Lab 3 - IAM Role for Ec2 and S3 Bucket

This repository contains Lab 3 for connecting an EC2 instance to an S3 bucket via an IAM role and policy. Follow the instructions below to complete the lab, earn points, and climb the leaderboard!

Now review the tasks I've laid out for you all. You can try this first from the GUI, if you have one with your AWS Free Tier. I'm not sure if Local Stack provides a GUI, but maybe it does. Anyway, we usually try to "play" with something that is new via "ClickOps" in the GUI in AWS. So you can follow some tutorial that shows you S3 bucket creation and go ahead if you want to visually see it come together. When engineers do automation, they write down all the variables and parameters they add into the new infrastructure or service they provision via the GUI. Once they are happy, they delete the test resources and write it into automation.

So there are some stretch goals here. And finally, I'm add a points system for a Leaderboard. You don't have to worry about points if you don't want to. I will try to think of something fun for the winners at the end of the 6 months.

Let me know if the lab is unclear or has any information that needs clarification.

---

1. Objective

- Understand how to grant permissions to EC2 and S3 via IAM roles.
- Learn about S3 buckets. How to create one and what they are used for. Also, why and how they can be a juicy security risk.
- Create and attach a least-privilege policy
- Associate the IAM role with your running EC2 instance.
- Verify access from the instance --> ec2-vm :> `aws s3 ls`

  You should see your bucket name listed

2. Prerequisites

- Lab 1: AWS Console navigation.
- Lab 2: Security Groups setup.
- AWS Free Tier account with IAM, EC2, and S3 permissions.
- AWS CLI installed and configured.

- Running Linux EC2 instance (t2.micro) and its ID.
- Existing S3 bucket (same Region).

Environment Setup

1. **Note your EC2 details**
   - Instance ID: `<EC2_INSTANCE_ID>`
   - Public IPv4 (for SSH): `<EC2_PUBLIC_IP>`
2. **Note your S3 bucket name**
   - Bucket: `<YOUR_BUCKET_NAME>`
3. **Verify CLI access**

   ```
   aws sts get-caller-identity
   ```
4. **Ensure Security Group allows outbound HTTPS/SSH** (from Lab 2).

---

4. Tasks

4.1. Create a Custom IAM Policy

1. **Console**
   - IAM → Policies → Create policy → JSON → paste: json CopyEdit `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "s3:ListBucket", "Resource": "arn:aws:s3:::<YOUR_BUCKET_NAME>" }, { "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::<YOUR_BUCKET_NAME>/*" } ] }`
   - Name: `EC2_S3_ListAndGet_<yourname>`
   - Create policy.
2. **CLI**

   ```
   aws iam create-policy \ --policy-name EC2_S3_ListAndGet_<yourname> \ --policy-document file://ec2-s3-policy.json
   ```

4.2. Create an IAM Role for EC2

1. **Console**
   - IAM → Roles → Create role → Select “AWS service” → EC2 → Next.
   - Attach the `EC2_S3_ListAndGet_<yourname>` policy.
   - Name: `EC2-S3-Access-Role-<yourname>` → Create role.

2. **CLI**

   `aws iam create-role \ --role-name EC2-S3-Access-Role-<yourname> \ --assume-role-policy-document file://ec2-trust-policy.json aws iam attach-role-policy \ --role-name EC2-S3-Access-Role-<yourname> \ --policy-arn arn:aws:iam::<ACCOUNT_ID>:policy/EC2_S3_ListAndGet_<yourname>`

   where `ec2-trust-policy.json` contains:

   `json

   { "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "Service": "ec2.amazonaws.com" }, "Action": "sts:AssumeRole" }] }
   `

4.3. Attach the Role to Your EC2 Instance

1. **Console**
   - EC2 → Instances → Select your instance → Actions → Security → Modify IAM role → Choose `EC2-S3-Access-Role-<yourname>` → Save.
2. **CLI**

   `aws ec2 associate-iam-instance-profile \ --instance-id <EC2_INSTANCE_ID> \ --iam-instance-profile Name=EC2-S3-Access-Role-<yourname>`

4.4. Verify Access from the EC2 Instance

`ssh -i /path/to/key.pem ec2-user@<EC2_PUBLIC_IP> # On the EC2 shell: aws s3 ls s3://<YOUR_BUCKET_NAME>`

You should see your bucket contents.

---

5. Stretch Goals

a. Deep Dive into Bucket Policies

1. **Create a bucket policy** that blocks all public access but allows your IAM role (try and format it correctly before pasting it in):

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

~~~
"Principal": "*",     "Action":"s3:*",     "Resource":["arn:aws:s3:::
<YOUR_BUCKET_NAME>","arn:aws:s3:::<YOUR_BUCKET_NAME>/*"],     "Condition":
{"Bool":{"aws:SecureTransport":"false"}}   }] }

    ```



2. **Experiment** with requiring MFA or VPC conditions.

3. **Host a static site**

    - Enable static website hosting (`index.html`).

    - Configure Route 53 alias or CNAME for `resume.<yourdomain>` to the
bucket endpoint.

    - (Optional) Deploy CloudFront with ACM certificate for HTTPS.


### b. Private "Invite-Only" Resume Hosting

1. **Pre-signed URLs**

    `aws s3 presign s3://<YOUR_BUCKET_NAME>/resume.pdf --expires-in 3600`

2. **IAM-only access**

    - Store under `private/`.

    - Write a bucket policy allowing only the role `EC2-S3-Access-Role-
<yourname>` to `GetObject`.

3. **Restrict to IP address:**
   ```json

