AWS re:Invent

SEC305

How to Become an IAM Policy Ninja in 60 Minutes or Less

Jeff Wierer, Senior AWS IAM Manager

October 2015

amazon web services

What to expect from this session

- Know more about securing your AWS resources
- Get a deeper understanding of the policy language
- Tips and tricks for common use cases
- Debugging and testing policies
- Keep this a lively session via demos
  - Amazon S3
  - AWS Identity and Access Management (IAM)
  - Amazon EC2

Limit Amazon EC2 instance types

Demo

Limit Amazon EC2 instance types

Demo

- Goal: Limit a user from starting an instance unless the instance is `t1.*`, `t2.*`, `m3.*`
- Let's try to:
  - Create a managed policy that attempts to limit starting an EC2 instance except for these instance types.
  - Attach that policy to an IAM user.

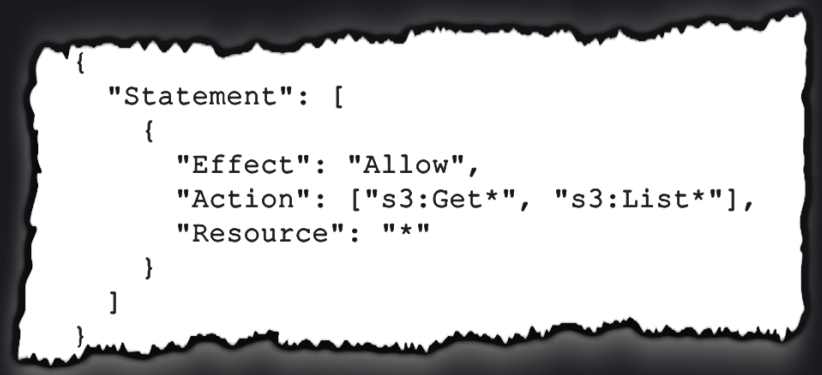
Limit Amazon EC2 instance types

Demo

- Goal: Limit a user from starting an instance unless the instance is `t1.*`, `t2.*`, `m3.*`
- Let's try to:
  - Create a managed policy that attempts to limit starting an EC2 instance except for these instance types.
  - Attach that policy to an IAM user.

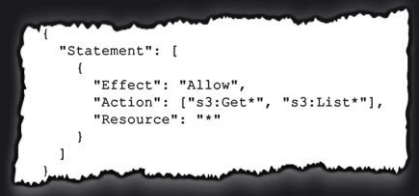
FAIL

The policy language



```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:Get*", "s3:List*"],
      "Resource": "*"
    }
  ]
}
```

The policy language



- Provides authorization
- Two facets:
  - **Specification**: *Defining* access policies
  - **Enforcement**: *Evaluating* policies

Policy specification basics

- ☑ JSON-formatted documents
- ☑ Contain a statement (permissions) that specifies:
  - Which actions a principal can perform
  - Which resources can be accessed

```
{
 "Statement":[{
   "Effect":"effect",
   "Principal":"principal",
   "Action":"action",
   "Resource":"arn",
   "Condition":{
     "condition":{
       "key":"value" }
     }
    }
   ]
 }
```

Principal
Action
Resource
Condition

You can have multiple statements and each statement is comprised of PARC.

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"
```

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}
```

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}

<!-- Individual IAM user -->
"Principal":"AWS":"arn:aws:iam::123456789012:user/username"
```

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}

<!-- Individual IAM user -->
"Principal":"AWS":"arn:aws:iam::123456789012:user/username"

<!-- Federated user (using web identity federation) -->
"Principal":{"Federated":"www.amazon.com"}
"Principal":{"Federated":"graph.facebook.com"}
"Principal":{"Federated":"accounts.google.com"}
```

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}

<!-- Individual IAM user -->
"Principal":"AWS":"arn:aws:iam::123456789012:user/username"

<!-- Federated user (using web identity federation) -->
"Principal":{"Federated":"www.amazon.com"}
"Principal":{"Federated":"graph.facebook.com"}
"Principal":{"Federated":"accounts.google.com"}

<!-- Specific role -->
"Principal":{"AWS":"arn:aws:iam::123456789012:role/rolename"}
```

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}

<!-- Individual IAM user -->
"Principal":"AWS":"arn:aws:iam::123456789012:user/username"

<!-- Federated user (using web identity federation) -->
"Principal":{"Federated":"www.amazon.com"}
"Principal":{"Federated":"graph.facebook.com"}
"Principal":{"Federated":"accounts.google.com"}

<!-- Specific role -->
"Principal":{"AWS":"arn:aws:iam::123456789012:role/rolename"}

<!-- Specific service -->
 "Principal":{"Service":"ec2.amazonaws.com"}
```

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}

<!-- Individual IAM user -->
"Principal":"AWS":"arn:aws:iam::123456789012:user/username"

<!-- Federated user (using web identity federation) -->
"Principal":{"Federated":"www.amazon.com"}
"Principal":{"Federated":"graph.facebook.com"}
"Principal":{"Federated":"accounts.google.com"}

<!-- Specific role -->
"Principal":{"AWS":"arn:aws:iam::123456789012:role/rolename"}

<!-- Specific service -->
 "Principal":{"Service":"ec2.amazonaws.com"}
```

Action – Examples

- Describes the type of access that should be allowed or denied
- You can find these in the docs or use the policy editor to get a drop-down list
- Statements must include either an `Action` or `NotAction` element

Action – Examples

- Describes the type of access that should be allowed or denied
- You can find these in the docs or use the policy editor to get a drop-down list
- Statements must include either an `Action` or `NotAction` element

```
<!-- EC2 action -->
"Action":"ec2:StartInstances"

<!-- IAM action -->
"Action":"iam:ChangePassword"

<!-- S3 action -->
"Action":"s3:GetObject"
```

Action – Examples

- Describes the type of access that should be allowed or denied
- You can find these in the docs or use the policy editor to get a drop-down list
- Statements must include either an `Action` or `NotAction` element

```
<!-- EC2 action -->
"Action":"ec2:StartInstances"

<!-- IAM action -->
"Action":"iam:ChangePassword"

<!-- S3 action -->
"Action":"s3:GetObject"

<!-- Specify multiple values for the Action element-->
"Action":["sqs:SendMessage","sqs:ReceiveMessage"]
```

Action – Examples

- Describes the type of access that should be allowed or denied
- You can find these in the docs or use the policy editor to get a drop-down list
- Statements must include either an `Action` or `NotAction` element

```
<!-- EC2 action -->
"Action":"ec2:StartInstances"

<!-- IAM action -->
"Action":"iam:ChangePassword"

<!-- S3 action -->
"Action":"s3:GetObject"

<!-- Specify multiple values for the Action element-->
"Action":["sqs:SendMessage","sqs:ReceiveMessage"]

<--Use wildcards (* or ?) as part of the action name.  This would cover Create/Delete/List/Update-->
"Action":"iam:*AccessKey*"
```

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let’s say you want to allow everything but IAM APIs

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let's say you want to allow everything but IAM APIs

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

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let's say you want to allow everything but IAM APIs

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

or

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let's say you want to allow everything but IAM APIs

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

or

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

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let's say you want to allow everything but IAM APIs

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


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

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let's say you want to allow everything but IAM APIs

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

This is not a **`Deny`**. A user could still have a separate policy that grants **`IAM:*`**

or

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

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let's say you want to allow everything but IAM APIs

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

This is not a **`Deny`**. A user could still have a separate policy that grants **`IAM:*`**

or

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

If you want to prevent the user from ever being able to call IAM APIs, use an explicit deny.

Resource – Examples

- The object or objects that are being requested
- Statements must include either a `Resource` or a `NotResource` element

Resource – Examples

- The object or objects that are being requested
- Statements must include either a `Resource` or a `NotResource` element

```
<-- S3 Bucket -->
"Resource":"arn:aws:s3:::my_corporate_bucket/*"

<-- SQS queue-->
"Resource":"arn:aws:sqs:us-west-2:123456789012:queue1"

<-- Multiple DynamoDB tables -->
"Resource":["arn:aws:dynamodb:us-west-2:123456789012:table/books_table",
           "arn:aws:dynamodb:us-west-2:123456789012:table/magazines_table"]

