
AWS
re:Inforce
JUNE 13 - 14, 2023 | ANAHEIM, CA

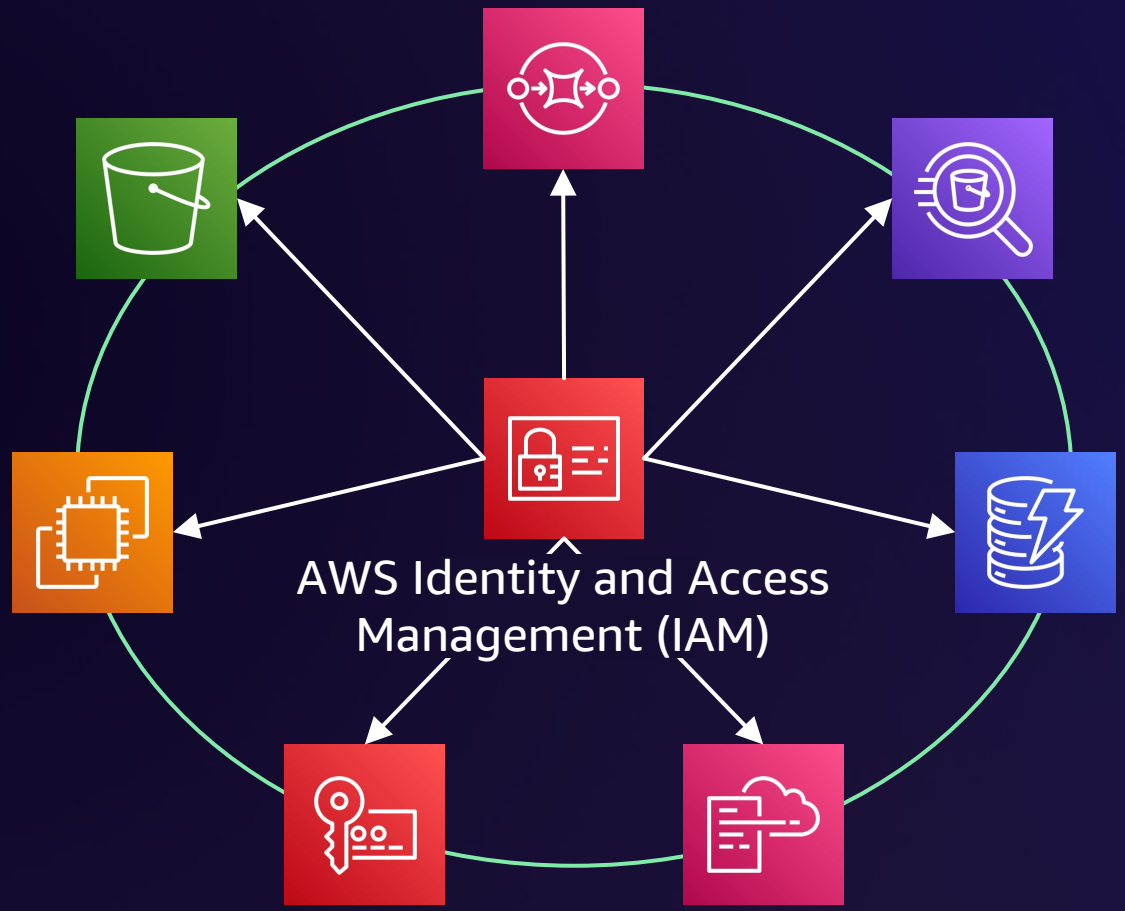
IAM201

A first-principles approach: AWS Identity and Access Management (IAM)

Becky Weiss

Senior Principal Engineer
Amazon Web Services

IAM: The most fundamental cloud skill



Agenda: Practical skills ++

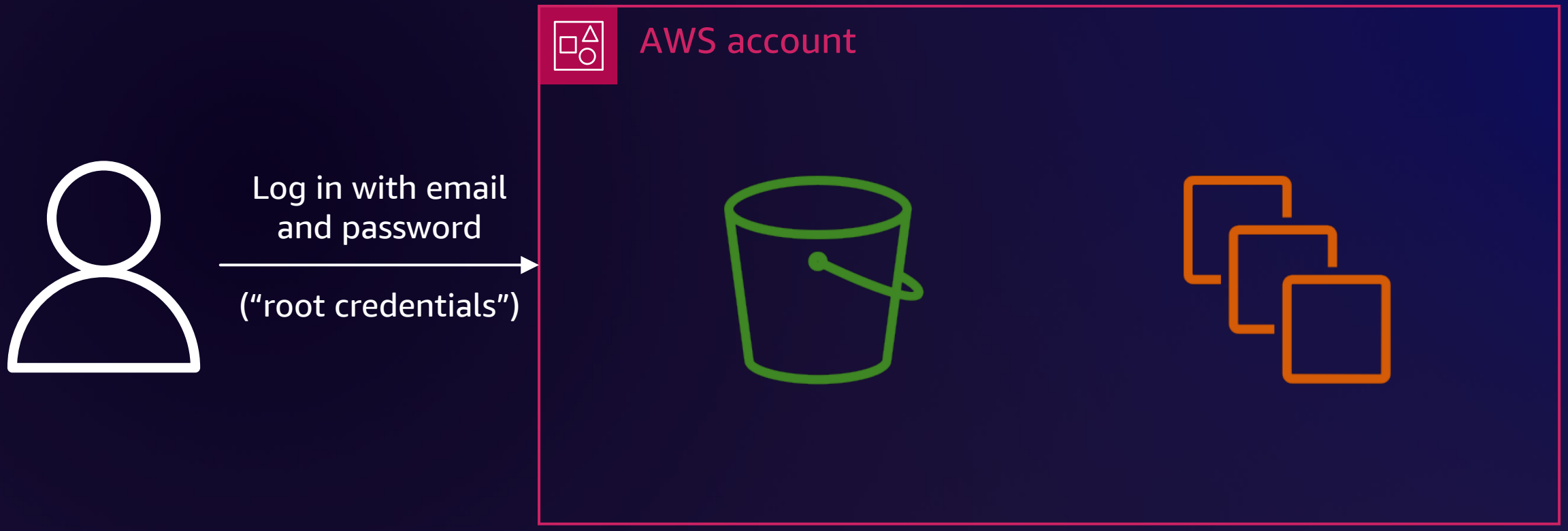
- The **what** of IAM: Skills for builders – Learn how to interpret and write good IAM policies
- The **why** of IAM: Working backward from customer needs over time
- The **how** of IAM: What happens in an authorization

Agenda: Practical skills ++

- The **what** of IAM: Skills for builders – Learn how to interpret and write good IAM policies
- The **why** of IAM: Working backward from customer needs over time
- The **how** of IAM: What happens in an authorization

The backstory

AWS existed before IAM



Launch of IAM: 2011

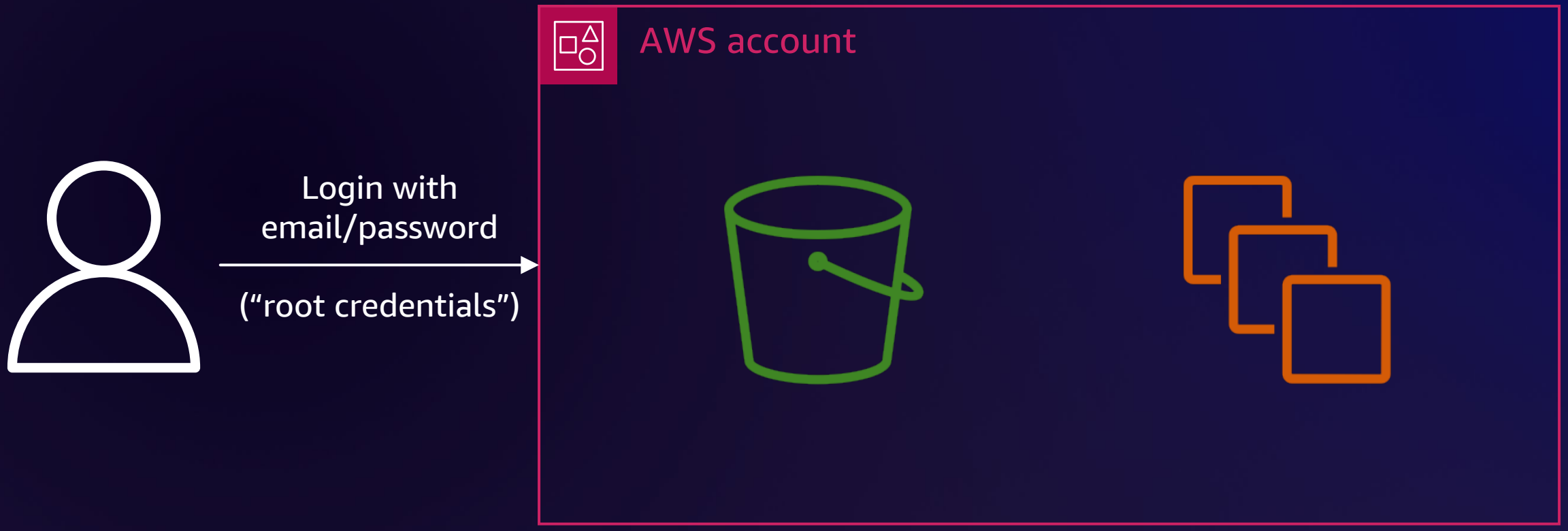
AWS Identity and Access Management (IAM) Announces General Availability and Support in the AWS Management Console

Posted On: May 3, 2011

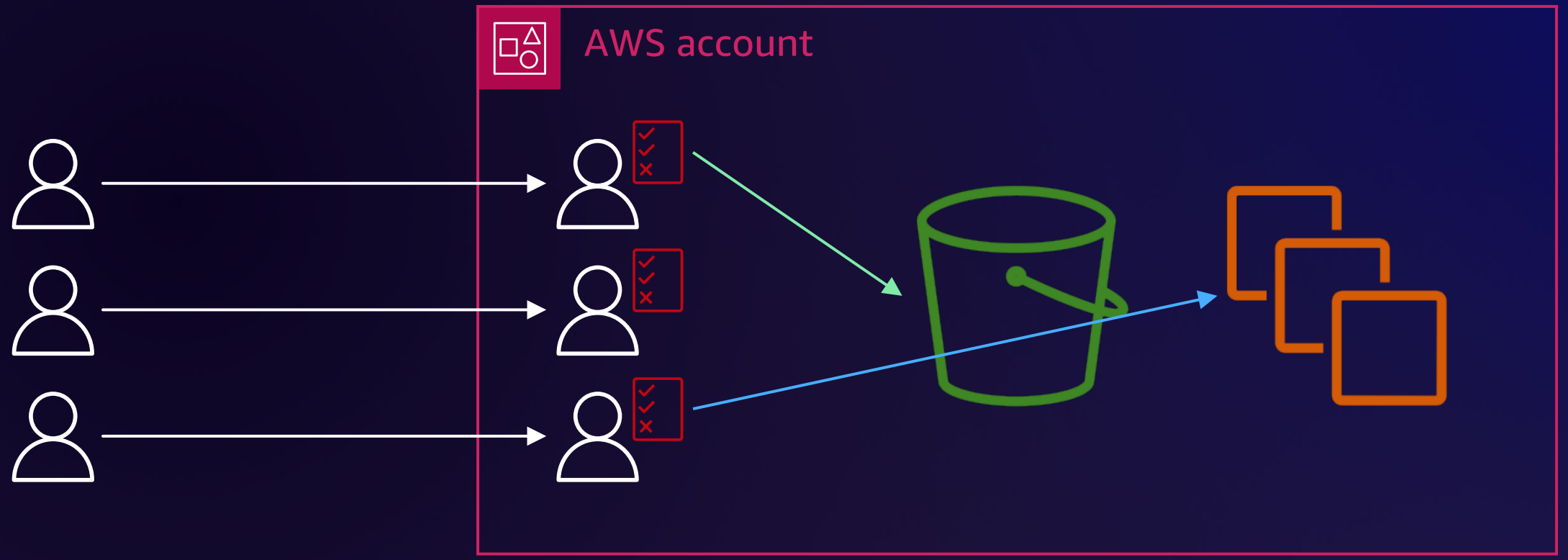
We are excited to announce the availability of AWS Identity and Access Management (IAM) in the AWS Management Console. You can now add users to your AWS Account, set groups and permissions for these users, and enable users to call AWS Service APIs, all from a simple point-and-click browser interface.

To learn more about the IAM console, please visit the Using IAM guide or open the IAM console.

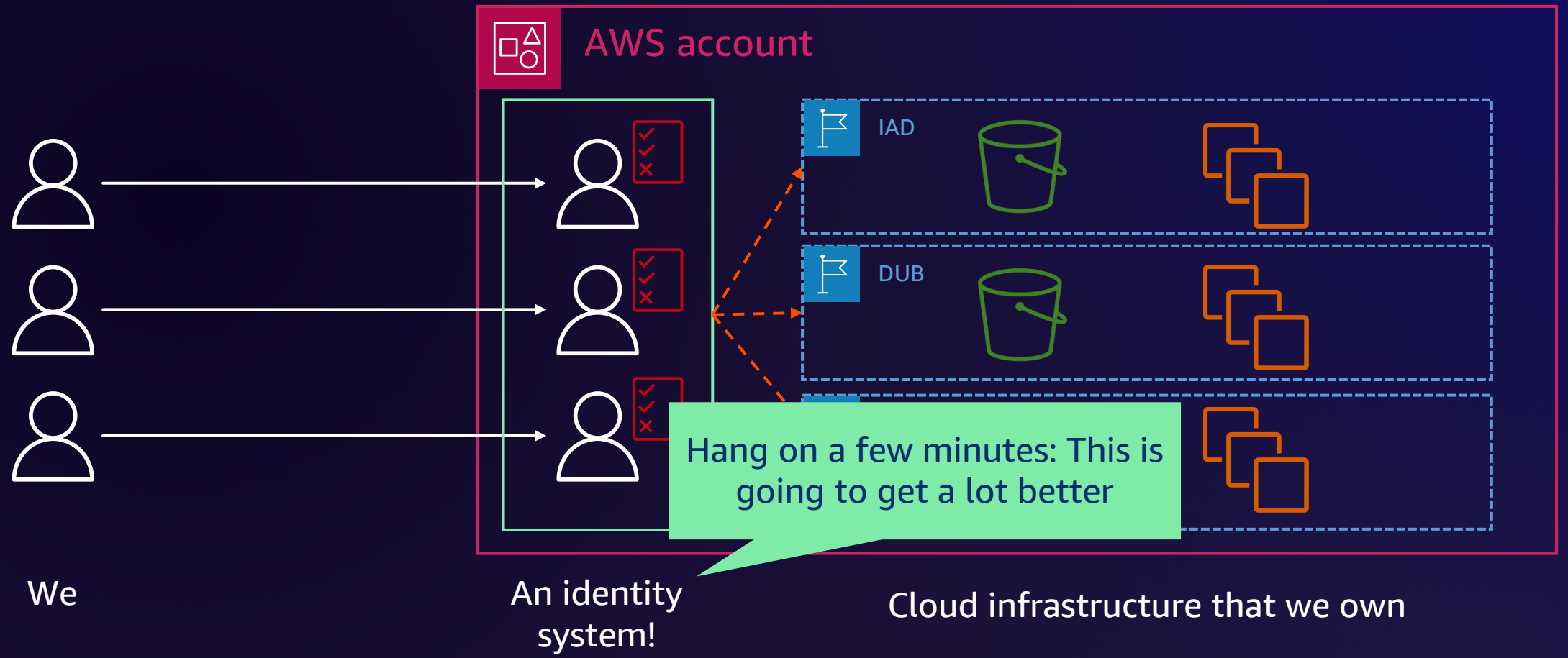
AWS without IAM: Me and my stuff



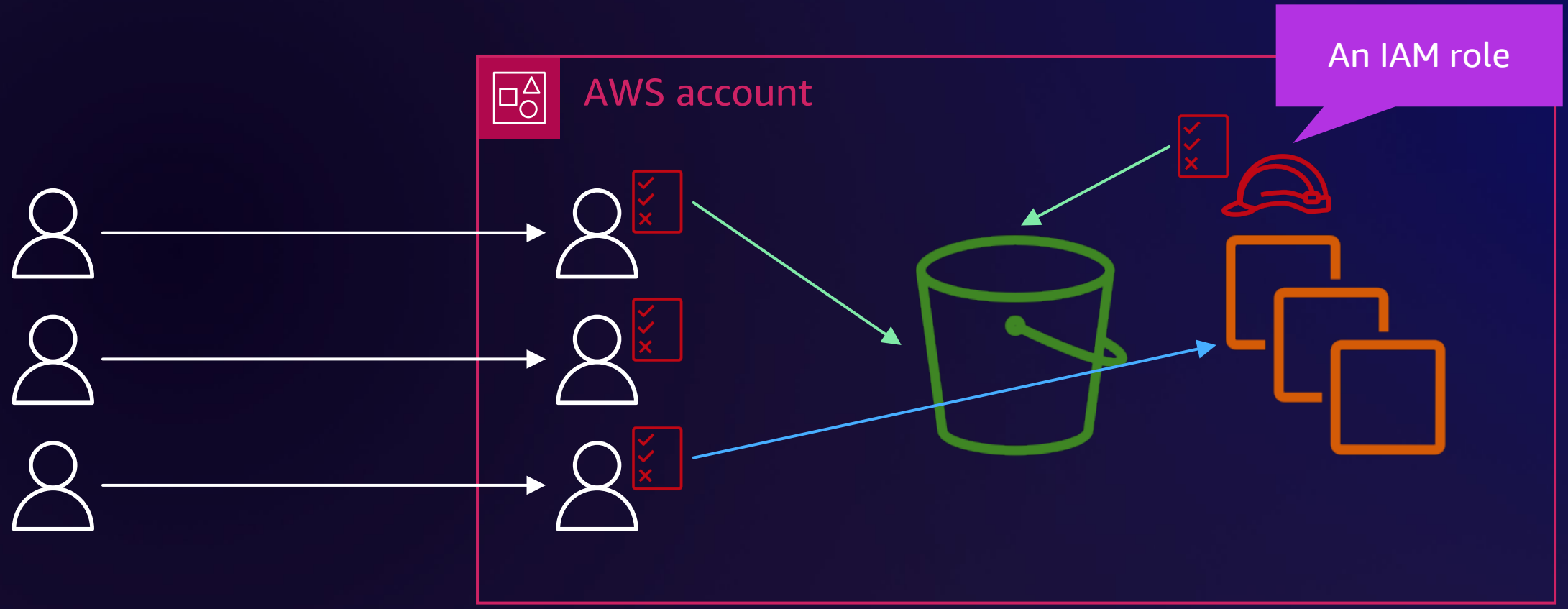
IAM circa 2011: We and our stuff



IAM circa 2011: An identity system for the cloud



IAM circa 2012: Identities for compute



Terminology: IAM principal

IAM princip n the IAM system

Appropriate for nearly all modern use cases

IAM roles: IAM principals that authenticate with temporary credentials

Use cases

- AWS compute environments (EC2 instances, Lambda functions, etc.)
- Non-AWS compute environments (IAM Roles Anywhere)
- Credentials for federated identities (e.g., users in a directory)

IAM users: IAM principals with long-term credentials

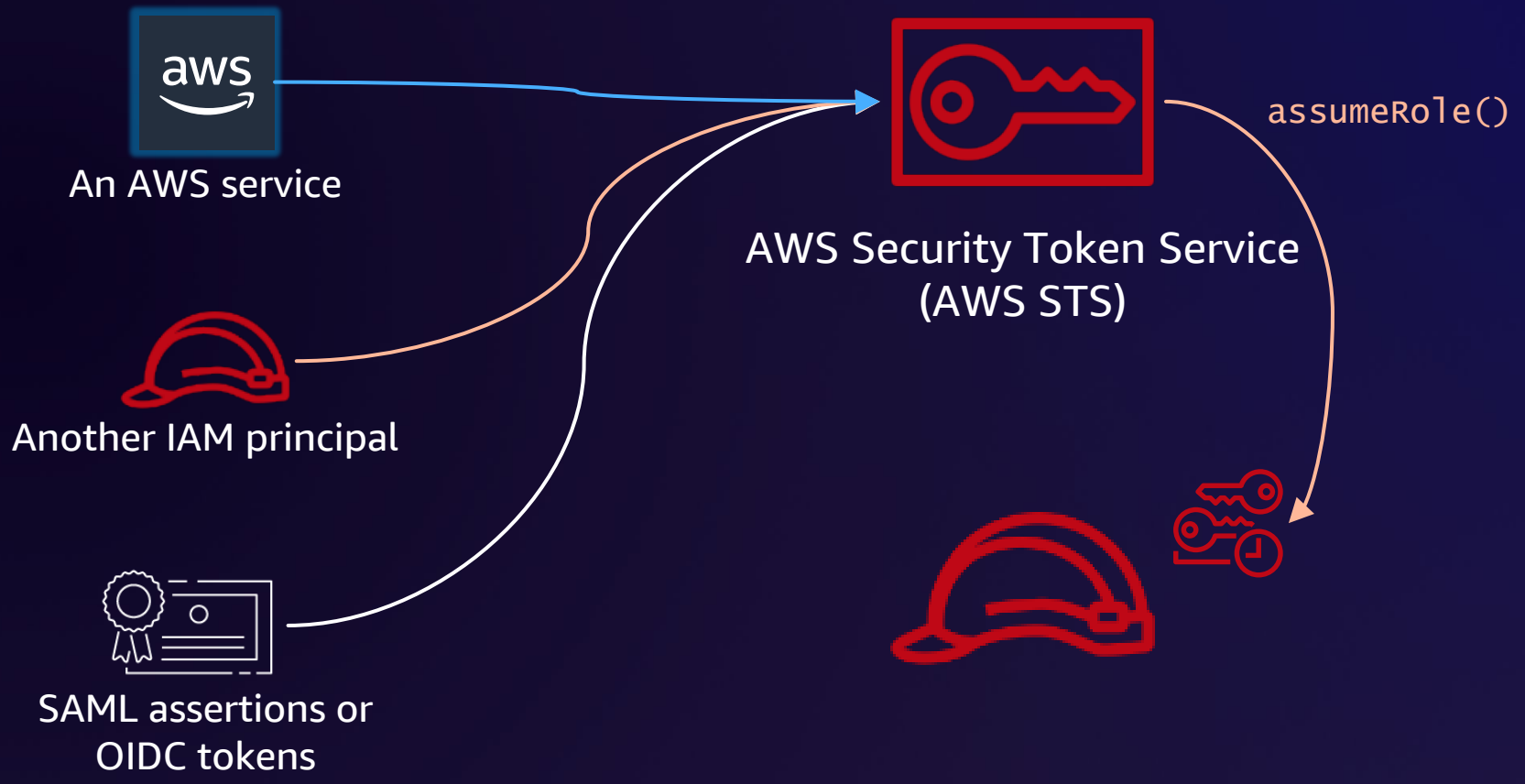
Principals and principles: Spell-check

Principal (n.)

