AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

STG328

Solving large-scale data access challenges with Amazon S3

Becky Weiss

Senior Principal Engineer
Amazon Web Services

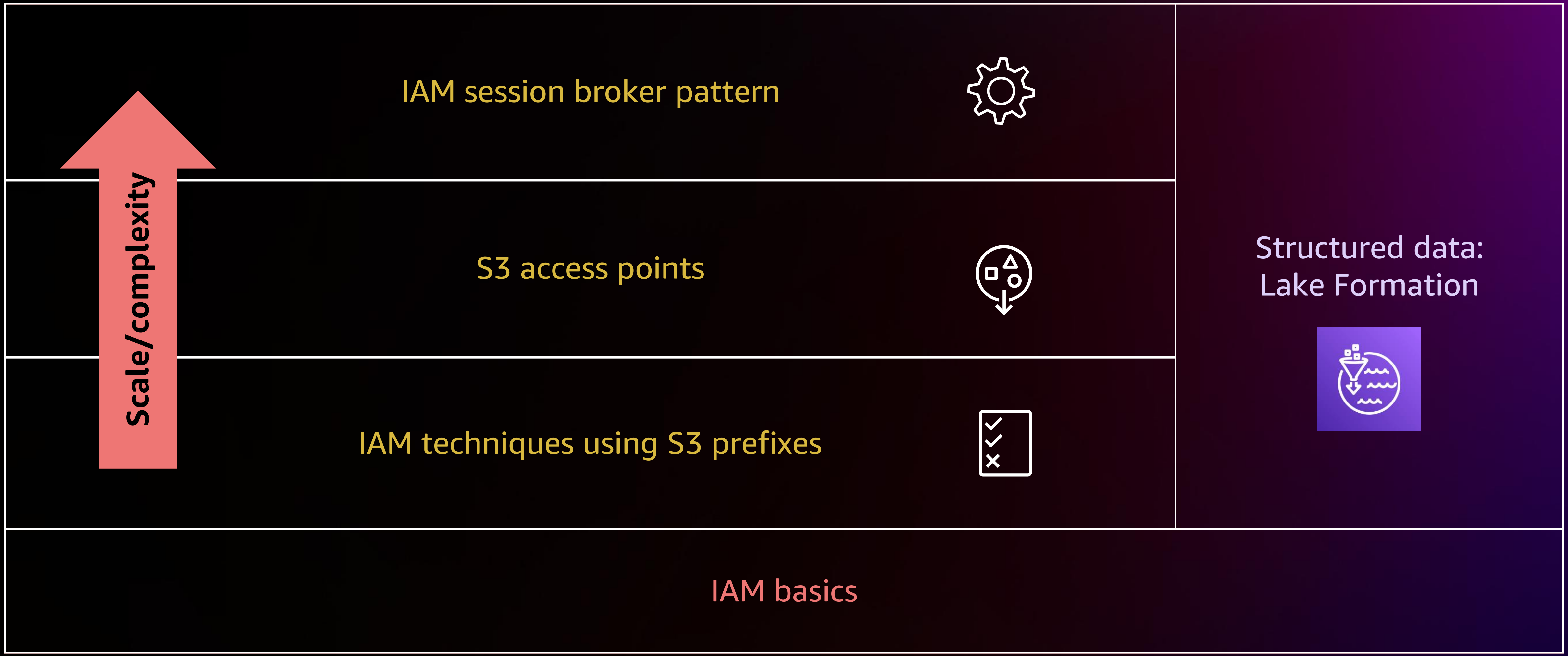
Nothing in this talk is hypothetical

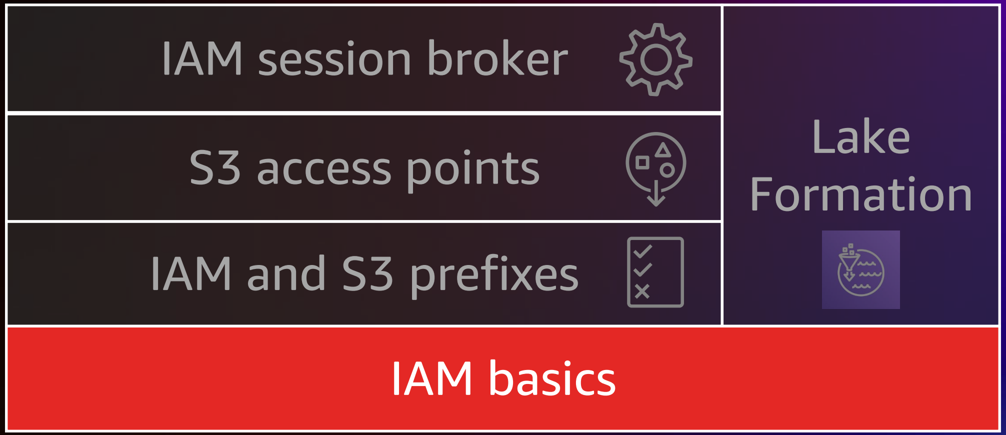
```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

Agenda: Real talk about large-scale data access in Amazon S3

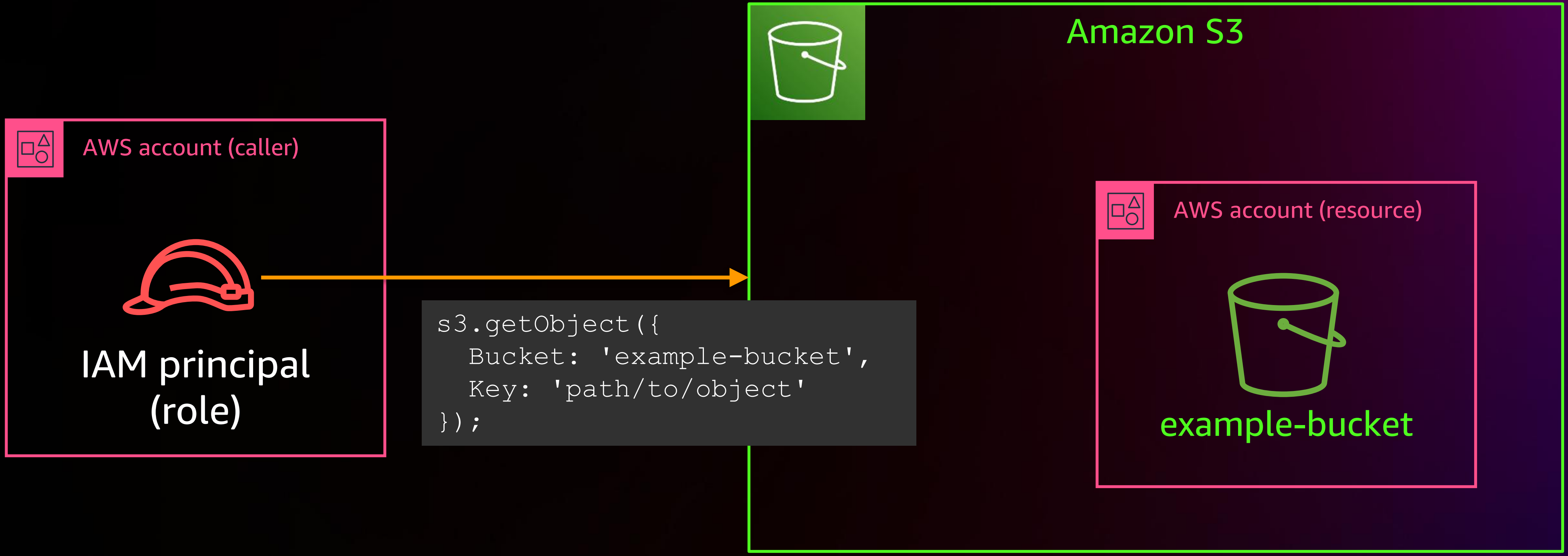
- The nuts and bolts of AWS Identity and Access Management (IAM) authorization in S3
- Getting the best scale out of IAM using S3 prefix patterns
- Modeling dynamic access schemes with IAM sessions
- Fine-grained access controls that follow the shape of your structured data, with AWS Lake Formation

Data access concepts in this session


Scale/complexity
IAM session broker pattern
S3 access points
IAM techniques using S3 prefixes
Structured data: Lake Formation
IAM basics

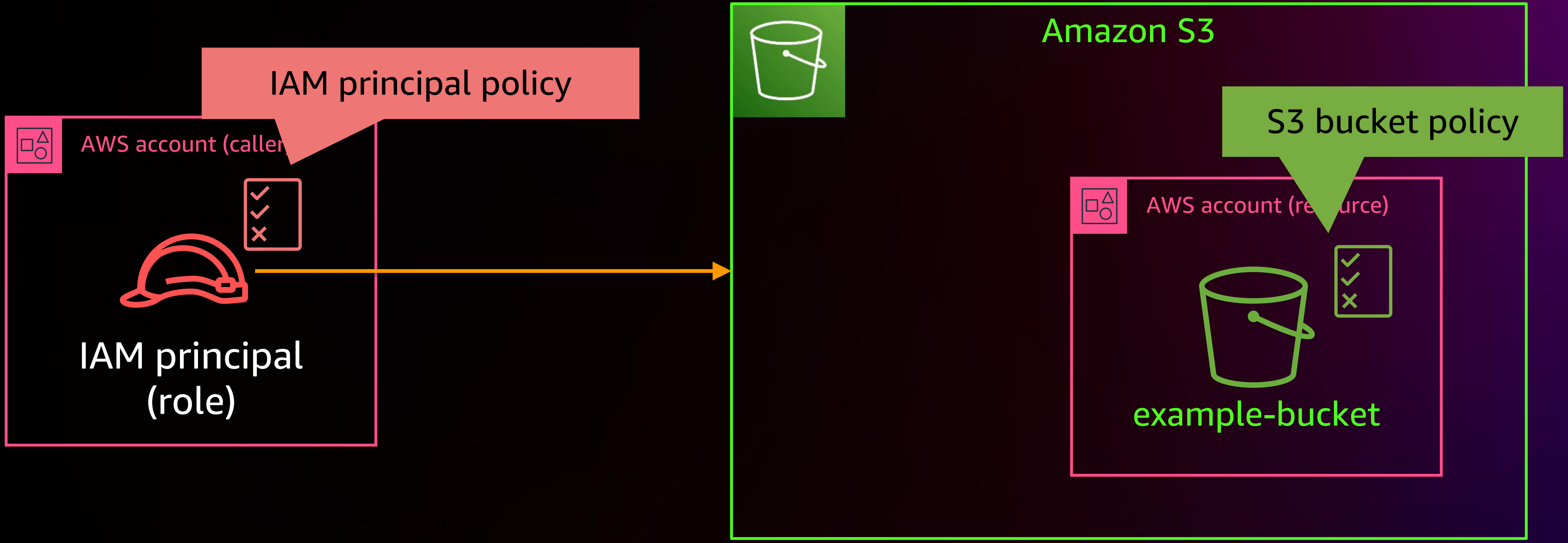

IAM session broker
S3 access points
IAM and S3 prefixes
Lake Formation
IAM basics

Background: An IAM authorization in S3