<-- All EC2 instances for an account in a region -->
 "Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*"
```

Conditions

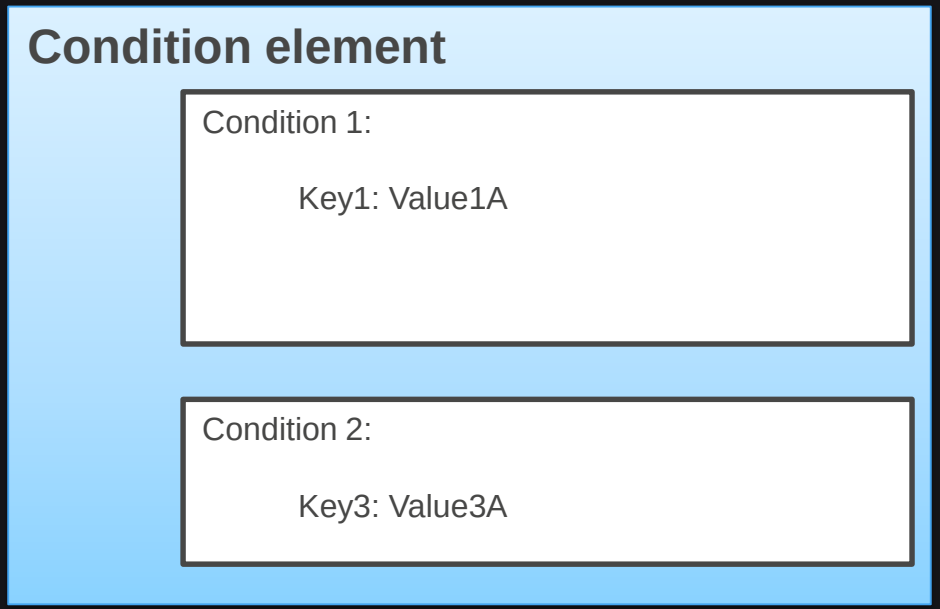
- Optional criteria that must evaluate to true for the policy to evaluate as true

  (ex: restrict to an IP address range)

Condition element

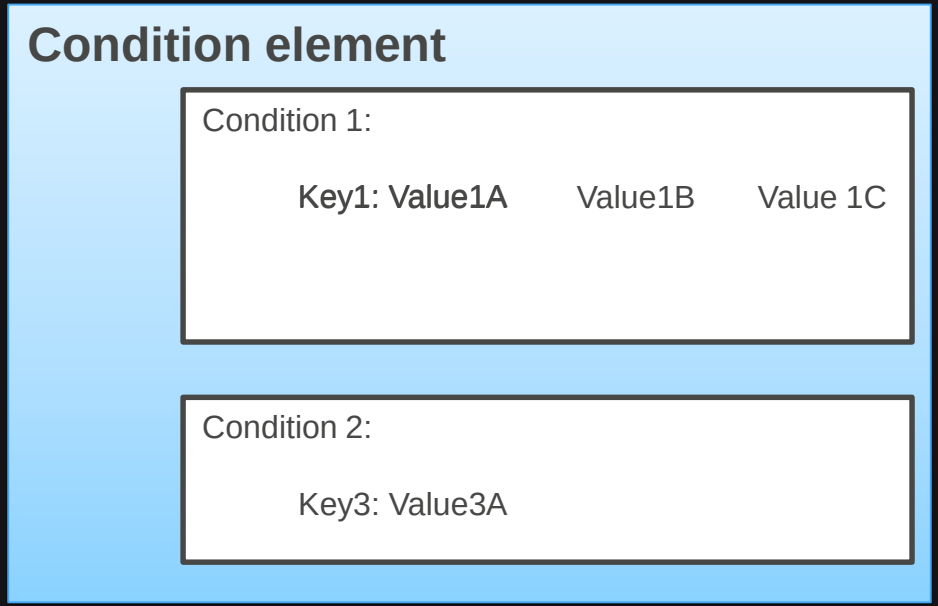
Conditions

- Optional criteria that must evaluate to true for the policy to evaluate as true
    - Ex: restrict to an IP address range
- Can contain multiple conditions



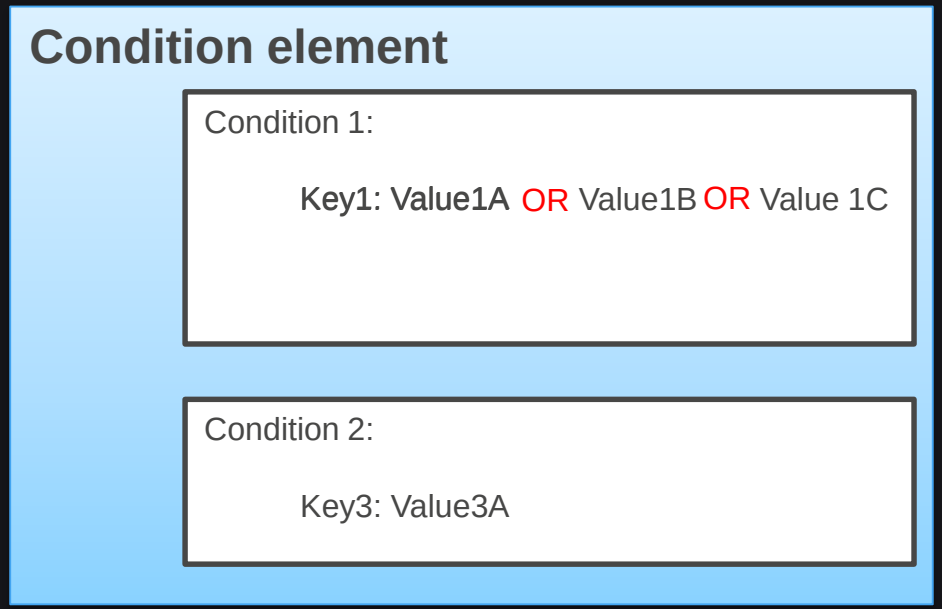
Conditions

- Optional criteria that must evaluate to true for the policy to evaluate as true
    - Ex: restrict to an IP address range
- Can contain multiple conditions
- Condition keys can contain multiple values



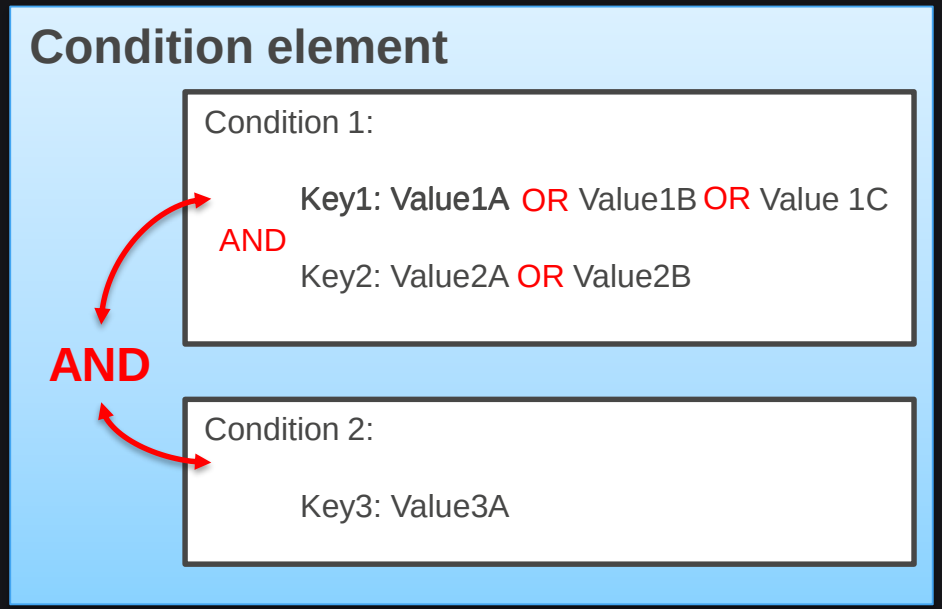
Conditions

- Optional criteria that must evaluate to true for the policy to evaluate as true
    - Ex: restrict to an IP address range
- Can contain multiple conditions
- Condition keys can contain multiple values
- If a single condition includes multiple values for one key, the condition is evaluated using logical OR
-



Conditions

- Optional criteria that must evaluate to true for the policy to evaluate as true
    - Ex: restrict to an IP address range
- Can contain multiple conditions
- Condition keys can contain multiple values
- If a single condition includes multiple values for one key, the condition is evaluated using logical OR
- Multiple conditions (or multiple keys in a single condition): the conditions are evaluated using logical AND



Condition example

What if you wanted to restrict access to a time frame and IP address range?