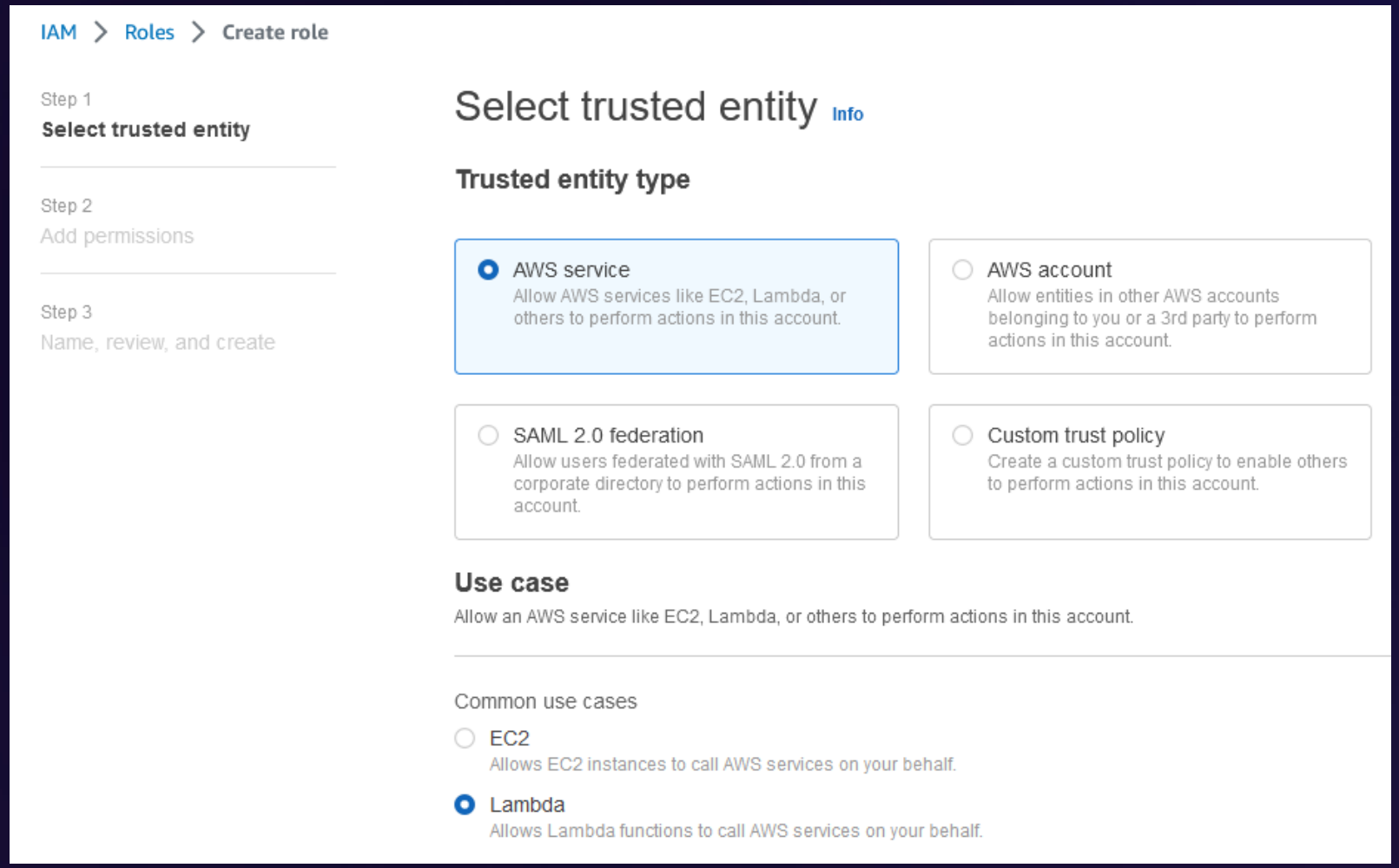
Principle (n.)

