**Topic 3: Identity and Access Management (IAM)**

* [Topic 3: Identity and Access Management (IAM)](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#topic-3-identity-and-access-management-iam)
  + [Conventions](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#conventions)
  + [Lesson 3.1: Introduction to Identity and Access Management](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lesson-31-introduction-to-identity-and-access-management)
    - [Principle 3.1](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#principle-31)
    - [Practice 3.1](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#practice-31)
      * [Lab 3.1.1: IAM Role](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-311-iam-role)
      * [Lab 3.1.2: Customer Managed Policy](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-312-customer-managed-policy)
      * [Lab 3.1.3: Customer Managed Policy Re-Use](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-313-customer-managed-policy-re-use)
      * [Lab 3.1.4: AWS-Managed Policies](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-314-aws-managed-policies)
      * [Lab 3.1.5: Policy Simulator](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-315-policy-simulator)
      * [Lab 3.1.6: Clean Up](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-316-clean-up)
    - [Retrospective 3.1](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#retrospective-31)
      * [Question: Stack Outputs](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#question-stack-outputs)
      * [Task: Stack Outputs](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#task-stack-outputs)
  + [Lesson 3.2: Trust Relationships & Assuming Roles](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lesson-32-trust-relationships--assuming-roles)
    - [Principle 3.2](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#principle-32)
    - [Practice 3.2](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#practice-32)
      * [Lab 3.2.1: Trust Policy](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-321-trust-policy)
      * [Lab 3.2.2: Explore the assumed role](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-322-explore-the-assumed-role)
      * [Lab 3.2.3: Add privileges to the role](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-323-add-privileges-to-the-role)
      * [Lab 3.2.4: Clean up](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-324-clean-up)
    - [Retrospective 3.2](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#retrospective-32)
      * [Question: Inline vs Customer Managed Policies](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#question-inline-vs-customer-managed-policies)
      * [Question: Role Assumption](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#question-role-assumption)
  + [Lesson 3.3: Fine-Grained Controls With Policies](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lesson-33-fine-grained-controls-with-policies)
    - [Principle 3.3](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#principle-33)
    - [Practice 3.3](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#practice-33)
      * [Lab 3.3.1: Unrestricted access to a service](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-331-unrestricted-access-to-a-service)
      * [Lab 3.3.2: Resource restrictions](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-332-resource-restrictions)
      * [Lab 3.3.3: Conditional restrictions](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#lab-333-conditional-restrictions)
    - [Retrospective](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#retrospective)
      * [Question: Positive and Negative Tests](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#question-positive-and-negative-tests)
      * [Task: Positive and Negative Tests](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#task-positive-and-negative-tests)
      * [Question: Limiting Uploads](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#question-limiting-uploads)
      * [Task: Limiting Uploads](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#task-limiting-uploads)
  + [Further Reading](https://github.com/Jahidul2543/IzaanSchoolDevOps/blob/master/03-iam/README.md#further-reading)

**Conventions**

* All CloudFormation templates should be written in YAML
* Do NOT copy and paste CloudFormation templates from the Internet at large
* DO use the [CloudFormation documentation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-reference.html)
* DO utilize every link in this document; note how the AWS documentation is laid out
* DO use the AWS CLI for [CloudFormation](https://docs.aws.amazon.com/cli/latest/reference/cloudformation/index.html) and [IAM](https://docs.aws.amazon.com/cli/latest/reference/iam/index.html) (NOT the Console) unless otherwise specified.

**Lesson 3.1: Introduction to Identity and Access Management**

**Principle 3.1**

*Identity and Access Management (IAM) is the****authentication and authorization service****used to control access to virtually everything in AWS.*

**Practice 3.1**

IAM consists of a set of services and resources that allow individuals (and services) to authenticate with AWS and then authorizes those entities to perform specific activities with specific services. Like most authentication/authorization systems, IAM deals with the *concepts* of users, groups and permissions, but not necessarily those precise *entities*.

**Lab 3.1.1: IAM Role**

Create a CFN template that specifies an IAM Role.

* Provide values only for required attributes.
* Using inline Policies, give the Role read-only access to all IAM resources.
* Create the Stack.