```
"Condition" :  {
   "DateGreaterThan" : {"aws:CurrentTime" : "2015-10-08T12:00:00Z"},
   "DateLessThan": {"aws:CurrentTime" : "2015-10-08T15:00:00Z"},
   "IpAddress" : {"aws:SourceIp" : ["192.0.2.0/24", "203.0.113.0/24"]}
}
```

AND

OR

- Allows a user to access a resource under the following conditions:
  - The time is after 12:00 P.M. on 10/8/2015 AND
  - The time is before 3:00 P.M. on 10/8/2015 AND
  - The request comes from an IP address in the 192.0.2.0 /24 OR 203.0.113.0 /24 range

All of these conditions must be met in order for the statement to evaluate to TRUE.

Policy variables

`${aws:userid}`

Policy variables

`${aws:userid}`

- Predefined variables based on service request context
  - Existing keys (`aws:SourceIP`, `aws:CurrentTime`, etc.)
  - Principal-specific keys (`aws:username, aws:userid, aws:principaltype`)
  - Provider-specific keys (`graph.facebook.com:id, www.amazon.com:user_id`)
  - SAML keys (`saml:aud, saml:iss`)
  - See documentation for service-specific variables
- Benefits
  - Simplifies policy management
  - Reduces the need for hard-coded, user-specific policies
- Use cases we'll look at
  - Easily set up user access to "home folder" in S3
  - Limit access to specific EC2 resources

The anatomy of a policy with variables

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Action": ["s3:ListBucket"],
     "Resource": ["arn:aws:s3:::myBucket"],
     "Condition":
             {"StringLike":
                {"s3:prefix":["home/${aws:username}/*"]}
       }
    },
    {
      "Effect":"Allow",
      "Action":["s3:*"],
      "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                   "arn:aws:s3:::myBucket/home/${aws:username}/*"]
    }
  ]
}
```

Grants a user access to a home directory in S3 that can be accessed programmatically

The anatomy of a policy with variables

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Action": ["s3:ListBucket"],
     "Resource": ["arn:aws:s3:::myBucket"],
     "Condition":
             {"StringLike":
                {"s3:prefix":["home/${aws:username}/*"]}
        }
    },
    {
      "Effect":"Allow",
      "Action":["s3:*"],
      "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                   "arn:aws:s3:::myBucket/home/${aws:username}/*"]
    }
  ]
}
```

Version is required

Grants a user access to a home directory in S3 that can be accessed programmatically

The anatomy of a policy with variables

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Action": ["s3:ListBucket"],
     "Resource": ["arn:aws:s3:::myBucket"],
     "Condition":
              {"StringLike":
                 {"s3:prefix":["home/${aws:username}/*"]}
       }
    },
    {
      "Effect":"Allow",
      "Action":["s3:*"],
      "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                   "arn:aws:s3:::myBucket/home/${aws:username}/*"]
    }
  ]
}
```

Version is required

Variable in conditions

Grants a user access to a home directory in S3 that can be accessed programmatically

The anatomy of a policy with variables

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Action": ["s3:ListBucket"],
     "Resource": ["arn:aws:s3:::myBucket"],
     "Condition":
              {"StringLike":
                 {"s3:prefix":["home/${aws:username}/*"]}
       }
    },
    {
      "Effect":"Allow",
      "Action":["s3:*"],
      "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                   "arn:aws:s3:::myBucket/home/${aws:username}/*"]
    }
  ]
}
```

Version is required

Variable in conditions

Variable in resource ARNs

Grants a user access to a home directory in S3 that can be accessed programmatically

Managing your policies

Managing your policies

- IAM policies
  - Managed policies
  - Inline polices
- Resource-based policies

IAM policies

- Managed policies (newer way)
  - Can be attached to multiple users, groups, and roles
  - AWS managed policies: Created and maintained by AWS
  - Customer managed policies: Created and maintained by you
    - Up to 5K per policy
    - Up to 5 versions of a policy so you can roll back to a prior version
  - You can attach 10 managed policies per user, group, or role
  - You can limit who can attach which managed policies

- Inline policies (older way)
  - You create and embed directly in a single user, group, or role
  - Variable policy size (2K per user, 5K per group, 10K per role)

Resource-based policies

IAM policies live with:

- IAM users
- IAM groups
- IAM roles

Some services allow storing policy with resources:

- S3 (bucket policy)
- Amazon Glacier (vault policy)
- Amazon SNS (topic policy)
- Amazon SQS (queue policy)

```
{
 "Statement":
  {
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": "sqs:SendMessage",
    "Resource":
       "arn:aws:sqs:us-east-1:444455556666:queue1"
  }
}
```

Resource-based policies

IAM policies live with:

- IAM users
- IAM groups
- IAM roles

Some services allow storing policy with resources:

- S3 (bucket policy)
- Amazon Glacier (vault policy)
- Amazon SNS (topic policy)
- Amazon SQS (queue policy)

Principal required here

```
{
  "Statement":
    {
      "Effect": "Allow",
      "Principal": {"AWS": "111122223333"},
      "Action": "sqs:SendMessage",
      "Resource":
         "arn:aws:sqs:us-east-1:444455556666:queue1"
    }
}
```

Resource-based policies

IAM policies live with:

- IAM users
- IAM groups
- IAM roles

Some services allow storing policy with resources:

- S3 (bucket policy)
- Amazon Glacier (vault policy)
- Amazon SNS (topic policy)
- Amazon SQS (queue policy)

Principal required here

```
{
  "Statement":
    {
      "Effect": "Allow",
      "Principal": {"AWS": "111122223333"},
      "Action": "sqs:SendMessage",
      "Resource":
         "arn:aws:sqs:us-east-1:444455556666:queue1"
    }
}
```

Managed policies apply only to users, groups, and roles—not resources

Resource-based policies

IAM policies live with:

- IAM users
- IAM groups
- IAM roles

Some services allow storing policy with resources:

- S3 (bucket policy)
- Amazon Glacier (vault policy)
- Amazon SNS (topic policy)
- Amazon SQS (queue policy)

Principal required here

```
{
  "Statement":
    {
      "Effect": "Allow",
      "Principal": {"AWS": "111122223333"},
      "Action": "sqs:SendMessage",
      "Resource":
         "arn:aws:sqs:us-east-1:444455556666:queue1"
    }
}
```

Enough already…
Let’s look at some examples

Enough already…
Let's look at some examples

S3

IAM

EC2

Creating a home directory using S3

Demo

Creating a home directory using S3
Demo

- Goal: create a managed policy that:
  - Limits access to a prefix in an S3 bucket
  - E.g., arn:aws:s3:::my_bucket/home/Bob/*
- We'll examine how to:
  - Create a managed policy that uses variables.
  - Enable users to list buckets in the S3 console.
  - Limit users' access to specific folders in a bucket.

Giving a user a home directory from the S3 console

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

Giving a user a home directory from the S3 console

```
{
"Version": "2012-10-17",
 "Statement": [
   {"Sid": "AllowGroupToSeeBucketListInTheManagementConsole",
     "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
     "Effect": "Allow",
     "Resource": ["arn:aws:s3:::*"]},
   {"Sid": "AllowRootLevelListingOfThisBucketAndHomePrefix",
     "Action": ["s3:ListBucket"],
     "Effect": "Allow",
     "Resource": ["arn:aws:s3:::myBucket"],
     "Condition":{"StringEquals":{"s3:prefix":["","home/"],"s3:delimiter":["/"]}}},
  ]
}
```

- Necessary to access the S3 console.

Giving a user a home directory from the S3 console

