LESSON WORKBOOK

# AWS Fundamentals 03 - IAM Roles and a Private S3 Workflow

## AWS Fundamentals

Study the lesson, work through the lab, and check your understanding with the quiz. Keep the reference sheet nearby as you practice.

Includes the complete lesson, guided lab, knowledge check, answer explanations, and quick reference.

# Lesson

## AWS Fundamentals 03 - IAM Roles and a Private S3 Workflow

**CloudComputingTutor · Beginner · 25-minute reading + 45-60-minute lab**
**Prerequisites:** `aws-fundamentals-02`; basic file paths and shell quoting. **Outcome:** Give a temporary EC2 role permission to write/read/delete objects under one S3 prefix, prove allowed and denied behavior, and remove the complete lab. **Publication state:** New teaching material; these steps were checked against AWS documentation, not run against a live account. Sources checked **2026-09-18**.

### 1. Shutterbird needs a permission boundary

Shutterbird's temporary machine can now be managed, but it cannot automatically access the company's photos. Today the fictional team gives it a smaller task: place a training note under `uploads/` in one private bucket, retrieve that note, and leave other prefixes alone.

The note represents an object-processing workflow without personal data or a production application. Success has two sides. The intended operation works. An operation outside the intended scope fails. Testing only the successful upload would miss half of the requirement.

You will create a dedicated bucket through the console, attach an original narrow policy to the lab role, use the AWS CLI on EC2, and inspect the results. No access key needs to be copied into a file. This is the first step toward an application identity whose job is written down precisely.

### 2. Role, trust, permission, and session

An IAM role describes permissions that an authorized principal can assume. The role's **trust policy** answers who may assume it. Its **permissions policies** answer what the resulting identity may do. An instance profile supplies the role to EC2. The AWS CLI can obtain temporary role credentials through EC2's metadata service automatically. You inspect the caller identity, not the underlying credentials. EC2 role workflow, CLI instance credentials

Our existing role trusts the EC2 service and has the SSM permissions needed for management. We will add an S3 policy for one lab bucket. The human who edits this role has a different identity from the machine that performs the object requests. Keep browser provisioning and terminal workload actions separate in your notes.

This lesson assumes the bucket and role belong to the same account and there are no additional bucket policies, permission boundaries, endpoint policies, session restrictions, or organization controls denying the requests. Those controls can further restrict access. The absence of an allow normally means denial; an applicable explicit deny wins. We do not remove organizational safeguards to force a tutorial to pass. S3 policy model

## 3. Buckets and keys: the addressing model

A bucket is an S3 container in a selected Region. An object has a key, such as `uploads/observation.txt`, plus its data and metadata. The slash makes the key convenient to browse, but S3 prefixes are not Linux directories. A console "folder" presentation should not lead you to assume Unix ownership or `chmod` controls S3 access. S3 prefixes

Two ARN shapes matter. The bucket itself is `arn:aws:s3:::BUCKET_NAME`. An object namespace beneath it is `arn:aws:s3:::BUCKET_NAME/uploads/*`. Listing operates on the bucket, with a prefix condition. Reading/writing/deleting an object uses the object ARN. Mixing those shapes is a common reason a plausible policy does not work.

Privacy and encryption also answer different questions. Encryption at rest protects stored data through an encryption mechanism. IAM and resource settings determine which requests may access it. An encrypted object does not automatically have the right access policy. We use SSE-S3 encryption and a private bucket so we can study the permission question without introducing KMS key permissions. Default encryption

## 4. Preflight and cost boundaries

Use the learning account, Region, budget, and network route from AWS Fundamentals 02. If that lesson's instance was terminated, follow its preflight and Steps 1-4 to recreate the dedicated instance, role/profile, and security group; stop before cleanup. If continuing immediately in the same supervised session, record the reused IDs. Never leave the machine running overnight just to preserve continuity.

Your console identity needs permission to create/configure/delete this lab bucket, view its contents, and add/remove an inline policy on the dedicated instance role. The EC2 profile must initially have only the lesson's SSM policy, with no broad S3 policy. Keep the existing no-inbound/outbound-HTTPS network design and functioning STS/S3 regional endpoint connectivity. If an instructor provisions resources, they must preserve these assumptions for the denied tests to mean anything.

Budget for EC2, EBS, public IPv4, S3 storage, S3 requests, and applicable transfer. Use one tiny text object and complete cleanup in this session. Current prices and account credits vary; budget notifications can be delayed. The role is intentionally allowed to delete its training objects for cleanup. A production uploader may need a different permission set.

## 5. Step 1 - Create a private training bucket

In the S3 console choose **Create bucket**, **General purpose**, and your recorded Region. Use a unique lowercase name such as `cct-shutterbird-lab-` followed by a random suffix; bucket names must meet AWS naming requirements and be available. Record the actual name without adding private personal information to it.

