IAM - Policies

What is a Policy?

A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents.

JSON stands for "JavaScript Object Notation." It is a lightweight data-interchange format.

The JSON file seen below contains the AdministratorAccess policy that we used when creating the first Admin user.

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

This Policy allows (Effect:"Allow") the user to take any action(Action:"*") on all of the resources (Resource: "*")

"*" refers to any/all in JSON format.

IAM Policy - JSON Identifiers

The keywords that make up the policy document are briefly described below. All policy components described below are mandatory keys for a policy. Apart from these, there are different keys that we can optionally use, but these are highly advance issues and are currently outside the scope of our topic.

We will use the AdministratorAccess policy again to describe the keywords.

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

Version:

```
"Version": "2012-10-17"
```

- Specifies the version of the policy document.
- The version in the example is the current version.

Statement Block:

```
"Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
```

- It is the basic building block of Policy and mandatory to be included in the policy document.
- A policy can have multiple statements.
- As shown in the example, it contains the **Effect**, **Actions** and **Resources** keys respectively.

Effect:

```
"Effect": "Allow"
```

- It determines what the statement actually does.
- It gets only the **Allow** or **Deny** values.
- In the example, the **Allow** value is assigned to the **Effect** key.

Actions:

```
"Action": "*"
```

- It determines what actions the identity can perform or not.
- In other words, according to the value of the Effect key, it refers to what actions are allowed or not.
- **"*"** is a special character and when assigned to **Action** key, it covers all or every action.

Resource:

```
"Resource": "*"
```

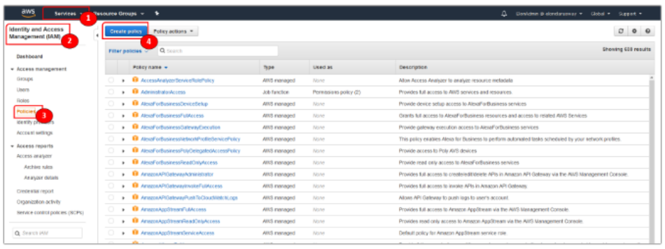
- It refers to the resources covered by the statements.
- In other words, it explains **in which AWS resources** the **Statement** will perform the operations specified in the **Actions** key.
- **"*"** is a special character and when assigned to the **Resource** key, it covers all the resources.

To summarize, the **AdministratorAccess** policy; is a policy that **allows** the identity to apply **all the actions** to **all AWS resources**.

Creating IAM Policies

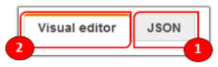
Now we will take a look at what actions we will take in the AWS Management Console to create an IAM policy.

- Sign in to the AWS Management Console.
- Open the **IAM** page using the **Services** tab from the menu bar.
- Click the **Policies** link from the menu on the left.
- If you have not created a policy before, the page will return as below.

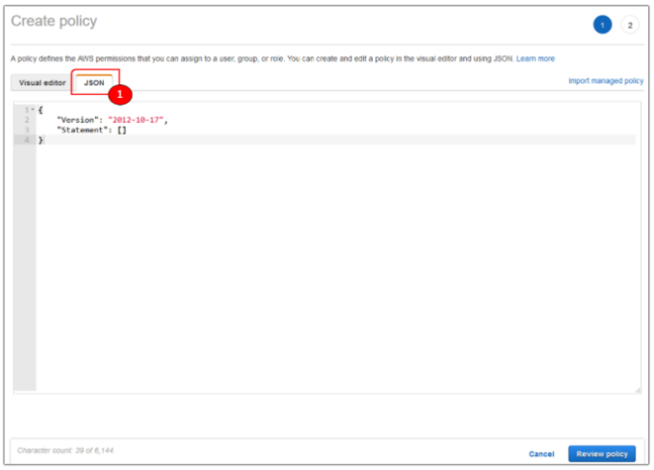


Policy name	Type	Used as	Description
AccessAnalyzerServiceRolePolicy	AWS managed	None	Allow Access Analyzer to analyze resource metadata
AdministratorAccess	Job function	Permissions policy (2)	Provides full access to AWS services and resources.
AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaForBusiness services
AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness resources and access to related AWS Services
AlexaForBusinessGatewayExecution	AWS managed	None	Provide gateway execution access to AlexaForBusiness services
AlexaForBusinessNetworkProfileServicePolicy	AWS managed	None	This policy enables Alexa for Business to perform automated tasks scheduled by your network profiles.
AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None	Provide access to Poly AWS devices
AlexaForBusinessReadOnlyAccess	AWS managed	None	Provide read only access to AlexaForBusiness services
AmazonAPIGatewayAdministrator	AWS managed	None	Provides full access to create/edit/delete APIs in Amazon API Gateway via the AWS Management Console.
AmazonAPIGatewayInvokeFullAccess	AWS managed	None	Provides full access to invoke APIs in Amazon API Gateway.
AmazonAPIGatewayPushToCloudWatchLogs	AWS managed	None	Allows API Gateway to push logs to user's account.
AmazonAppStreamFullAccess	AWS managed	None	Provides full access to Amazon AppStream via the AWS Management Console.
AmazonAppStreamReadOnlyAccess	AWS managed	None	Provides read only access to Amazon AppStream via the AWS Management Console.
AmazonAppStreamServiceAccess	AWS managed	None	Default policy for Amazon AppStream service role.

Let's see how to create our first policy by clicking the **"Create policy"** tab at the top of the page.