```
s3.getObject({
  Bucket: 'example-bucket',
  Key: 'path/to/object'
});
```

Background: An IAM authorization in S3



Object actions: Reading and writing data

```
+ example-bucket/
  + red/
    - alert.csv
    - firetruck.txt
  + yellow/
    - dandelion.jpg
    - submarine.json
  + blue/
    - moon.csv
```

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject*",
    "s3:PutObject*",
    … others
  ],
  "Resource": "arn:aws:s3:::example-bucket/yellow/submarine.json"
}
```

Amazon Resource Name (ARN) for an S3 object resource

Object actions: Reading and writing data

```
+ example-bucket/
  + red/
    - alert.csv
    - firetruck.txt
  + yellow/
    - dandelion.jpg
    - submarine.json
  + blue/
    - moon.csv
```

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject*",
    "s3:PutObject*",
    … others
  ],
  "Resource": "arn:aws:s3:::example-bucket/yellow/*"
}
```

Amazon Resource Name (ARN) for an S3 object resource, with wildcards

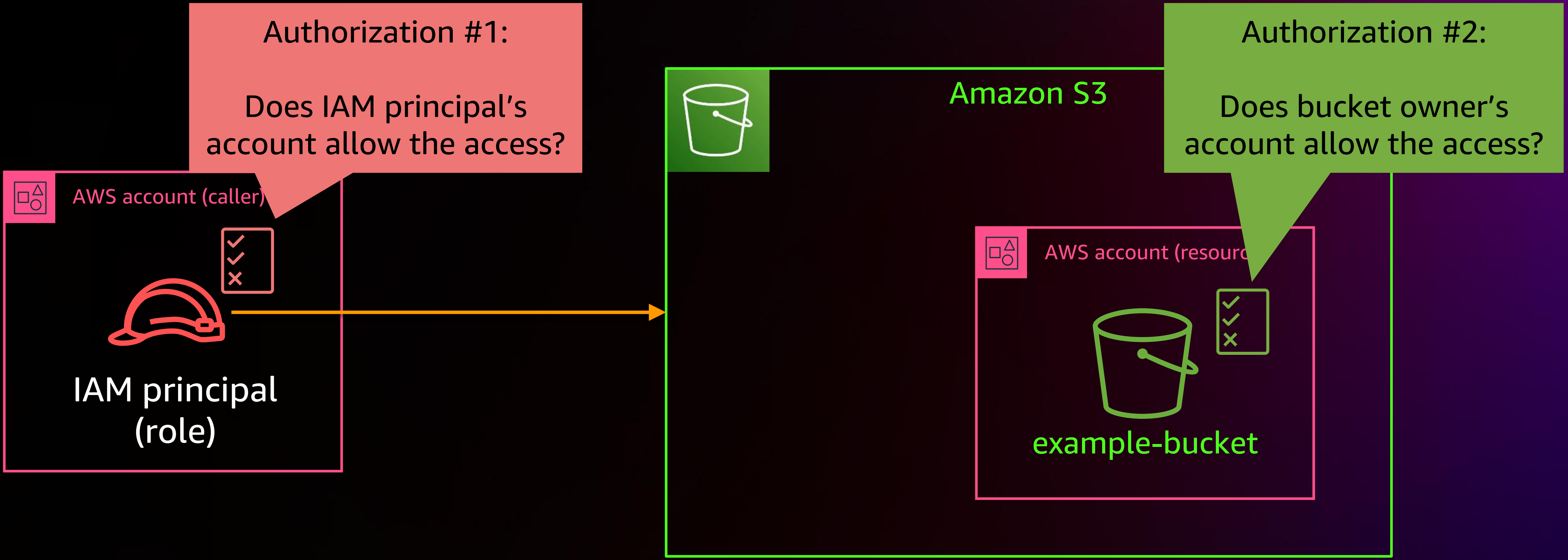
Object actions: Listing objects

Amazon Resource Name (ARN) for an S3 bucket resource

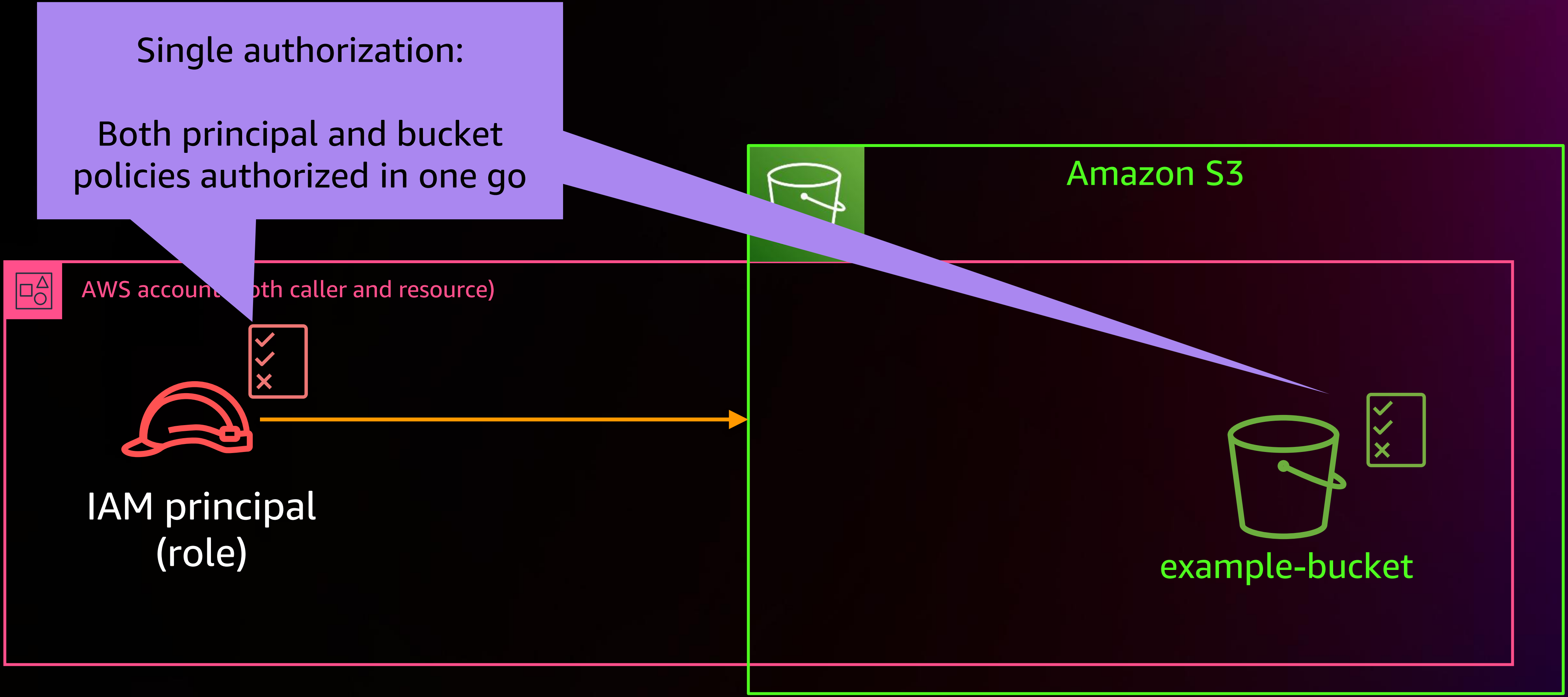
```
{
  "Effect": "Allow",
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket"
}
```