Principal principles: Getting IAM role credentials

IAM role credentials are temporary



How AWS services assume IAM roles



AWS Lambda service

Lambda function runtime environment

How AWS services assume IAM roles

```
// Role Trust Policy, a.k.a. Assume-Role Policy.
//
// This policy is attached to the IAM Role itself and
// says who can obtain its credentials (the AWS Lambda
// service in this case)

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

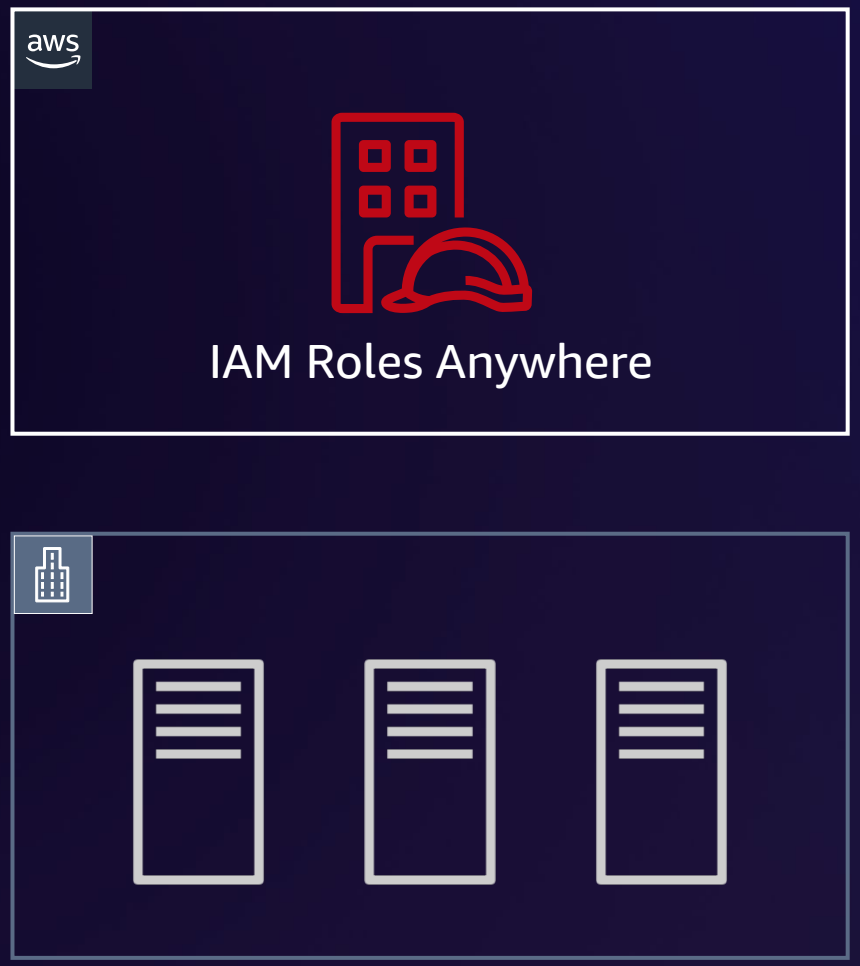
Service principal:

Identifies an AWS service acting on your resources

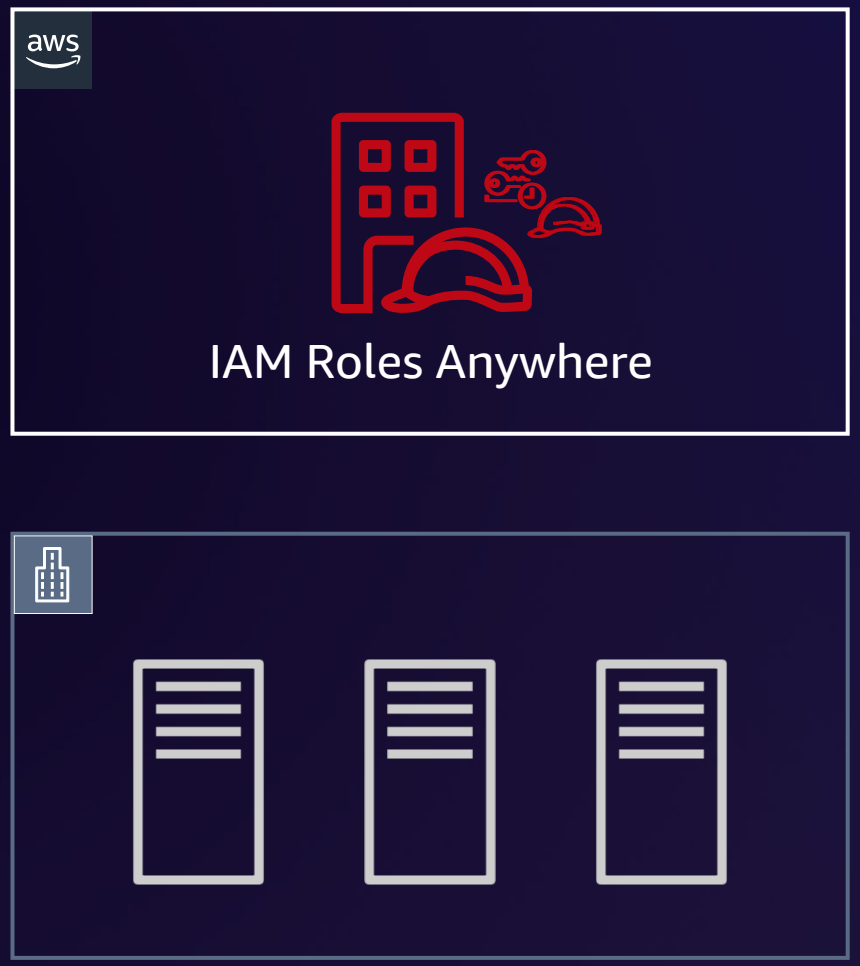
AWS Lambda service

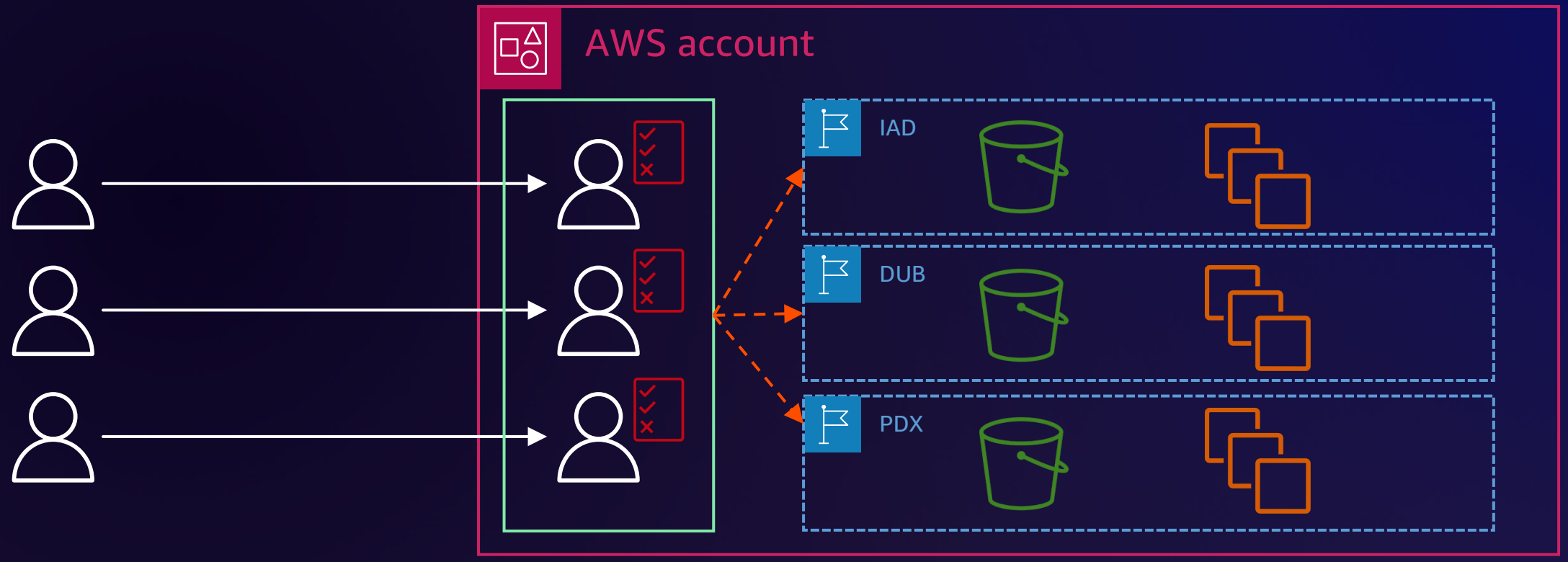
Lambda function runtime environment

IAM Roles Anywhere: IAM roles for workloads outside AWS



IAM Roles Anywhere: IAM roles for workloads outside AWS

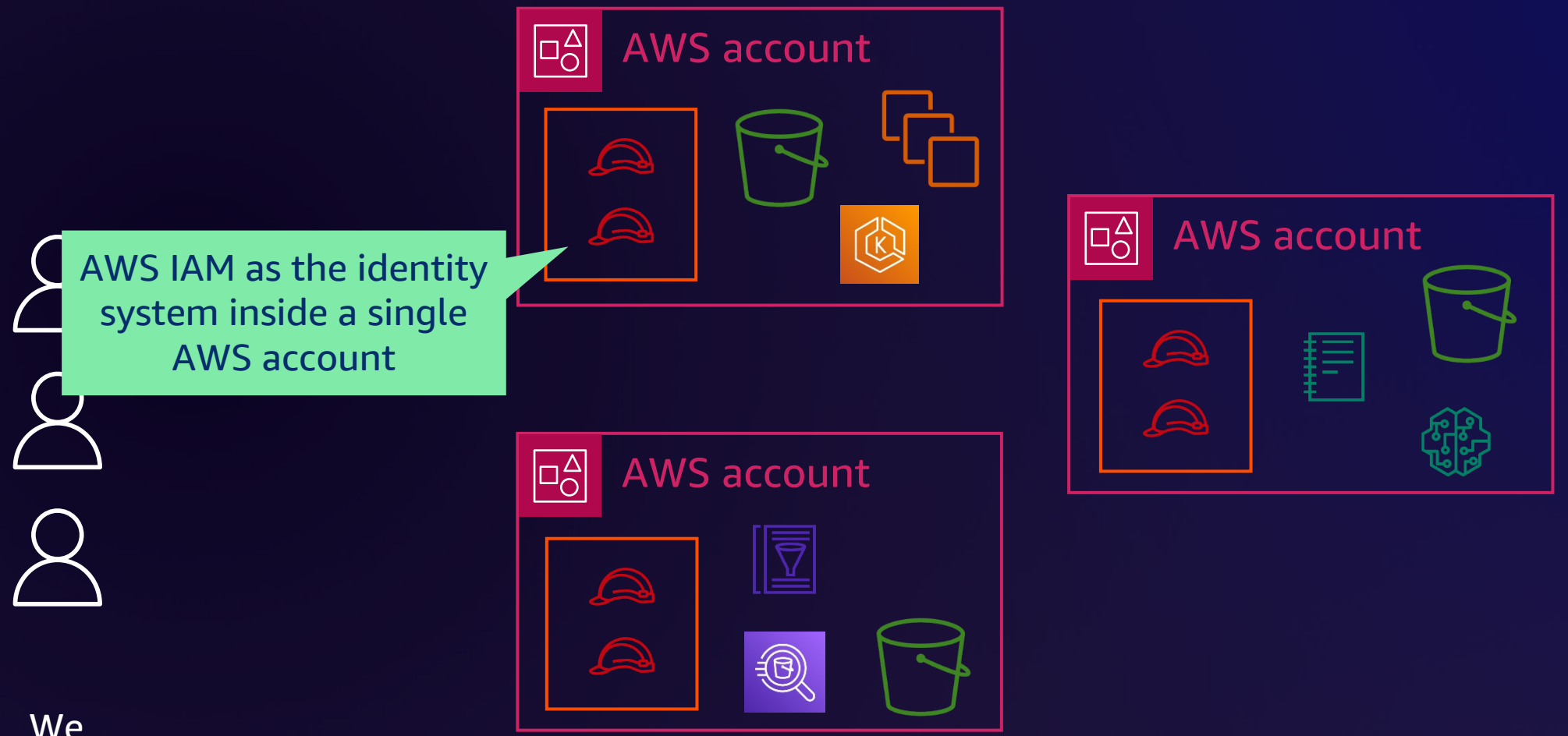


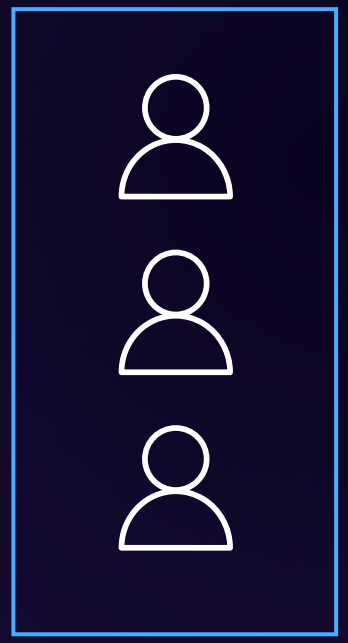


We

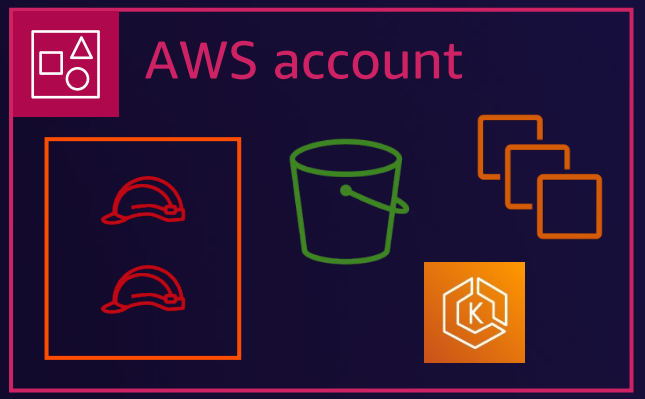
An identity system!

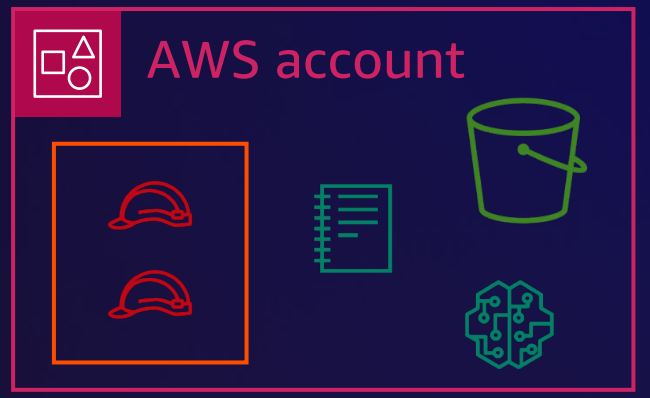
Cloud infrastructure that we own

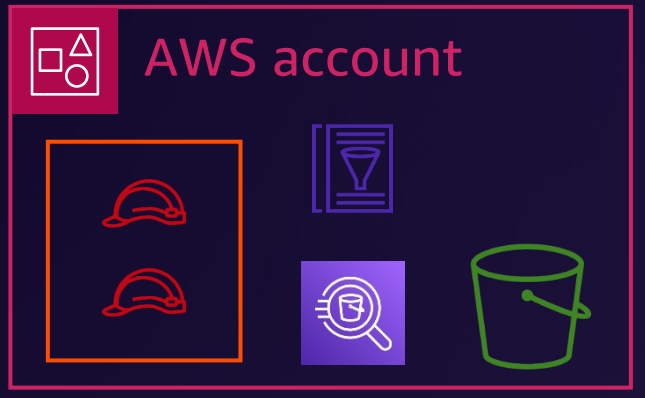




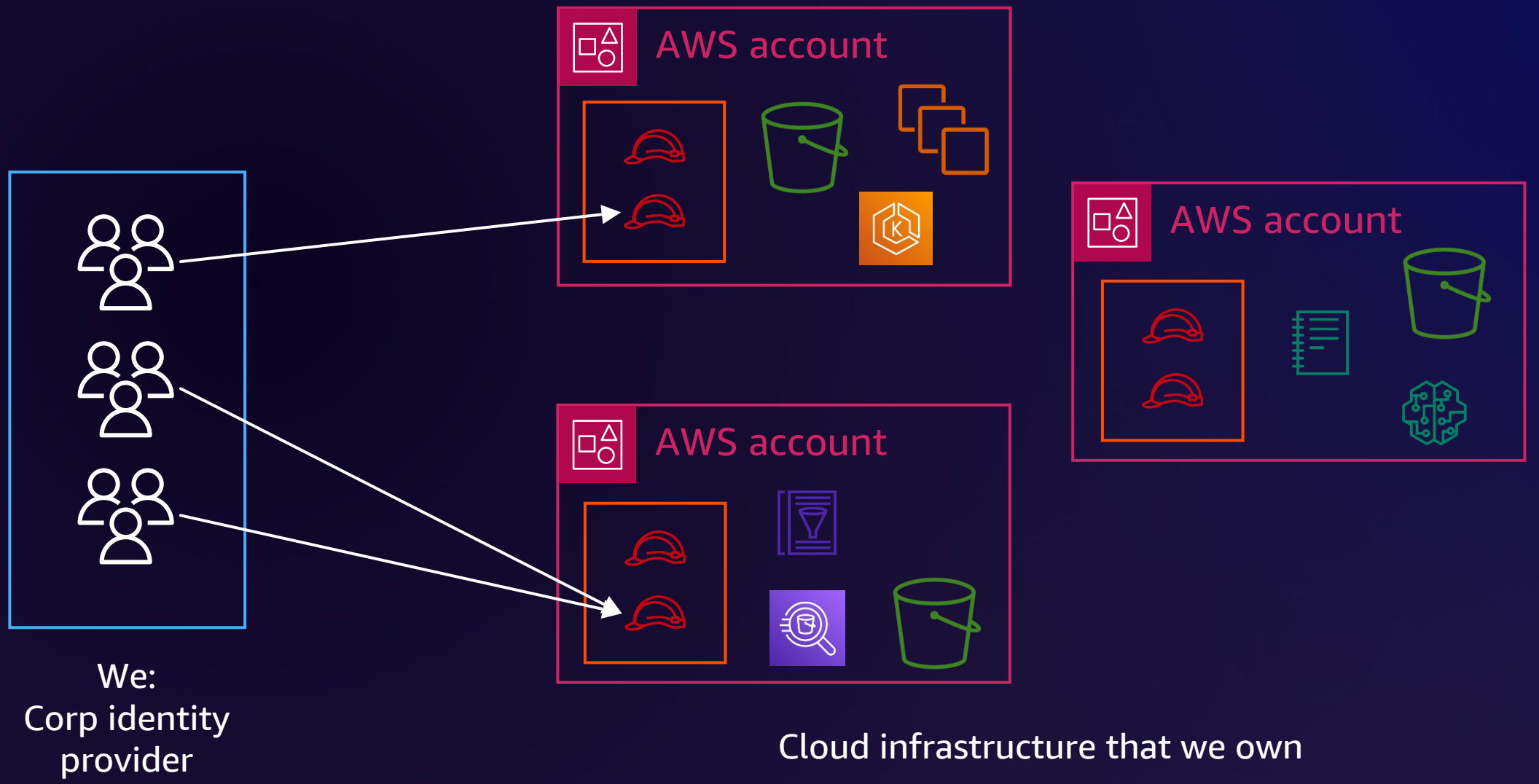
We:
Corp identity provider



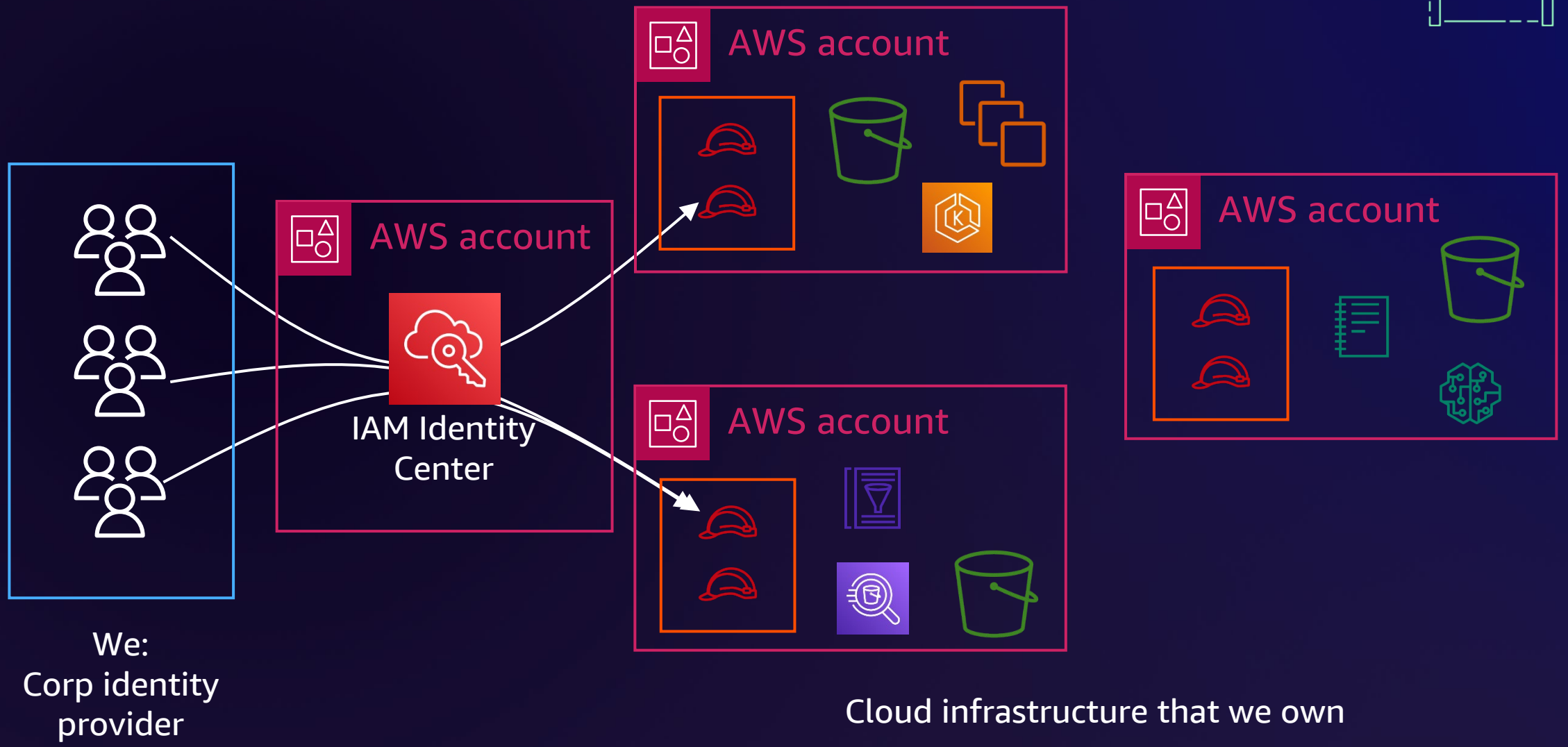




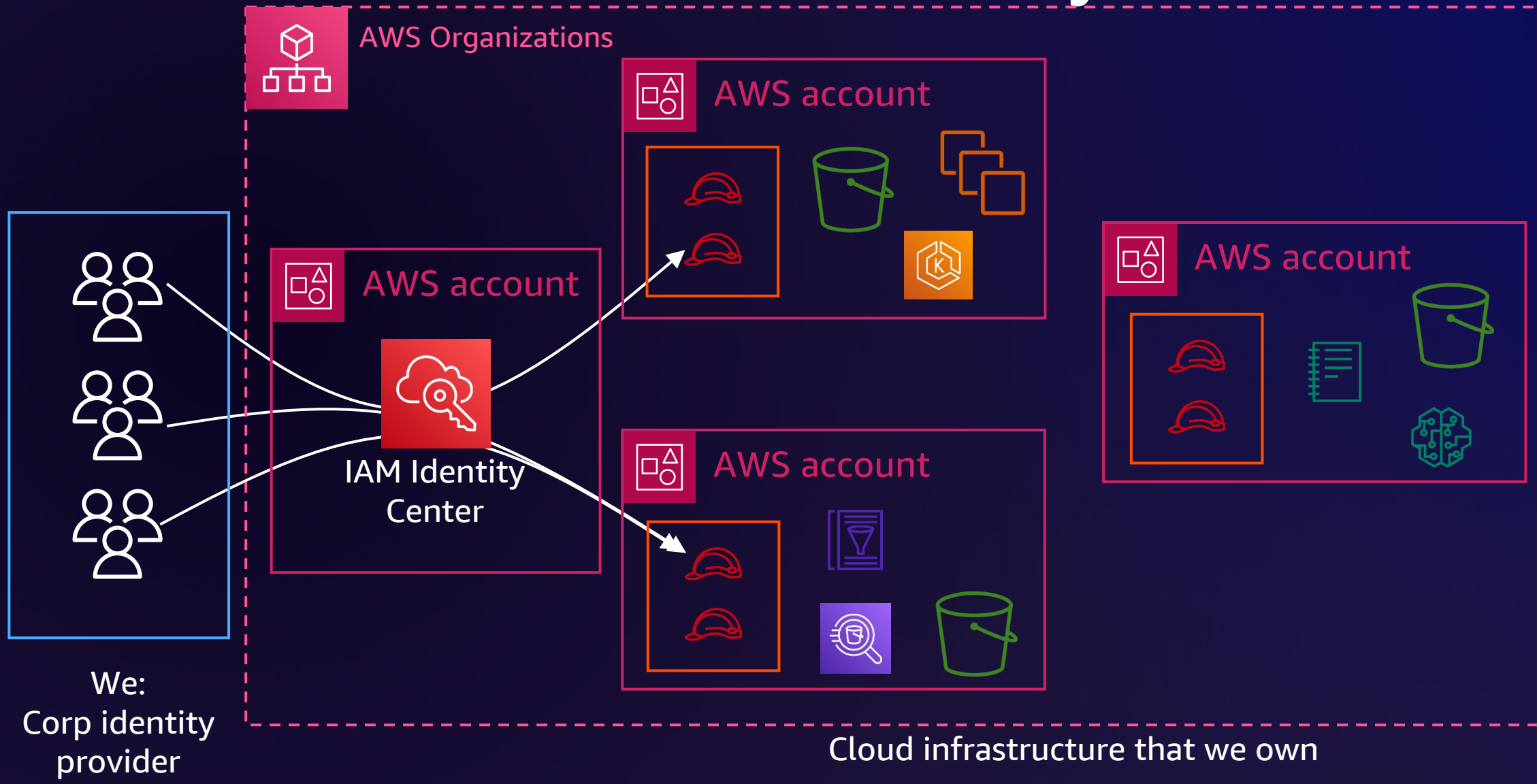
Cloud infrastructure that we own



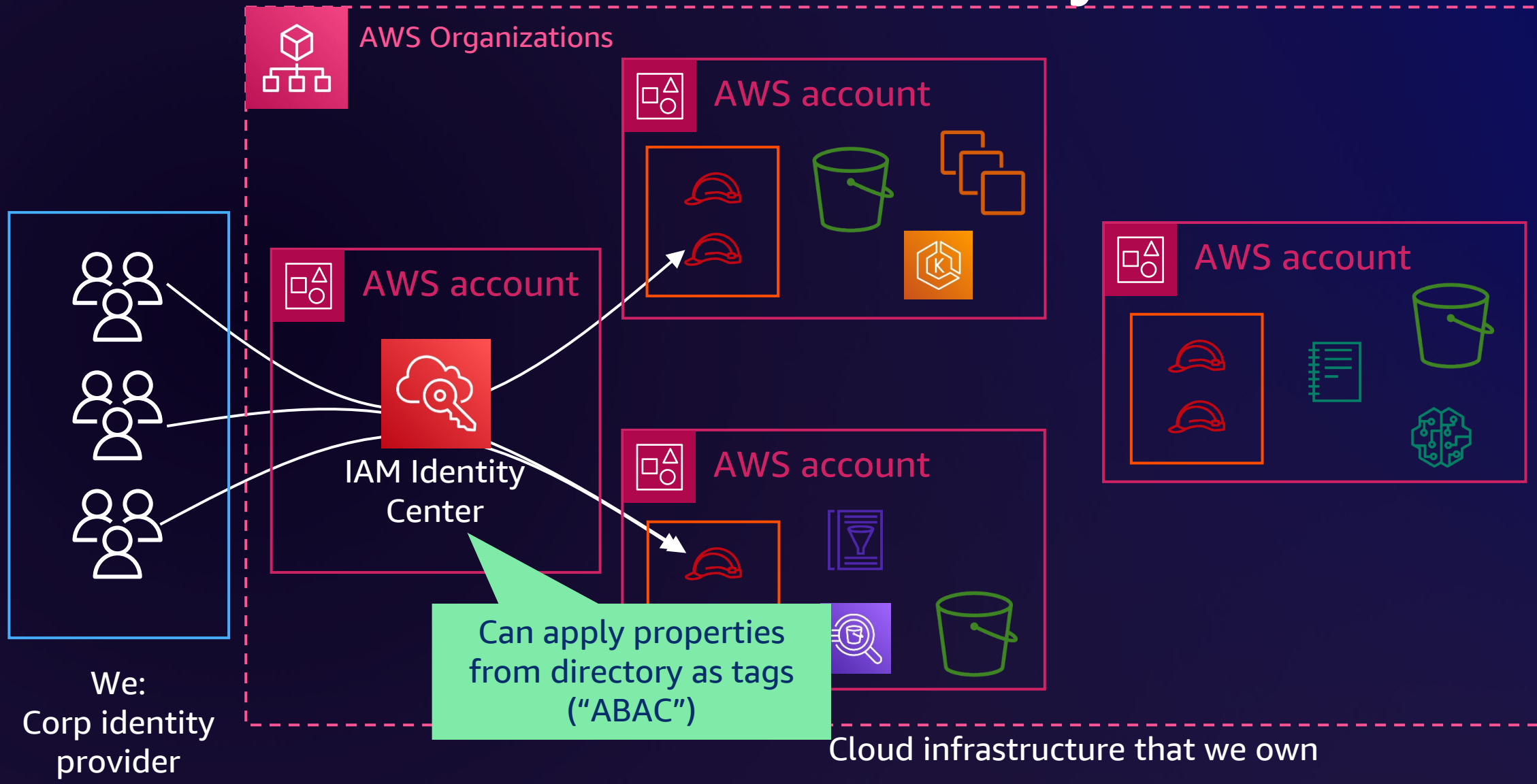
User access to IAM roles: IAM Identity Center



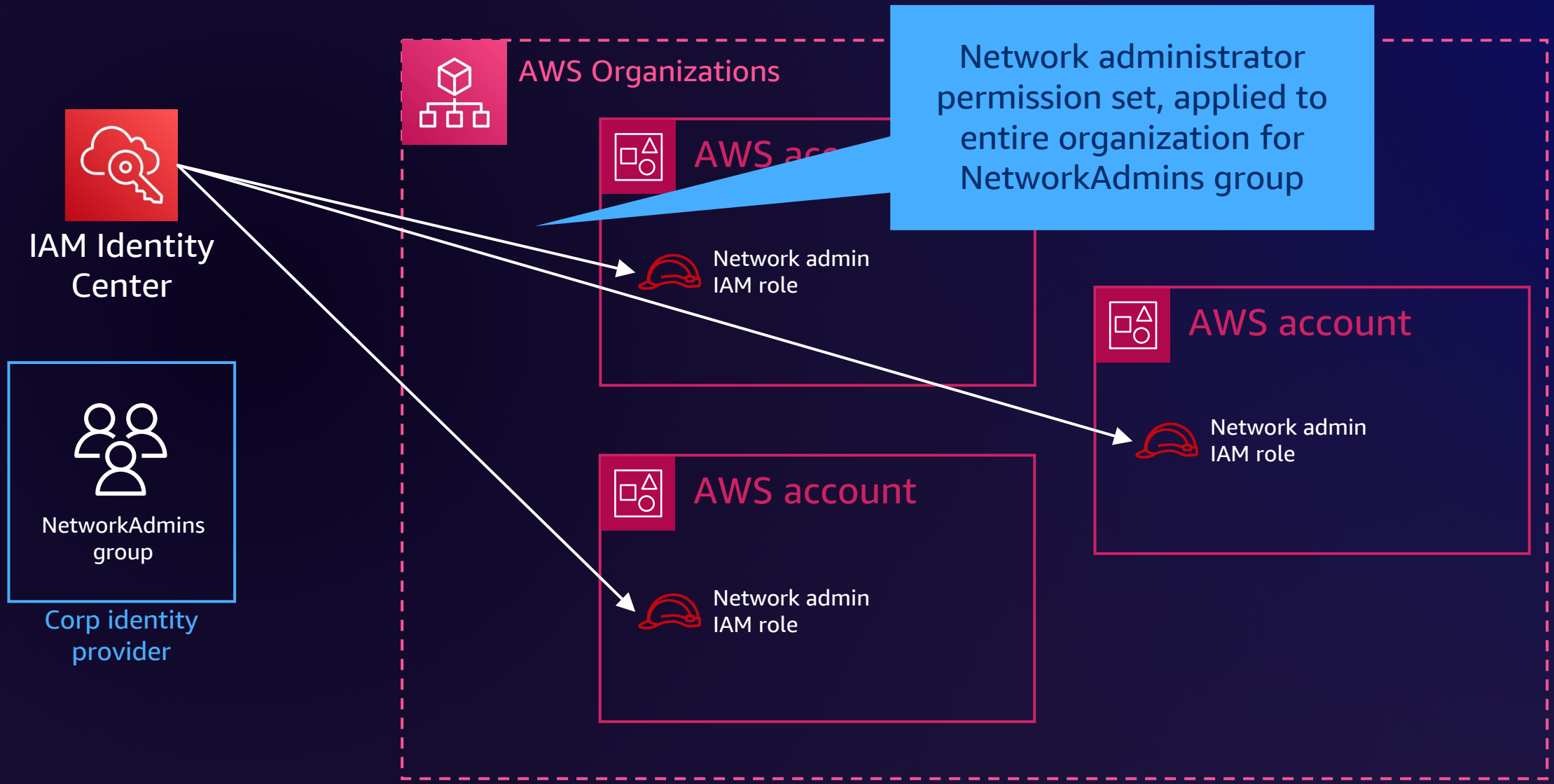
User access to IAM roles: IAM Identity Center



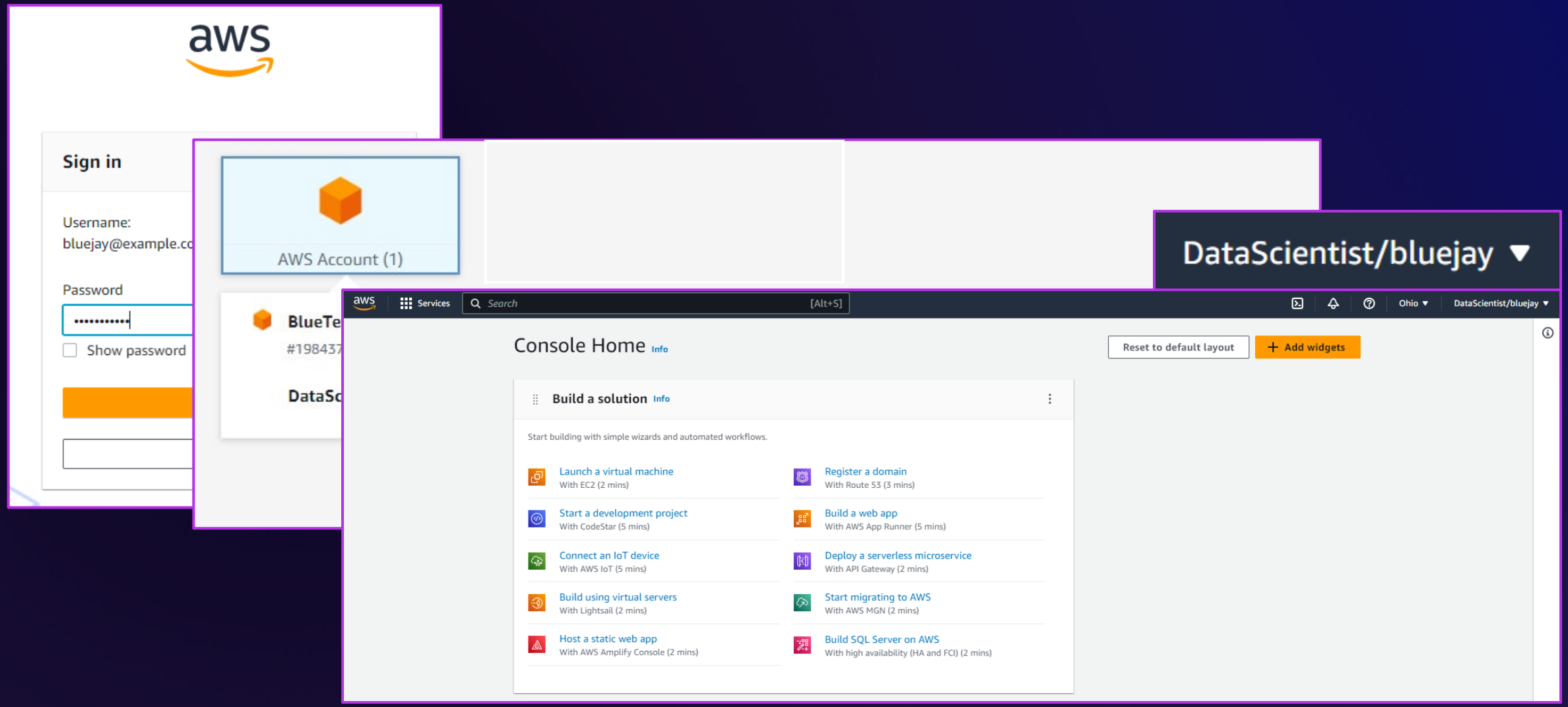
User access to IAM roles: IAM Identity Center



IAM Identity Center permission sets

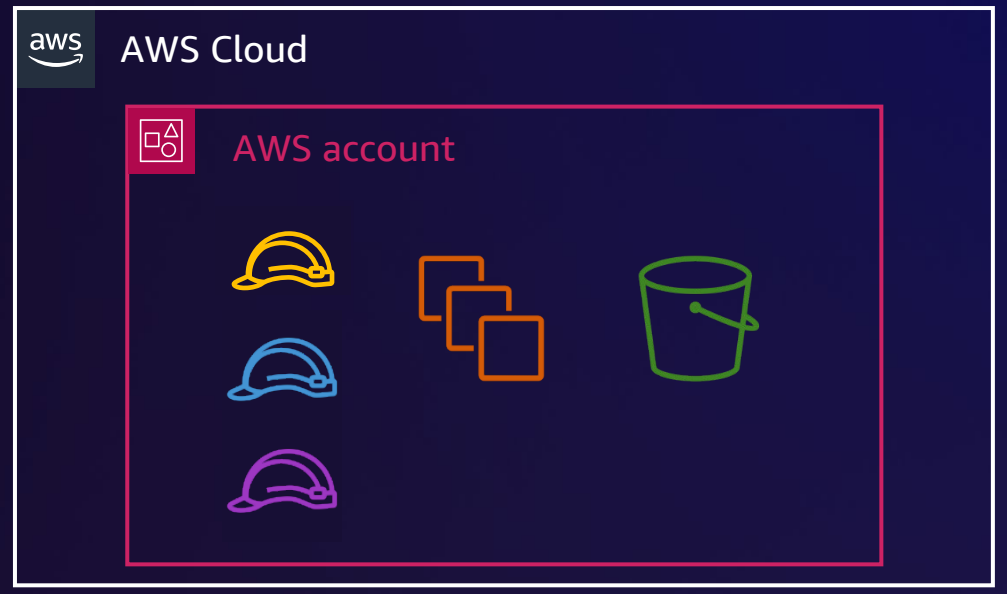


User access to AWS via IAM Identity Center



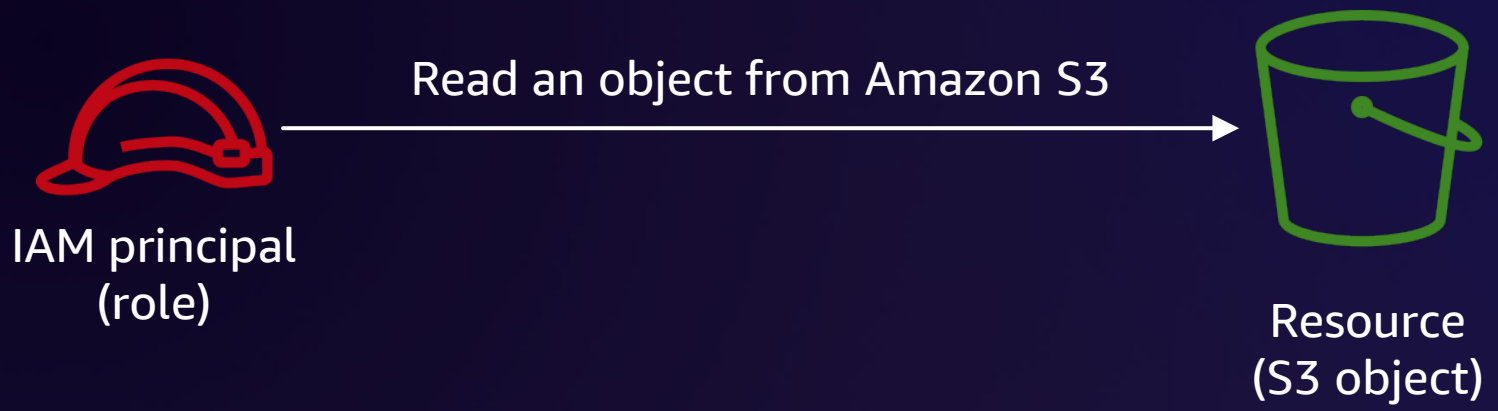
Principles of principals: Summary

- IAM roles for AWS compute environments
- IAM roles for non-AWS compute environments via IAM Roles Anywhere
- IAM roles for federated access by cloud operators (e.g., through IAM Identity Center)



Fundamentals of IAM: Authentication

Authentication example: A request to Amazon S3



Authentication example: A request to Amazon S3