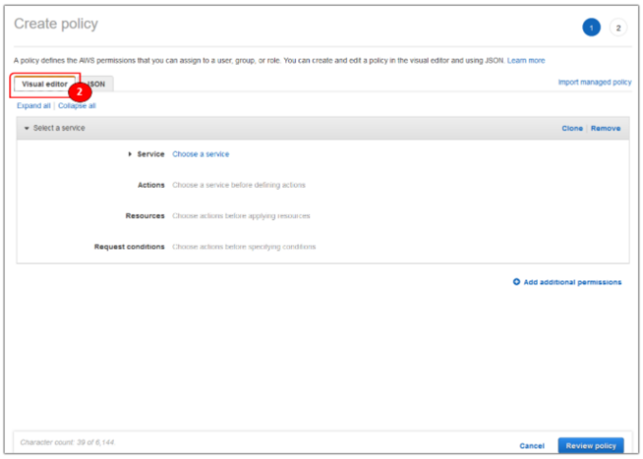
As you can see, we can use one of two different methods to create IAM Policy which are JSON and Visual Editor options.



A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

```
{
    "Version": "2012-10-17",
    "Statement": []
}
```

Character count: 39 of 6,144



Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor | JSON | Import managed policy

Expand all | Collapse all

Select a service | Clone | Remove

Service: Choose a service

Actions: Choose a service before defining actions

Resources: Choose actions before applying resources

Request conditions: Choose actions before specifying conditions

Add additional permissions

Character count: 39 of 6,144

Cancel | Review policy

We will learn these methods by applying them in the following lessons, respectively.

Creating IAM Policies - JSON

We will now see how to create a slightly more detailed JSON file policy than the AdministrativeAccess policy we saw before.

In this example, we will create a policy that allows users to change their own passwords. We will use 2 different statements.

- The first one is the statement that will allow the user to access the Account Password Policy.
- The second statement will allow changing the password.

Version:

- We are specifying the current version in AWS.
- We should not forget to start creating the policy with a curly brace and separate each key-value pairs with a comma.

```
{
    "Version": "2012-10-17",
}
```

Statement Block and Keys:

- We create the statement block where we will create 2 different statements.
- Both statements will include the Effect, Actions and Resource keys and their values.
- So let's write these keys first, then assign the values.

```
{
    {
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "",
    "Action": "",
    "Resource": ""
  },
  {
    "Effect": "",
    "Action": "",
    "Resource": ""
    }
  ]
}
```

First Statement:

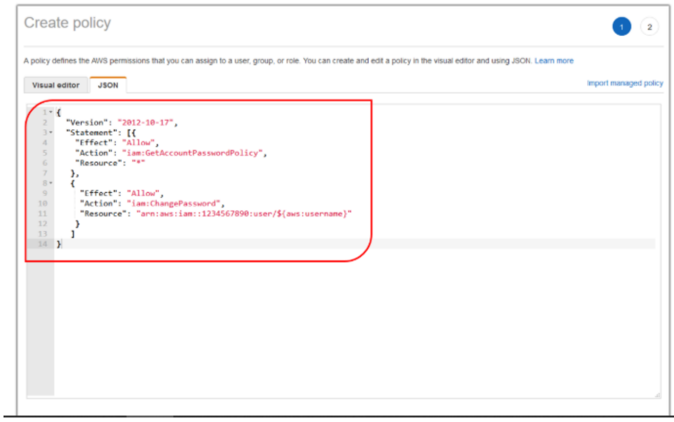
- The statement will allow the action to be taken. So we have to assign the value of **Allow** to the **Effect** key.
- As an **Action**, access to the password policy in the IAM service will be defined. So after writing the IAM service, we use colons and then we define action as **"iam:GetAccountPasswordPolicy"**.
- Finally, as **Resource**, we use the special character **'*'** to express all.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:GetAccountPasswordPolicy",
    "Resource": "*"
  },
  {
    "Effect": "",
    "Action": "",
    "Resource": ""
    }
  ]
}
```

Second Statement:

- The second statement will also allow the action to be taken. So we have to assign the value of **Allow** to the **Effect** key again.
- The password change action in the IAM service should be defined as **Action** like **"iam:ChangePassword".**
- Finally, we determine the source on which the user will perform this action.
- We should define this part to include the username so that the user can only access his/her own resource: **"arn:aws:iam::1234567890:user/${aws:username}"**

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:GetAccountPasswordPolicy",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "iam:ChangePassword",
    "Resource": "arn:aws:iam::1234567890:user/${aws:username}"
    }
  ]
}
```



Create policy

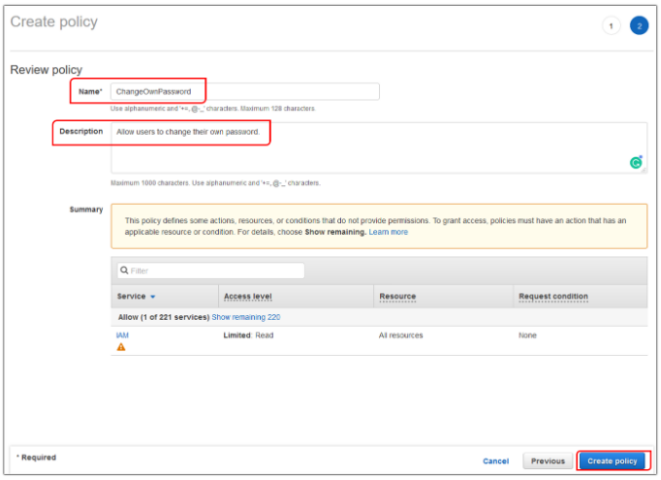
1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor | JSON | Import managed policy