```
+ example-bucket/
  + red/
    - alert.csv
    - firetruck.txt
  + yellow/
    - dandelion.jpg
    - submarine.json
  + blue/
    - moon.csv
```

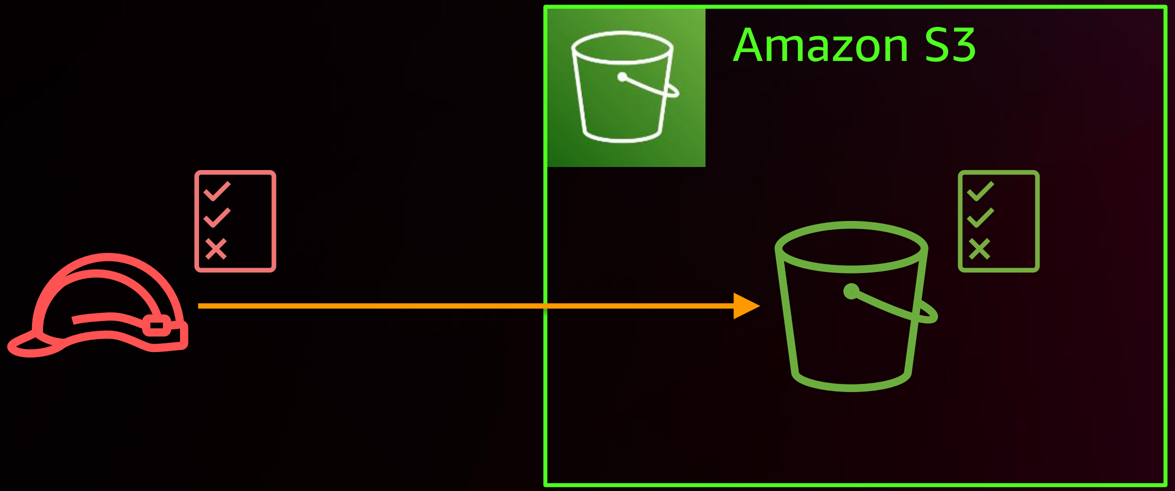
A cross-account access: Two authorizations



A same-account access: One authorization


Single authorization:
Both principal and bucket policies authorized in one go
AWS account (both caller and resource)
IAM principal (role)
Amazon S3
example-bucket

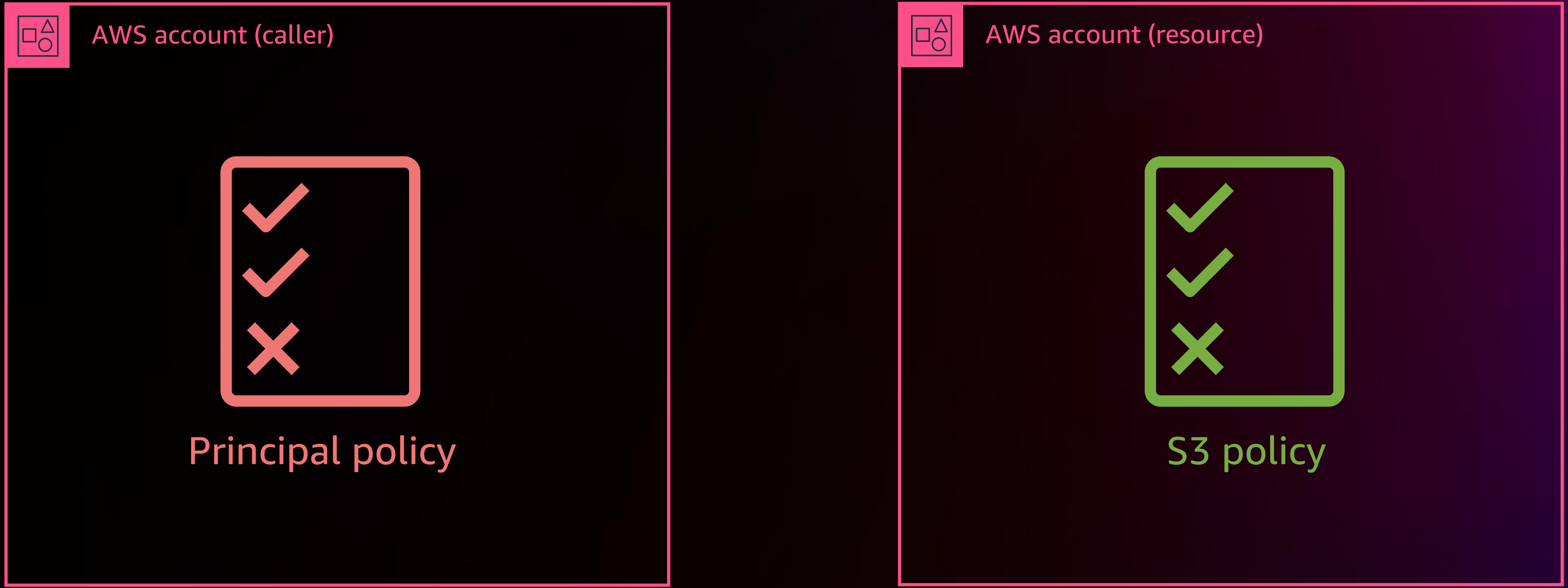
Recap: IAM basics



IAM principal policy

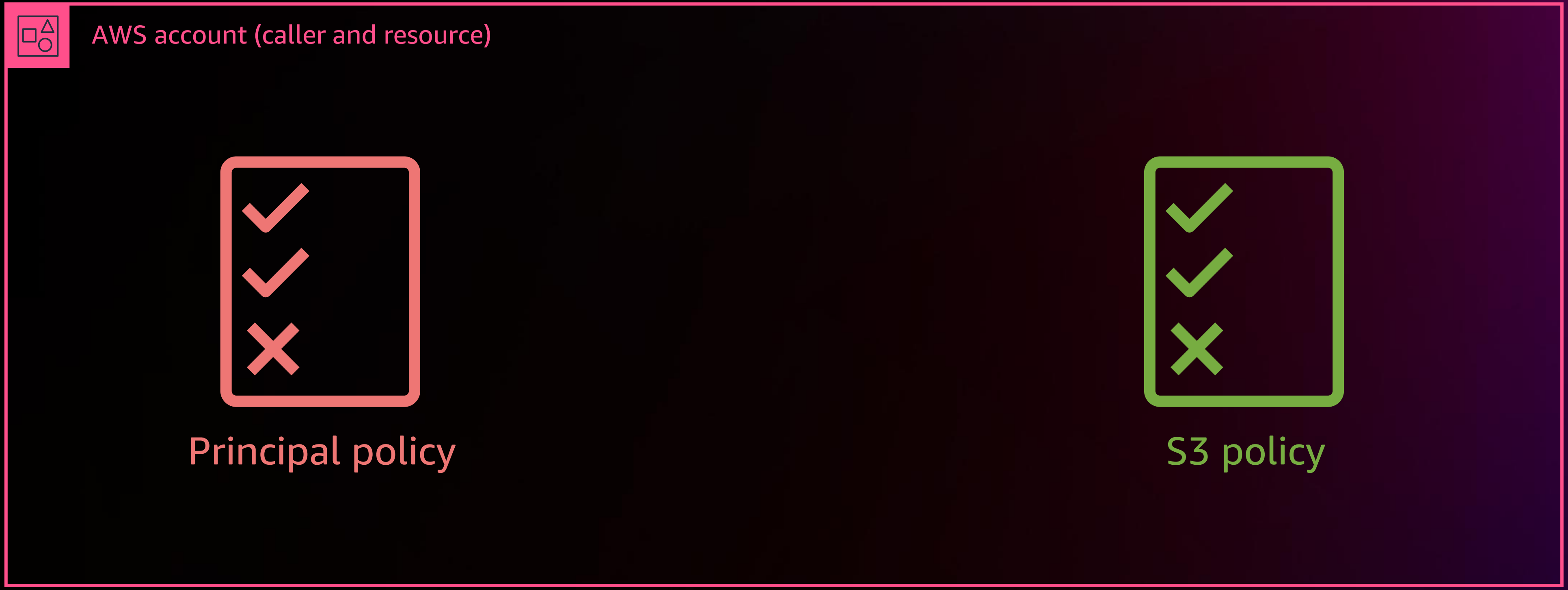
S3 bucket policy

Recap: IAM basics

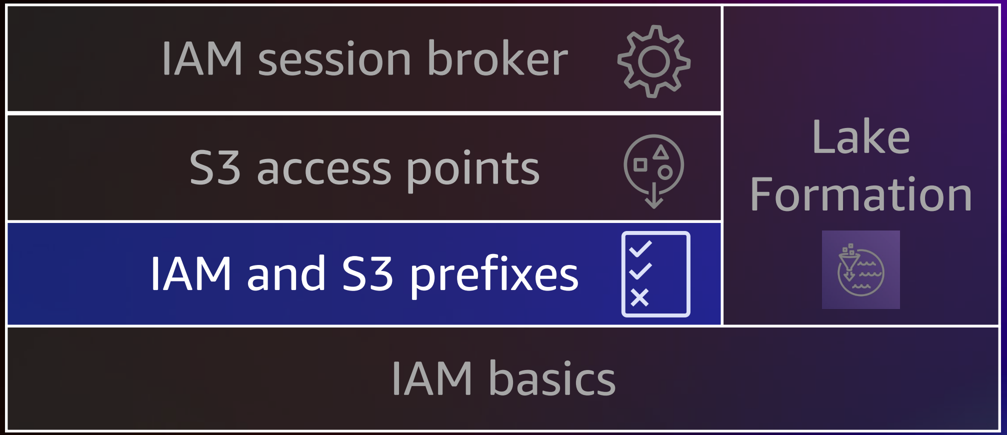


Two accounts: Two separate authorizations (AND)

Recap: IAM basics



One account: One authorization


IAM session broker
S3 access points
IAM and S3 prefixes
Lake Formation
IAM basics

Prefix-based access for object read/write