```
{
"Version": "2012-10-17",
  "Statement": [
    {"Sid": "AllowGroupToSeeBucketListInTheManagementConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]},
    {"Sid": "AllowRootLevelListingOfThisBucketAndHomePrefix",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::myBucket"],
      "Condition":{"StringEquals":{"s3:prefix":["","home/"],"s3:delimiter":["/"]}}},
    {"Sid": "AllowListBucketofASpecificUserPrefix",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::myBucket"],
      "Condition":{"StringLike":{"s3:prefix":["home/${aws:username}/*"]}}},
  ]
}
```

- Allows listing all objects in a folder and its subfolders.

amazon web services

Giving a user a home directory from the S3 console

```
{
"Version": "2012-10-17",
  "Statement": [
    {"Sid": "AllowGroupToSeeBucketListInTheManagementConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]},
    {"Sid": "AllowRootLevelListingOfThisBucketAndHomePrefix",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::myBucket"],
      "Condition":{"StringEquals":{"s3:prefix":["","home/"],"s3:delimiter":["/"]}}},
    {"Sid": "AllowListBucketofASpecificUserPrefix",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::myBucket"],
      "Condition":{"StringLike":{"s3:prefix":["home/${aws:username}/*"]}}},
    {"Sid":"AllowUserFullAccesstoJustSpecificUserPrefix",
       "Action":["s3:*"],
       "Effect":"Allow",
       "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                    "arn:aws:s3:::myBucket/home/${aws:username}/*"]}
  ]
}
```

- Allows modifying objects in the folder and subfolders.

amazon web services

Creating a “limited” IAM administrator

Demo

Creating a “limited” IAM administrator
Demo

- Goal
  - Create a limited administrator who can use the IAM console to manage users, but only using certain policies

Creating a “limited” IAM administrator
Demo

- Goal
  - Create a limited administrator who can use the IAM console to manage users, but only using certain policies
- We’ll examine group policies that use variables to:
  - Grant admin full access to Amazon DynamoDB and read/write access to an S3 bucket.
  - Grant admin access to the IAM console to be able to create users and generate access keys.
  - Limit admin to a well-defined set of managed policies.

Create a “limited” IAM administrator

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Sid": "ManageUsersPermissions",
      "Effect": "Allow",
      "Action": ["iam:ChangePasword", "iam:CreateAccessKey", "iam:CreateLoginProfile",
                 "iam:CreateUser", "iam:DeleteAccessKey", "iam:DeleteLoginProfile",
                 "iam:DeleteUser", "iam:UpdateAccessKey", "iam:ListAttachedUserPolicies",
                 "iam:ListPolicies"],
      "Resource": "*"
    },
    {
      "Sid": "LimitedAttachmentPermissions",
      "Effect": "Allow",
      "Action": ["iam:AttachUserPolicy","iam:DetachUserPolicy"],
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "iam:PolicyArn": [
            "arn:aws:iam::123456789012:policy/reInvent2015_S3_Home_Folder",
            "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
          ]
        }
      }
    }
  ]
}
```

See AWS Security Blog post http://amzn.to/1Hf2XRl

Create a “limited” IAM administrator

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Sid": "ManageUsersPermissions",
      "Effect": "Allow",
      "Action": ["iam:ChangePasword", "iam:CreateAccessKey", "iam:CreateLoginProfile",
                 "iam:CreateUser", "iam:DeleteAccessKey", "iam:DeleteLoginProfile",
                 "iam:DeleteUser", "iam:UpdateAccessKey", "iam:ListAttachedUserPolicies",
                 "iam:ListPolicies"],
      "Resource": "*"
    },
    {
      "Sid": "LimitedAttachmentPermissions",
      "Effect": "Allow",
      "Action": ["iam:AttachUserPolicy","iam:DetachUserPolicy"],
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "iam:PolicyArn": [
            "arn:aws:iam::123456789012:policy/reInvent2015_S3_Home_Folder",
            "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
          ]
        }
      }
    }
  ]
}
```

- Allows creating users, managing keys, and setting passwords.

See AWS Security Blog post http://amzn.to/1Hf2XRl

Create a “limited” IAM administrator

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Sid": "ManageUsersPermissions",
      "Effect": "Allow",
      "Action": ["iam:ChangePasword", "iam:CreateAccessKey", "iam:CreateLoginProfile",
                 "iam:CreateUser", "iam:DeleteAccessKey", "iam:DeleteLoginProfile",
                 "iam:DeleteUser", "iam:UpdateAccessKey", "iam:ListAttachedUserPolicies",
                 "iam:ListPolicies"],
      "Resource": "*"
    },
    {
      "Sid": "LimitedAttachmentPermissions",
      "Effect": "Allow",
      "Action": ["iam:AttachUserPolicy","iam:DetachUserPolicy"],
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "iam:PolicyArn": [
            "arn:aws:iam::123456789012:policy/reInvent2015_S3_Home_Folder",
            "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
          ]
        }
      }
    }
  ]
}
```

- Limits attaching only these two policies.

See AWS Security Blog post http://amzn.to/1Hf2XRl

Grant a user access to the IAM console

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Sid": "ViewListOfAllUsers",
     "Action": "iam:ListUsers",
     "Effect": "Allow",
     "Resource": "arn:aws:iam::123456789012:user/*"
    },
    {
     "Sid": "AllowAdmintoAccessUser",
     "Effect": "Allow",
     "Action": ["iam:GetUser","iam:GetLoginProfile",
                "iam:ListGroupsForUser","iam:ListAccessKeys"],
     "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
    }
  ]
}
```

- Underneath the covers, the IAM console calls these APIs to view user settings.
- The user will be able to view details about all users.
- Doesn’t enable adding/removing MFA.

EC2 resource-level permissions

EC2 resource-level permissions

- Previously policies applied to all EC2 resources
- Permissions can be set per resource
- Ex: assign which users can stop, start, or terminate a particular instance

EC2 policies before resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":"*"
   }
  ]
}
```

EC2 policies before resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":"*"
   }
  ]
}
```

EC2 policies after resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":
        "arn:aws:ec2:us-east-1:123456789012:instance/i-abc12345"
     }
  ]
}
```

EC2 policies after resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":
        "arn:aws:ec2:us-east-1:123456789012:instance/*"
     }
  ]
}
```

EC2 policies after resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":
        "arn:aws:ec2:us-east-1:123456789012:instance/*",
      "Condition": {
        "StringEquals": {"ec2:ResourceTag/department": "dev"}
      }
    }
  ]
}
```

Supported EC2 resource types

Supports many different resource types, including:

- Customer gateway
- DHCP options set
- Image
- Instance
- Instance profile
- Internet gateway
- Key pair
- Network ACL
- Network interface
- Placement group
- Route table
- Security group
- Snapshot
- Subnet
- Volume
- VPC
- VPC peering connection

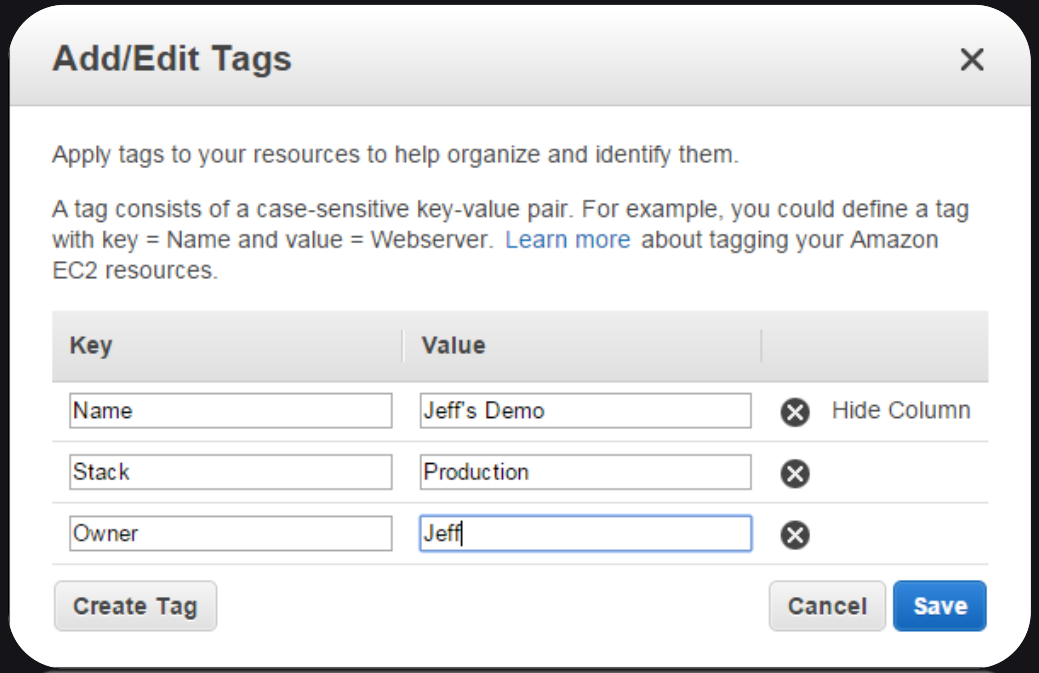
Supported EC2 actions

Note: This is only a subset of all possible EC2 actions.