Keep **Object Ownership: Bucket owner enforced**, which disables ACLs. Keep all four **Block Public Access** settings enabled. Leave versioning disabled and Object Lock off for this disposable exercise. Use the default **SSE-S3** encryption setting. Leave static website hosting disabled and do not add a bucket policy. Add a project tag if your permissions allow it. Create the bucket and inspect the saved settings. S3 getting started, ownership, Block Public Access

Expected evidence: one empty general-purpose bucket, correct Region, no public access, ACLs disabled, SSE-S3, and no versioning. If an organization mandates versioning or KMS, use the tabletop route or an instructor-adjusted lab with its corresponding permissions and cleanup steps. Do not override those controls.

## 6. Step 2 - Identify the caller and observe the baseline

Open the EC2 browser Session Manager shell. The standard Amazon Linux 2023 image includes AWS CLI v2; confirm with `aws --version`. If unavailable, stop and check the selected AMI with your instructor rather than copying an unreviewed installer. AL2023 CLI

Replace the two example values below with your recorded Region and actual bucket name, keeping quotes:

```
LAB_REGION='us-east-1'
LAB_BUCKET='replace-with-your-actual-bucket-name'
aws sts get-caller-identity --region "$LAB_REGION" --no-cli-pager
aws s3api list-objects-v2 \
  --bucket "$LAB_BUCKET" \
  --prefix 'uploads/' \
  --region "$LAB_REGION" \
  --no-cli-pager
```

Expected identity: an `assumed-role` ARN containing your dedicated EC2 role name. The listing should return **AccessDenied** before adding the S3 policy. If it succeeds, investigate unexpected existing permission; do not claim a valid baseline. Record the error without exposing sensitive identifiers publicly. `get-caller-identity` tells you who sent the request; it does not enumerate every permission that identity possesses. Caller identity reference

## 7. Step 3 - Add a policy for one prefix

In the browser IAM console, open the **dedicated EC2 role** from your inventory. Choose Add permissions -> Create inline policy -> JSON. Replace **both** instances of `REPLACE_WITH_LAB_BUCKET` below with the exact bucket name. Review the JSON and save it as `CCTUploadsOnly`. Leave the SSM managed policy attached.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListUploadsPrefix",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::REPLACE_WITH_LAB_BUCKET",
```

```
      "Condition": {
        "StringLike": {
          "s3:prefix": ["uploads/*"]
        }
      }
    },
    {
      "Sid": "ManageTrainingObjects",
      "Effect": "Allow",
      "Action": ["s3:PutObject", "s3:GetObject", "s3:DeleteObject"],
      "Resource": "arn:aws:s3:::REPLACE_WITH_LAB_BUCKET/uploads/*"
    }
  ]
}
```

Read the policy aloud as two sentences. First: this role may list this bucket when the requested prefix starts with `uploads/`. Second: it may put, get, and delete objects whose keys start with `uploads/`. There is no permission to list all account buckets, modify the bucket's privacy settings, or create another bucket. Wait briefly for IAM changes to propagate, then retry the previous listing. An empty successful response is expected. A broader console browsing permission is unnecessary for these specific API calls. S3 identity-policy examples

## 8. Step 4 - Write, list, retrieve, compare

Use a new local workspace and one harmless sentence. Run each command separately:

```
LAB_DIR=$(mktemp -d /tmp/cct-s3-workflow.XXXXXX)
printf '%s\n' 'Shutterbird training object: role-based access works.' > "$LAB_DIR/observation.txt"
aws s3api put-object \
  --bucket "$LAB_BUCKET" \
  --key 'uploads/observation.txt' \
  --body "$LAB_DIR/observation.txt" \
  --region "$LAB_REGION" \
  --no-cli-pager
aws s3api list-objects-v2 \
  --bucket "$LAB_BUCKET" \
  --prefix 'uploads/' \
  --region "$LAB_REGION" \
  --no-cli-pager
aws s3api get-object \
  --bucket "$LAB_BUCKET" \
  --key 'uploads/observation.txt' \
  --region "$LAB_REGION" \
  --no-cli-pager "$LAB_DIR/download.txt"
cmp "$LAB_DIR/observation.txt" "$LAB_DIR/download.txt"
cat "$LAB_DIR/download.txt"
```

Expected: upload returns response metadata; listing includes the exact key; retrieval writes the local download; `cmp` succeeds silently with exit status zero; `cat` shows your sentence. Record the commands and their observed outcomes. A returned ETag is useful metadata, but do not assume every S3 ETag is an MD5 digest in all encryption/upload configurations. Here the byte comparison is the test you actually performed. PutObject, GetObject

## 9. Step 5 - Prove the boundary

Keep the same role, bucket, and session. Request a listing outside the permitted prefix:

```
aws s3api list-objects-v2 \
  --bucket "$LAB_BUCKET" \
  --prefix 'private/' \
  --region "$LAB_REGION" \
  --no-cli-pager
```

Expected: **AccessDenied**. It should not merely return an empty list. This test is read-only and creates no outside-prefix data. It checks the listing boundary; the policy review separately shows the object-action boundary. An additional optional negative test is `get-object` for a known instructor-created object outside `uploads/`, but do not create extra objects or infer missing-object behavior from ambiguous status codes in this introductory lab.

If an allowed request fails, inspect exact bucket spelling, ARN shape, prefix case, Region, active role, policy propagation, and higher-level restrictions. If the forbidden listing succeeds, inspect unexpected permissions before continuing. Never disable Block Public Access to fix a role-based request. Public access is not required for an authorized role to read a private object. ListObjectsV2 reference

## 10. Step 6 - Delete exactly what you created

Save your redacted evidence. Delete the exact training object and list the same prefix again:

```
aws s3api delete-object \
  --bucket "$LAB_BUCKET" \
  --key 'uploads/observation.txt' \
  --region "$LAB_REGION" \
  --no-cli-pager
aws s3api list-objects-v2 \
  --bucket "$LAB_BUCKET" \
  --prefix 'uploads/' \
  --region "$LAB_REGION" \
  --no-cli-pager
```

Expected: the key is absent. Through the console, confirm the dedicated bucket is empty and delete that exact bucket. The instance role deliberately lacks bucket deletion permission, so use your authorized provisioning identity. This procedure assumes versioning was never enabled; versioned buckets require deleting versions and delete markers too. Object deletion, bucket deletion

Remove `CCTUploadsOnly` from the lab role. End the session and complete AWS Fundamentals 02's cleanup: terminate the recorded instance, verify EBS removal/address release, and remove only the dedicated security group and unused role/profile. Keep shared networking and the budget. Record IDs and completion time, then revisit cost data after it updates.

## 11. Offline route and assessment

Without AWS, use the policy as a tabletop case. Make a request table with columns for identity, action, resource, prefix, predicted result, and reason. Evaluate: list `uploads/`; list `private/`; get `uploads/observation.txt`; put `archive/observation.txt`; delete the bucket. Predicted results are allow, deny, allow, deny, deny under the stated assumptions. Explain why listing and object actions use different ARN shapes.

On any local terminal with `printf`, `mktemp`, and `cmp`, you may create and compare two local text files to practice the observation method. That is a local file exercise, not proof of S3 behavior. Label every unexecuted API result "predicted."

Submit the identity diagram, original role-policy JSON with your private identifiers redacted for sharing, baseline denial, allowed object round trip, forbidden-prefix denial, and teardown record. Tabletop learners submit predicted equivalents. Next topics build on this foundation: VPC routing, CloudWatch observations, and repeatable infrastructure. The central skill is already present: state a narrow job, implement its permission boundary, test both sides, and remove the experiment.

# Knowledge check and answers

## AWS Fundamentals 03 - Scenario Quiz

Answer before reading the explanation. Score one point per correct choice; aim for 6/8 and explain both permission statements without looking.

### 1. One machine, one job

Shutterbird's EC2 worker needs to upload training objects. What is the appropriate credential approach for this exercise?

A. Put the root access key into a script. B. Create a public bucket. C. Let the AWS CLI use temporary credentials from the attached EC2 role. D. Email an administrator password to the worker.

**Answer: C.** The instance profile makes the role available to EC2, and the CLI can use its temporary credentials automatically. No permanent key is required.

### 2. Trust versus permission

The role's trust policy allows EC2 to assume it. Does that alone allow S3 uploads?

A. Yes, every assumed role may upload to S3. B. No; the role also needs an applicable permission for the operation and object resource. C. Only if the bucket is public. D. Only if the filename ends in `.jpg`.

**Answer: B.** Trust describes who can assume the role. A permissions policy describes what the resulting identity may do, subject to other applicable controls.

### 3. The ARN mismatch

A policy grants `s3:ListBucket` on `arn:aws:s3:::example-bucket/uploads/*`. Why does that not correctly grant listing?

A. S3 supports no lists. B. `ListBucket` targets the bucket ARN; a prefix condition narrows the listing. C. Prefixes must be IAM users. D. All ARNs require an EC2 Region.

**Answer: B.** The listing action belongs to the bucket resource. Get/Put/DeleteObject target object resources. The lesson uses separate statements for those scopes.

### 4. A private object round trip

Block Public Access is enabled, yet the authorized instance role successfully uploads and downloads the note. What does this show?