```
s3.getObject({
  Bucket: 'example-bucket',
  Key: 'blue/ocean.jpg'
});
```

```
// Policy granting object access for blue/ prefix
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject*",
    "s3:PutObject*"
  ],
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
}
```

```
+ example-bucket/
   - red/
   - yellow/
   - blue/
```

Prefix-based access for listing objects

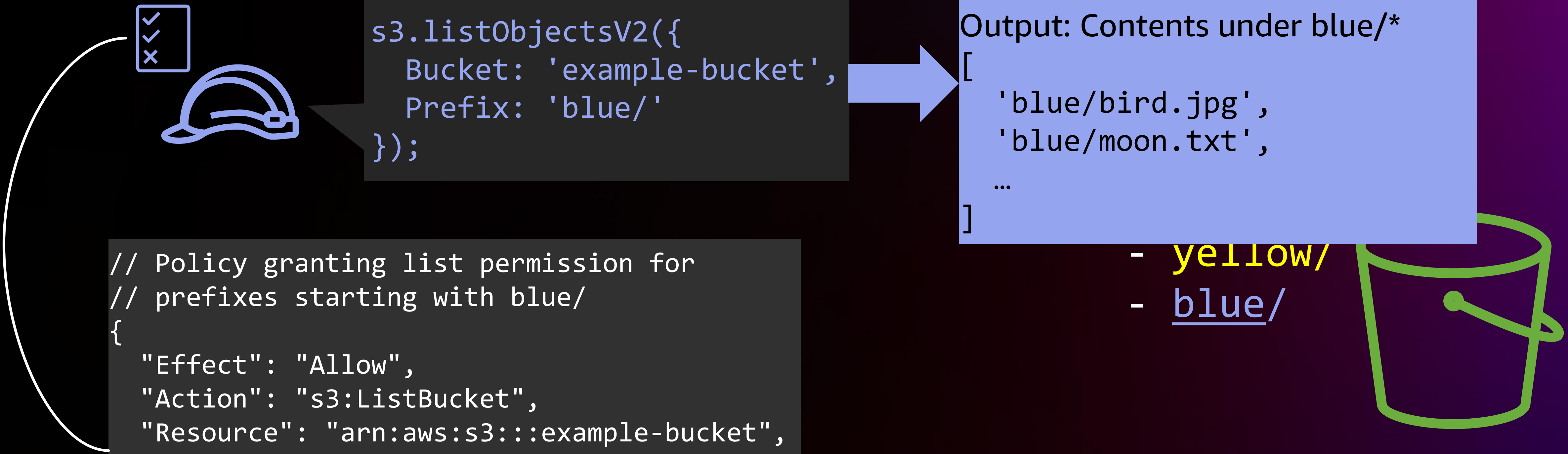
```
s3.listObjectsV2({
  Bucket: 'example-bucket'
});
```

Output: All bucket contents