```
// AWS Javascript V3 SDK example
import {S3Client, GetObjectCommand} from "@aws-sdk/client-s3";

const s3Client = new S3Client({});

const obj = await s3Client.send(new GetObjectCommand({
  Bucket: 'example-bucket',
  Key: 'path/to/obj'
});

// Object contents will be in obj.Body
```

IAM principal
(Lambda function role)

Resource
(S3 object)

Authentication example: A request to Amazon S3

```
// AWS Javascript V3 SDK example
import {S3Client, GetObjectCommand} from "@aws-sdk/client-s3";

const s3Client = new S3Client({});

const obj = await s3Client.send(new GetObj      mmand({
  Bucket: 'example-bucket',
  Key: 'path/to/obj'
});

// Object contents will be in obj.Body
```

IAM principal
(Lambda function role)

The AWS SDK typically knows where to find the right credentials, though you can also configure them manually here

Resource
(S3 object)

Authentication example: A request to Amazon S3

```
# AWS command-line interface (CLI)

> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out
```

IAM
Identity Center

IAM principal
(federated identity)

Resource
(S3 object)

Authentication example: A request to Amazon S3

IAM
Identity Center

```
# AWS command-line interface (CLI)

> aws sso login
… IAM Identity Center login workflow …

> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out
```

Configures your CLI session with credentials from IAM Identity Center

IAM principal
(federated identity)

Resource
(S3 object)

Authentication example: A request to Amazon S3

```
> aws s3api get-object --bucket example-bucket --key path/to/obj --debug /tmp/obj.out

… Lots of debug output…
```

IAM principal

Resource
(S3 object)

Authentication example: A request to Amazon S3

```
> aws s3api get-object --bucket example-bucket --key path/to/obj --debug /tmp/obj.out

…
2023-04-20 13:57:04,144 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
GET
/path/to/obj

host:example-bucket.s3.amazonaws.com
…
```

IAM principal

Resource
(S3 object)

Authentication example: A request to Amazon S3

```
> aws s3api get-object --bucket example-bucket --key path/to/obj --debug /tmp/obj.out

…
2023-04-20 13:57:04,144 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
GET
/path/to/obj

host:example-bucket.s3.amazonaws.com
…
```

IAM principal

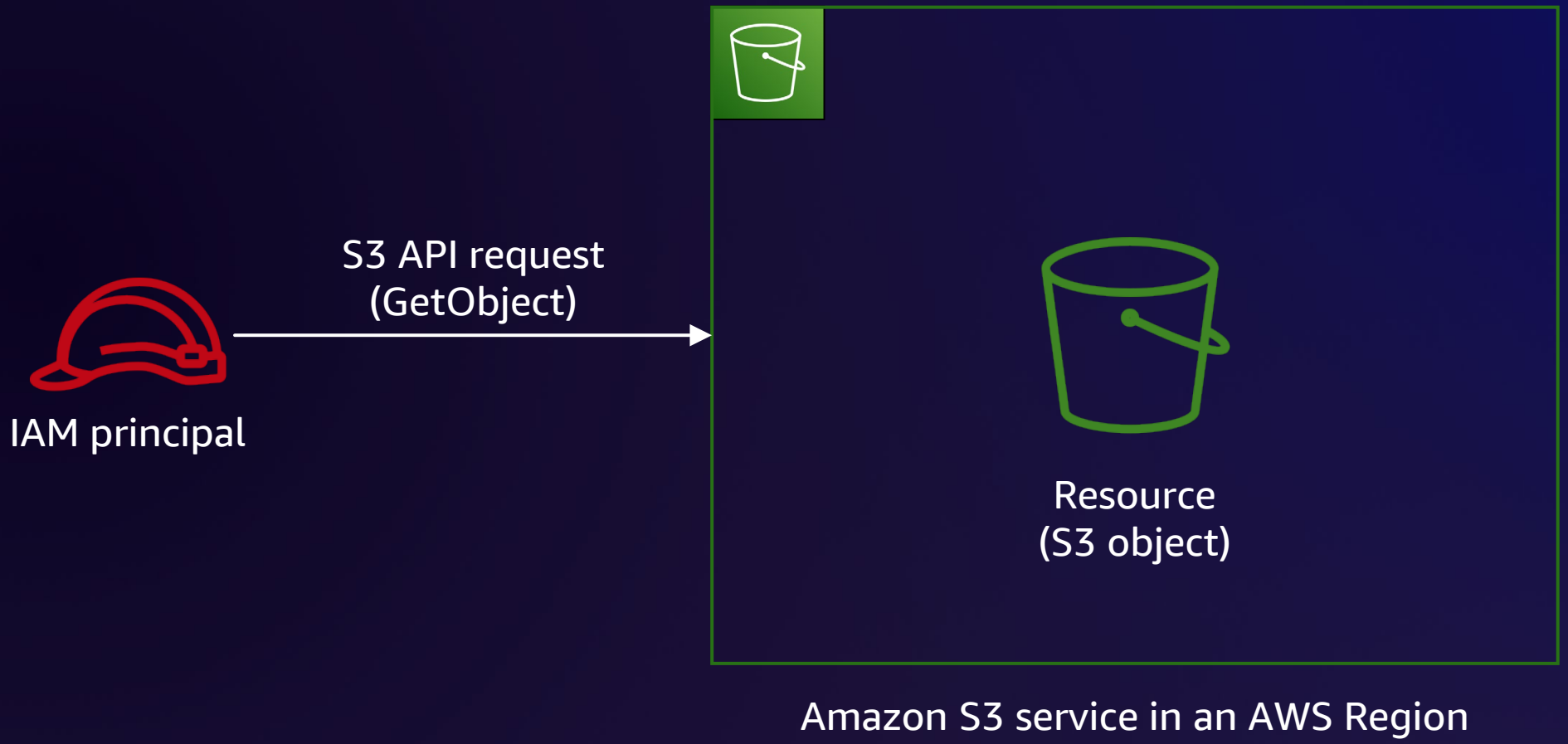
```
> dig example-bucket.s3.amazonaws.com

…
;; ANSWER SECTION:
example-bucket.s3.amazonaws.com. 42821 IN CNAME s3-1-w.amazonaws.com.
s3-1-w.amazonaws.com.    60      IN      CNAME   s3-w.us-east-1.amazonaws.com.
s3-w.us-east-1.amazonaws.com. 2 IN      A       52.216.208.41
… more records…
```

Resc
(S3 o

Amazon S3 in us-east-1

Authentication example: A request to Amazon S3



Authentication example: A request to Amazon S3

```
> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out

…
2023-04-20 13:57:04,144 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
GET
/path/to/obj

host:example-bucket.s3.amazonaws.com
…

2023-04-20 13:57:04,145 - MainThread - botocore.endpoint - DEBUG - Sending http request:
<AWSPreparedRequest stream_output=True, method=GET, url=https://example-
bucket/path/to/obj, headers={ … , 'Authorization': 'AWS4-HMAC-SHA256
Credential=ASIAEXAMPLE/20230420/us-east-1/s3/aws4_request, SignedHeaders=host;x-amz-
content-sha256;x-amz-date;x-amz-security-token, Signature=a76b58…8', 'X-Amz-Security-
Token': 'a-base64-value', … }>
```

IAM princip

Resource
(S3 object)

Amazon S3 service in an AWS Region

Authentication example: A request to Amazon S3

Access key ID:
Identifies the requester

```
> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out

…
2023-04-20 13:57:04,144 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
GET
/path/to/obj

host:example-bucket.s3.amazonaws.com
…

2023-04-20 13:57:04,145 - MainThread - botocore.endpoint - DEBUG - Sending http request:
<AWSPreparedRequest stream_output=True, method=GET, url=https://example-
bucket/path/to/obj, headers={ … , 'Authorization': 'AWS4-HMAC-SHA256
Credential=ASIAEXAMPLE/20230420/us-east-1/s3/aws4_request, SignedHeaders=host;x-amz-
content-sha256;x-amz-date;x-amz-security-token, Signature=a76b58…8', 'X-Amz-Security-
Token': 'a-base64-value', … }>
```

IAM princip

Resource
(S3 object)

Amazon S3 service in an AWS Region

Authentication example: A request to Amazon S3

Access key ID:
Identifies the requester

```
> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out

…
2023-04-20 13:57:04,144 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
GET
/path/to/obj

host:example-bucket.s3.amazonaws.com
…

2023-04-20 13:57:04,145 - MainThread - botocore.endpoint - DEBUG - Sending http request:
<AWSPreparedRequest stream_output=True, method=GET, url=https://example-
bucket/path/to/obj, headers={ … , 'Authorization': 'AWS4-HMAC-SHA256
Credential=ASIAEXAMPLE/20230420/us-east-1/s3/aws4_request, SignedHeaders=host;x-amz-
content-sha256;x-amz-date;x-amz-security-token, Signature=a76b58…8', 'X-Amz-Security-
Token': 'a-base64-value', … }>
```

Signature of request:
Produced client-side (CLI)
with session credentials

IAM principal

Resource
(S3 object)

Amazon S3 service in an AWS Region

Authentication example: A request to Amazon S3

```
> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out

…
2023-04-20 13:57:04,144 - MainThread - botocore.auth - DEBUG - CanonicalRequest:
GET
/path/to/obj

host:example-bucket.s3.amazonaws.com
…

2023-04-20 13:57:04,145 - MainThread - botocore.endpoint - DEBUG - Sending http request:
<AWSPreparedRequest stream_output=True, method=GET, url=https://example-
bucket/path/to/obj, headers={ … , 'Authorization': 'AWS4-HMAC-SHA256
Credential=ASIAEXAMPLE/20230420/us-east-1/s3/aws4_request, SignedHeaders=host;x-amz-
content-sha256;x-amz-date;x-amz-security-token, Signature=a76b58…8', 'X-Amz-Security-
Token': 'a-base64-value', … }>
```

Access key ID:
Identifies the requester

Signature of request:
Produced client-side (CLI)
with session credentials

IAM principal

Resource
(S3 object)

Token associated with the
current role session

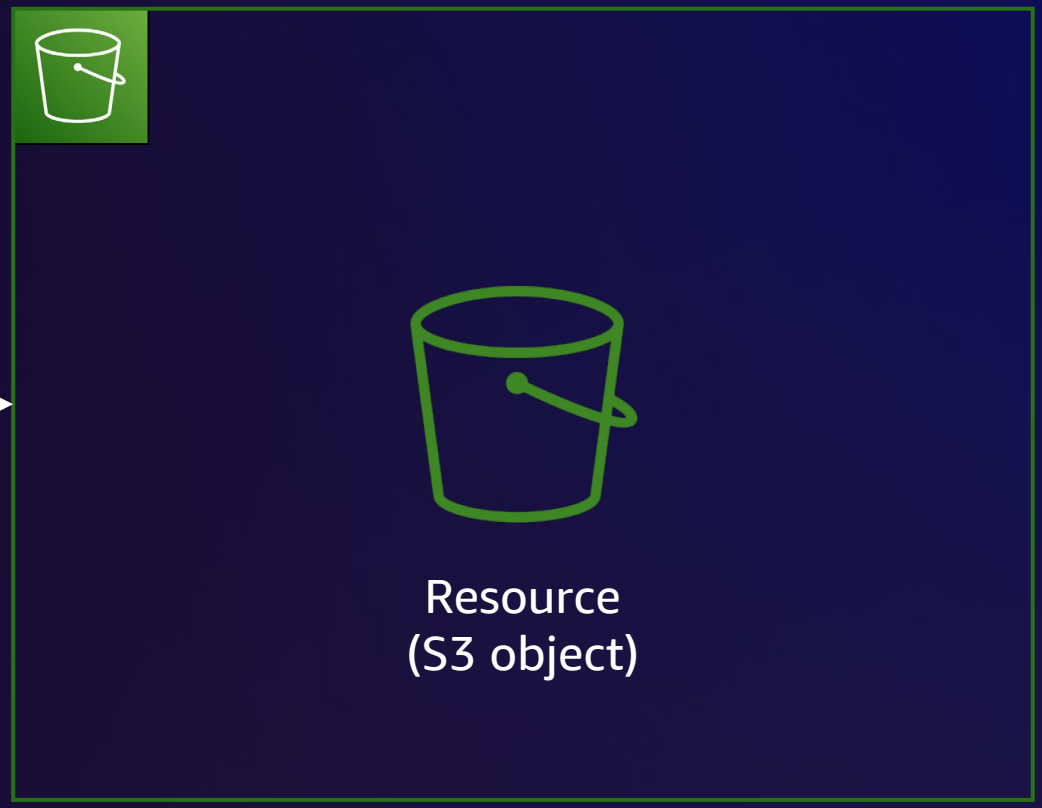
Amazon S3 service in an AWS Region

Authentication example: A request to Amazon S3

```
> aws s3api get-object --bucket example-bucket \
                       --key path/to/obj \
                       /tmp/obj.out
```

IAM principal

Amazon S3 API request (GetObject)



Amazon S3 service in an AWS Region

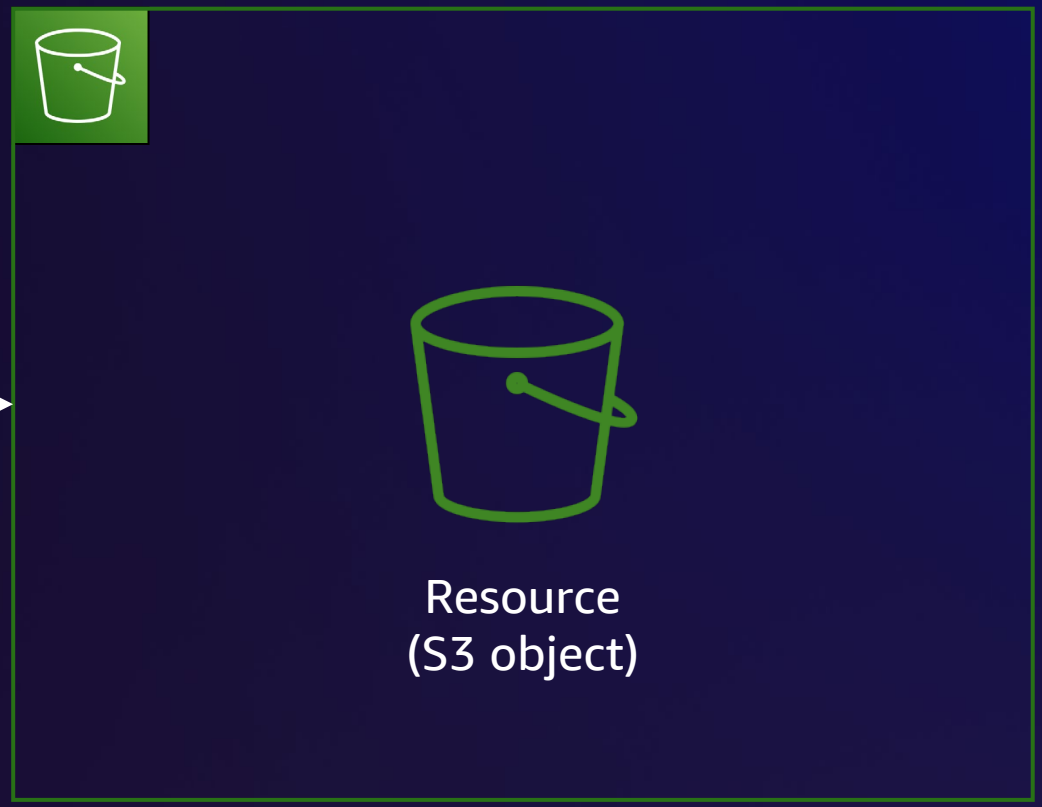
Authentication example: A request to Amazon S3

IAM data plane
in the region

```
> aws s3api get-object --bucket example-bucket \
                       --key path/to/obj \
                       /tmp/obj.out
```

IAM principal

SigV4-signed
request



Resource
(S3 object)

Amazon S3 service in an AWS Region

Authentication example: A request to Amazon S3

IAM data plane
in the region

