Instant hands-on su AWS policy

AWS policies are written in JSON format, so it's essential to know this as a premise.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::my-example-bucket"
    }
    ]
}
```

As we can see, policies are characterized by various keys that together form the policy. The first two keys that jump out are Version, which is the policy's version, and Statement, where each statement block starts with curly braces {}.

Statement is particularly important because it divides the different actions of allow and deny within a file, making the document more organized. It's not possible to have 1 action of deny and 1 of allow in the same statement.

```
{
    "Version": "2012-10-17",
    "id": "number123456",
    "Statement": [
    {
        "Sid": "Usabile come commento",
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::my-example-bucket"
    },
    {
        "Sid": "o come identificativo111222333",
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition":{
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.168.44.0/24",
                    "192.168.30.0/24"
                ]
            }
        }
    }
```

From another example, we can notice the two distinct statement blocks, just look at the curly braces { } in the same column as the word statement. Another thing we notice are **id** and **Sid**; we can say that these two new keys can be used either as identifiers or as descriptors, allowing us to insert comments on what that statement or policy does.
The symbol "*" means any option.

Effect: Can be "Allow" or "Deny" to permit or deny access.
Action: The operations that can be performed on AWS resources (e.g., s3:GetObject).
Resource: The specific AWS resources to which the action applies (e.g., arn:aws:s3:::your-bucket-name/*).
Condition: (Optional) Specifies the conditions under which the policy is in effect.
Among the keys we mentioned earlier, we also have the NOT versions of keys:

Action vs NotAction:

- Action specifies the permitted operations. Example: allow only the action s3:GetObject.
- NotAction specifies all operations except those listed. Example: allow all actions except s3:DeleteBucket.

Resource vs NotResource:

- Resource defines which resources the actions apply to. Example: apply rules to a specific S3 bucket.
- NotResource specifies all resources except those listed. Example: apply rules to all resources except a specific S3 bucket.

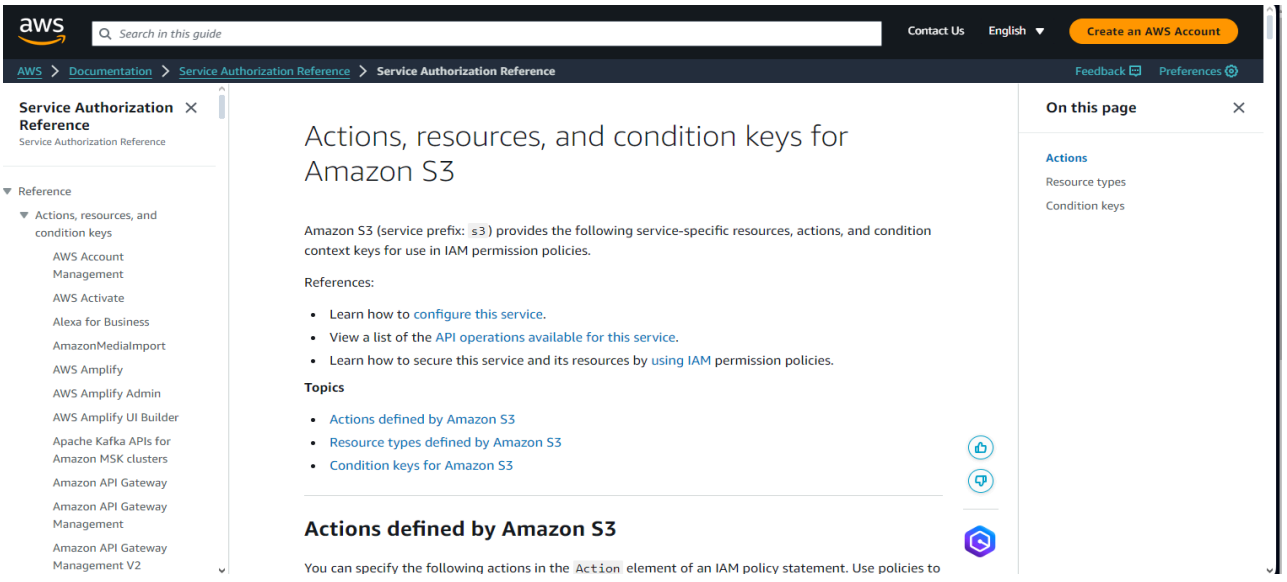
Principal vs NotPrincipal:

- Principal indicates who is authorized to perform the actions. Example: allow access only to a specific IAM user.
- NotPrincipal specifies who is excluded from the granted actions. Example: allow access to everyone except a specific IAM user.

Now, after briefly and roughly describing the various keys, let's search for the elements we need for the construction of our policy.

How to Use the Guide for Action, Resource, and Condition:

As previously mentioned, in this section, we insert all the actions we would like our policy to perform, and the best way to find them is on: https://docs.aws.amazon.com/service-authorization/latest/reference/reference_policies_actions-resources-contextkeys.html



On the left, we have all the services from which to extract the actions. Once we have found and opened the AWS service we wish to use for our policy, the page will consist of three sections: **Actions, Resource types, and Condition keys.**

Actions, Resource Types, Condition, and Variables:

Here we are at the beginning of the page where the prefix will be indicated: Amazon S3 (service prefix: s3) provides the keys for use in IAM permission policies.

This prefix should be used at the beginning of every action command we find in the table below, so the formula is:`"Action": "{prefix}:{ActionComand}"`

example: `"Action": "s3:ListAllMyBuckets"`

The table consists of:

Actions	Description	Access level	Resource types (*required)	Condition keys	Dependent actions

1 Actions

"Actions" specify the operations that can be performed on an AWS resource. For example, s3:PutObject is an action that allows uploading a file to an S3 bucket. These actions are what are actually granted or denied in an IAM policy.

2 Description

"Description" provides an explanation of the action.

3 Access Level

"Access Level" categorizes actions based on the type of access they grant:

- List: Allows listing resources or information about them.
- Read: Allows reading contents and attributes of resources.
- Write: Allows creating, modifying, or deleting resources.
- Permissions management: Allows managing access permissions to resources.
- Tagging: Allows assigning or removing tags from resources.

4. *Resource Types (required)

The 'Resource Types' indicate which types of resources an action can be performed on. Some actions can be performed on multiple types of resources, while others are specific to a single type. The symbol " indicates that the action requires a specific resource to operate on. As we can see from the example on the left, for actions such as CreateVirtualMFADevice, an mfa* resource is required, whereas for actions like EnableMFADevice, a user* resource is required. Also, for resources, there is a table which includes elements such as: Condition type, ARN, and Condition key."