Graphical user interface, text, application, email

Description automatically generatedA screenshot of a computer

Description automatically generated

* Use the [awscli](https://docs.aws.amazon.com/cli/latest/reference/iam/index.html) to query the IAM service twice:
  + List all the Roles

aws iam list-roles

Text

Description automatically generatedText

Description automatically generated

* + Describe the specific Role your Stack created.

aws iam get-role --role-name Lab3.1.1

Text

Description automatically generated

**Lab 3.1.2: Customer Managed Policy**

Update the template and the corresponding Stack to make the IAM Role's inline policy more generally usable:

* Convert the IAM Role's inline Policies array to a separate [customer managed policy](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies) resource.
* Attach the new resource to the IAM Role.
* Update the Stack using the modified template.

**Lab 3.1.3: Customer Managed Policy Re-Use**

Update the template further to demonstrate reuse of the customer managed policy:

* Add another IAM Role.
* Attach the customer managed policy resource to the new role.
* Be sure that you're not referencing an AWS managed policy in the role.
* **Add/Update the Description of the customer managed policy to indicate the re-use of the policy.**
* Update the Stack. *Did the stack update work?*
  + Query the stack to determine its state.
  + If the stack update was not successful, [troubleshoot and determine why](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-update-behaviors.html#update-replacement).

**Lab 3.1.4: AWS-Managed Policies**

Replace the customer managed policy with [AWS managed policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies).

* To both roles, replace the customer managed policy reference with the corresponding AWS managed policy granting Read permissions to the IAM service.
* To the second role, add an additional AWS managed policy to grant Read permissions to the EC2 service.
* Update the stack.

**Lab 3.1.5: Policy Simulator**

Read about the [AWS Policy Simulator](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html) tool and practice using it.

* Using the two roles in your stack, simulate the ability of each role to perform the following actions (using the AWS CLI):
  + iam:CreateRole
  + iam:ListRoles
  + iam:SimulatePrincipalPolicy
  + ec2:DescribeImages
  + ec2:RunInstances
  + ec2:DescribeSecurityGroups

**Lab 3.1.6: Clean Up**

Clean up after yourself by deleting the stack.

**Retrospective 3.1**

**Question: Stack Outputs**

*In Lab 3.1.5, you had to determine the Amazon resource Names (ARN) of the stack's two roles in order to pass those values to the CLI function. You probably used the AWS web console to get the ARN for each role. What could you have done to your CFN template to make that unnecessary?*

**Task: Stack Outputs**

Institute that change from the Question above. Recreate the stack as per Lab 3.1.5, and demonstrate how to retrieve the ARNs.

**Lesson 3.2: Trust Relationships & Assuming Roles**

**Principle 3.2**

*AWS service roles and other IAM principals can assume customer created roles, enabling a principle-of-least-privilege of permissions for AWS services and applications.*

**Practice 3.2**

An IAM Role has two kinds of policies. The first we've worked with already and this policy type (whether inline or managed) describes permissions the role has. The second is a trust policy, describing which AWS principles (services, roles and users) are allowed to masquerade as that role.

For example, an AWS Lambda Function requires an execution role that defines the permissions the function will have when it executes. To provide those permissions, the role must trust the AWS Lambda service to assume it, and this trust must be granted explicitly by the role.

In these labs, you will use your IAM User to assume roles in order to explore policies and permissions.

**Lab 3.2.1: Trust Policy**

Create a CFN template that creates an IAM Role and makes it possible for your User to assume that role.

* The role should reference the AWS managed policy ReadOnlyAccess.
* Add a trust relationship to the role that enables your specific IAM user to assume that role.
* Create the stack.
* Using the AWS CLI, assume that new role. If this fails, take note of the error you receive, diagnose the issue, and fix it.

*Hint: Instead of setting up a new profile in your ~/.aws/credentials file, use [aws sts assume-role](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp_use-resources.html" \l "using-temp-creds-sdk-cli). It's a valuable mechanism you'll use often through the API, and it's good to know how to do it from the CLI as well.*

**Lab 3.2.2: Explore the assumed role**

Test the capabilities of this new Role.