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:GetAccountPasswordPolicy",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "iam:ChangePassword",
    "Resource": "arn:aws:iam::1234567890:user/${aws:username}"
  }
  ]
}
```

- Then go to the review page by pressing the **Review Policy** tab.



Create policy

1 2

Review policy

Name*: ChangeOwnPassword

Use alphanumeric and '+=,.@-_' characters. Maximum 128 characters.

Description: Allow users to change their own password.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Summary

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose **Show remaining.** Learn more

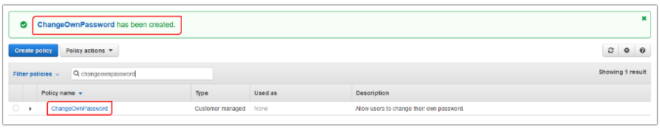
Service	Access level	Resource	Request condition
Allow (1 of 221 services) Show remaining 220			
IAM	Limited: Read	All resources	None

* Required

Cancel | Previous | Create policy

In this page;

- First, you have to define a policy **name**.
- Then, you can define a **description** for the policy. This is an optional but recommended step.
- Finally, select **Create policy** tab to create your first policy.



ChangeOwnPassword has been created.

Create policy | Policy actions

Filter policies | changeownpassword | Showing 1 result

Policy name	Type	Used as	Description
ChangeOwnPassword	Customer managed	None	Allow users to change their own password.

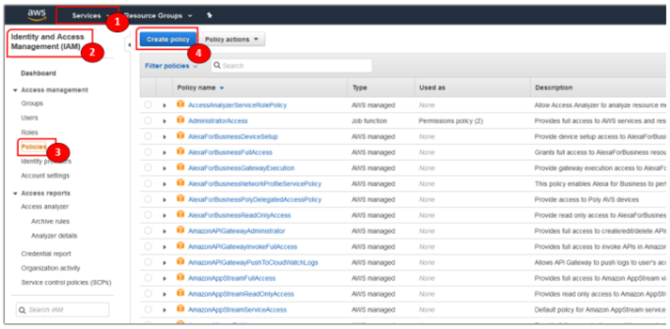
Congratulations! We have created our first policy in line with the objectives we originally determined.

- As you can see, it is very easy to understand and apply JSON syntax with mandatory key-value pairs to prepare the policy.
- However, it requires slightly more extensive AWS knowledge to determine the values to be assigned to the Action and Resource keys. As you learn more about AWS architecture and resources, your competencies in this area will also improve.

Creating IAM Policies - Visual Editor

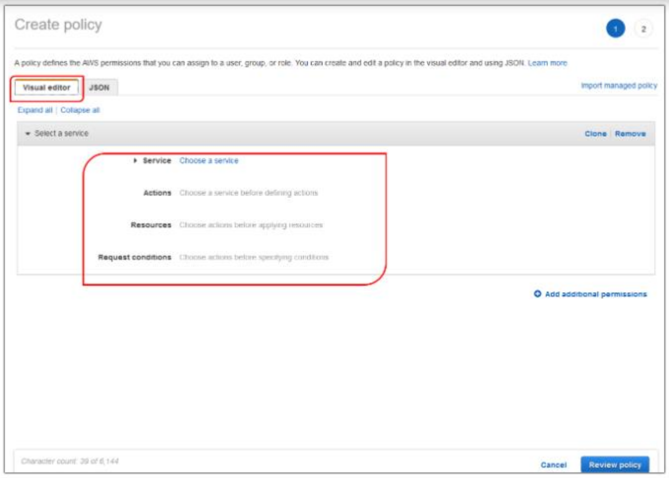
Now we will see how to create a policy using a visual editor.

- Sign in to the AWS Management Console.
- Open the **IAM** page using the **Services** tab from the menu bar.
- Click the **Policies** link from the menu on the left.
- Click **Create policy** tab on top of the page.



Policy name	Type	Used as	Description
AccessAnalyzerServiceRolePolicy	AWS managed	None	Allow Access Analyzer to analyze resource m
AdministratorAccess	Job function	Permissions policy (2)	Provides full access to AWS services and res
AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaForBusi
AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness resou
AlexaForBusinessGatewayExecution	AWS managed	None	Provide gateway execution access to AlexaFo
AlexaForBusinessNetworkProfileServicePolicy	AWS managed	None	This policy enables Alexa for Business to perf
AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None	Provide access to Poly AVS devices
AlexaForBusinessReadOnlyAccess	AWS managed	None	Provide read only access to AlexaForBusines
AmazonAPIGatewayAdministrator	AWS managed	None	Provides full access to create/edit/delete APIs