Type of Resource	Actions — Accurate as of 10/1/2015
EC2 instances	RebootInstances, RunInstance, StartInstances, StopInstances, TerminateInstances, AttachClassicLinkVpc, AttachVolume, DetachClassicLinkVpc, DetachVolume,
Customer gateway	DeleteCustomerGateway
DHCP options sets	DeleteDhcpOptions
Internet gateways	DeleteInternetGateway
Network ACLs	DeleteNetworkAcl, DeleteNetworkAclEntry
Route tables	DeleteRoute, DeleteRouteTable
Security groups	AuthorizeSecurityGroupEgress, AuthorizeSecurityGroupIngress, DeleteSecurityGroup, RevokeSecurityGroupEgress, RevokeSecurityGroupIngress, AttachClassicLinkVpc, RunInstances
Volumes	AttachVolume, DeleteVolume, DetachVolume, RunInstances
VPC peering connections	AcceptVpcPeeringConnection, CreateVpcPeeringConnection, DeleteVpcPeeringConnection, RejectVpcPeeringConnection, DisableVpcClassicLink, EnableVpcClassicLink

Categorize your EC2 resources

Use tags as a resource attribute



- Allows user-defined models
- “Prod”/”Dev”
- “Cost Center X”
- “Department Y”
- “Project reInvent”

EC2 resource-level permissions

Demo

EC2 resource-level permissions

Demo

- Goal: Limit a user from starting, stopping, or terminating an instance unless the instance is owned by the user.
-

EC2 resource-level permissions

Demo

- Goal: Limit a user from starting, stopping, or terminating an instance unless the instance is owned by the user.
- We'll examine:
  - Adding an owner tag to an EC2 instance.
  - A policy that grants a user access to the EC2 console.
  - A policy that uses variables to limit access based on an owner tag.

Locking down access to EC2 instances

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Sid": "THISALLOWSEC2READACCESS",
    "Effect": "Allow",
    "Action": ["ec2:Describe*","elasticloadbalancing:Describe*",
               "cloudwatch:ListMetrics","cloudwatch:GetMetricStatistics",
              "cloudwatch:Describe*","autoscaling:Describe*"],
     "Resource": "*"
     },
     {
    "Sid": "THISLIMITSACCESSTOOWNINSTANCES",
    "Effect": "Allow",
    "Action": ["ec2:RebootInstances","ec2:StartInstances",
              "ec2:StopInstances","ec2:TerminateInstances"],
    "Resource":"arn:aws:ec2:us-east-1:123456789012:instance/*",
    "Condition": {"StringEquals":
                  {"ec2:ResourceTag/Owner": "${aws:username}"}}
      }
    ]
}
```

Locking down access to EC2 instances

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Sid": "THISALLOWSEC2READACCESS",
    "Effect": "Allow",
    "Action": ["ec2:Describe*","elasticloadbalancing:Describe*",
               "cloudwatch:ListMetrics","cloudwatch:GetMetricStatistics",
              "cloudwatch:Describe*","autoscaling:Describe*"],
     "Resource": "*"
     },
     {
    "Sid": "THISLIMITSACCESSTOOWNINSTANCES",
    "Effect": "Allow",
    "Action": ["ec2:RebootInstances","ec2:StartInstances",
              "ec2:StopInstances","ec2:TerminateInstances"],
    "Resource":"arn:aws:ec2:us-east-1:123456789012:instance/*",
    "Condition": {"StringEquals":
                  {"ec2:ResourceTag/Owner": "${aws:username}"}}
      }
    ]
}
```

Version is required here because we're using variables

Locking down access to EC2 instances

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Sid": "THISALLOWSEC2READACCESS",
    "Effect": "Allow",
    "Action": ["ec2:Describe*","elasticloadbalancing:Describe*",
               "cloudwatch:ListMetrics","cloudwatch:GetMetricStatistics",
              "cloudwatch:Describe*","autoscaling:Describe*"],
     "Resource": "*"
     },
     {
    "Sid": "THISLIMITSACCESSTOOWNINSTANCES",
    "Effect": "Allow",
    "Action": ["ec2:RebootInstances","ec2:StartInstances",
              "ec2:StopInstances","ec2:TerminateInstances"],
    "Resource":"arn:aws:ec2:us-east-1:123456789012:instance/*",
    "Condition": {"StringEquals":
                  {"ec2:ResourceTag/Owner": "${aws:username}"}}
      }
    ]
}
```

Version is required here because we're using variables

Allows seeing everything from the EC2 console

Locking down access to EC2 instances

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Sid": "THISALLOWSEC2READACCESS",
    "Effect": "Allow",
    "Action": ["ec2:Describe*","elasticloadbalancing:Describe*",
               "cloudwatch:ListMetrics","cloudwatch:GetMetricStatistics",
              "cloudwatch:Describe*","autoscaling:Describe*"],
     "Resource": "*"
     },
     {
    "Sid": "THISLIMITSACCESSTOOWNINSTANCES",
    "Effect": "Allow",
    "Action": ["ec2:RebootInstances","ec2:StartInstances",
              "ec2:StopInstances","ec2:TerminateInstances"],
    "Resource":"arn:aws:ec2:us-east-1:123456789012:instance/*",
    "Condition": {"StringEquals":
                  {"ec2:ResourceTag/Owner": "${aws:username}"}}
      }
    ]
}
```

Version is required here because we're using variables

Allows seeing everything from the EC2 console

Only allowed if this tag condition is true

Locking down access to EC2 instances

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
    "Sid": "THISALLOWSEC2READACCESS",
    "Effect": "Allow",
    "Action": ["ec2:Describe*","elasticloadbalancing:Describe*",
               "cloudwatch:ListMetrics","cloudwatch:GetMetricStatistics",
              "cloudwatch:Describe*","autoscaling:Describe*"],
     "Resource": "*"
     },
     {
    "Sid": "THISLIMITSACCESSTOOWNINSTANCES",
    "Effect": "Allow",
    "Action": ["ec2:RebootInstances","ec2:StartInstances",
              "ec2:StopInstances","ec2:TerminateInstances"],
    "Resource":"arn:aws:ec2:us-east-1:123456789012:instance/*",
    "Condition": {"StringEquals":
                  {"ec2:ResourceTag/Owner": "${aws:username}"}}
      }
    ]
}
```

Version is required here because we're using variables

Allows seeing everything from the EC2 console

Only allowed if this tag condition is true

Specify the tag key and value here

Limit EC2 instance types

Demo

Limit EC2 instance types

Demo

- Goal: Limit a user from starting an instance unless the instance is `t1.*`, `t2.*`, `m3.*`

Limit EC2 instance types

Demo

- Goal: Limit a user from starting an instance unless the instance is `t1.*`, `t2.*`, `m3.*`
- We'll do the following:
  - Create a new IAM group.
  - Create a managed policy that limits starting EC2 instances to specific instance types.
  - Attach that managed policy to the IAM group.

Locking down access to instance types

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "NotAction": ["iam:*","ec2:RunInstances"],
            "Resource": "*"},
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "NotResource": [
                "arn:aws:ec2:us-east-1:012345678912:instance/*",
                "arn:aws:ec2:eu-west-1:012345678912:instance/*"]},
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:us-east-1:012345678912:instance/*",
                "arn:aws:ec2:eu-west-1:012345678912:instance/*"],
            "Condition": {
                "StringLike": {"ec2:InstanceType": ["t1.*","t2.*","m3.*"]}
            }
        }
    ]
}
```

Include all services/actions you want to exclude!