```
[
  'blue/bird.jpg',
  'blue/moon.txt',
  …
  'red/alert.jpg',
  'red/fire.txt',
  …
]
```

```
// Policy granting general list permission
{
  "Effect": "Allow",
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket"
}
```

Prefix-based access for listing objects



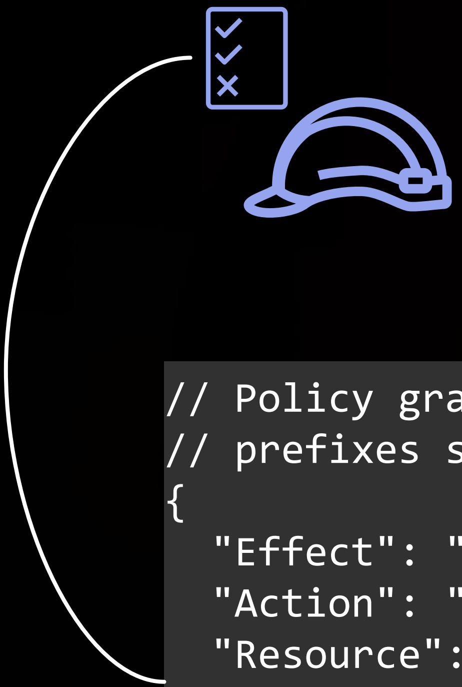
```
s3.listObjectsV2({
  Bucket: 'example-bucket',
  Prefix: 'blue/'
});
```

Output: Contents under blue/*

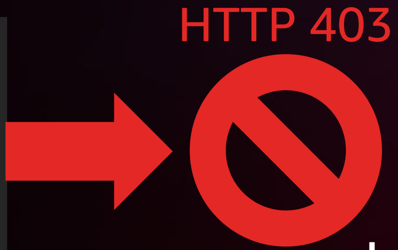
```
[
  'blue/bird.jpg',
  'blue/moon.txt',
  …
]
```

```
// Policy granting list permission for
// prefixes starting with blue/
{
  "Effect": "Allow",
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket",
  "Condition": {
    "StringLike": {
      "s3:prefix": "blue/*"
    }
  }
}
```

Prefix-based access for listing objects



```
s3.listObjectsV2({
  Bucket: 'example-bucket'
});
```

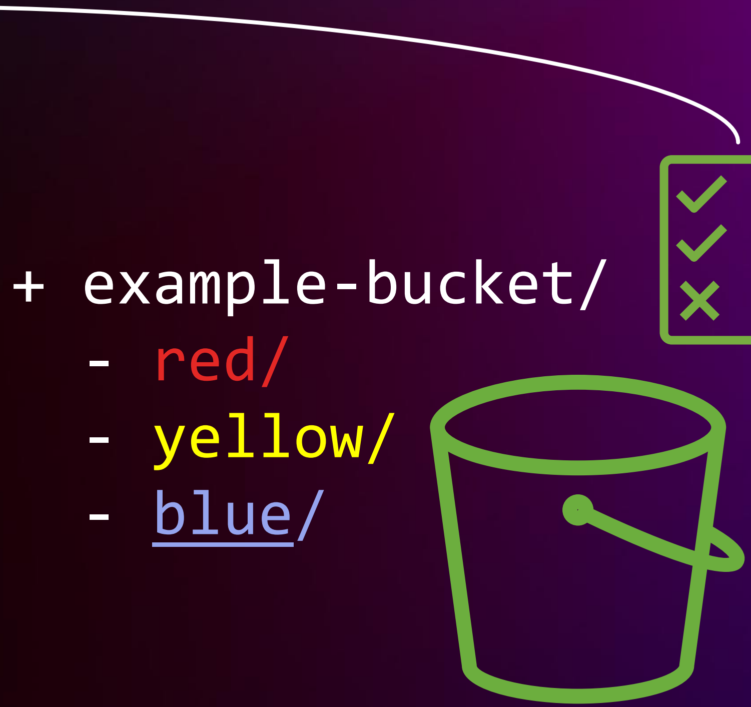


```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

```
// Policy granting list permission for
// prefixes starting with blue/
{
  "Effect": "Allow",
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket",
  "Condition": {
    "StringLike": {
      "s3:prefix": "blue/*"
    }
  }
}
```

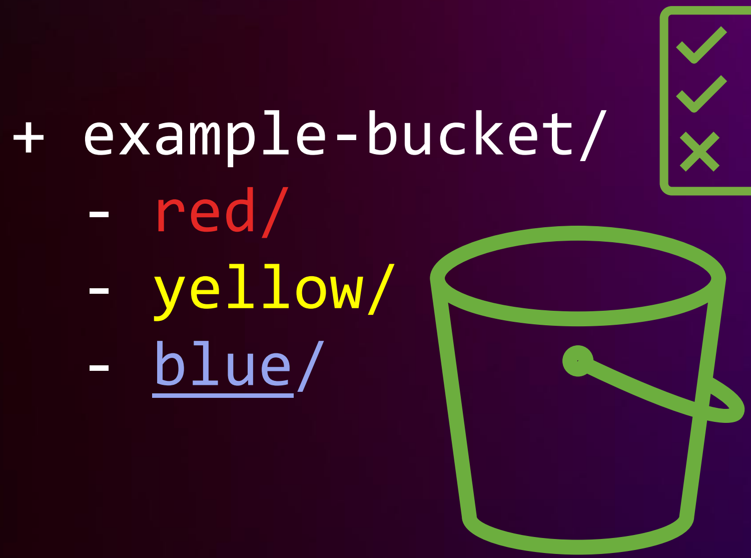
Counting up policy space

```
// Bucket policy with read permissions
// for blue/ prefix
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "ArnEquals": {
      "aws:PrincipalArn":
        "arn:aws:iam::111122223333:role/Blue"
      }
    }
  }
}

// [251 characters]
```



Counting up policy space

```
// Bucket policy with read and list permissions
// for blue/ prefix
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "ArnEquals": {
      "aws:PrincipalArn":
        "arn:aws:iam::111122223333:role/Blue"
…
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::example-bucket",
  "Condition": {
    "ArnEquals": {
      "aws:PrincipalArn":
        "arn:aws:iam::111122223333:role/Blue"
      }
    },
    "StringLike": {
      "s3:prefix": "blue/*"
…

// [554 characters]
```



Doing the math

- Estimate ~300 characters for a read policy statement
- Estimate ~300 characters for a list policy statement
- Bucket policy max: 20KB