```
> aws s3api get-object --bucket example-bucket \
                       --key path/to/obj \
                       /tmp/obj.out
```

IAM principal

Authenticated identity

Resource
(S3 object)

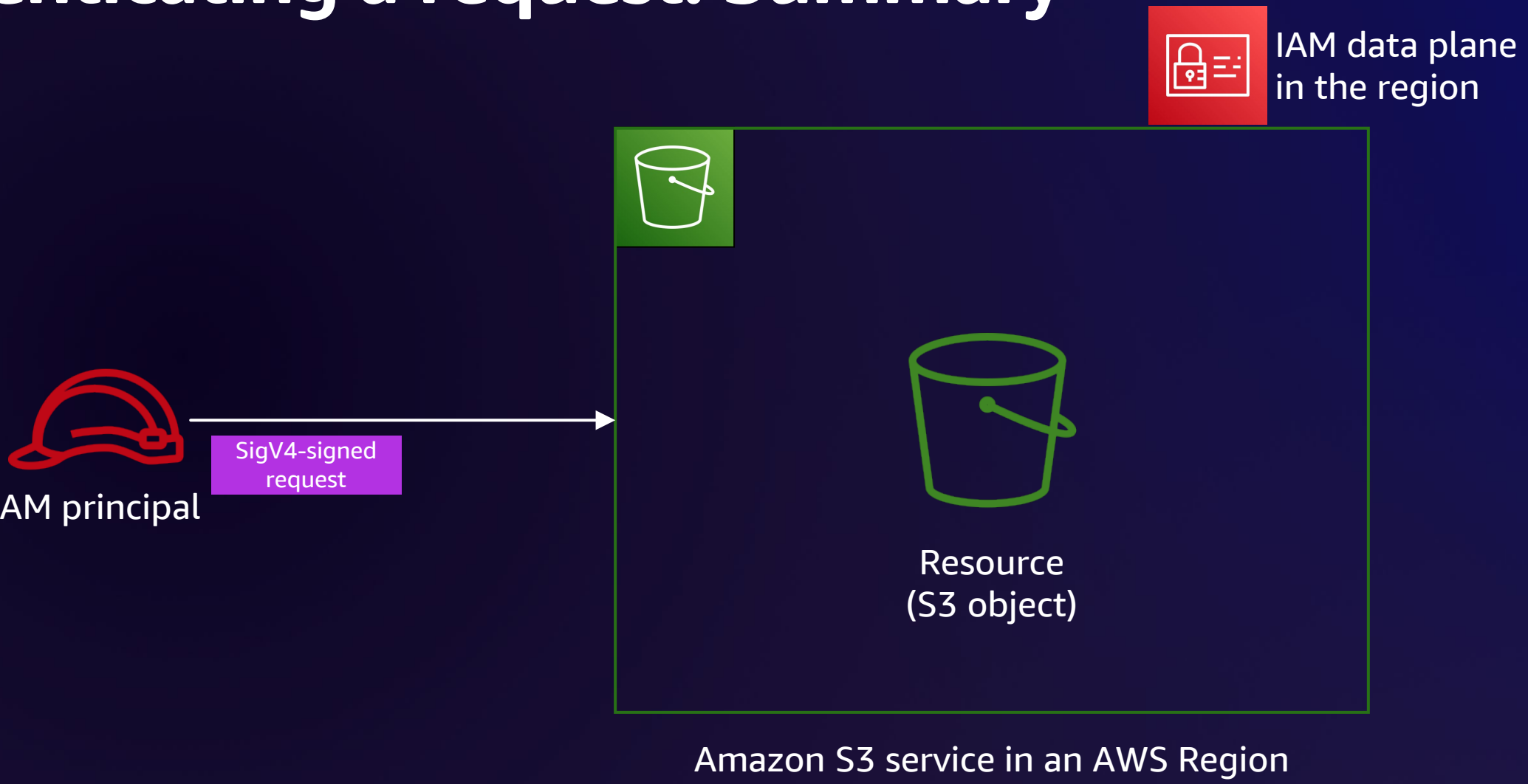
Amazon S3 service in an AWS Region

Successful authentication ➔ Identity properties

Successful authentication ➔ Identity policies

Amazon S3 service in an AWS Region

Authenticating a request: Summary

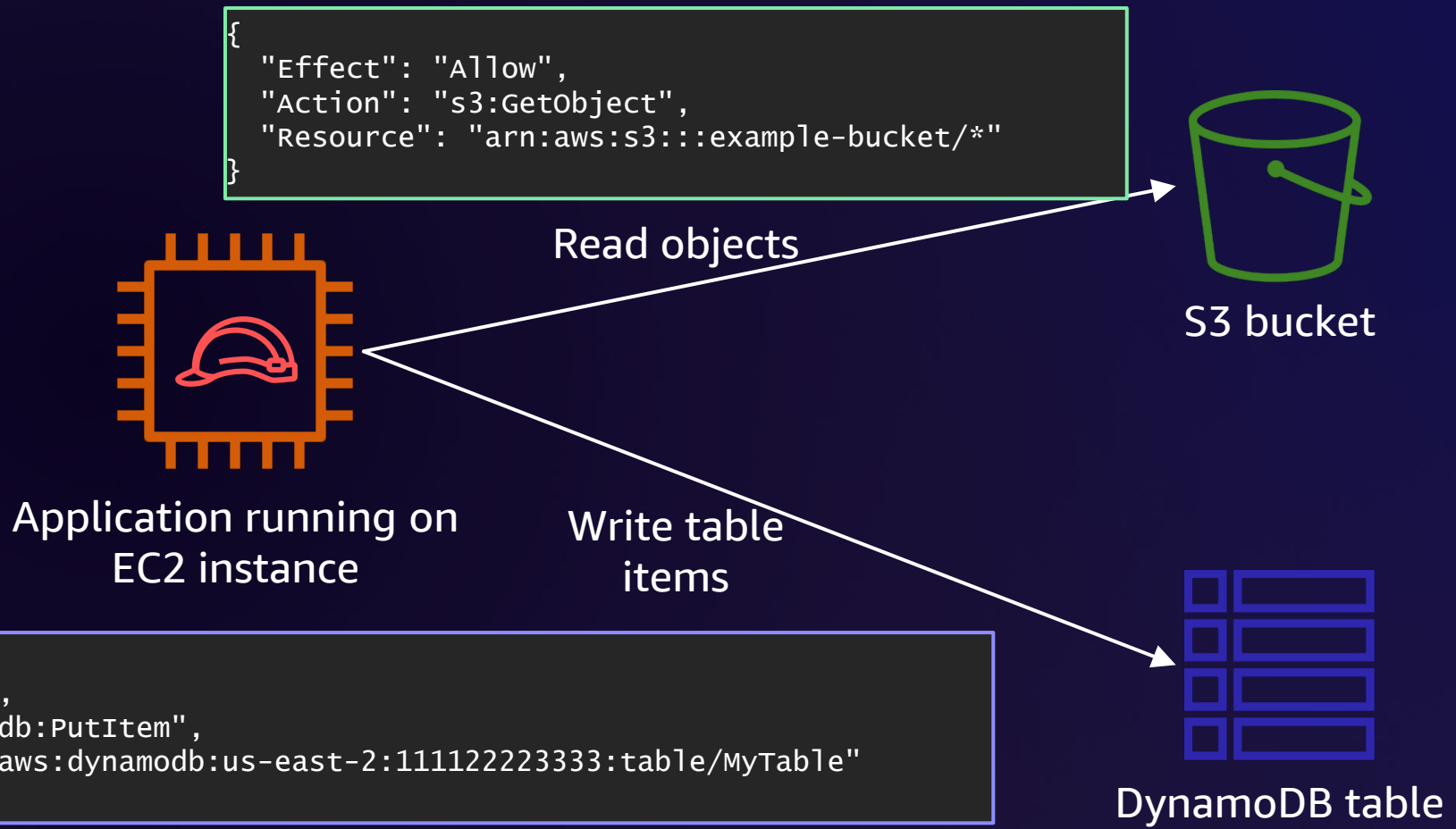


Fundamentals of IAM: Authorization

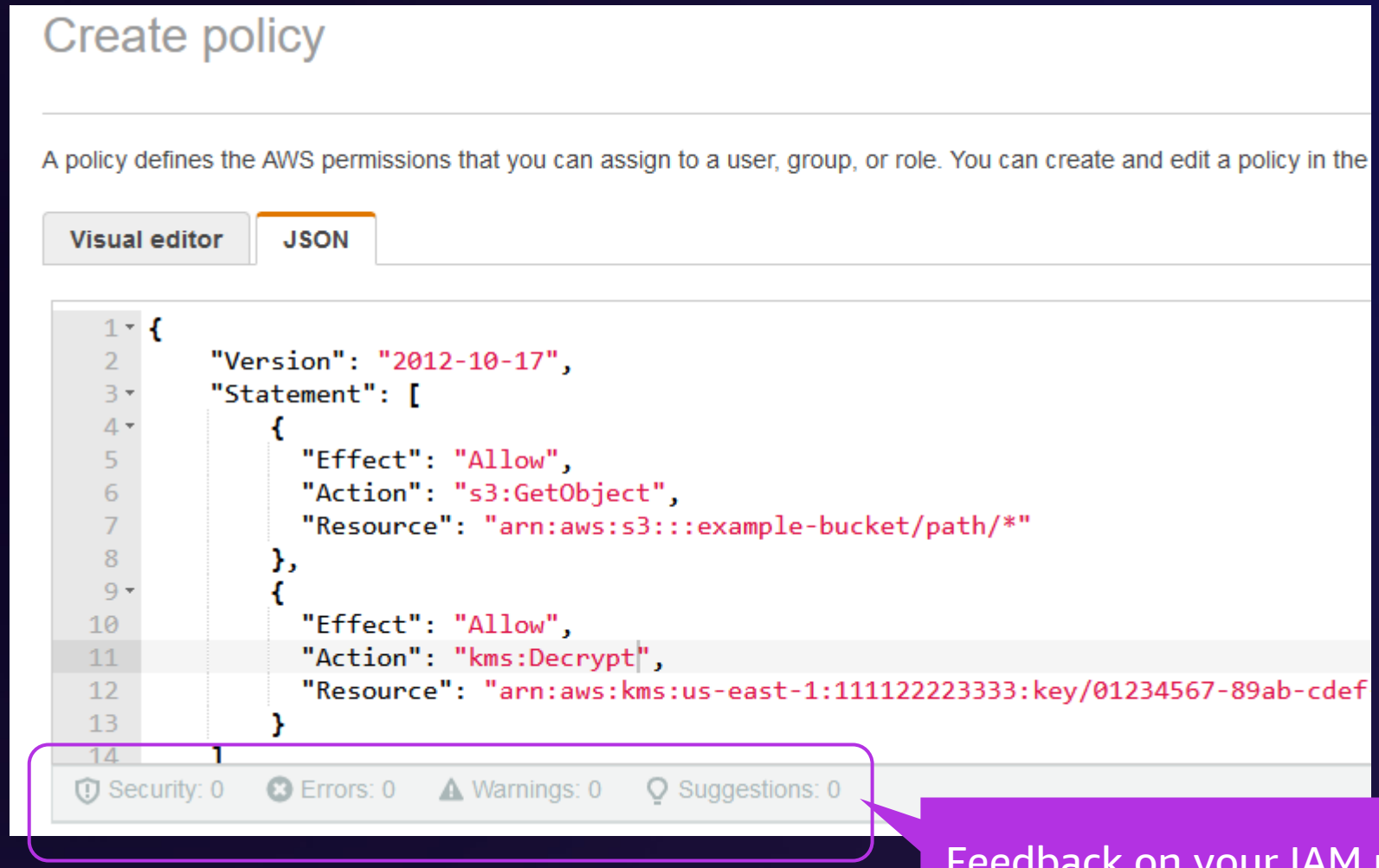
IAM's little secret: Authorization is string-matching

Policy document[s]

Workloads tend to need specific IAM permissions

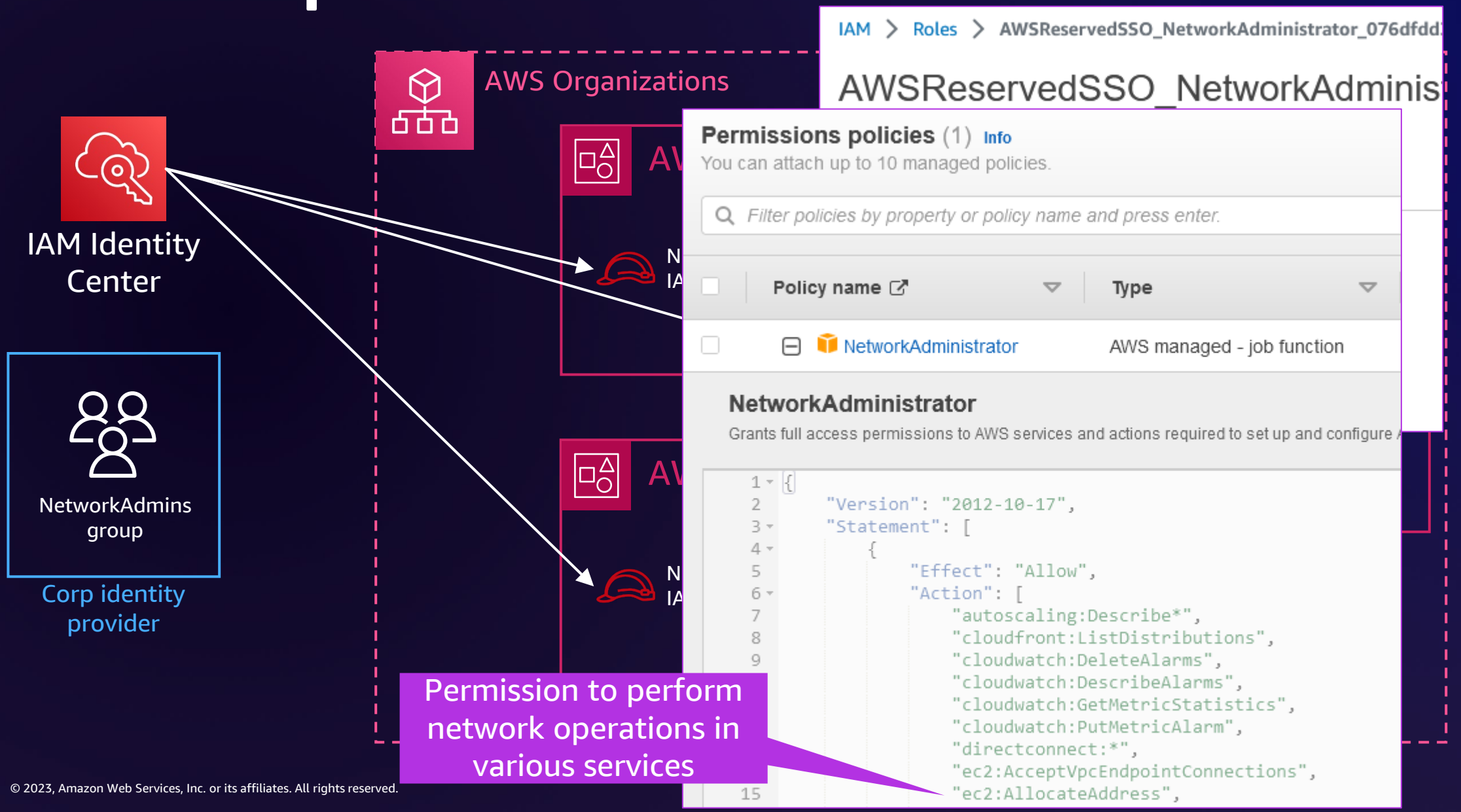


Writing custom IAM policies

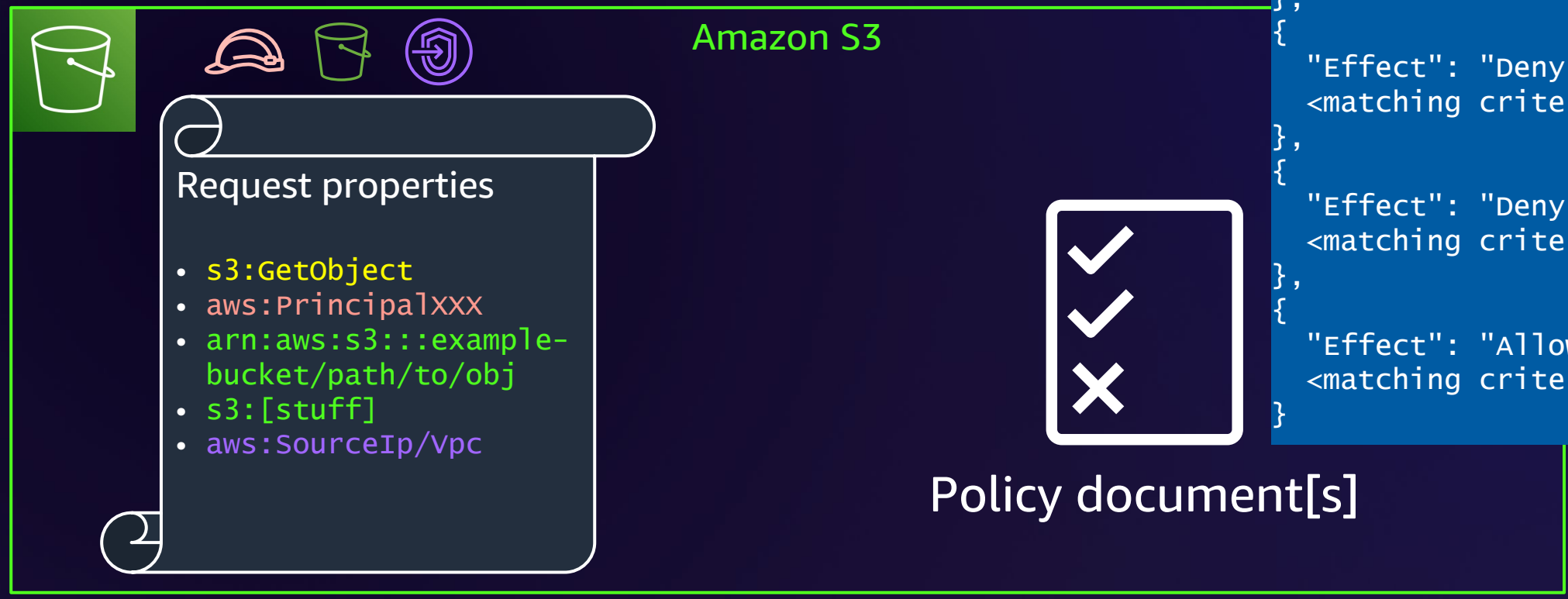


Feedback on your IAM policy

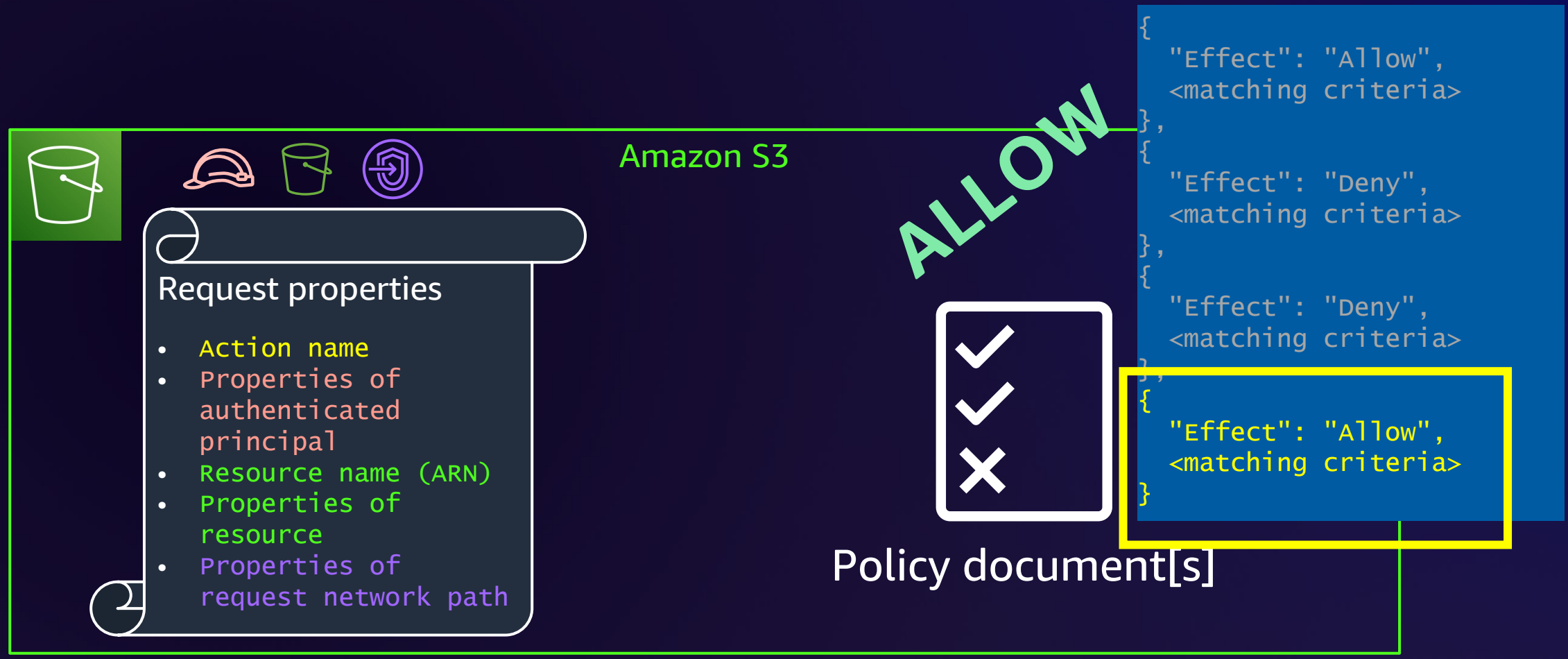
Policies for operator access: Permission sets



IAM's little secret: Authorization is string-matching



IAM's little secret: Authorization is string-matching



IAM's little secret: Authorization is string-matching

IAM's little secret: Authorization is string-matching

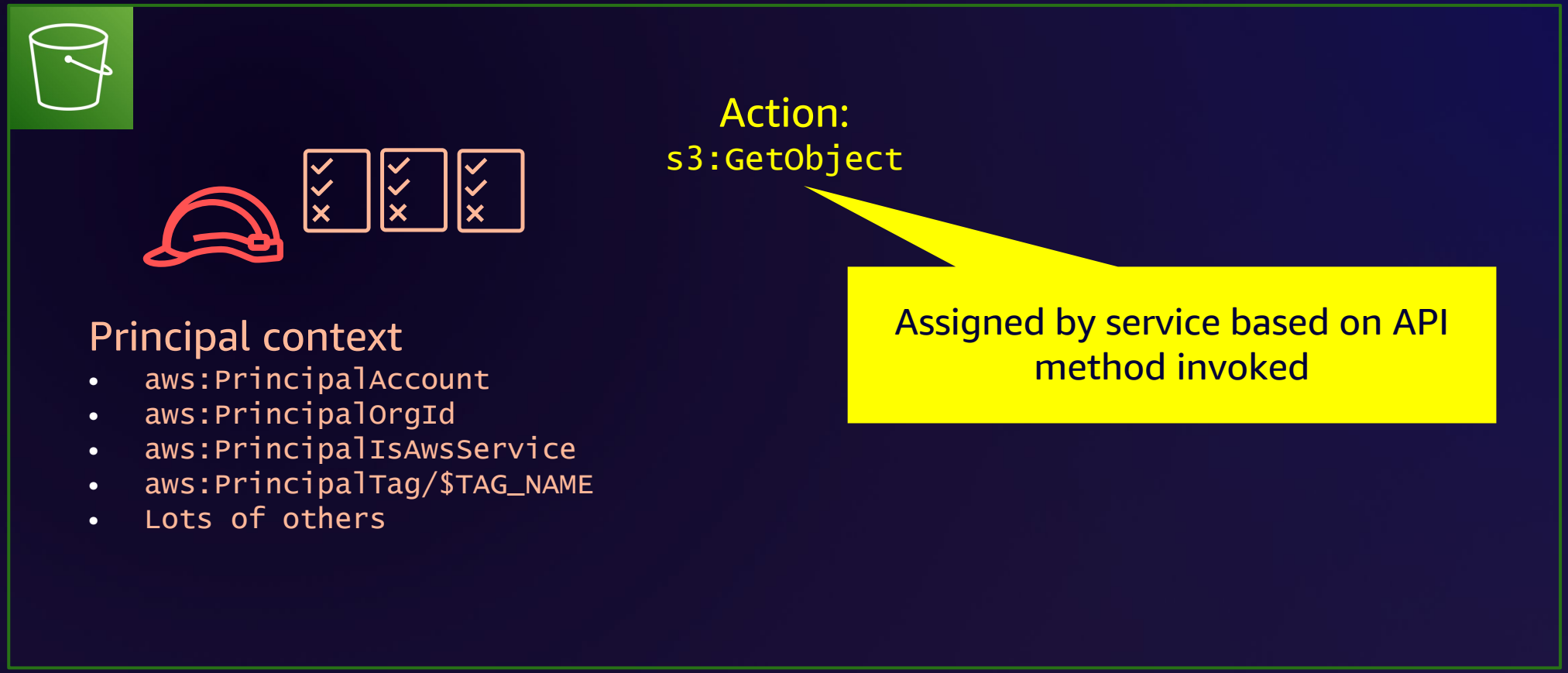