Grants access to everything you need to launch an instance, except the actual instance

Lock down types here

Take advantage of `IfExists` conditional operator

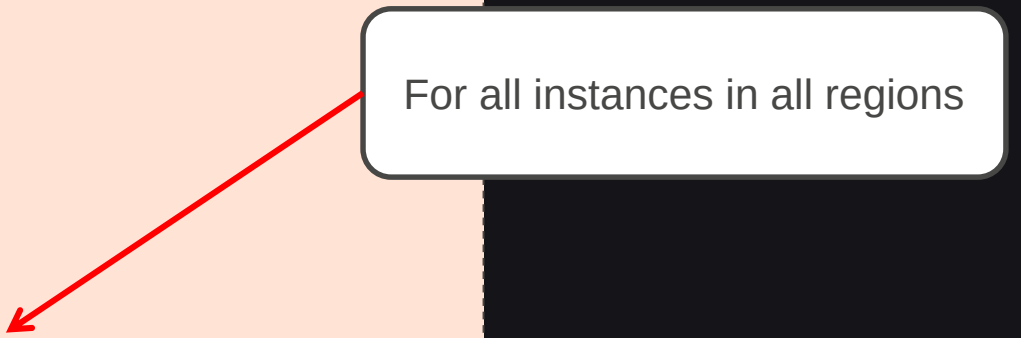
- Many condition keys only exist for certain resource types.
- If you test for a nonexistent key, your policy will fail to evaluate (i.e., access denied).
- You can add `IfExists` at the end of any condition operator except the `Null` condition (e.g., `StringLikeIfExists`).
- Allows you to create policies that “don’t care” if the key is not present.

StringNotLikeIfExists Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:012345678901:instance/*",
            "Condition": {
                "StringNotLikeIfExists": {
                    "ec2:InstanceType": ["t1.*", "t2.*", "m3.*"]
                }
            }
        }
    ]
}
```

StringNotLikeIfExists Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:012345678901:instance/*",
            "Condition": {
                "StringNotLikeIfExists": {
                    "ec2:InstanceType": ["t1.*", "t2.*", "m3.*"]
                }
            }
        }
    ]
}
```



StringNotLikeIfExists Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:012345678901:instance/*",
            "Condition": {
                "StringNotLikeIfExists": {
                    "ec2:InstanceType": ["t1.*", "t2.*", "m3.*"]
                }
            }
        }
    ]
}
```

For all instances in all regions

Only apply this condition if this `InstanceType` key exists

Testing and debugging

Testing and debugging

- Authoring – Policy editor
- Testing – Policy simulator
- Debugging – Encoded authorization message (EC2)

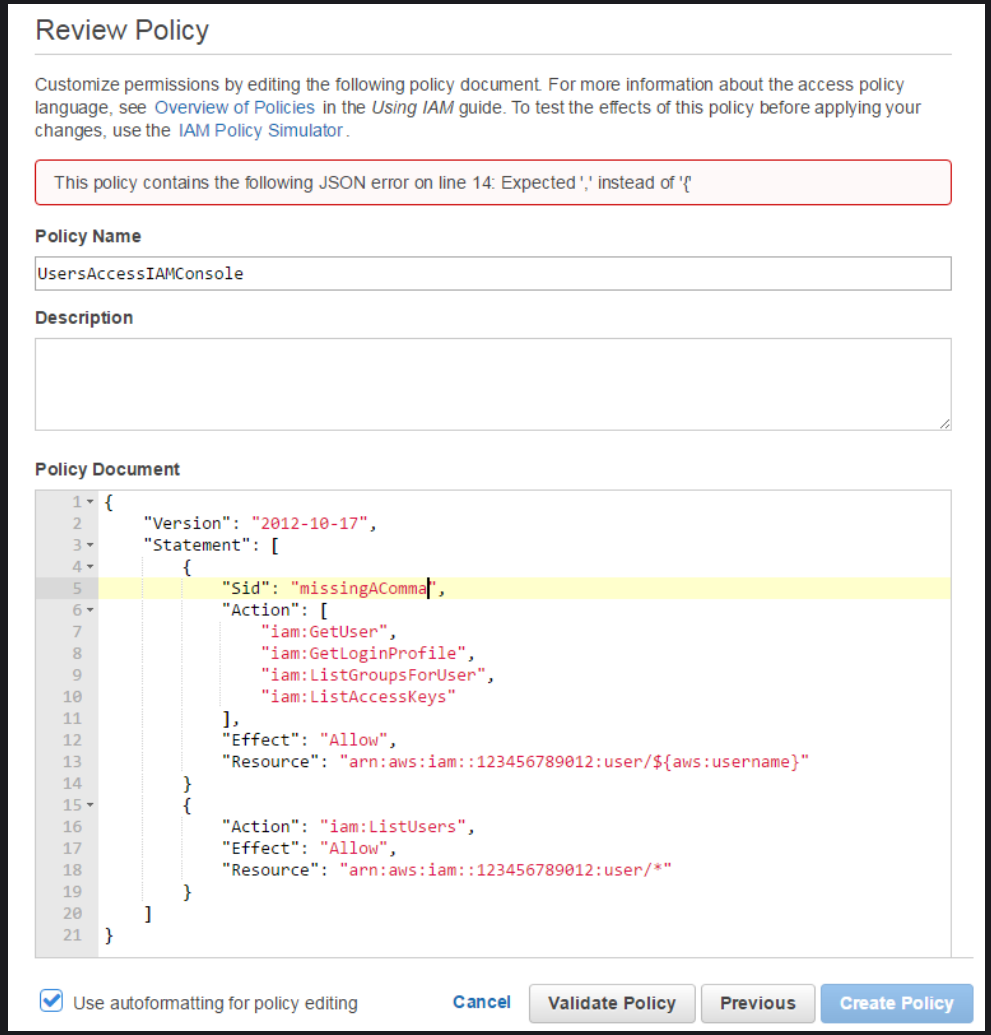
Policy editor

Policy validation checks:

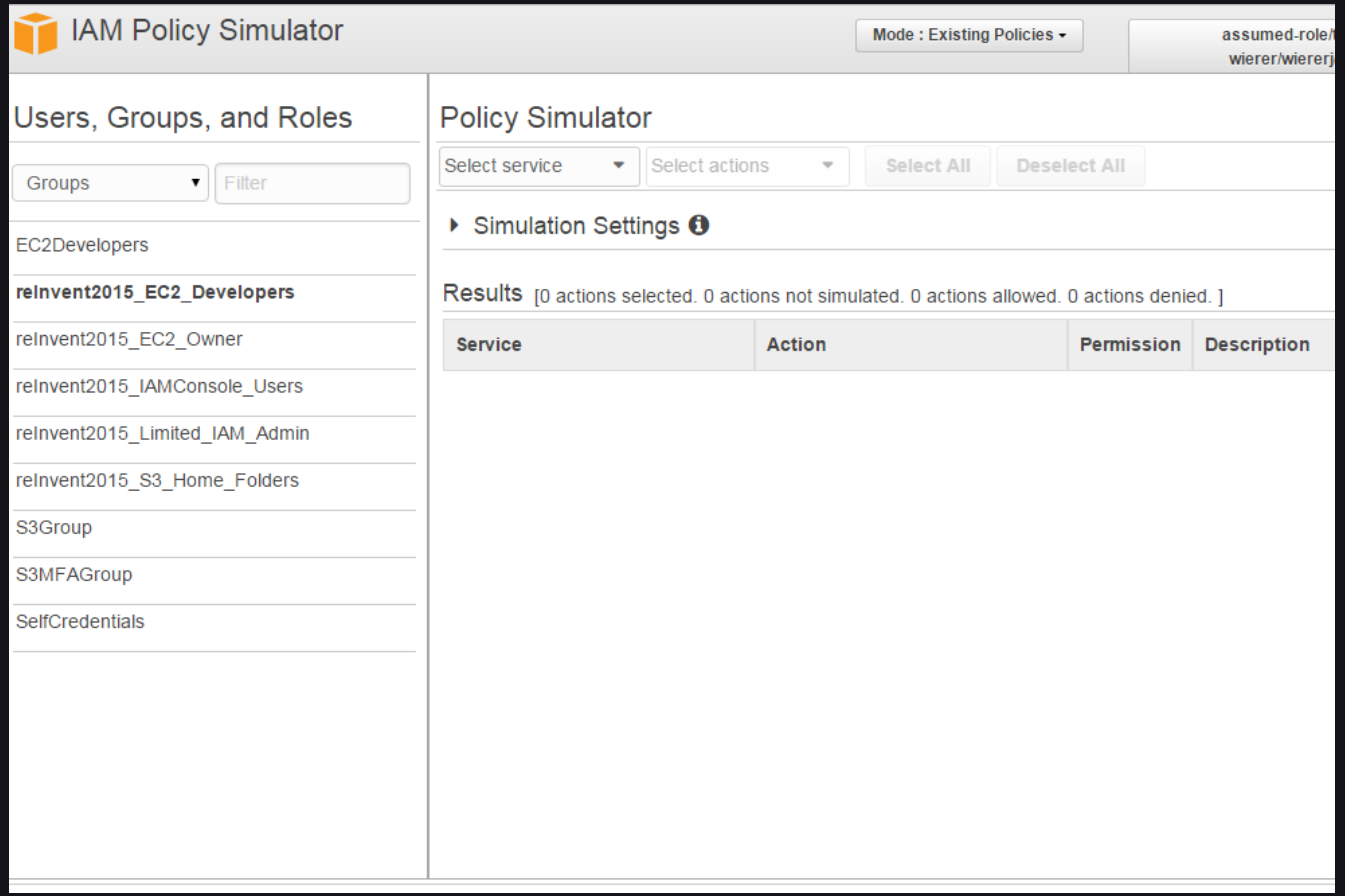
- JSON errors
- Policy grammar errors

Policy formatting:

- On-demand
- Autoformatting



Policy simulator



Decoding the EC2 authorization message

- Additional information about the authorization status of a request

Decoding the EC2 authorization message

- Additional information about the authorization status of a request
- The decoded message includes:
  - Whether the request was denied due to an explicit deny or absence of an explicit allow.
  - The principal who made the request.
  - The requested action.
  - The requested resource.
  - The values of condition keys in the context of the user's request.

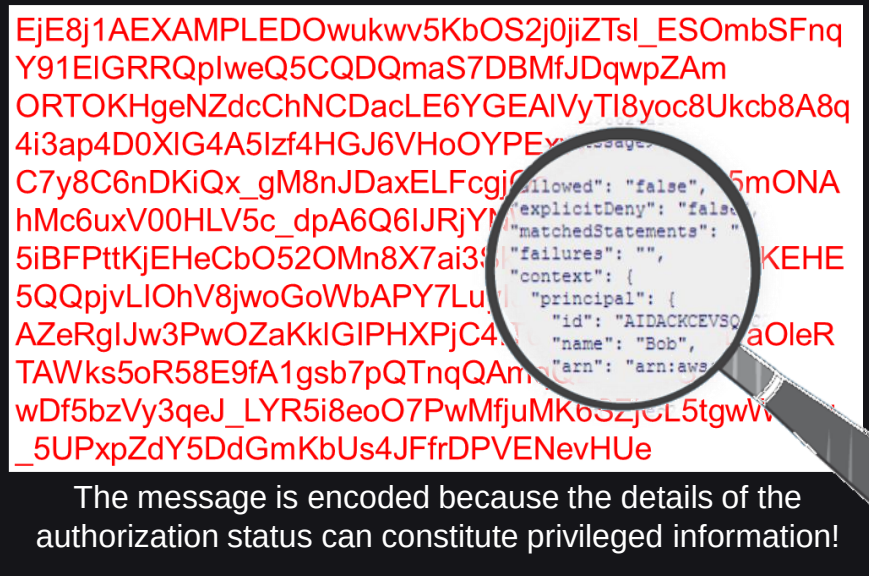
Output