   {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetListFromSpecificIP",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:ListBucket",
        "s3:GetObject"
~~~

```
      ],
      "Resource": [
        "arn:aws:s3:::your-bucket-name",
        "arn:aws:s3:::your-bucket-name/*"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "YOUR IP YOU LOOKED UP/32"
        }
      }
    }
  ]
}
```

What this does:

- **Principal**: `*` means "anyone" (we lock it down by IP). **Please** do some reading around Principal/Action/Resource/Condition structure
- **Action**: Allows only `ListBucket` on the bucket itself and `GetObject` on objects.
- **Resource**: Targets both the bucket and all its objects.
- **Condition**:
  - Uses the `IpAddress` operator.
  - `aws:SourceIp` must exactly match `203.0.113.5/32` (replace with your actual public IP, in CIDR notation).

Steps to apply in the console

1. **Go to S3** → select your bucket → **Permissions** → **Bucket policy** → **Edit**.
2. Paste in the above JSON, replacing:
   - `your-bucket-name` with the real bucket name.
   - `203.0.113.5/32` with your (e.g. `198.51.100.24/32`).
3. **Save**.

Now only requests coming from that IP can list or read objects in the bucket.

CLI alternative

Save the JSON above to a file `bucket-ip-policy.json`, then run:

`aws s3api put-bucket-policy \ --bucket your-bucket-name \ --policy file://bucket-ip-policy.json`

Extending for multiple IPs

If you want to allow more than one IP (e.g. home and office), simply supply a list in the json policy:

`"Condition": { "IpAddress": { "aws:SourceIp": [ "yourIP/32", "198.51.100.24/32" ] } }`

That way, both addresses are permitted.

c. Further EC2 Exploration on Free Tier

1. **Snapshots & AMIs**
   - Create an EBS snapshot of `/dev/xvda`.
   - Register or create an AMI from that snapshot. Understand how you can "version" a server with snapshots. Why this is useful.
   - Launch a new instance from your AMI.
2. **Linux & Security Tooling**
   - Use `ss -tulpn`, `lsof`, and `auditctl` to inspect services and audit.
   - Install and run:
     - `nmap localhost`
     - `tcpdump -c 20 -ni eth0`
     - `lynis audit system`
     - `fail2ban-client status`
     - OSSEC/Wazuh or ClamAV.
3. **Scripting & Automation**
   - Bash: report world-writable files (research any commands you don't know!): `find / -perm -002 -type f > ww-files.txt`
   - Python with `boto3`: list snapshots, start/stop instances.

---

6. Submit the Completed Lab

- **Screenshot**: IAM Role’s **Trust relationships** and **Permissions** tab.
- **Terminal output**: `aws s3 ls` success.

- **Commands** used. with section and number from this lab sheet referenced.
- **Mask** any sensitive ARNs, IDs, or IPs. And remember, no keys!!! Check your code for keys before committing to GitHub. Soon we will build a pipeline to keep you safe.

---

7. Further Reading and Resources for Study

- **IAM Roles for EC2**: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html
- **Creating IAM Policies**: https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html
- **S3 Bucket Policies**: https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html
- **Static Website Hosting on S3**: https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteHosting.html
- **AWS CLI Reference**: https://docs.aws.amazon.com/cli/latest/reference/

- **S3 Security Best Practices**: https://cloudsecurityalliance.org/blog/2024/06/10/aws-s3-bucket-security-the-top-cspm-practices#
- **AWS EC2 Security**: https://aws.plainenglish.io/271-essential-ec2-security-checks-best-practices-for-a-secure-aws-instance-7b10219c8273

Submission Guidelines

- Include screenshots of IAM role trust & permissions tabs.
- Paste your terminal session showing successful `aws s3 ls`.
- List all commands used.
- Redact any sensitive ARNs, IDs, or IPs.

---

8. Reflection (1–2 sentences)

- **What you built**: an IAM role granting your EC2 instance least-privilege access to list and read objects in a specific S3 bucket.

- **Challenges**: navigating trust policies, JSON syntax errors, and verifying role association on a running instance.
- **Security concerns:** Ok, you've got two pieces of infrastructure in place now. Regardless of how much you did on this lab, I want you to know that this is a lot. Especially, if you did some or all of the stretch goals. That's a lot!

  Reflect on **scale and security at scale.** Are you surprised or overwhelmed at the kinds of issues you might potentially see here? Did you do some research on security issues for Ec2s and S3s? How can we operate security at the kind of scale where we have hundreds of buckets, hundreds of Ec2s and 40 groups, all those AWS roles and policies and 100 users?

  Don't let your head explode. Just reflect on that and give yourself a pat on the back for getting this far.

  --> Now make Lab structure in your **GitHub or Gitea** (Lab 1, Lab 2, Lab 3) and upload your lab work and artifacts.

Good luck and happy hacking! 🎉