AmazonAPIGatewayInvokeFullAccess	AWS managed	None	Provides full access to invoke APIs in Amazon
AmazonAPIGatewayPushToCloudWatchLogs	AWS managed	None	Allows API Gateway to push logs to user's ac
AmazonAppStreamFullAccess	AWS managed	None	Provides full access to Amazon AppStream vi
AmazonAppStreamReadOnlyAccess	AWS managed	None	Provides read only access to Amazon AppStr
AmazonAppStreamServiceAccess	AWS managed	None	Default policy for Amazon AppStream service

The page that we will create a policy via visual editor will be displayed as below.

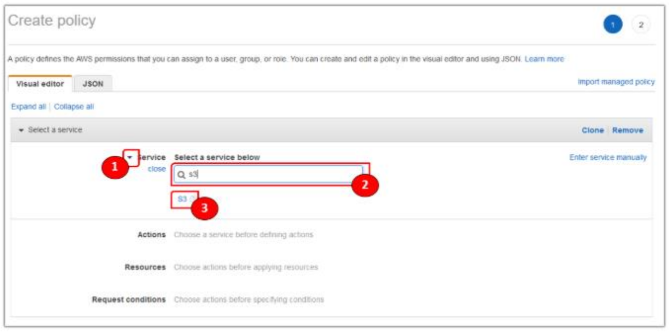


A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Service: Choose a service
Actions: Choose a service before defining actions
Resources: Choose actions before applying resources
Request conditions: Choose actions before specifying conditions

Character count: 39 of 6,144

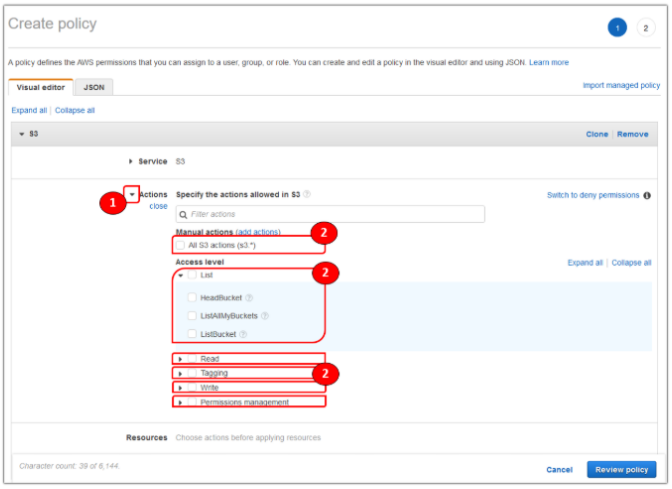
- First of all, we need to choose the service we want to authorize.
- Let's click on the Service tab and type S3 in the search bar and then select it.



A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Actions: Choose a service before defining actions
Resources: Choose actions before applying resources
Request conditions: Choose actions before specifying conditions

- So here we determine that this policy will be related to S3 service.
- Now, we should determine which actions we will allow for this service.



A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Service: S3

Actions: Specify the actions allowed in S3

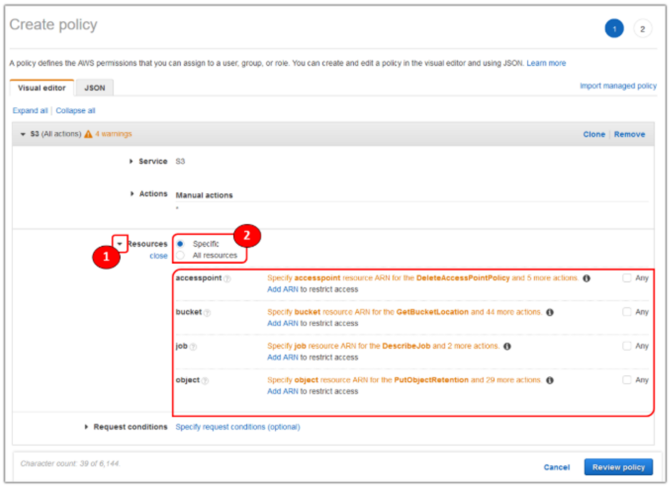
Manual actions (add actions)
- All S3 actions (s3:*)

Access level
- List
  - HeadBucket
  - ListAllMyBuckets
  - ListBucket
- Read
- Tagging
- Write
- Permissions management

Resources: Choose actions before applying resources

Character count: 39 of 6,144

- Firstly, click the **Actions** tab.
- Secondly, choose which actions or access levels to be allowed. You can select **all** option to allow all actions, or you can specify access levels such as list, read, write or you can select each component below these access levels one by one.
- For this example, let's select **All S3 actions (s3: *)** which means allowing all actions in the S3 service.
- Now, we need to determine on which resources the action we provide in the S3 service will be used. To do this, click the **Resources** tab.



A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

S3 (All actions) 4 warnings

Service: S3

Actions: Manual actions
*

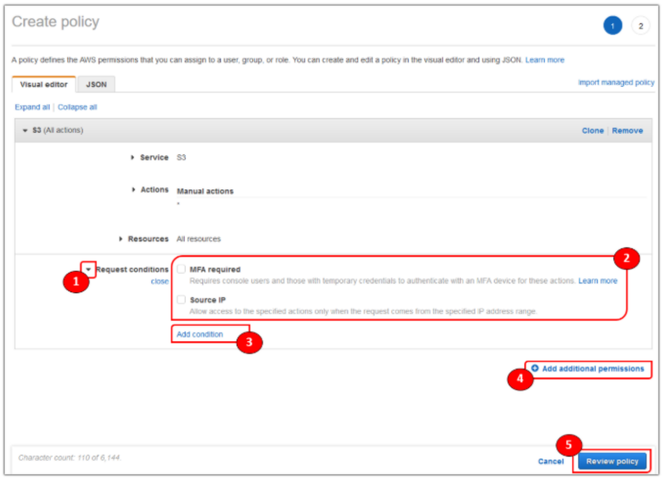
Resources: [✓ Specific] All resources

accesspoint	Specify **accesspoint** resource ARN for the **DeleteAccessPointPolicy** and 5 more actions. Add ARN to restrict access	Any
bucket	Specify **bucket** resource ARN for the **GetBucketLocation** and 44 more actions. Add ARN to restrict access	Any
job	Specify **job** resource ARN for the **DescribeJob** and 2 more actions. Add ARN to restrict access	Any
object	Specify **object** resource ARN for the **PutObjectRetention** and 29 more actions. Add ARN to restrict access	Any