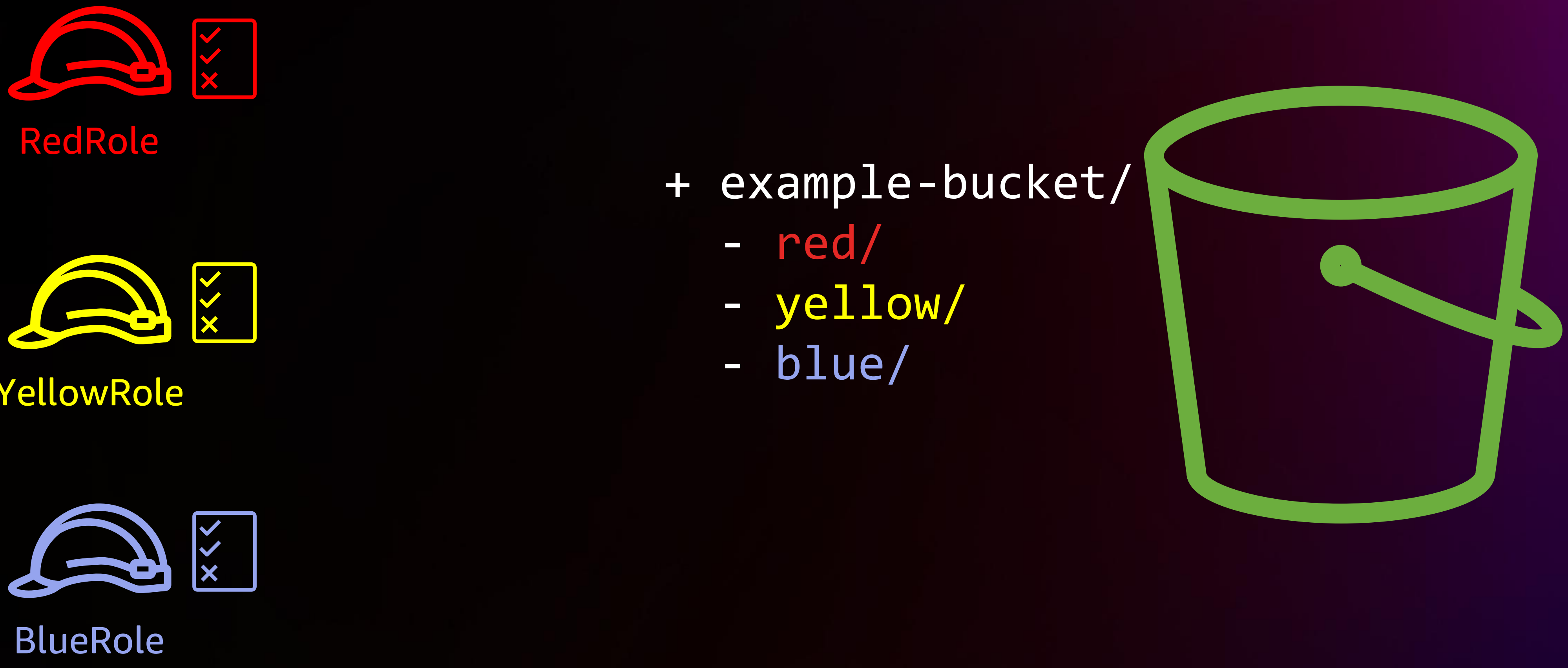
➔ ~30 prefixes with distinct access patterns fit in a bucket policy

➔ (About twice that if there are no list permissions)

Primary constraint: Number of prefixes ➔ policy sizes

Example: Prefix policies on IAM principals

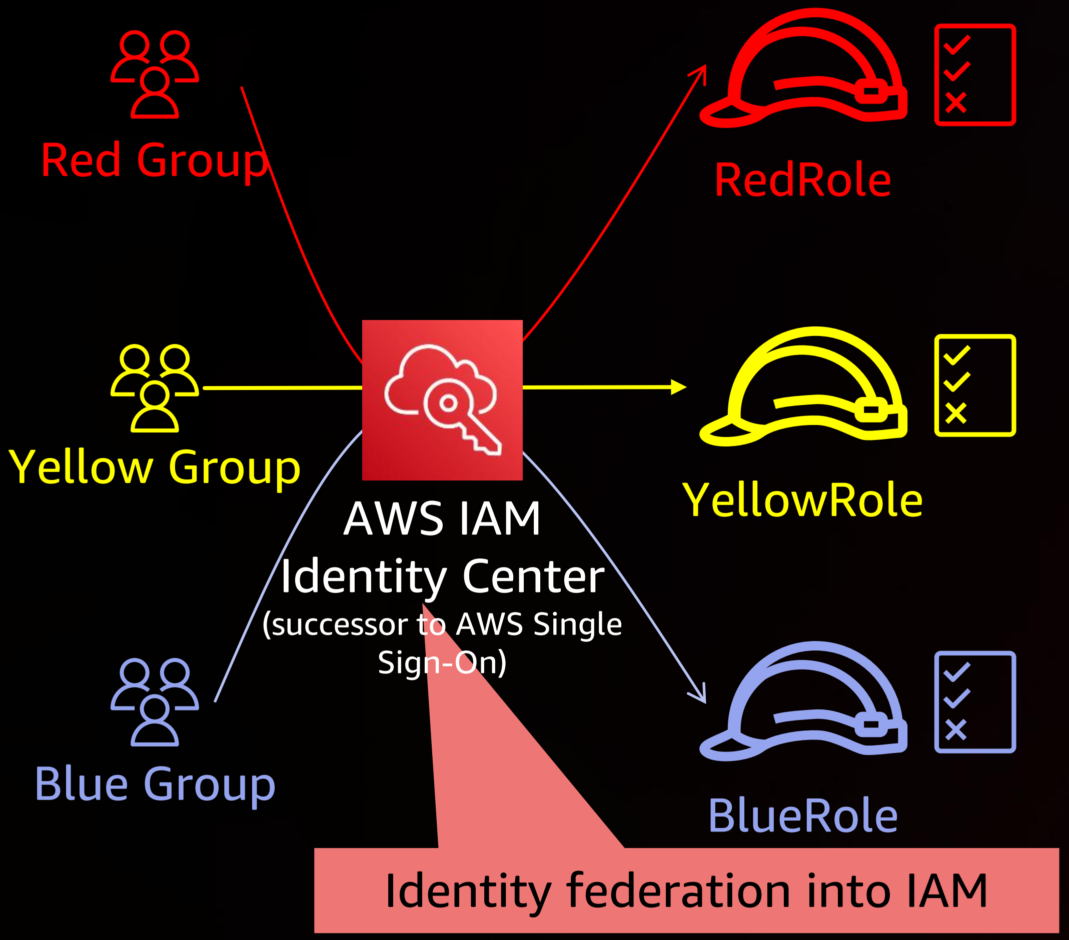
[SAME-ACCOUNT USE CASES]



```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

Example: Prefix policies on IAM principals

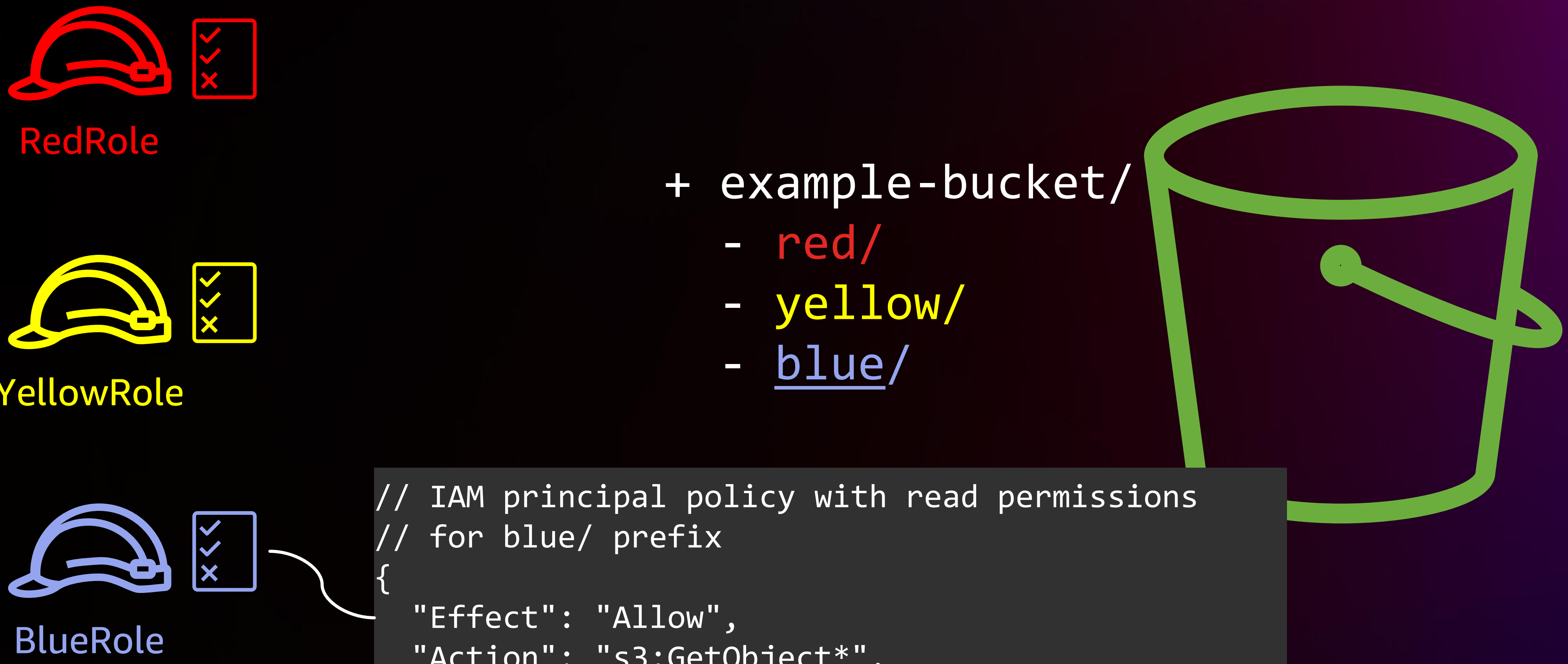
[SAME-ACCOUNT USE CASES]


Red Group
Yellow Group
Blue Group
AWS IAM Identity Center (successor to AWS Single Sign-On)
RedRole
YellowRole
BlueRole
Identity federation into IAM