```
{
    "Sid": "AllowManageOwnMFADevices",
    "Effect": "Allow",
    "Action": [
        "iam:CreateVirtualMFADevice",
        "iam:DeleteVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:ResyncMFADevice"
    ],
    "Resource": [
        "arn:aws:iam::123456789012:mfa/${aws:username}",
        "arn:aws:iam::123456789012:user/${aws:username}"
    ],
```

Resource types	ARN	Condition keys
mfa	arn:${Partition}:iam::${Account}:mfa/${MfaTokenIdWithPath}	aws:ResourceTag/${TagKey}
user	arn:${Partition}:iam::${Account}:user/${UserNameWithPath}	aws:ResourceTag/${TagKey} iam:ResourceTag/${TagKey}

4.5 Variables

Variables are a concept that lies between condition keys and resources. Looking at the resource table indicated above, the ARN identifying the specific resource is composed of variables that we must insert, such as **${Partition}, ${Account}, ${UsernameWithPath},** etc.
${Partition}: typically replaced with AWS unless we are in China or a government entity
Other variables can be specified by us or be assumed from other variables (condition key).

```
"Resource": [
    "arn:aws:iam::123456789012:user/${aws:username}"
    "arn:aws:iam::${aws:PrincipalAccount}:user/${aws:username}"
```

These two code strings can indicate the same thing if executed from the **account 123456789012**. Here, we are working on ${Account} as we can notice. And to find the "table" that contains all the variables, we need to look for the GLOBAL CONDITION KEYS: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html#condition-keys-principalaccount

5. Condition e Condition Keys

Conditions are the tool that allows us to impose conditions before executing certain "Actions." The formula is:

```
"Condition" : { "{condition-operator}" : { "{condition-key}" : "{condition-value}" }}
```

Condition Operators:

Found in https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html
These describe the logic of how the operator should read and treat the value of such a condition, among the most common we have:
StringEquals,StringLike,Bool,IpAddress,NotIpAddress ecc…

Condition keys:

"Condition Keys" allow defining the conditions under which the action is allowed or denied, and as previously indicated, we can say there are two types: **GLOBAL CONDITION KEY** and **Resource Condition Key**. **GLOBAL CONDITION KEYS** are conditions we can use on any resource and are used for security controls on user/account, role, network, request type, resource attributes, etc.

Resource Condition Key:

Are conditions specifically usable for that resource, which allow us to make specific checks on that resource, as in the example →

where we use s3:prefix that we can use in this specific action that has that specific resource with its own Resource condition key. Therefore, users can only view their prefix in the example bucket.

6. Dependent Actions

"Dependent Actions" are actions that must be allowed for the principal action to be executable. For example, to perform an action that reads data from an S3 bucket, it may also be necessary to have permission to perform an action that lists the buckets (such as s3:ListBucket), depending on the specific policy configuration.