How authorization works on AWS

Principal context

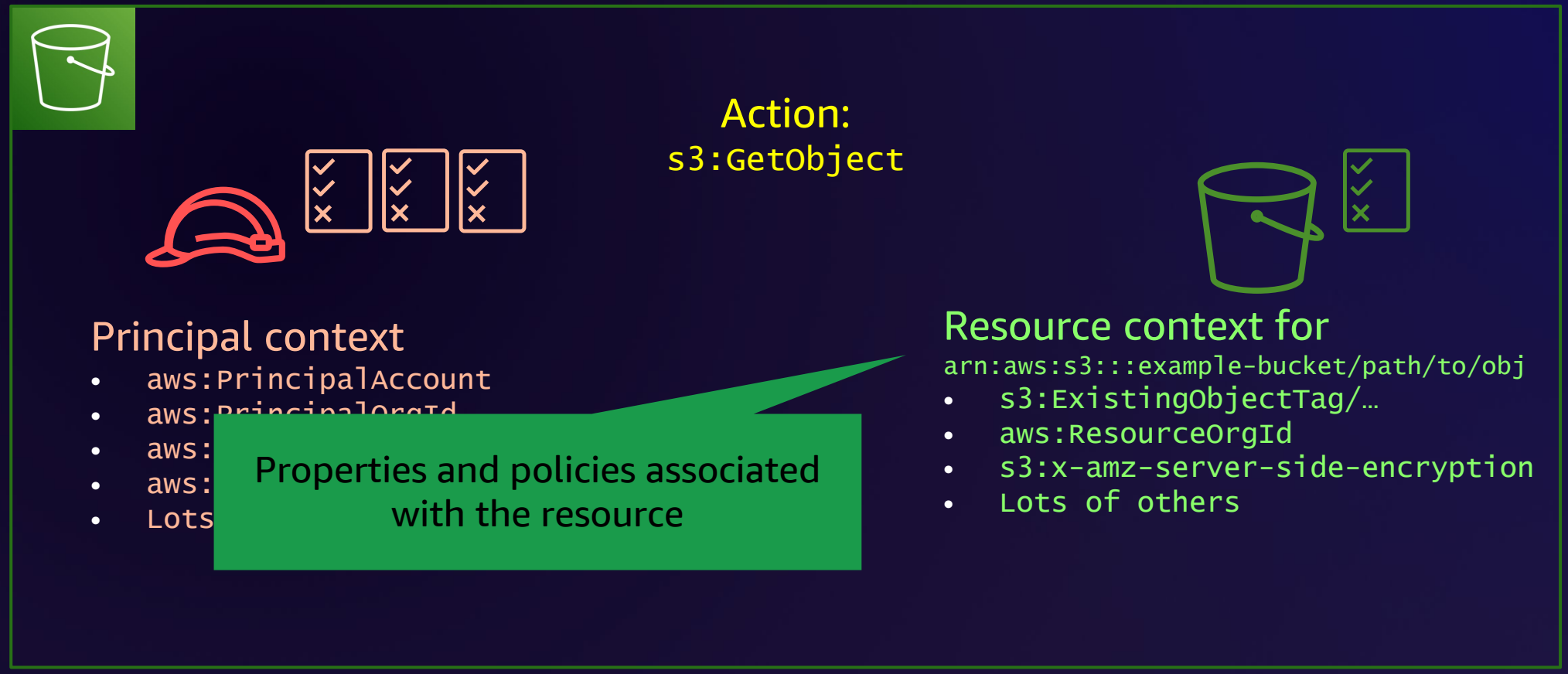
- aws:PrincipalAccount
- aws:PrincipalOrgId
- aws:PrincipalIsAwsService
- aws:PrincipalTag/$TAG_NAME
- Lots of others

Result of successful authentication:
Properties and policies of the caller

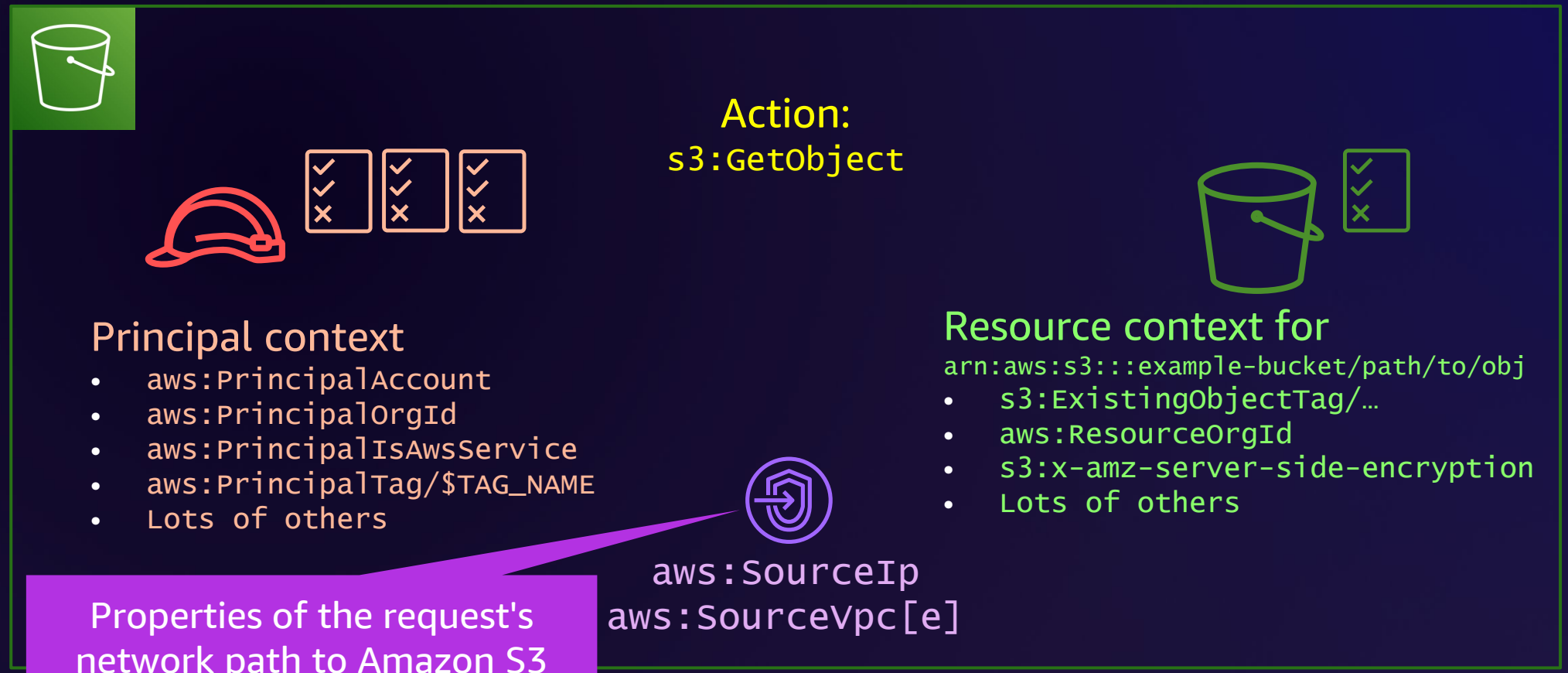
How authorization works on AWS



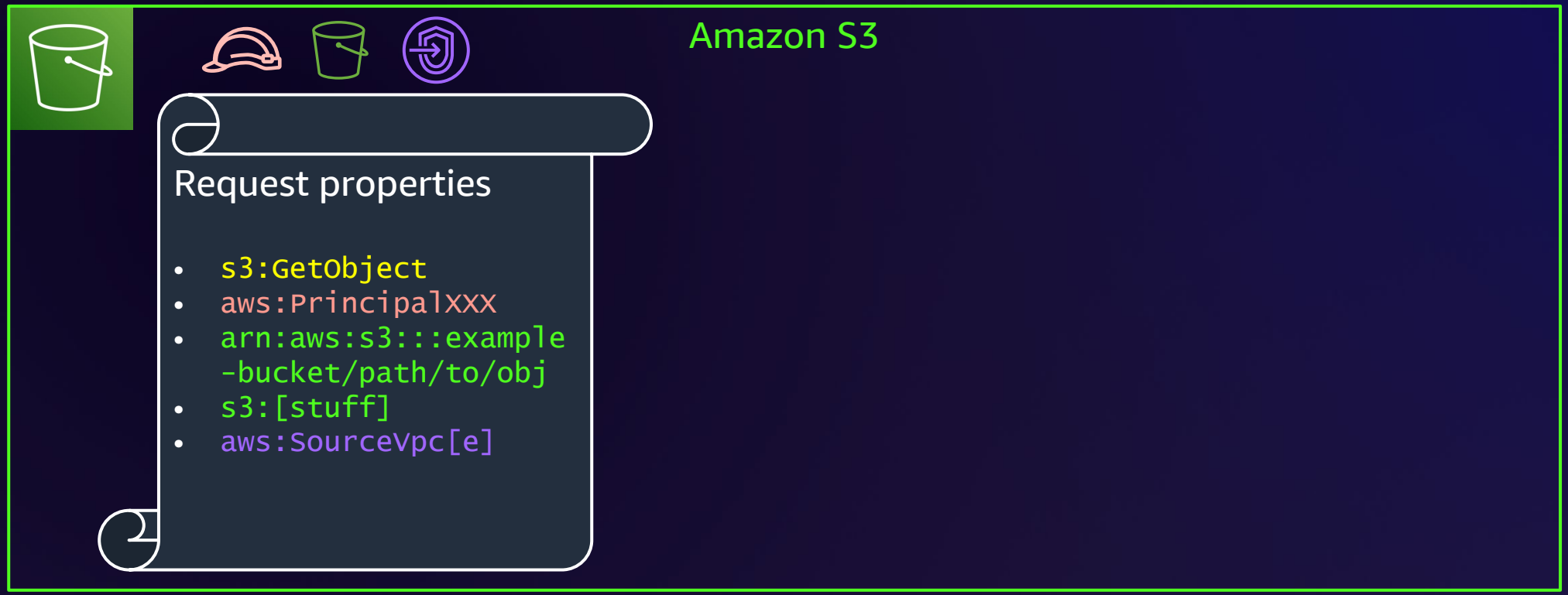
How authorization works on AWS



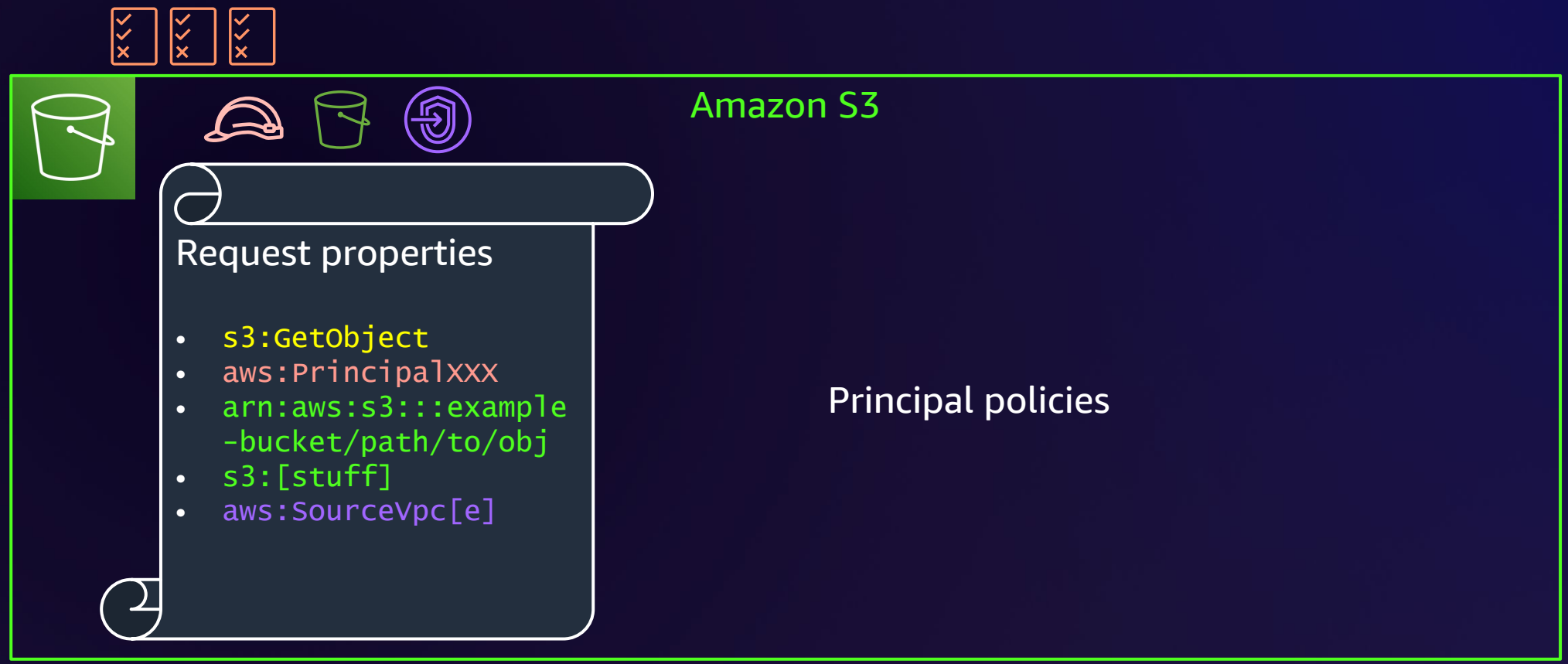
How authorization works on AWS



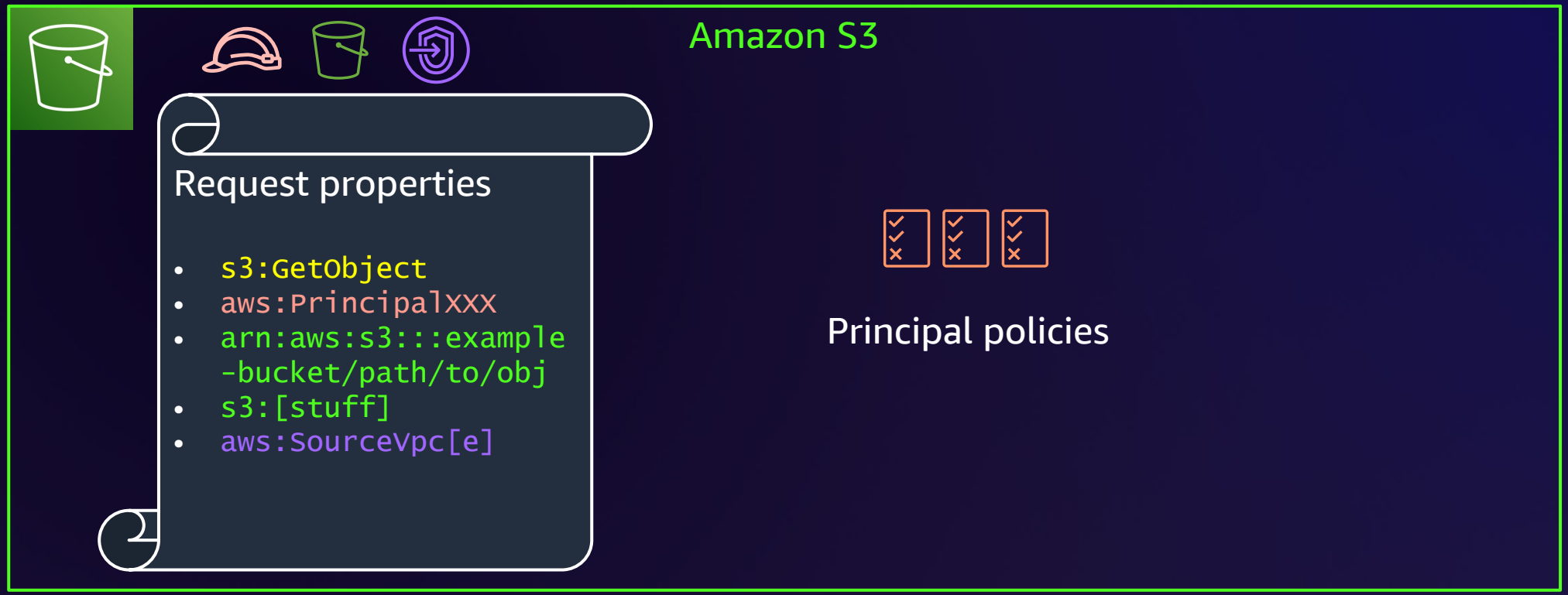
The authorization request context



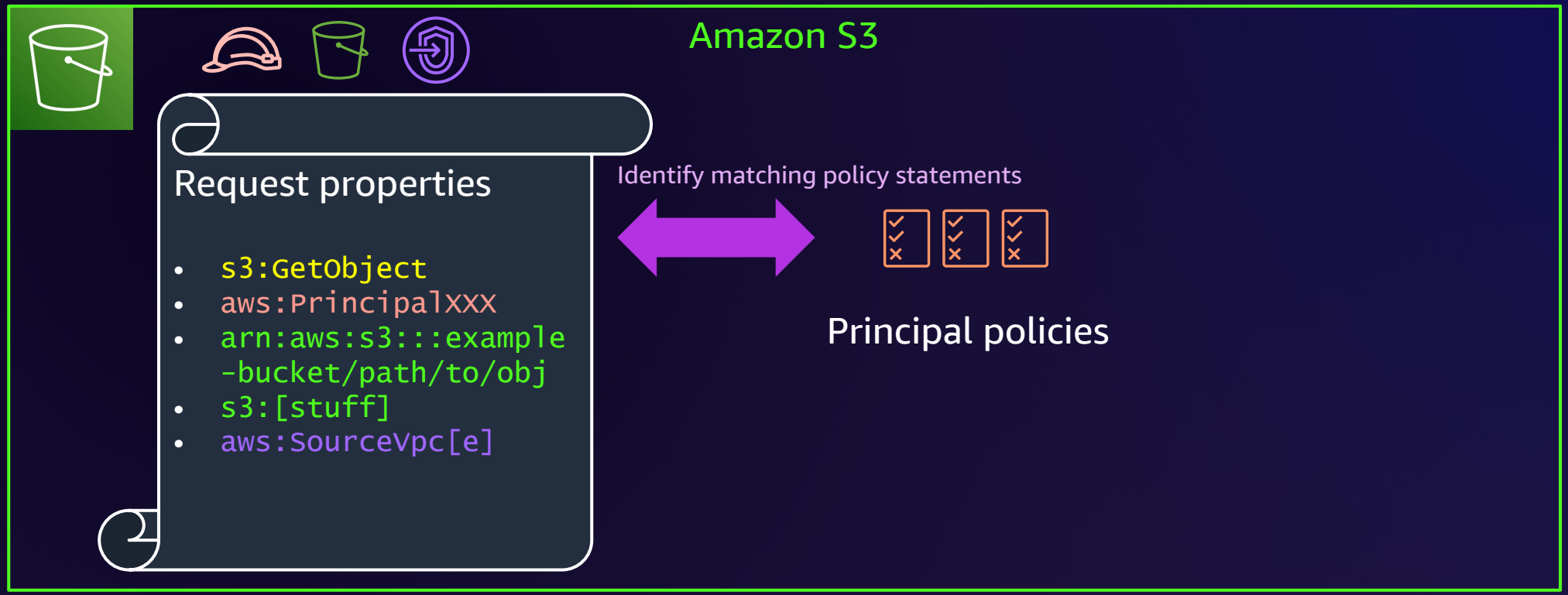
Authorization: Matching a request to policy statements



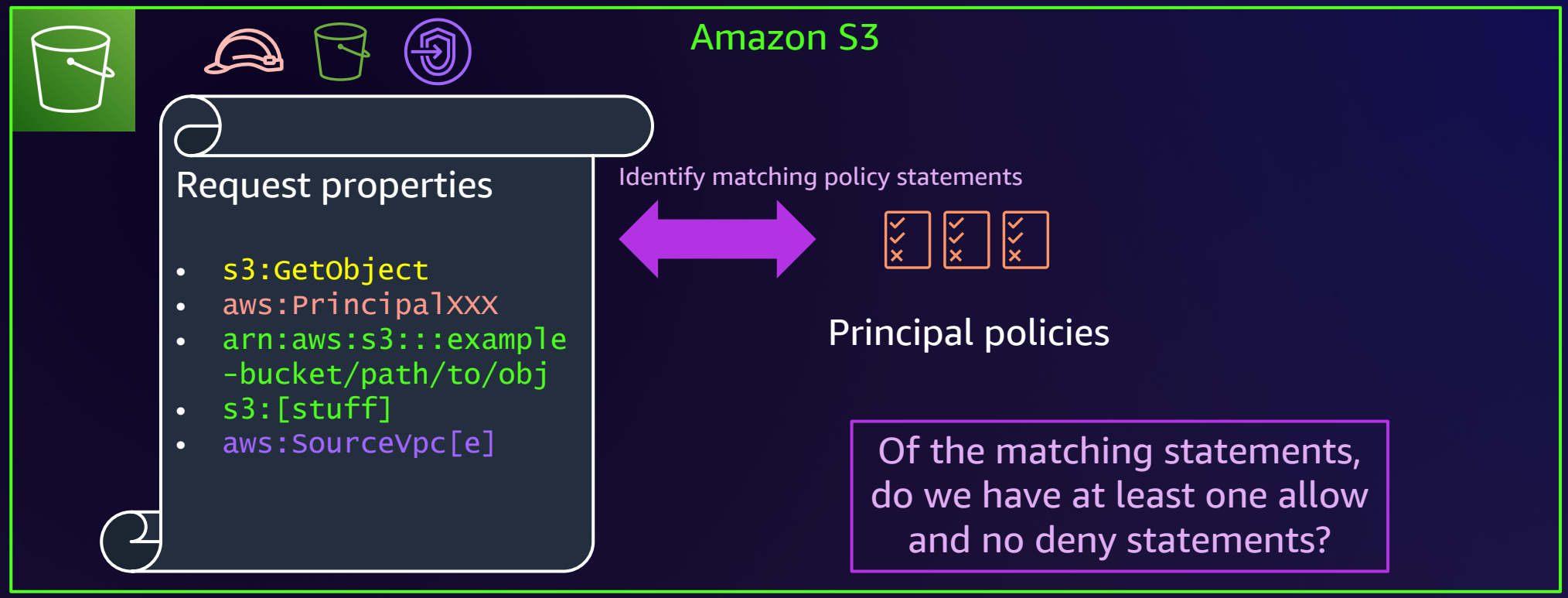
Authorization: Matching a request to policy statements



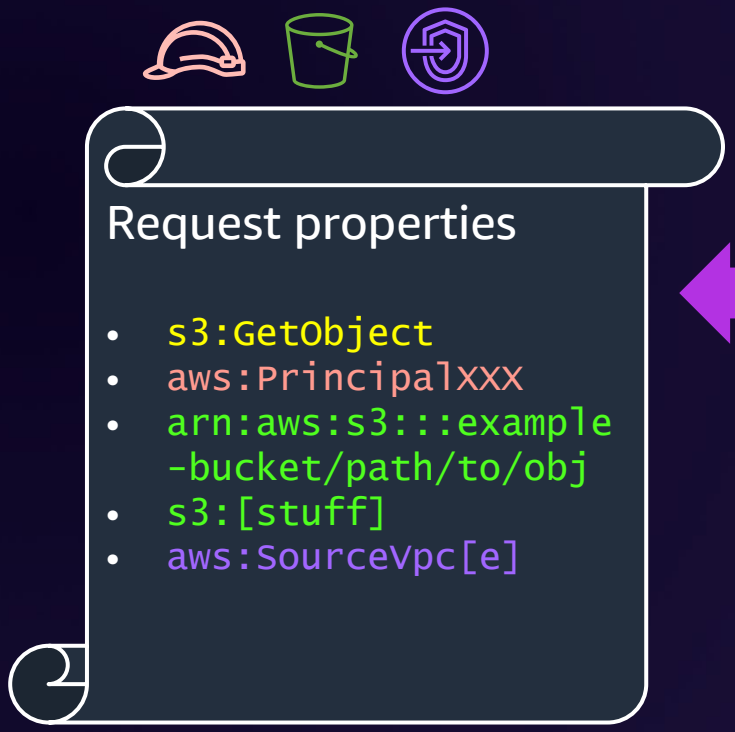
Authorization: Matching a request to policy statements



Authorization: Matching a request to policy statements



Authorizing a request: Summary



Principal policies

ALLOW?

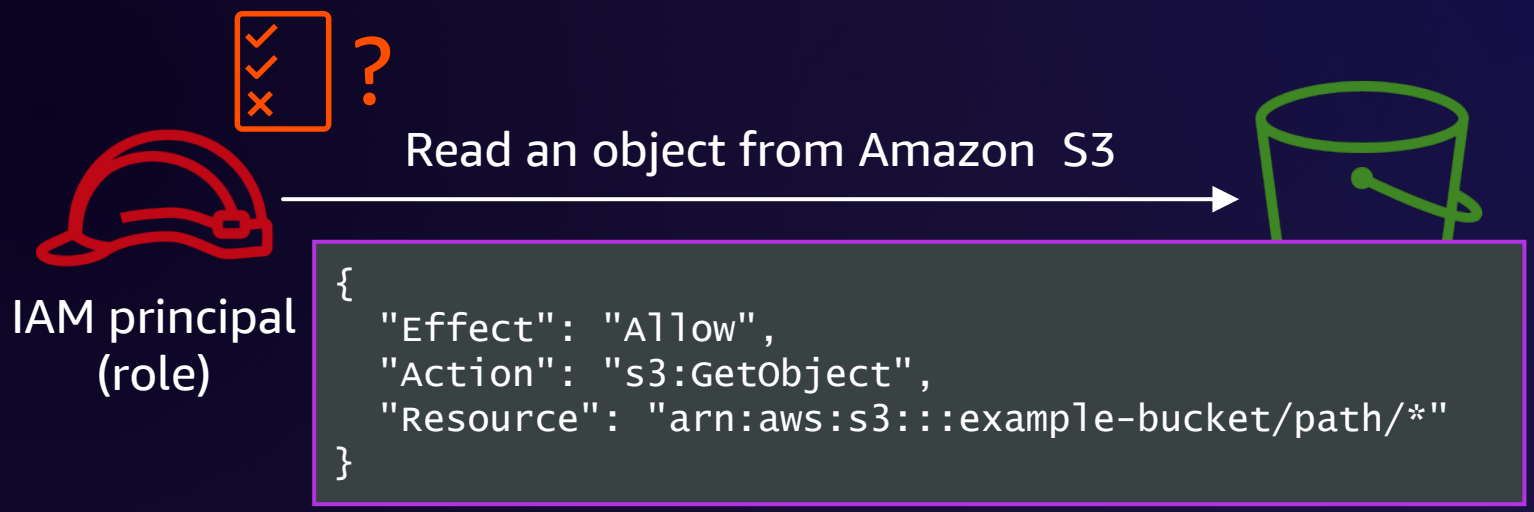
!DENY?

IAM policy-writing for builders

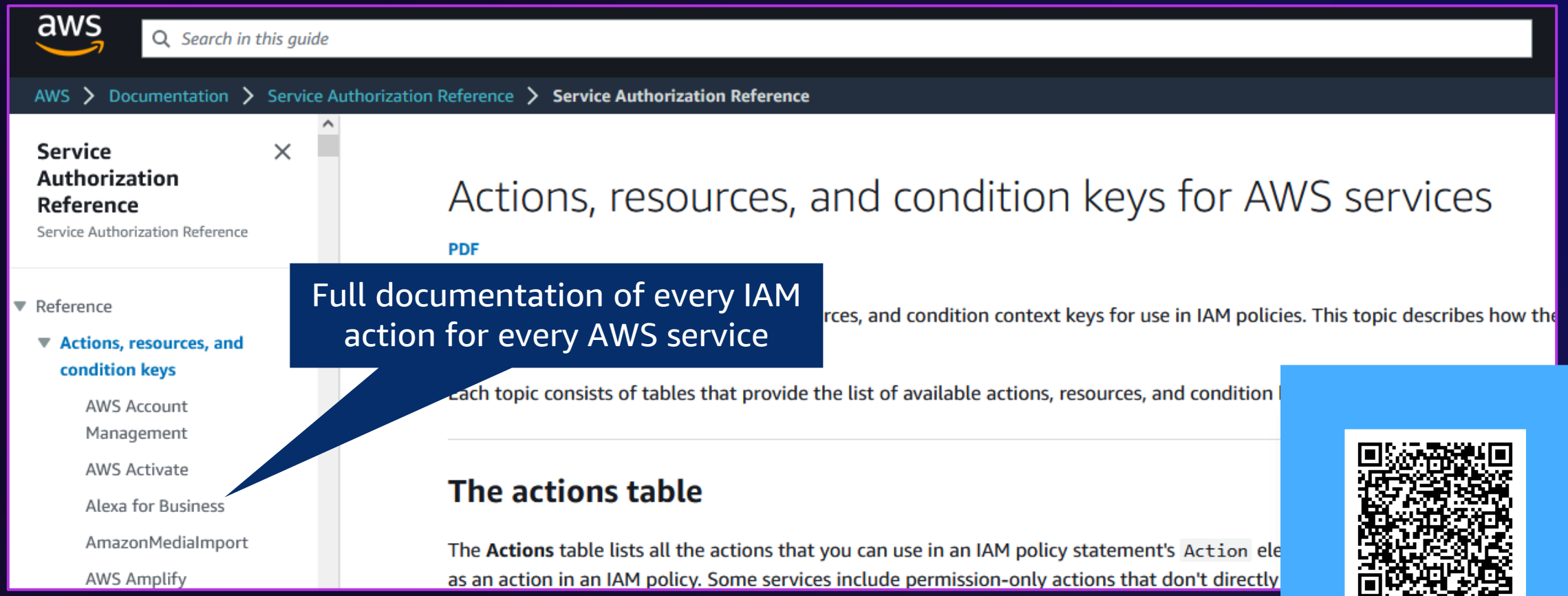
Example: Read-only access to part of an S3 bucket