Request conditions: Specify request conditions (optional)

Character count: 39 of 6,144

- In the **Resources** section, as in the **Actions** section, we can either select all resources or specify specific resources one by one.
- For this example, let's select the **All resources** option which means allowing all actions in the S3 service on all resources.
- The **Service, Actions, Resources** sections were mandatory sections that we had to determine on this page.
- Now let's move on to the **Request conditions** section that is the last section on this page, which is also optional.



A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

S3 (All actions)

Service: S3

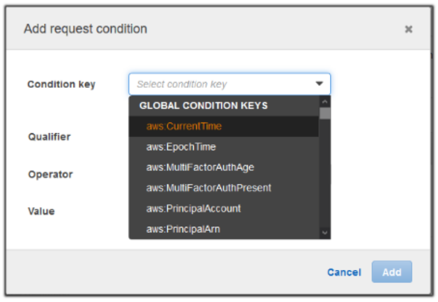
Actions: Manual actions
*

Resources: All resources

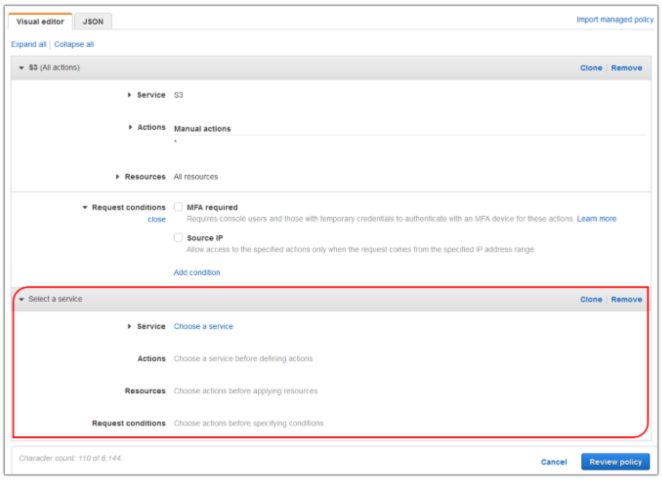
Request conditions:
- MFA required
  Requires console users and those with temporary credentials to authenticate with an MFA device for these actions. Learn more
- Source IP
  Allow access to the specified actions only when the request comes from the specified IP address range.

Character count: 110 of 6,144

- Here, AWS offers 2 ready-made options: **MFA required** or **Source IP**.
- If we select the **MFA required** option, the user to whom we assign this policy cannot use the powers we give with this policy if MFA is not defined.
- With the **Source IP** option, we can ensure that the user can access these services only over specified IPs. For example, if we only want users to access this information from company computers, we can create an IP Range using this option.
- For this example, let's not select either.
- Here, if we want, we can also determine a new condition as below by clicking on the **Add condition** tab or a new permission statement by clicking **Add additional permissions** tab.



Add condition



Visual editor | JSON | Import managed policy

Expand all | Collapse all

S3 (All actions) — Clone | Remove

Service: S3

Actions: Manual actions

*

Resources: All resources

Request conditions (close): MFA required — Requires console users and those with temporary credentials to authenticate with an MFA device for these actions. Learn more

Source IP — Allow access to the specified actions only when the request comes from the specified IP address range.

Add condition

Select a service — Clone | Remove

Service: Choose a service

Actions: Choose a service before defining actions

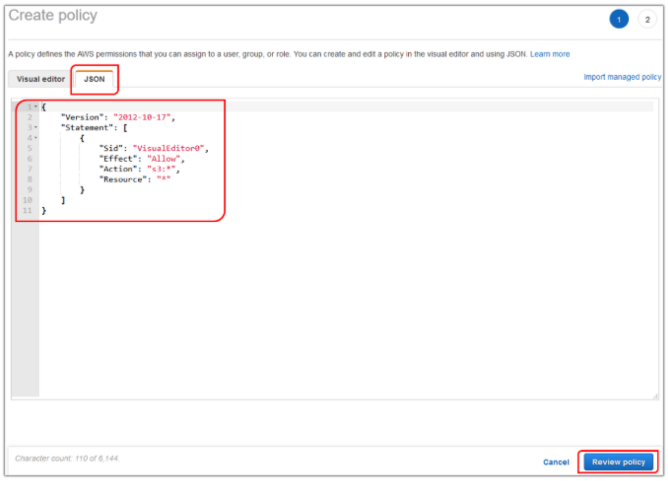
Resources: Choose actions before applying resources

Request conditions: Choose actions before specifying conditions

Character count: 110 of 6,144. — Cancel — Review policy

Add additional permissions

- At the same time, we can see the JSON version of the policy we prepared with the visual editor by clicking the JSON tab located at the top left of the page whenever we want.



Create policy (1, 2)

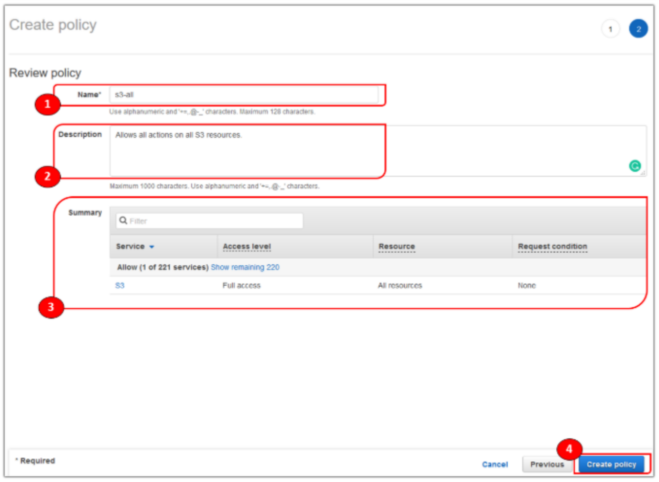
A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor | JSON | Import managed policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        }
    ]
}
```