A. The bucket has become public. B. Private objects can be accessed by authenticated identities with the correct permission. C. Encryption was disabled. D. The console has ignored IAM.

**Answer: B.** Public access and authorized role access are different. A private object need not be inaccessible to every identity.

## 5. An outside-prefix listing

The only S3 listing allow is conditioned on `uploads/*`. The role requests `private/` in the same bucket and receives an empty successful result. What should the learner do?

A. Declare the denied test passed. B. Disable encryption. C. Investigate another effective permission because the expected outcome was AccessDenied. D. Add AdministratorAccess.

**Answer: C.** Empty and forbidden are different outcomes. The negative test checks authorization, not whether objects happen to exist.

## 6. Encryption confusion

A teammate says, "The bucket uses SSE-S3, so the IAM policy doesn't matter." What is the correction?

A. Encryption at rest and authorization solve different problems; access permissions still matter. B. SSE-S3 grants anonymous read. C. IAM only applies to EC2. D. An object key is a password.

**Answer: A.** Encryption describes stored-data protection. IAM and resource controls determine whether a request is authorized.

## 7. Bucket deletion fails from the worker

After deleting the training note, the EC2 role cannot delete the empty bucket. What best explains this?

A. S3 buckets cannot be deleted. B. The role has object permissions and a scoped list permission, but no DeleteBucket permission. C. A running instance always locks all buckets. D. The Linux file mode is wrong.

**Answer: B.** That failure matches the intended boundary. The authorized provisioning identity deletes the dedicated empty bucket through the console.

## 8. A versioning difference

An instructor changes the exercise to a versioned bucket. Is deleting the current object key always enough to empty it?

A. Yes, every version disappears automatically. B. No; versions and delete markers can remain and need their own authorized cleanup. C. Only public buckets have versions. D. Terminating EC2 deletes S3 versions.

**Answer: B.** This lesson deliberately uses an unversioned disposable bucket. A versioned variation requires an updated permission and cleanup design.

**Instructor extension:** Given `GetObject uploads/a.txt`, `PutObject archive/a.txt`, `ListBucket prefix=uploads/`, and `DeleteBucket`, ask learners to identify the relevant statement or the absence of an allow. Keep the same-account/no-additional-deny assumptions explicit.

# Quick reference

## IAM + Private S3 - Desk Reference

**Shutterbird task:** manage a training note under `uploads/` in one private bucket. **Source check:** 2026-09-18. Follow lesson.md for complete prerequisites and policy JSON.

| Question | Concept |
|---|---|
| Who may assume the role? | Trust policy |
| What may that identity do? | Permissions policies, subject to other controls |
| How does EC2 receive a role? | Instance profile |
| How does the CLI authenticate here? | Automatically obtained temporary role credentials |
| What names one stored object? | Bucket + object key |
| Is `uploads/` a Linux directory? | No; it is an S3 key prefix |

| Lab request | Intended result | Permission shape |
|---|---|---|
| List `uploads/` | Allow | ListBucket on bucket ARN + prefix condition |
| Get/put/delete `uploads/note.txt` | Allow | Object actions on bucket/uploads/* ARN |
| List `private/` | Deny | No matching list condition |
| Put `archive/note.txt` | Deny | No matching object resource |
| Delete bucket | Deny for instance role | Provisioning identity handles teardown |

**Private bucket settings:** General purpose; recorded Region; unique name; Bucket owner enforced; all Block Public Access enabled; SSE-S3; versioning disabled; Object Lock off; no website hosting or bucket policy.

**Before the workflow:** `aws --version`; set `LAB_REGION` and `LAB_BUCKET`; verify `aws sts get-caller-identity`; observe the pre-policy denial. Never print or save underlying role credentials.

**Workflow:** create local note -> `put-object` -> `list-objects-v2` with `--prefix uploads/` -> `get-object` -> `cmp` original/download -> denied listing for `private/` -> delete exact object -> verify empty -> provisioning identity deletes bucket.

**Policy review:** ListBucket uses `arn:aws:s3:::BUCKET`. Object operations use `arn:aws:s3:::BUCKET/uploads/*`. Keep the SSM policy; add the narrow inline policy only to the dedicated role.

**Investigate:** active identity, bucket name, prefix/case, ARN shape, Region, policy propagation, and additional policies/denies. Empty success is not AccessDenied. Keep Block Public Access enabled.

**Cleanup:** save evidence; delete exact object and dedicated empty bucket; remove lab inline policy; terminate EC2; verify EBS/address cleanup; remove dedicated unused role/profile and security group; preserve shared network/budget; review delayed costs. Versioning changes the deletion procedure.

Sources: EC2 roles, prefixes, policy model, private access controls, bucket cleanup.