```
# AWS command-line interface (CLI)

> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out
```



Bookmark this page: IAM policy documentation



Full documentation of every IAM action for every AWS service

https://docs.aws.amazon.com/service-authorization/latest/reference/reference_policies_actions-resources-contextkeys.html

IAM policy documentation for Amazon S3

IAM policy documentation for Amazon S3

Resource types defined by Amazon S3

The following resource types are defined by this service and can be used in the `Resource` element of IAM permission policy statements. Each actio
that can be specified with that action. A resource type can also define which condition keys you can include in a policy. These keys are displayed in t
columns in the following table, see Resource types table.

Resource types	ARN
accesspoint	arn:${Partition}:s3:${Region}:${Account}:accesspoint/${AccessPointName}
bucket	arn:${Partition}:s3:::${BucketName}
object	arn:${Partition}:s3:::${BucketName}/${ObjectName}
job	arn:${Partition}:s3:${ }:${Account}:job/${JobId}
storagelensconfiguration	arn:
objectlambdaaccesspoint	arn: ssPointName}

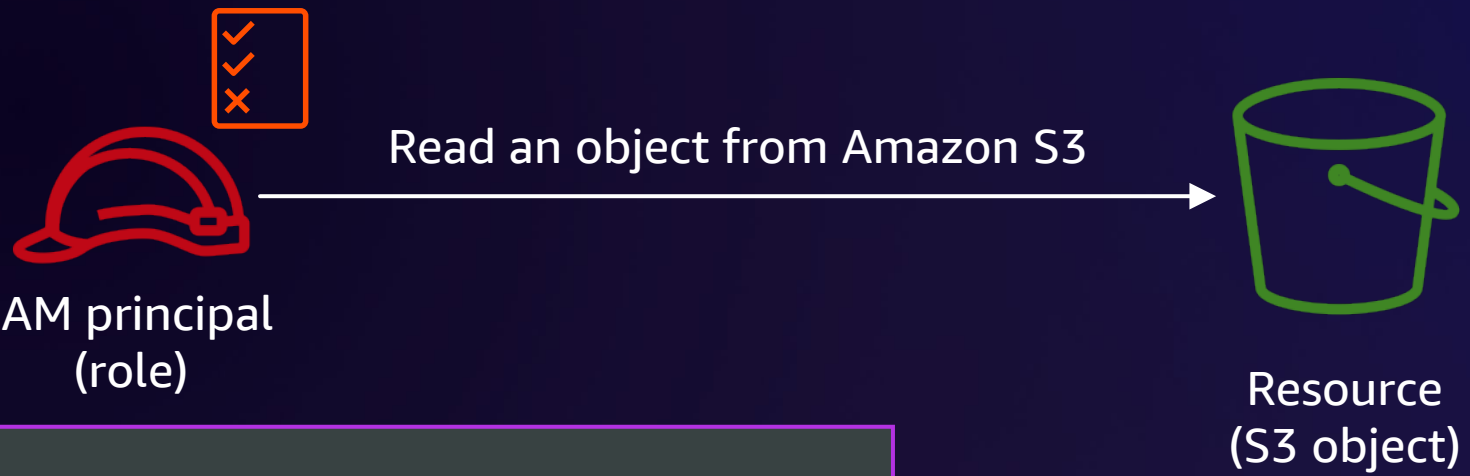
Amazon Resource Name (ARN) for an S3 object:

`arn:aws:s3:::example-bucket/path/to/obj`

Example: Read-only access to part of an S3 bucket

```
# AWS command-line interface (CLI)

> aws s3api get-object --bucket example-bucket --key path/to/obj /tmp/obj.out
```

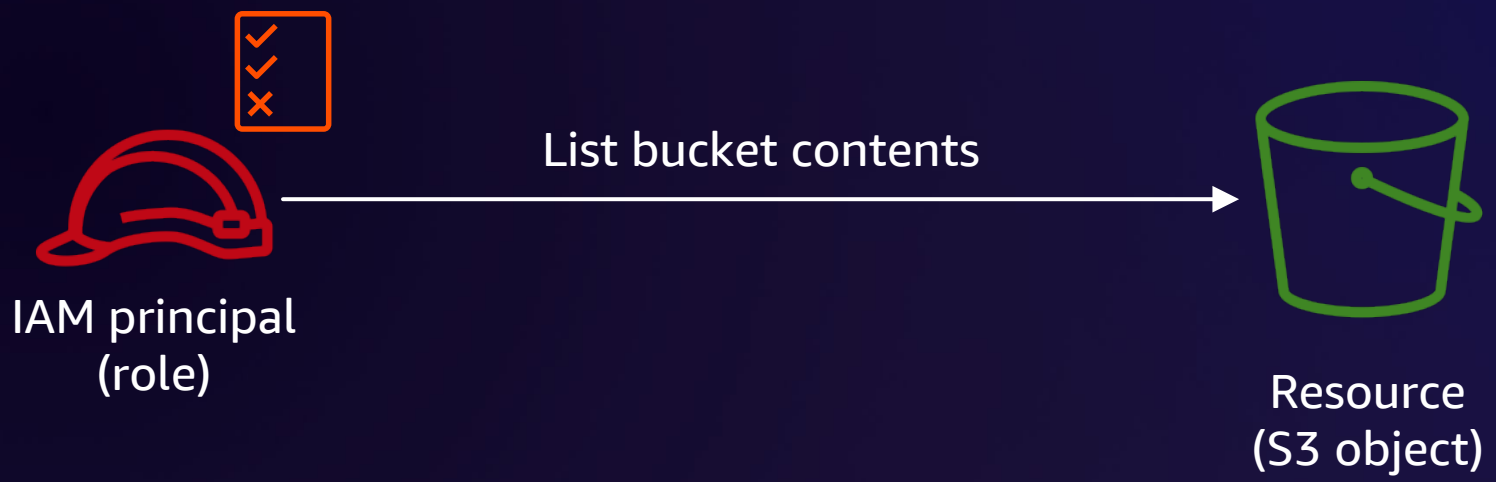



```
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/path/*"
}
```

Example: Read-only access to Amazon S3, including list

```
# AWS command-line interface (CLI)

> aws s3api list-objects-v2 --bucket example-bucket
```



IAM policy documentation for Amazon S3

	ListBucket	Grants permission to list some or all of the objects in an Amazon S3 bucket (up to 1000)	List	bucket*	s3:DataAccessPointAccount s3:DataAccessPointArn s3:AccessPointNetworkOrigin s3:authType s3:delimiter s3:max-keys s3:prefix s3:ResourceAccount s3:signatureAge s3:signatureversion s3:TlsVersion s3:x-amz-content-sha256

Example: Read-only access to Amazon S3, including list

```
// Statements from IAM policy
…
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/path/*"
},
{
  "Effect": "Allow",
  "Action: "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket"
}
```

```
# AWS command-line interface (CLI)

> aws s3api list-objects-v2 --bucket example-bucket \
    --query 'Contents[].{Key:Key}' --output text

other-path/object1.txt
other-path/object2.txt
path/hello.jpg
yet-another-path/mydata.cs
```

What happens here?

ListObjects response shows all objects in the bucket

Example: Read-only access to Amazon S3, including list

```
// Statements from IAM policy
…
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/path/*"
},
{
  "Effect": "Allow",
  "Action: "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket",
  "Condition": {
    "StringLike": {
      "s3:prefix": "path/*"
    }
  }
}
```

Example: Read-only access to Amazon S3, including list

```
// Statements from IAM policy
…
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/path/*"
},
{
  "Effect": "Allow",
  "Action: "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket",
  "Condition": {
    "StringLike": {
      "s3:prefix": "path/*"
    }
  }
}
```

```
# AWS command-line interface (CLI)

> aws s3api list-objects-v2 --bucket example-bucket \
    --query 'Contents[].{Key:Key}' --output text

An error occurred (Access[illegible]d) when calling the
ListObjectsV2 operation: A[illegible]ss Denied
```

What happens here?

No prefix was specified, so the allow statement does not match

Example: Read-only access to Amazon S3, including list

```
// Statements from IAM policy
…
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/path/*"
},
{
  "Effect": "Allow",
  "Action: "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket",
  "Condition": {
    "StringLike": {
      "s3:prefix": "path/*"
    }
  }
}
```

```
# AWS command-line interface (CLI)

➢ aws s3api list-objects-v2 --bucket example-bucket \
    --prefix path/
    --query 'Contents[].{Key:Key}' --output text

path/hello.jpg
```

What happens here?

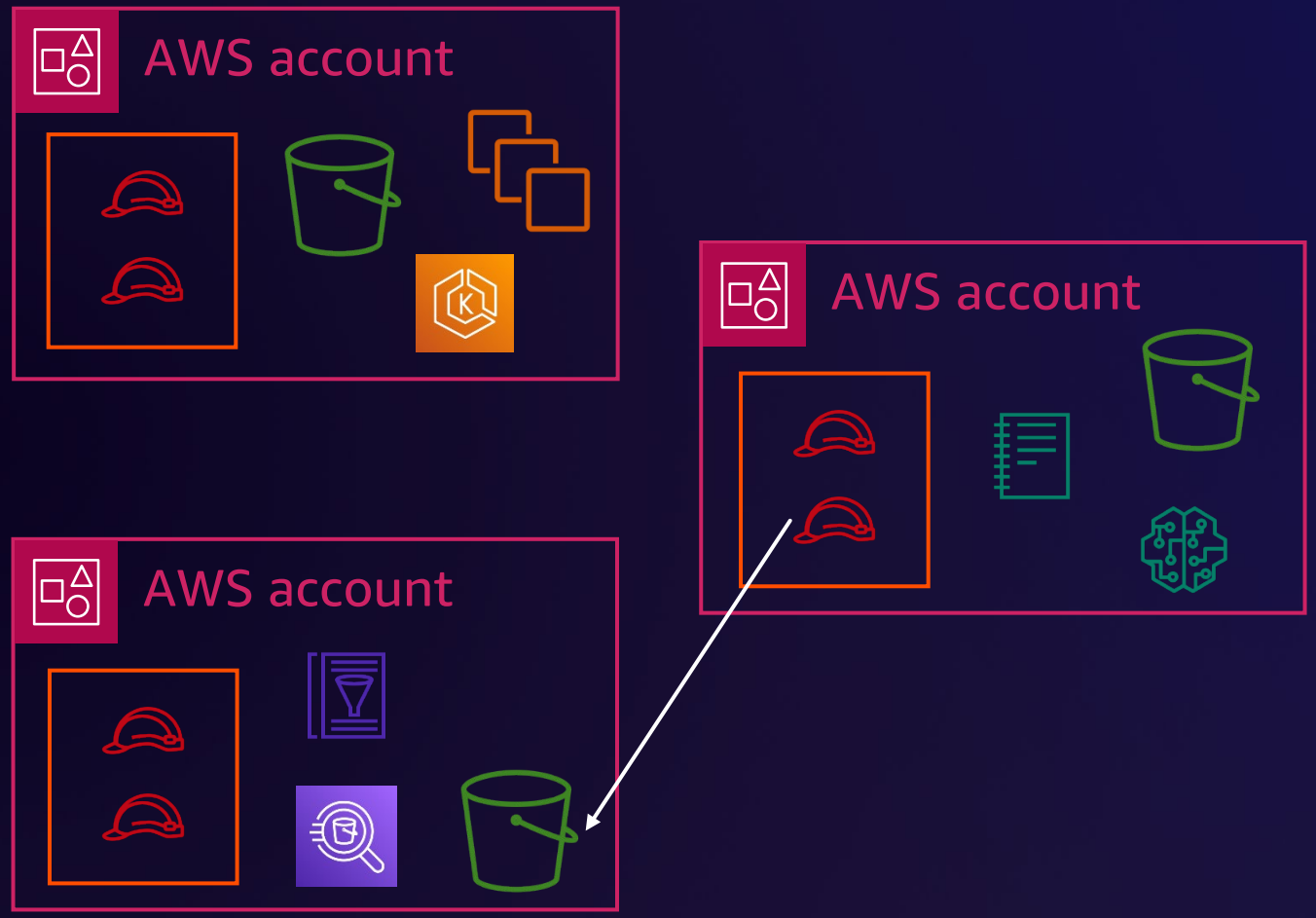
The request is specifying a prefix that matches the condition in the allow statement

Example: Invoke a Lambda function