Character count: 110 of 6,144. — Cancel — Review policy

Then, click the **Review policy** tab.



Create policy (1, 2)

Review policy

1 Name*: s3-all

Use alphanumeric and '+=,.@-_' characters. Maximum 128 characters.

2 Description: Allows all actions on all S3 resources.

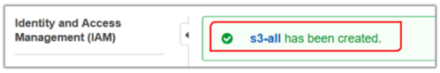
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

3 Summary

Service	Access level	Resource	Request condition
Allow (1 of 221 services) Show remaining 220			
S3	Full access	All resources	None

* Required — Cancel — Previous — 4 Create policy

- Define a name for the policy. This is required.
  You can optionally write a description.
- Review the policy.
- Finally, create the policy by clicking the **Create policy** tab.



Identity and Access Management (IAM)

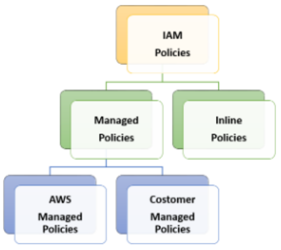
s3-all has been created.

Congratulations! We have successfully created a policy using the visual editor.

IAM Policy Types

Now that we have knowledge about policy formulation, let's get some more detailed information about these policy types.

IAM policies are basically two types: Managed and inline policies.



Let's look at the general definitions of these policy types.

Managed policies:
IAM managed policy is a standalone policy that can be attached to multiple entities (user, group of users, or role) in an AWS account. They can only be applied to entities and not to resources. There are two types of managed policies:

- **AWS managed policies:**

AWS managed policies are managed policies, created and managed by AWS. If you are new to using policies, AWS recommends that you start by using AWS managed policies.

- **Customer managed policies:**

Costomer managed policies are managed policies, created and managed by users (AWS costomers) in their AWS account. They provide more precise control over policies than AWS managed policies. You can create and edit an IAM policy in the visual editor or by creating the JSON policy document directly.

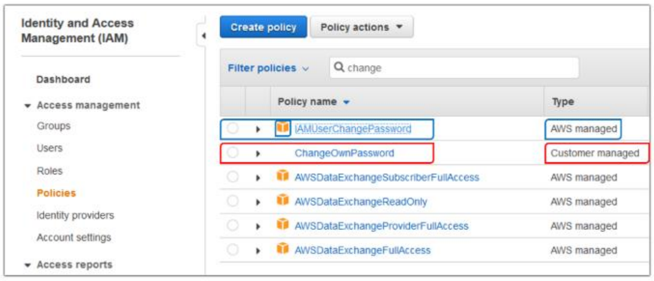
Inline policies:
Inline policies are policies that you create and manage and embedded directly into a single user, group, or role. In most cases, AWS does not recommend using inline policies. These policies are useful if the user wishes to maintain a strict one-to-one relationship between a policy and the main entity which it applies to. Therefore, the deletion of the entity or resource will also lead to the deletion of the inline policy.

Recap : Policy Types

Now let's try to understand the types of policies better by comparing the definition of policy types we learned with the processes we have done so far.

We've prepared two different policies so far: **ChangeOwnPassword**, and **s3-all**.
- The **ChangeOwnPassword** policy was created via **JSON**.
- The **s3-all** policy was created via **Visual editor**.
- They are both **Customer Managed Policy** and created by us.

Now let's look at the image below.



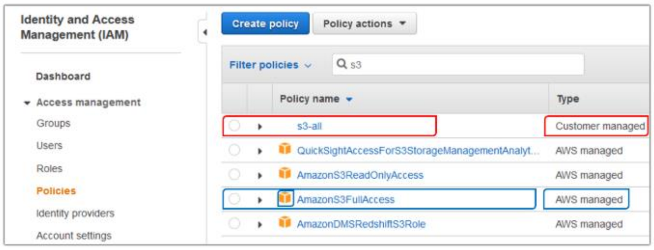
Identity and Access Management (IAM)

Dashboard — Access management: Groups, Users, Roles, Policies, Identity providers, Account settings — Access reports

Create policy | Policy actions

Filter policies: change

Policy name	Type
IAMUserChangePassword	AWS managed
ChangeOwnPassword	Customer managed
AWSDataExchangeSubscriberFullAccess	AWS managed
AWSDataExchangeReadOnly	AWS managed
AWSDataExchangeProviderFullAccess	AWS managed
AWSDataExchangeFullAccess	AWS managed



Identity and Access Management (IAM)

Dashboard — Access management: Groups, Users, Roles, Policies, Identity providers, Account settings

Create policy | Policy actions

Filter policies: s3

Policy name	Type
s3-all	Customer managed
QuickSightAccessForS3StorageManagementAnalyt...	AWS managed
AmazonS3ReadOnlyAccess	AWS managed
AmazonS3FullAccess	AWS managed
AmazonDMSRedshiftS3Role	AWS managed

When we look at the type column of the policies;

- We can see the policies that we have prepared are **Customer Managed Policies**.
- There are also AmazonS3FullAccess and IAMUserChangePassword policies which are framed in blue. You can see that they have an **AWS Managed Policy** type.
- If you select **ChangeOwnPassword** policy and **IAMUserChangePassword** policy one-by-one and look at JSON files, you can see they are nearly the same policies. The same is true for **AmazonS3FullAccess** and **s3-all** policies.

So what is the difference between them?

- Customer Managed Policies are created by us according to our needs. While preparing the policy, we can determine every detail as we want according to our needs and the characteristics of the conditions. So they give us elasticity and freedom. However, AWS architecture should be mastered in order to prepare comprehensive and specific detailed policies.
- On the other hand, AWS managed policies, as the name suggests, are policies created by AWS. In many cases, you can find an AWS managed policy that suits your needs. But of course, not in some cases.
- Therefore, it is recommended by AWS to use AWS Managed Policy, especially for beginners.