```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

Example: Prefix policies on IAM principals

[SAME-ACCOUNT USE CASES]

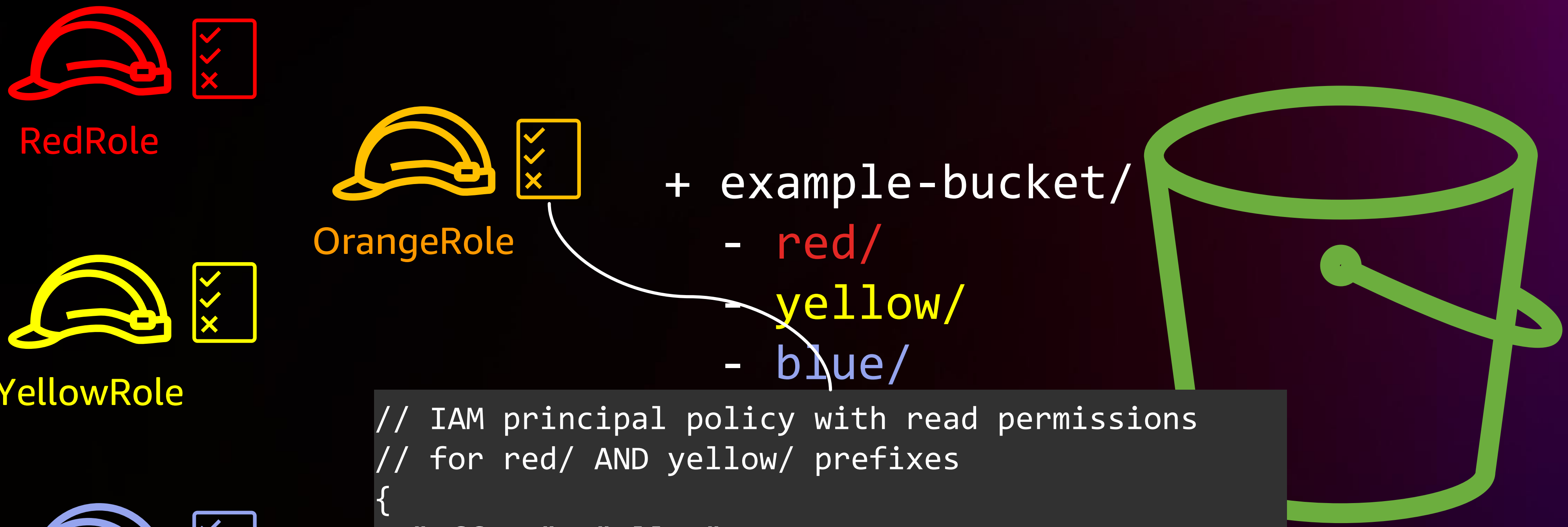


```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

```
// IAM principal policy with read permissions
// for blue/ prefix
{
  "Effect": "Allow",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
}
```

Example: Prefix policies on IAM principals

[SAME-ACCOUNT USE CASES]



```
// IAM principal policy with read permissions
// for red/ AND yellow/ prefixes
{
  "Effect": "Allow",
  "Action": "s3:GetObject*",
  "Resource": [
    "arn:aws:s3:::example-bucket/red/*",
    "arn:aws:s3:::example-bucket/yellow/*"
  ]
}
```

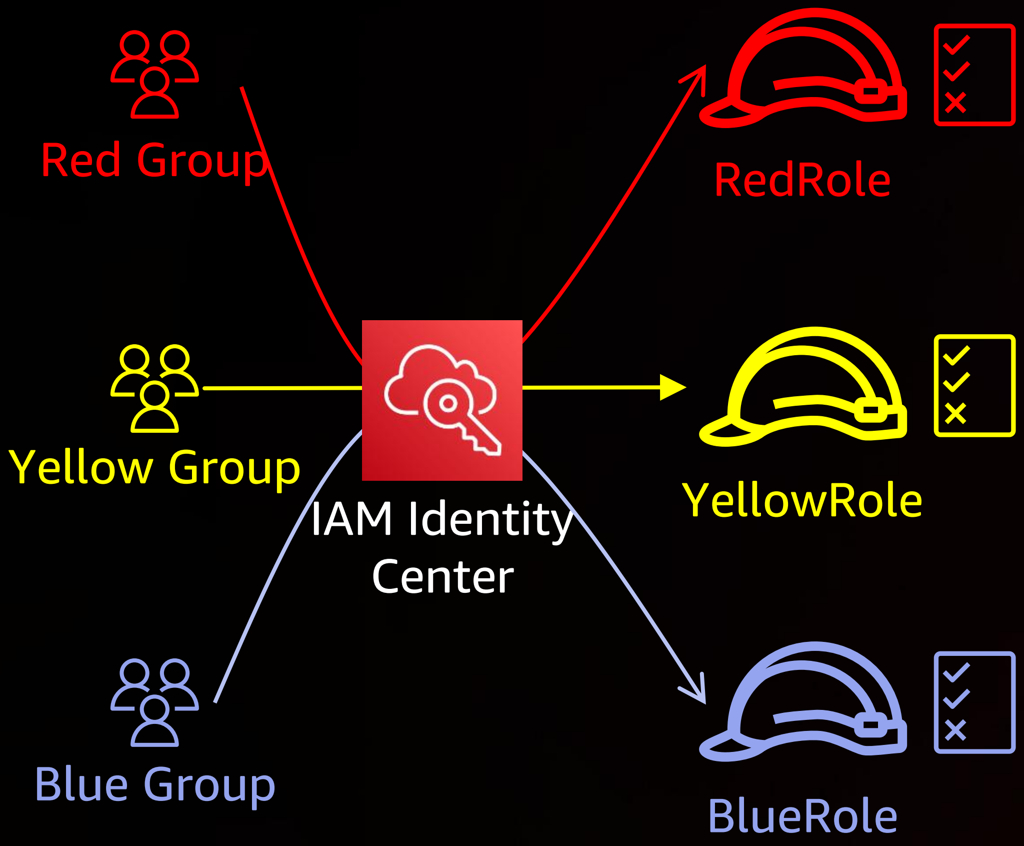
Doing the math

- This approach requires callers and bucket in same account, or a small number of accounts
- Each distinct combination of permissions needs an IAM role
  - IAM roles per account: 1,000 (5,000 max)
  - This works well for strictly role-based access schemes

- **Primary constraint:** Number of unique permissions use cases ➔ IAM role count

Tagging of IAM principals

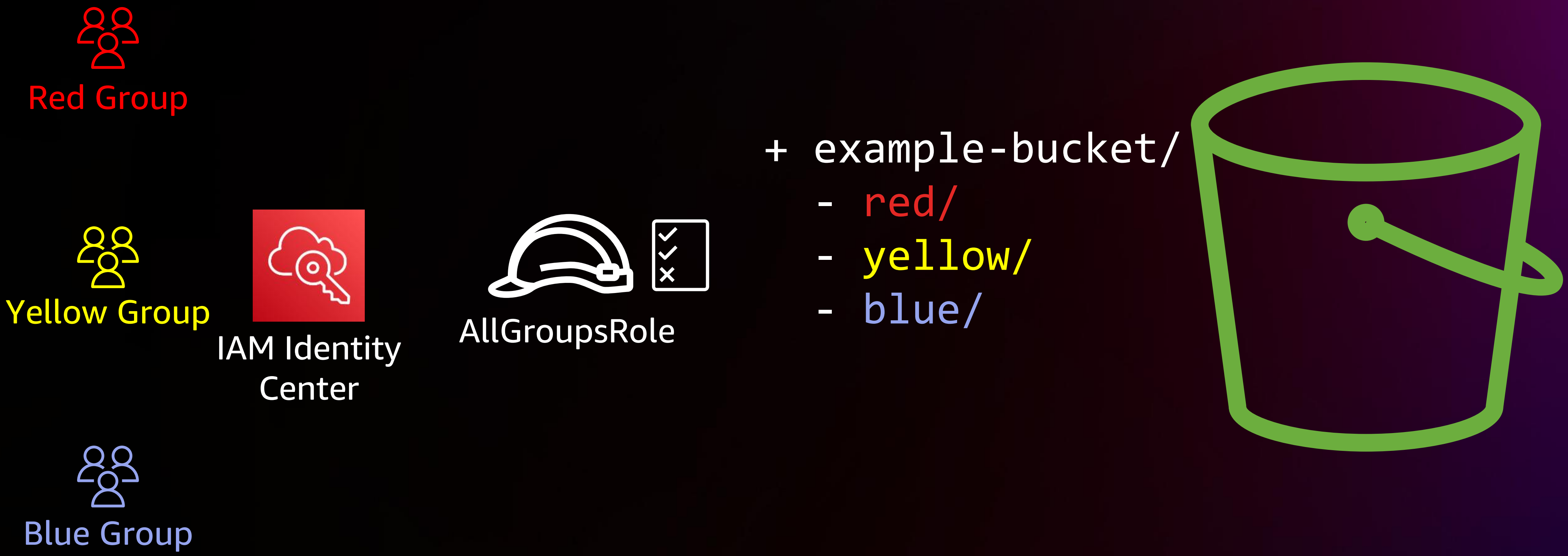
[SAME-ACCOUNT USE CASES]