```
// Statements from IAM policy
…
{
  "Effect": "Allow",
  "Action": "lambda:InvokeFunction",
  "Resource": "arn:aws:lambda:us-east-2:111122223333:function:MyFunction"
};
```

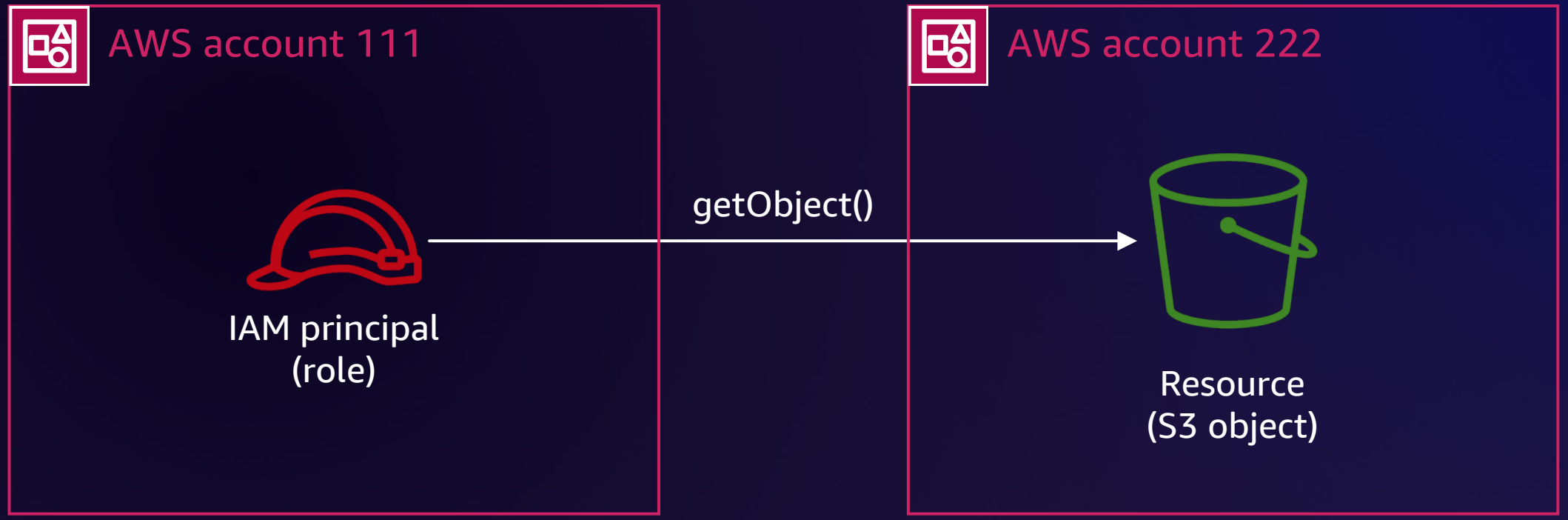
ARN of a Lambda function

Authorizing requests made by other AWS accounts

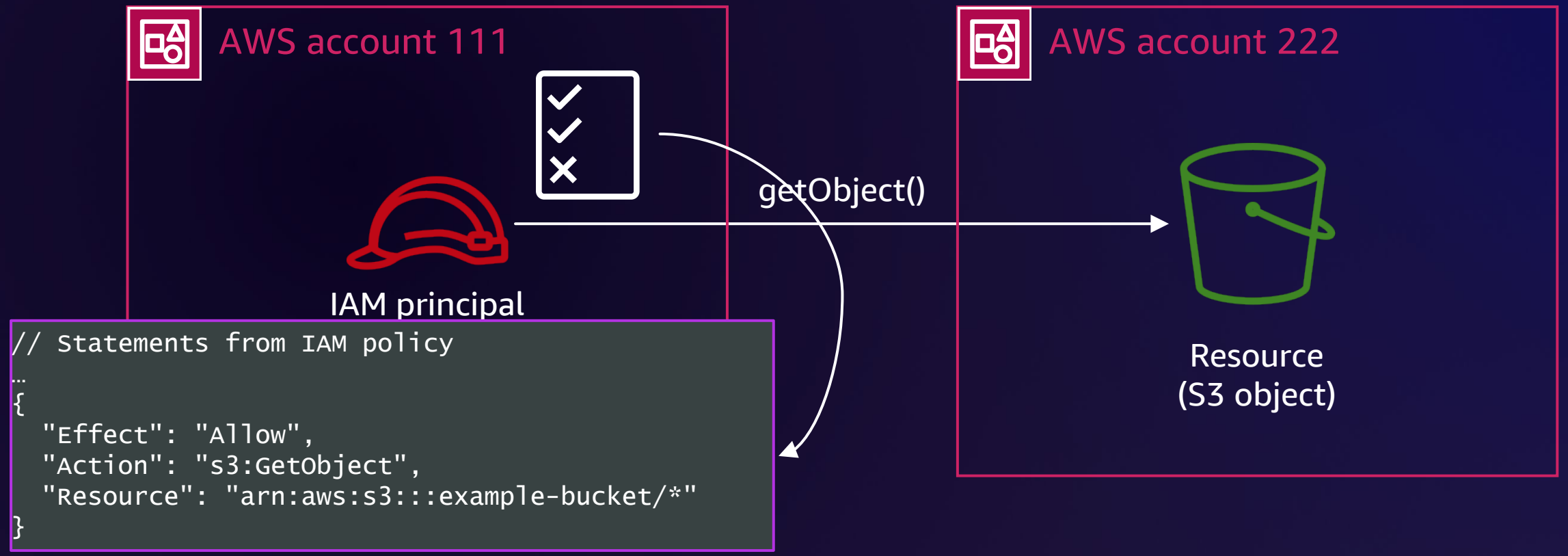


Cloud infrastructure that we own

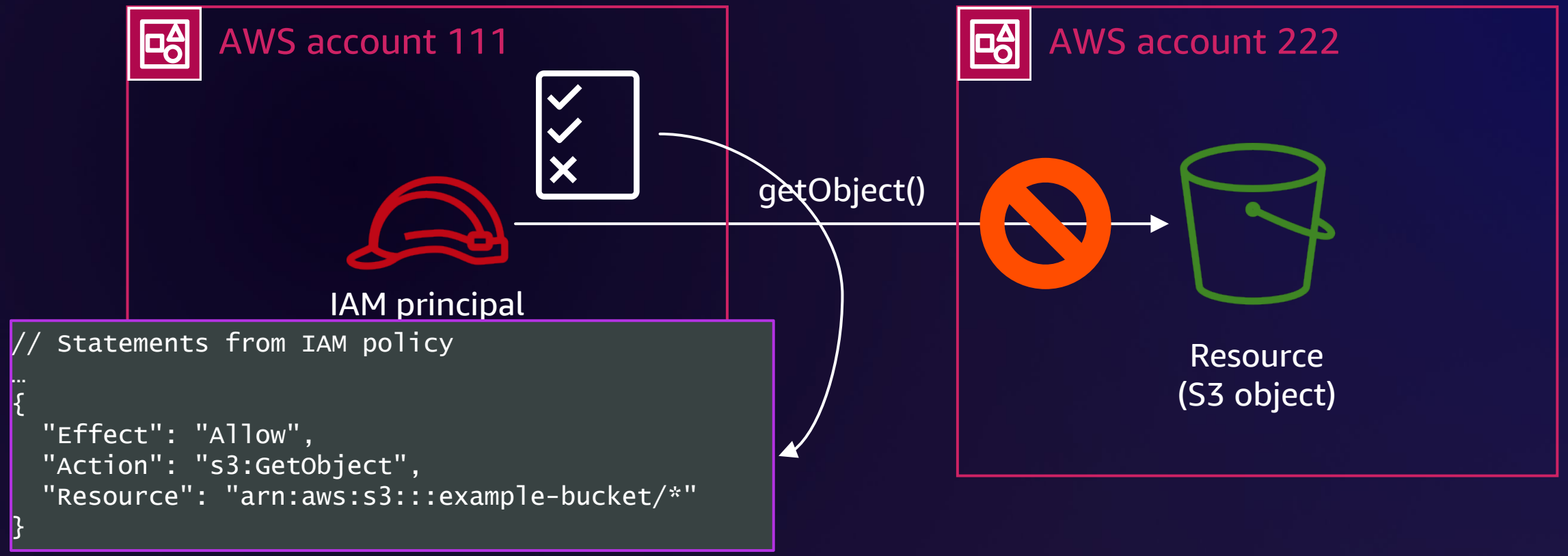
Reading an object from an Amazon S3 bucket in another account



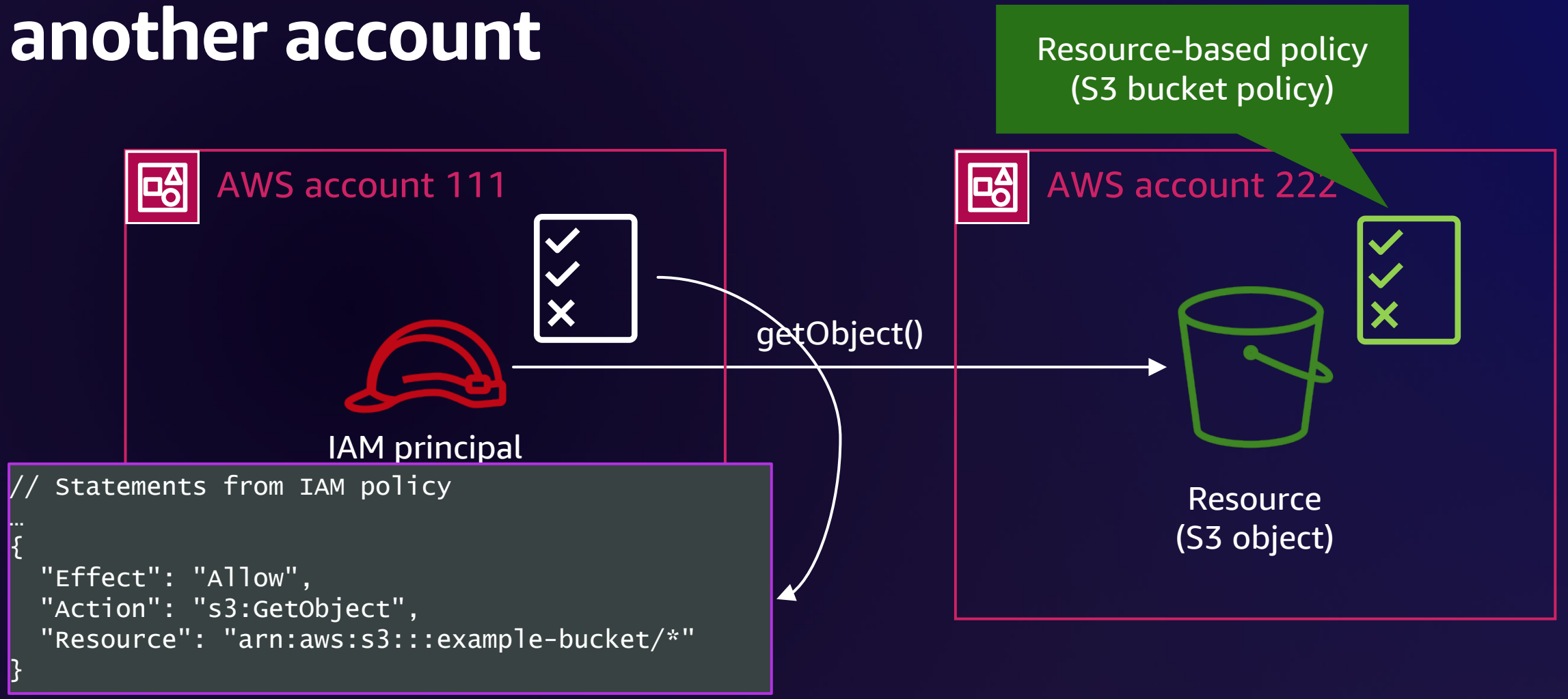
Reading an object from an Amazon S3 bucket in another account



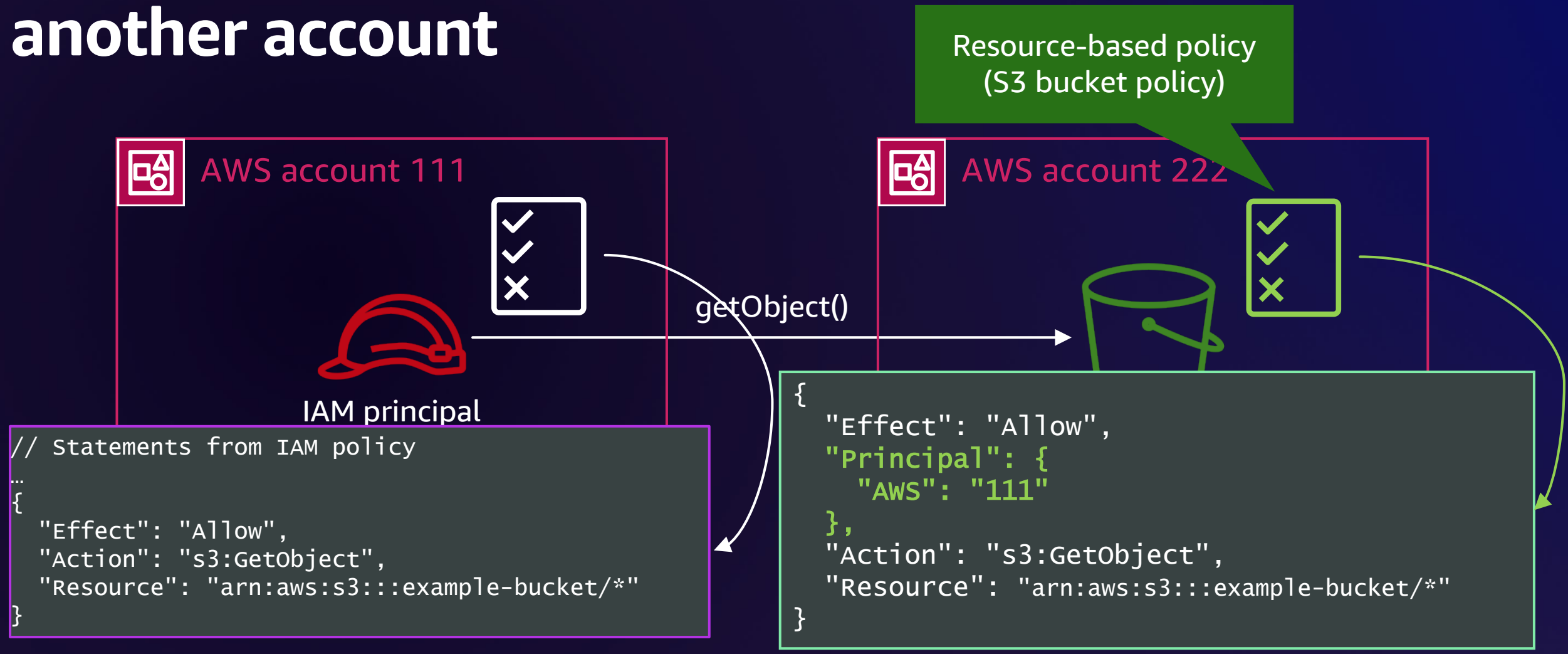
Reading an object from an Amazon S3 bucket in another account



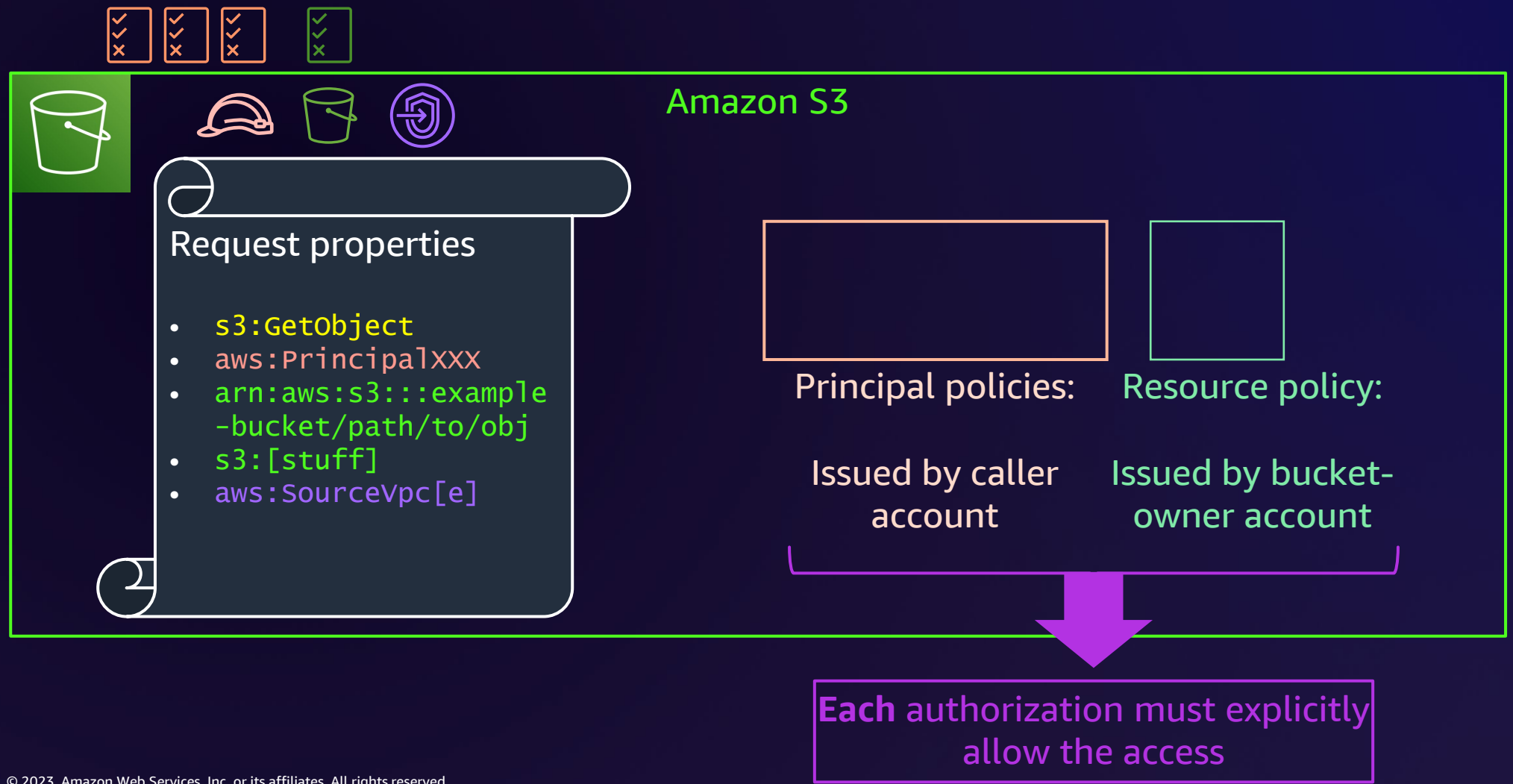
Reading an object from an Amazon S3 bucket in another account



Reading an object from an Amazon S3 bucket in another account



Cross-account authorization



Resource-based policy example: Amazon S3 cross-account

```
// Necessary, but not sufficient

{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/path/*"
}
```

Principal policy:

Issued by caller account
111

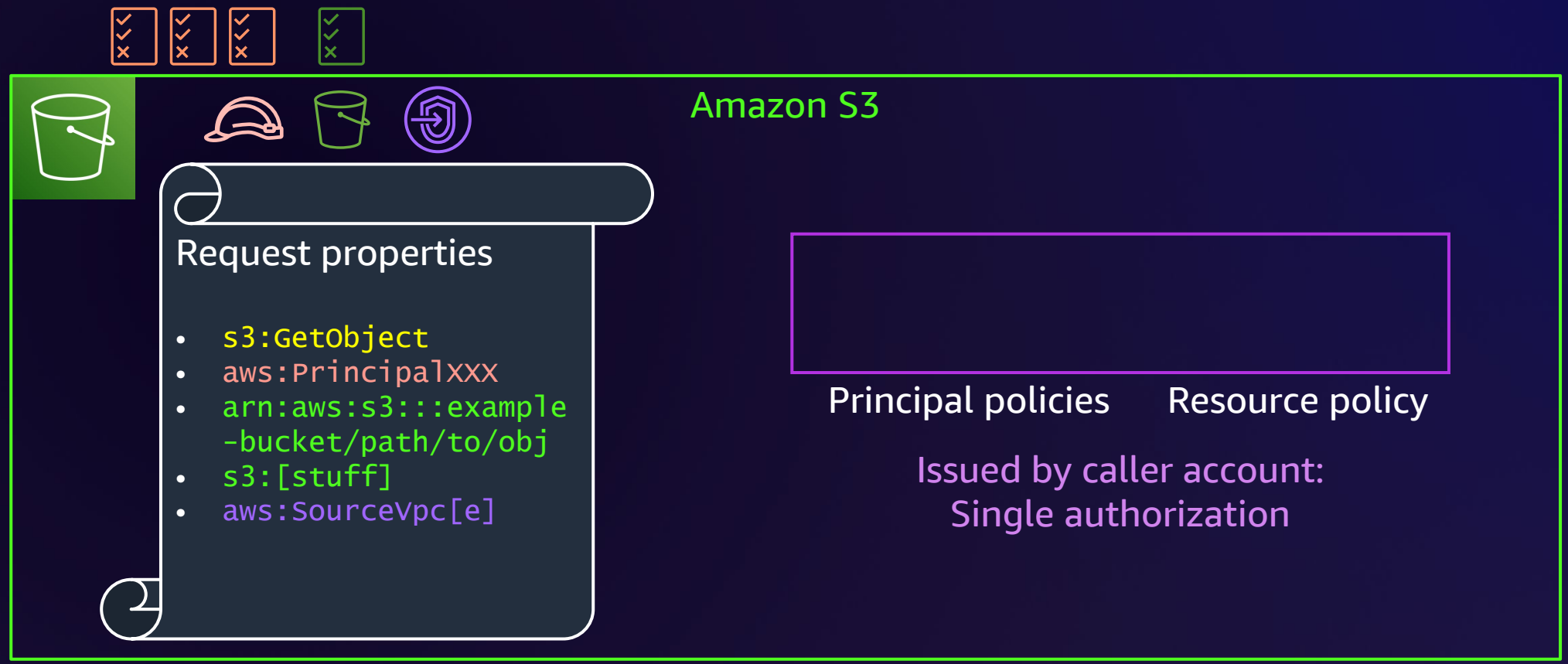
```
// Necessary, but not sufficient

{
  "Effect": "Allow",
  "Principal": {
    "AWS": "111"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/*"
}
```

Resource policy:

Issued by bucket-owner account
222

Same-account authorization with resource policy



Resource-based policy example: Assume-role policies on IAM accounts themselves

```
// Role Trust Policy, a.k.a. Assume-Role Policy.
//
// This policy is attached to the IAM Role itself and
// says who can obtain its credentials (the AWS Lambda
// service in this case)

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Service principal:

Identifies an AWS service acting on your resources

What AWS Lambda can do:
Obtain credentials for this role

AWS Lambda service

Lambda function runtime environment

Resource-based policy example: Amazon S3 permission to send events to an Amazon SQS queue

Amazon S3, sending event notifications to an Amazon SQS queue

Amazon SQS queue

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "s3.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "arn:aws:sqs:us-east-2:111122223333:my-queue",
  "Condition": {
    "ArnEquals": {
      "aws:SourceAccount": "111122223333",
      "aws:SourceArn": "arn:aws:s3:::example-bucket"
    }
  }
}
```

The Amazon S3 service can send messages to this Amazon SQS queue, as long as those messages correspond to this bucket and account

What we didn't cover

Concepts to learn about next

- AWS Organizations and service control policies
- VPC endpoints and VPC endpoint policies
- Building a data perimeter on AWS

Learning more about IAM

- Data perimeters https://docs.aws.amazon.com/whitepapers/latest/building-a-data-perimeter-on-aws/building-a-data-perimeter-on-aws.html
- Build some things on AWS ← This technique works best

AWS IAM recognized by Gartner Peer Insights

Thank you!

Please complete the session survey in the mobile app

Becky Weiss

becky@amazon.com