam::${data.aws_caller_identity.current.account_id}:root"
```

(Terraform configuration here)

Alternative approaches

This solution requires your application code to be modified to perform *assume role*

There needs to be some mechanism to check the expiry of the temporary token

Alternative approaches include *metadataproxy* and *kube2iam*

Useful links

Blog post: Setting up AWS EC2 Assume Role with Terraform

Terraform configuration for the demos:
https://github.com/amitsaha/aws-assume-role-demo

AWS Assume Role

Questions?