* Using the AWS CLI, assume that updated role and list the S3 buckets in the us-east-1 region.
* Acting as this role, try to create an S3 bucket using the AWS CLI.
  + Did it succeed? It should not have!
  + If it succeeded, troubleshoot how Read access allowed the role to create a bucket.

**Lab 3.2.3: Add privileges to the role**

Update the CFN template to give this role the ability to upload to S3 buckets.

* Create an S3 bucket.
* Using either an inline policy or an AWS managed policy, provide the role with S3 full access
* Update the stack.
* Assuming this role again, try to upload a text file to the bucket.
* If it failed, troubleshoot the error iteratively until the role is able to upload a file to the bucket.

**Lab 3.2.4: Clean up**

Clean up. Take the actions necessary to delete the stack.

**Retrospective 3.2**

**Question: Inline vs Customer Managed Policies**

*In the context of an AWS User or Role, what is the difference between an inline policy and a customer managed policy? What are the differences between a customer managed policy and an AWS managed policy?*

**Question: Role Assumption**

*When assuming a role, are the permissions of the initial principal mixed with those of the role being assumed? Describe how that could easily be demonstrated with both a*[*positive and negative testing*](https://www.guru99.com/positive-vs-negative-testing.html)*approach.*

**Lesson 3.3: Fine-Grained Controls With Policies**

**Principle 3.3**

*AWS policies can provide fine-grained access control to specific resources using specific conditions.*

**Practice 3.3**

So far we have only provided service-level IAM policy controls, but IAM policies generally should be more specific than that. For example, a service role for an application will generally only need read/write access to those specific resources that the application uses, and even then that resource might only be accessible under certain conditions. [Actions, Resources, and Condition Keys for AWS Services](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_actions-resources-contextkeys.html) introduces the topic. We'll be exploring Resource restrictions and Condition keys in this lesson.

Keep in mind that not all resource types support resource-level restrictions. See the Resource-level permissions information in [AWS Services That Work with IAM](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html) for details.

**Lab 3.3.1: Unrestricted access to a service**

Create a CFN template that generates two S3 buckets and a Role, and demonstrate you have full access to each bucket with this new role.

* Code a Role your User can assume with a customer managed policy that allows full access to the S3 service.
* Create the stack.
* As your User:
  + list the contents of your 2 new buckets
  + upload a file to each new bucket
* Assume the new role and repeat those two checks as that role.

**Lab 3.3.2: Resource restrictions**

Add a resource restriction to the role's policy that limits full access to the S3 service for just one of the two buckets and allows only read-only access to the other.

* Update the stack.
* Assume the new role and perform these steps as that role:
  + List the contents of your 2 new buckets.
  + Upload a file to each new bucket.

*Were there any errors? If so, take note of them.*

*What were the results you expected, based on the role's policy?*

**Lab 3.3.3: Conditional restrictions**

Add a conditional restriction to the role's policy. Provide a condition that grants list access only to objects that start with "lebowski/".

* Update the stack.
* Assume the new role and perform the remaining directives as that role.
* Try to list a file in the root of the available bucket
  + If it *worked*, fix your policy and update the stack until this fails.
* Try to list that same file but now with the proper object key prefix.
  + If it *doesn't work*, troubleshoot why and fix either the role's policy or the list command syntax until you are able to list a file.

**Retrospective**

**Question: Positive and Negative Tests**

*Were the tests you ran for resource- and condition-specific restrictions exhaustive? Did you consider additional*[*[positive and/or negative tests]*](https://smartbear.com/learn/automated-testing/negative-testing/)*that could be automated in order to confirm the permissions for the Role?*

**Task: Positive and Negative Tests**

Code at least one new positive and one new negative test.

**Question: Limiting Uploads**

*Is it possible to limit uploads of objects with a specific prefix (e.g. starting with "lebowski/") to an S3 bucket using IAM conditions? If not, how else could this be accomplished?*

**Task: Limiting Uploads**

Research and review the best method to limit uploads with a specific prefix to an S3 bucket.

**Further Reading**

* Read through the [IAM Best Practices](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html) and be sure you're familiar with the ideas there.