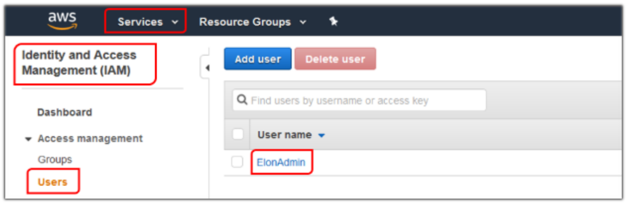
Well, so, what is the Inline Policy? Let's see What it is as an example in the next lesson.

IAM Inline Policies

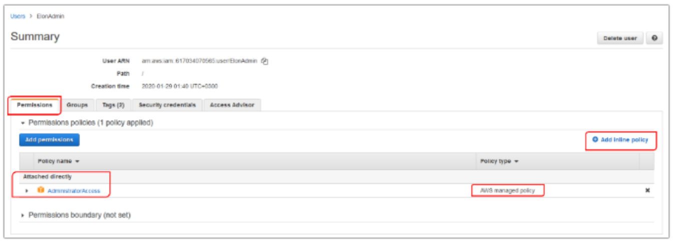
As we mentioned before, AWS IAM inline policies are created and managed by users and embedded directly into a single entity (user, group or role).

You can create a policy and embed it in an entity, either when you create the principal entity or later. Since we haven't created a group or role yet, we will create our inline policy example on the Admin user we created earlier.

- Sign in to the AWS Management Console.
- Open the **IAM** page using the **Services** tab from the menu bar.
- Click the **Users** link from the menu on the left.



- Then click the user name.



Users > ElonAdmin

Summary

User ARN arn:aws:iam::617034070565:user/ElonAdmin
Path /
Creation time 2020-01-29 01:40 UTC+0300

Permissions | Groups | Tags (3) | Security credentials | Access Advisor

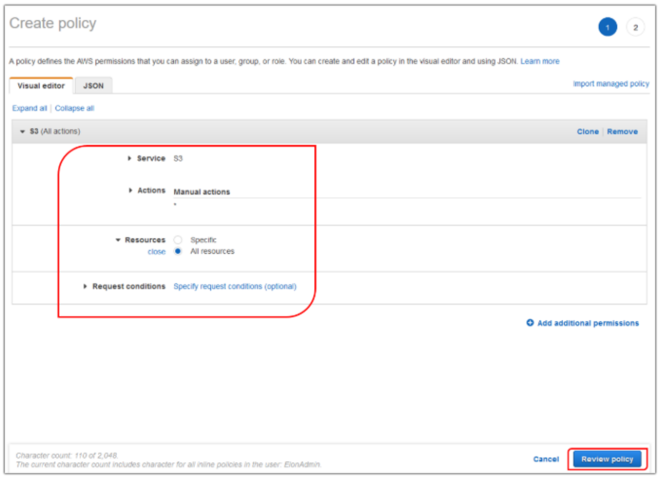
Permissions policies (1 policy applied)

Add permissions | Add inline policy

Policy name	Policy type
Attached directly	
AdministratorAccess	AWS managed policy

Permissions boundary (not set)

- Under the **Permissions** tab, there are policies owned by the user.
- As you can see, currently, there is only an Administrative Access policy that we assign to the user only when creating it.
- Then, click **Add inline policy** tab on the right of the page.



Create policy

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more

Visual editor | JSON | Import managed policy

Expand all | Collapse all

S3 (All actions) — Clone | Remove

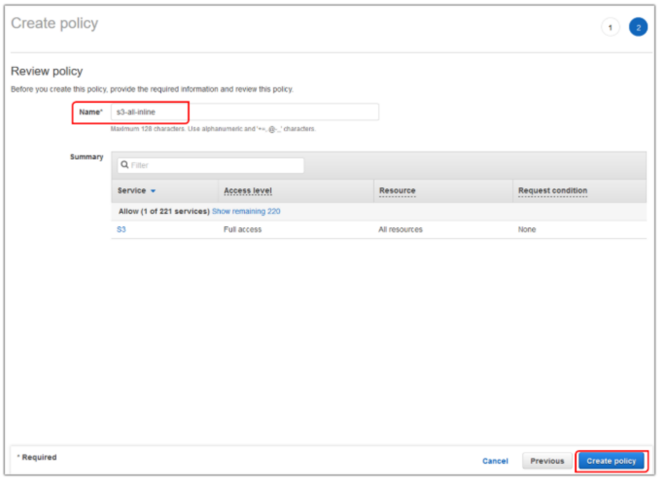
Service: S3
Actions: Manual actions *
Resources: ○ Specific ● All resources (close)
Request conditions: Specify request conditions (optional)

Add additional permissions

Character count: 110 of 2,048.
The current character count includes character for all inline policies in the user: ElonAdmin.

Cancel | Review policy

- Create the same **s3-all** policy that we created in Creating IAM Policies - Visual Editor lesson.
- Click **Review policy** tab.



Create policy

Review policy

Before you create this policy, provide the required information and review this policy.

Name* s3-all-inline
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.

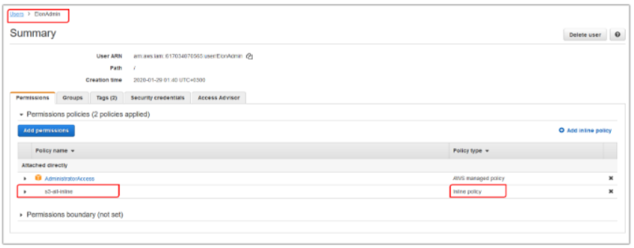
Summary

Service	Access level	Resource	Request condition
Allow (1 of 221 services) Show remaining 220			
S3	Full access	All resources	None

* Required

Cancel | Previous | Create policy

- This time, name it **s3-all-inline**.
- Click **Create policy** tab.