```
EjE8j1AEXAMPLEDOwukwv5KbOS2j0jiZTsl_ESOmbSFnq
Y91ElGRRQpIweQ5CQDQmaS7DBMfJDqwpZAm
ORTOKHgeNZdcChNCDacLE6YGEAlVyTI8yoc8Ukcb8A8q
4i3ap4D0XlG4A5Izf4HGJ6VHoOYPExvwVcDy
C7y8C6nDKiQx_gM8nJDaxELFcgjOa4RxfsDcpPe5mONA
hMc6uxV00HLV5c_dpA6Q6IJRjYNWxjNEEtky
5iBFPttKjEHeCbO52OMn8X7ai3SkRS7V33dpxcwpaKEHE
5QQpjvLIOhV8jwoGoWbAPY7LuyIJqtDysfP
AZeRgIJw3PwOZaKkIGIPHXPjC4IT63ttMvTObDdDaOleR
TAWks5oR58E9fA1gsb7pQTnqQAmqQBhvxWS
wDf5bzVy3qeJ_LYR5i8eoO7PwMfjuMK6SZjCL5tgwWRqu
_5UPxpZdY5DdGmKbUs4JFfrDPVENevHUe
```

Decoding the EC2 authorization message

- Additional information about the authorization status of a request
- The decoded message includes:
  - Whether the request was denied due to an explicit deny or absence of an explicit allow.
  - The principal who made the request.
  - The requested action.
  - The requested resource.
  - The values of condition keys in the context of the user's request.

Output

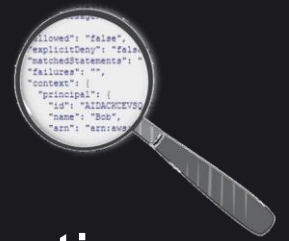


The message is encoded because the details of the authorization status can constitute privileged information!

Decoding the EC2 authorization message

Demo

Decoding the EC2 authorization message
Demo



- Goal: Determine what caused an EC2 authorization failure
- We'll do the following:
  - Try to call an EC2 action
  - Capture the EC2 Authorization Failure in JSON format
  - Remove the DecodeMessage "wrapper"
  - Format the message to identify the missing permission

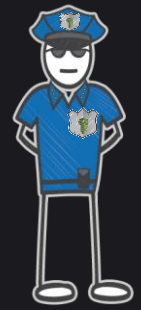
Policy enforcement

Policy enforcement

Policy enforcement

1. Decision starts at Deny

Policy enforcement

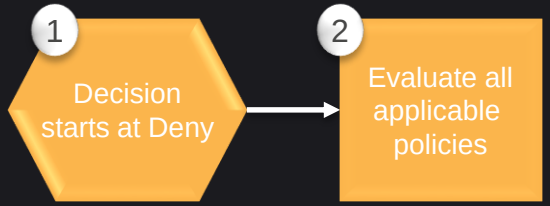


1

Decision starts at Deny

- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

Policy enforcement

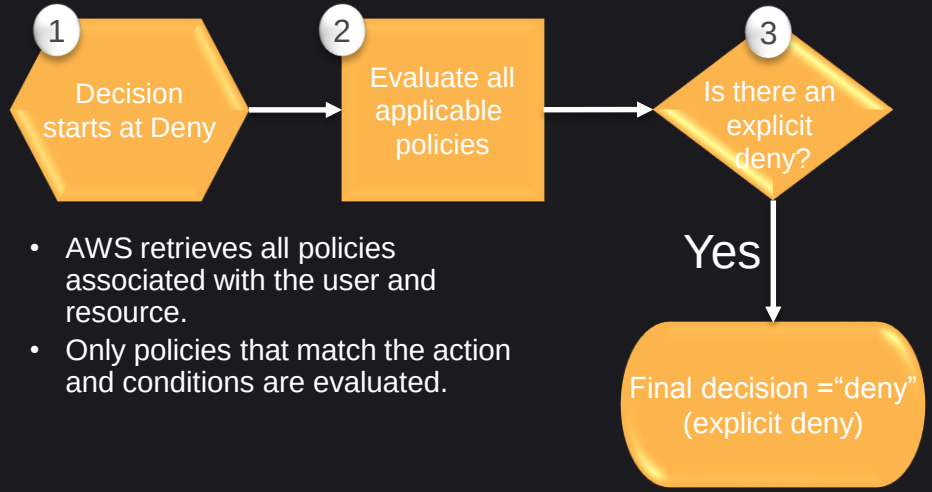


- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

Policy enforcement

- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

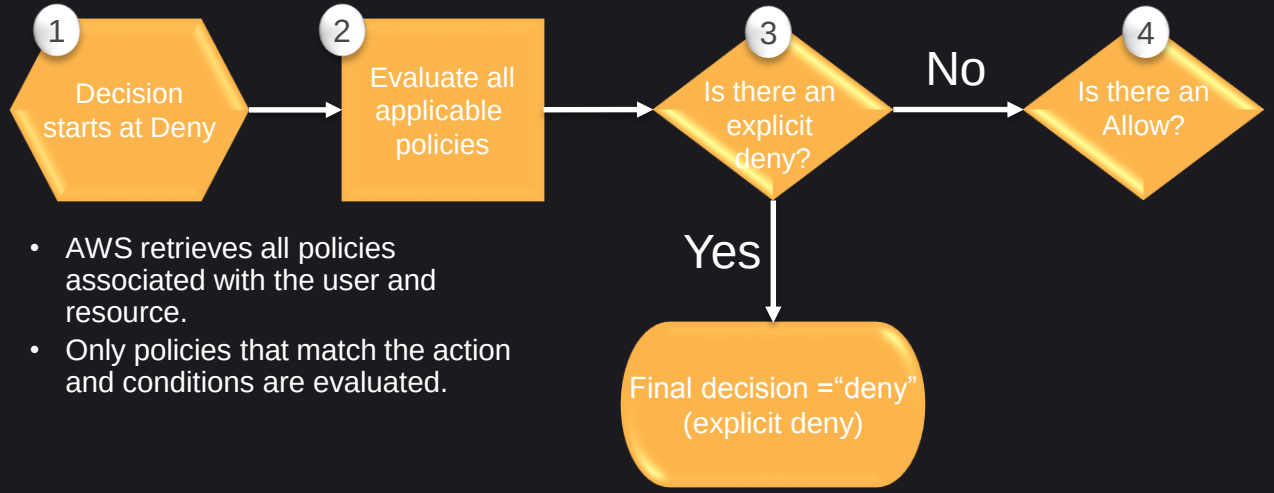
Policy enforcement



- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

