AWS IAM Policy and CLI Operations Guide

Task 1: Custom IAM Policy for EC2 Restriction

Policy Creation

Here's the JSON for a custom IAM policy that restricts EC2 launches to only T2.micro instances in the Mumbai region:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "ec2:RunInstances",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "ec2:InstanceType": "t2.micro"
            },
            "ArnLike": {
               "aws:RequestedRegion": "ap-south-1"
            }
         }
      },
      {
         "Effect": "Allow",
         "Action": [
            "ec2:DescribeInstances",
            "ec2:DescribeImages",
            "ec2:DescribeKeyPairs",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs"
```

```
        ],
        "Resource": "*"
      }
    ]
}
```

Implementation Steps:

1. Navigate to IAM Console → Policies → Create Policy
2. Select JSON tab and paste the above policy
3. Name the policy "EC2_T2Micro_Mumbai_Only"
4. Attach this policy to the desired IAM user/group

Validation:

- Attempt to launch a T2.micro in ap-south-1 (should succeed)
- Attempt to launch any other instance type (should fail)
- Attempt to launch in any other region (should fail)

Task 2: AWS CLI IAM Operations

Step-by-Step Commands:

```
# Create user
aws iam create-user --user-name DevOps_Intern

# Create group
aws iam create-group --group-name DevTeam

# Add user to group
aws iam add-user-to-group --user-name DevOps_Intern --group-name DevTe
am

# Create policy (save the JSON from Task 1 as policy.json)
```

```
aws iam create-policy --policy-name EC2_T2Micro_Restrict --policy-documen
t file://policy.json

# Note the Policy ARN from output, then attach to group
aws iam attach-group-policy --group-name DevTeam --policy-arn "arn:aws:ia
m::ACCOUNT_ID:policy/EC2_T2Micro_Restrict"
```

Additional Exploration Tasks

1. Time-Based Access Policies

Example Policy for 9 AM–6 PM Access:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "*",
         "Resource": "*",
         "Condition": {
            "DateGreaterThan": {"aws:CurrentTime": "2024-01-01T09:00:00Z"},
            "DateLessThan": {"aws:CurrentTime": "2024-01-01T18:00:00Z"}
         }
      }
   ]
}
```

Key Points:

- Uses `aws:CurrentTime` condition key
- Times are in UTC (adjust accordingly)
- Can be combined with other conditions
- Useful for temporary access or contractor accounts

2. Permission Boundaries

Scenario Diagram:

```
IAM User (Developer)
 |
 ├─ Permissions (EC2 Full Access)
 |   ├─ ec2:*
 |   └─ related services
 |
 └─ Permission Boundary (Limit)
    ├─ Deny iam:*
    └─ Deny escalation actions
```

Key Differences:

Feature	Managed Policy	Permission Boundary
Purpose	Grants permissions	Limits maximum permissions
Inheritance	Applied directly or via group	Must be attached to user
Effect	Allows actions	Restricts what can be granted

3. Policy Writing Practice

S3 Read-Only Policy:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:Get*",
            "s3:List*"
         ],
         "Resource": "*"
      }
```

```
  ]
}
```

Deny RDS Deletion Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "rds:DeleteDBInstance",
        "rds:DeleteDBCluster"
      ],
      "Resource": "*"
    }
  ]
}
```

4. CLI Simulation

Dry-run Examples:

```
# Dry-run user creation
aws iam create-user --user-name TestUser --dry-run

# Expected output shows validation without execution
```

Key Flags:

- `-dry-run`: Validates command without execution
- `-no-cli-pager`: Disables output pagination
- `-query`: Filters output
- `-output`: Changes output format (json, text, table)

5. Real-World Policy Analysis

AmazonS3ReadOnlyAccess Analysis:

- Allows: ListAllBuckets, GetBucketLocation, GetObject, ListBucket
- Denies: All other S3 actions
- Use Case: Auditors, reporting tools

PowerUserAccess Analysis:

- Allows: Full access to AWS services except IAM
- Denies: IAM user/role management
- Use Case: Senior developers needing broad access

Comparison Table:

Policy	Services Covered	Typical User	Key Restrictions
S3ReadOnly	S3 only	Auditors	No write access
PowerUser	All except IAM	Team leads	No IAM changes
Admin	All services	Sysadmins	None