Users > ElonAdmin

Summary

User ARN arn:aws:iam::617034070565:user/ElonAdmin
Path /
Creation time 2020-01-29 01:40 UTC+0300

Permissions | Groups | Tags (2) | Security credentials | Access Advisor

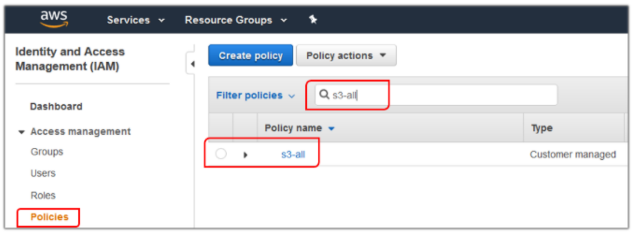
Permissions policies (2 policies applied)

Add permissions | Add inline policy

Policy name	Policy type
Attached directly	
AdministratorAccess	AWS managed policy
s3-all-inline	Inline policy

Permissions boundary (not set)

As seen in the image, the newly created policy will automatically attach to the user in the inline policy type.
Now let's click on the Policies tab on the left menu again.



Type **s3-all** into the policy search bar.

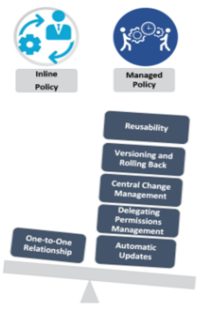
- As you will see, the newly created s3-all-inline policy does not appear.
- Because inline policies are not reusable. They are assigned to an entity (user, group, role) and exist only with that entity. You cannot reassign it to another entity later.
- If this entity is deleted, the inline policy will also be deleted.

Managed Policies vs Inline Policies

While setting identity permission in IAM, you need to decide whether to use an AWS-managed policy, a customer-managed policy or an inline policy. So:

- Which one is better and why?
- When to use managed policy or inline policy?

A managed policy and an inline policy are for different use cases. In most cases, AWS recommends to use managed policies instead of inline policies. Because they have more useful features and characteristics such as reusability, versioning, rolling back, etc.



Inline Policy
Managed Policy
Reusability
Versioning and Rolling Back
Central Change Management
Delegating Permissions Management
Automatic Updates
One-to-One Relationship

Managed policies:

Managed policies have characteristics as follows in AWS and those are the advantages of managed policies on inline policies:

- Reusability
- Central change management
- Versioning and rolling back
- Delegating permissions management
- Automatic updates for AWS managed policies

Inline policies:

Inline policies are useful if you want a direct one-to-one relationship between a policy and an entity.

By using an inline policy, you can not unintentionally attach the policy to the wrong entity.

Because the entity and the policy are part of each other, when you delete an entity, the policies embedded in the entity are deleted as well.

Job Function Policies

The AWS managed job function policies are designed to fit closely with common IT job functions.

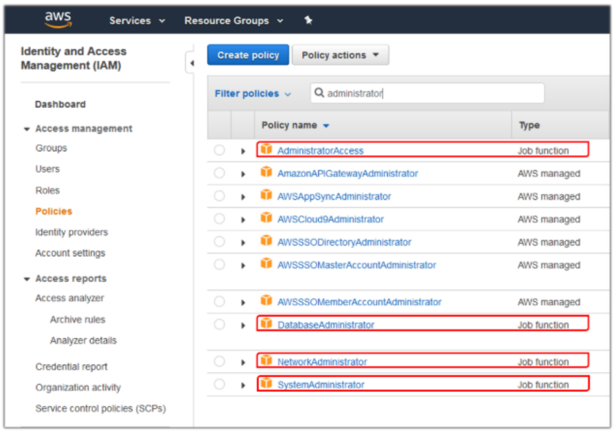
- Such policies can be used to quickly grant the necessary permissions for someone to carry out the expected job function.
- These policies consolidate permissions for many services into a single policy that's easier to work with than having permissions scattered across many policies.
- For many services, these policies combine permissions into a single policy that is easier to work with than getting permissions spread across many policies.
- You can attach these policies for job functions to any group, user, or role.
- These policies can not be updated by customers and are all controlled by AWS and are kept up to date as they are implemented by AWS to include support for new features and new capabilities.
- For example, the Network Administrator policy allows a user with the policy to pass a role named "flow-logs-vpc" to the Amazon CloudWatch service. CloudWatch uses that role to log and capture IP traffic for VPCs created by the user.

Managed policies in job function status are listed below:

- Administrator
- Billing
- Database Administrator
- Data Scientist
- Developer Power User
- Network Administrator
- Security Auditor
- Support User
- System Administrator
- View-Only User

Example Job Function Policy: AdministratorAccess

- This policy grants all actions for all the AWS services and all the account resources.
- We used this policy in the Admin user creation process we created from root account user.



administrator

Policy name	Type
AdministratorAccess	Job function
AmazonAPIGatewayAdministrator	AWS managed
AWSAppSyncAdministrator	AWS managed
AWSCloud9Administrator	AWS managed
AWSSSODirectoryAdministrator	AWS managed
AWSSSOMasterAccountAdministrator	AWS managed
AWSSSOMemberAccountAdministrator	AWS managed
DatabaseAdministrator	Job function
NetworkAdministrator	Job function
SystemAdministrator	Job function