- If a policy statement has a deny, it trumps all other policy statements.

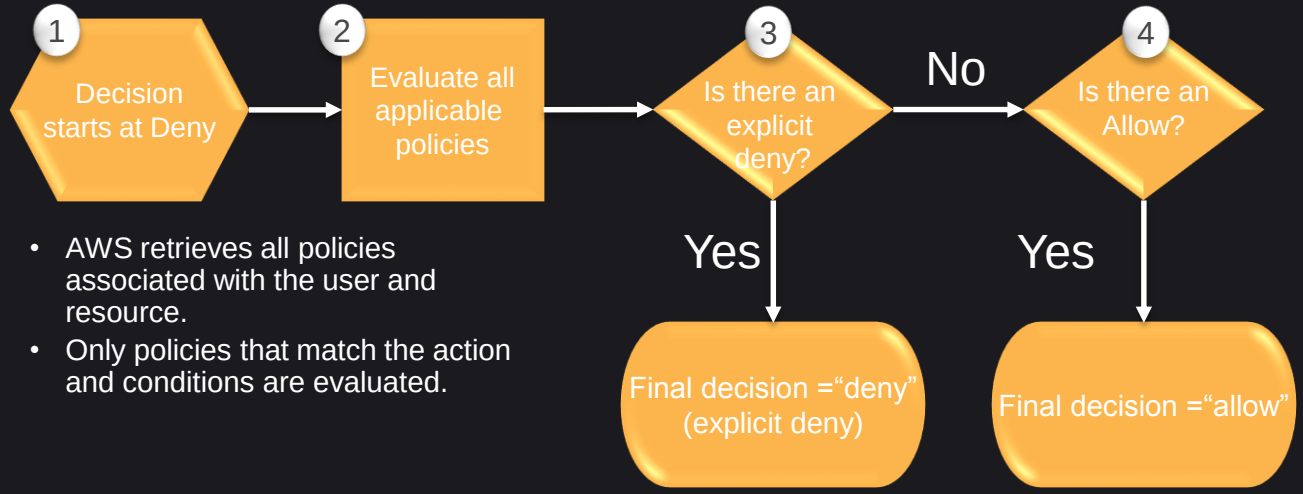
Policy enforcement



- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

- If a policy statement has a deny, it trumps all other policy statements.

Policy enforcement

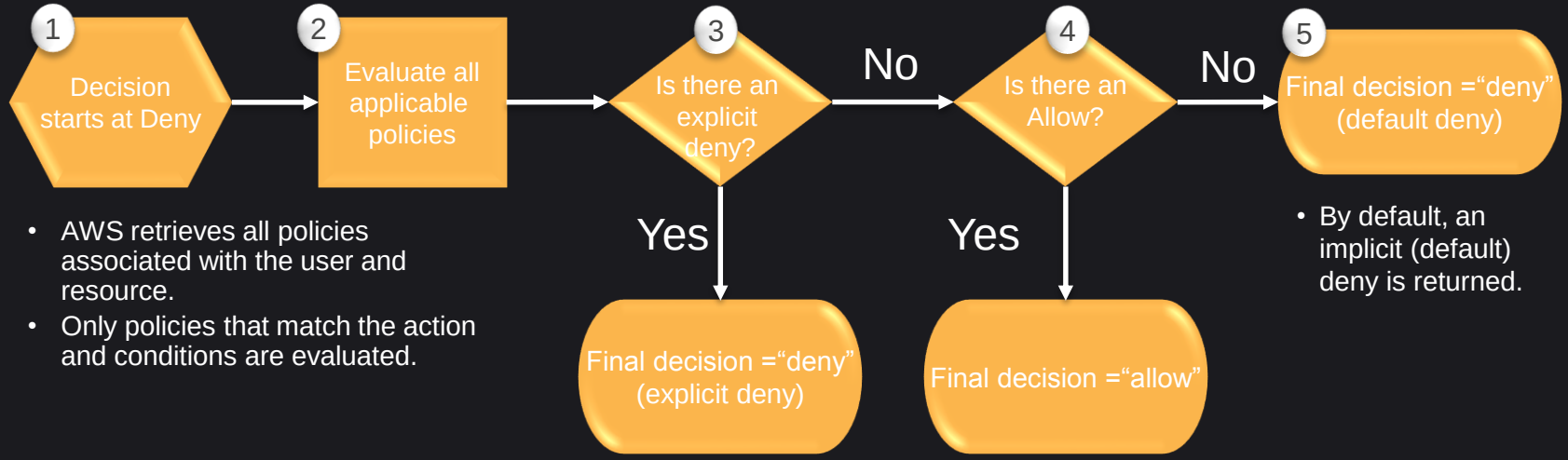


- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

- If a policy statement has a deny, it trumps all other policy statements.

- Access is granted if there is an explicit allow and no deny.

Policy enforcement



- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

- If a policy statement has a deny, it trumps all other policy statements.

- Access is granted if there is an explicit allow and no deny.

- By default, an implicit (default) deny is returned.

Summary

- IAM provides access control for your AWS account.
- The policy language authorizes that access.
- All applicable policies are evaluated.
  - Users are denied access by default.
  - A deny always trumps an allow.
- Use policy variables and remember the version!
- Keep in mind which EC2 actions or resources are currently supported.

Additional resources

- Documentation
  - http://aws.amazon.com/documentation/iam/
  - http://docs.aws.amazon.com/AWSEC2/latest/APIReference/ec2-api-permissions.html
- AWS Security Blog (blogs.aws.amazon.com/security)
  - http://blogs.aws.amazon.com/security/post/Tx2KPWZJJ4S26H6/Demystifying-EC2-Resource-Level-Permissions
  - http://blogs.aws.amazon.com/security/post/Tx29ZC3VE9SQGQM/Granting-Users-Permission-to-Work-in-the-Amazon-EC2-Console
- http://aws.amazon.com/iam
- https://forums.aws.amazon.com/forum.jspa?forumID=76
- Twitter: @AWSIdentity

Related sessions

Wednesday, 1:30–2:30 P.M.

SEC302 – IAM Best Practices to Live By

Thursday, 1:30–2:30 P.M.

SEC307 – A Progressive Journey Through AWS IAM Federation Options: From Roles to SAML to Custom Identity Brokers

Remember to complete your evaluations!

Thank you!