```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

Tagging of IAM principals

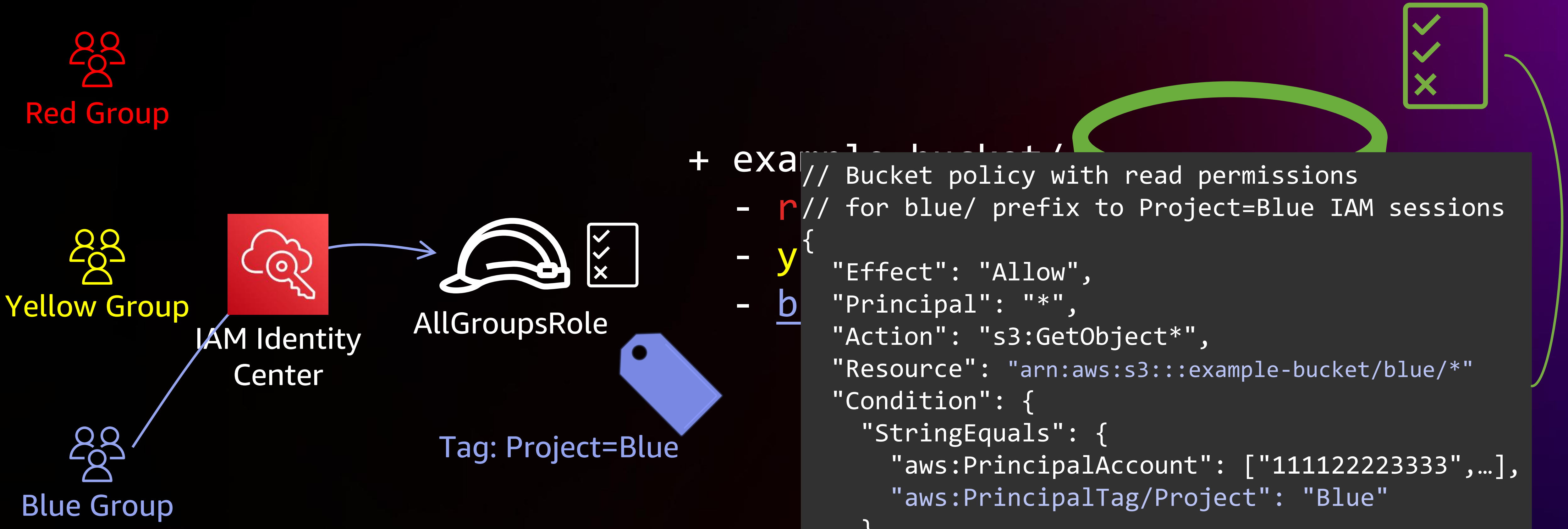
[SAME-ACCOUNT USE CASES]



```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

Tagging of IAM principals

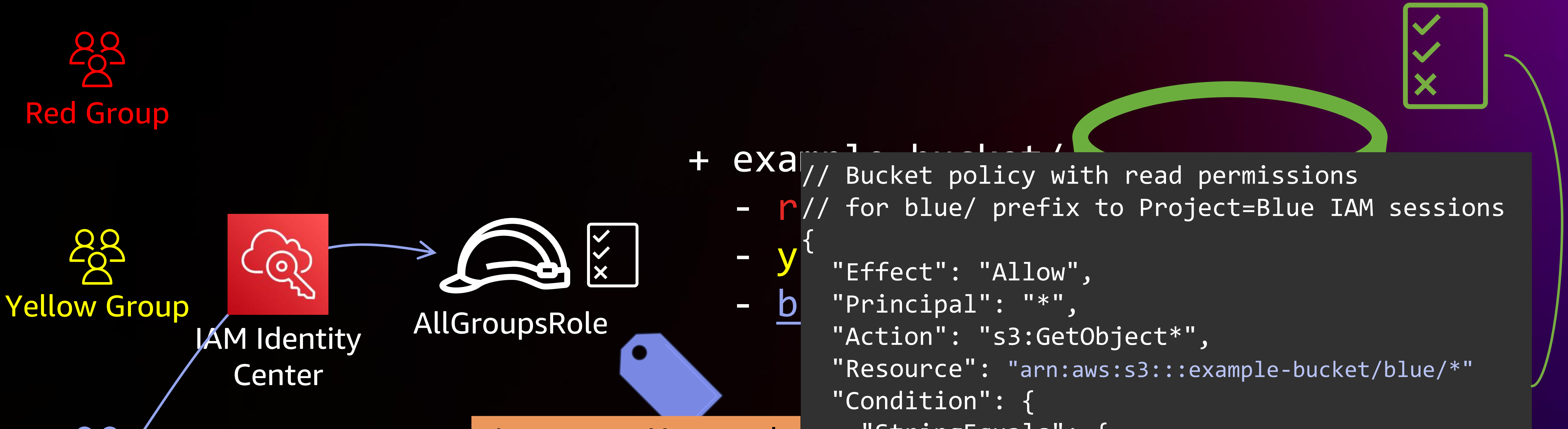
[SAME-ACCOUNT USE CASES]



```
// Bucket policy with read permissions
// for blue/ prefix to Project=Blue IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…],
      "aws:PrincipalTag/Project": "Blue"
    }
  }
}
```

Tagging of IAM principals

[SAME-ACCOUNT USE CASES]

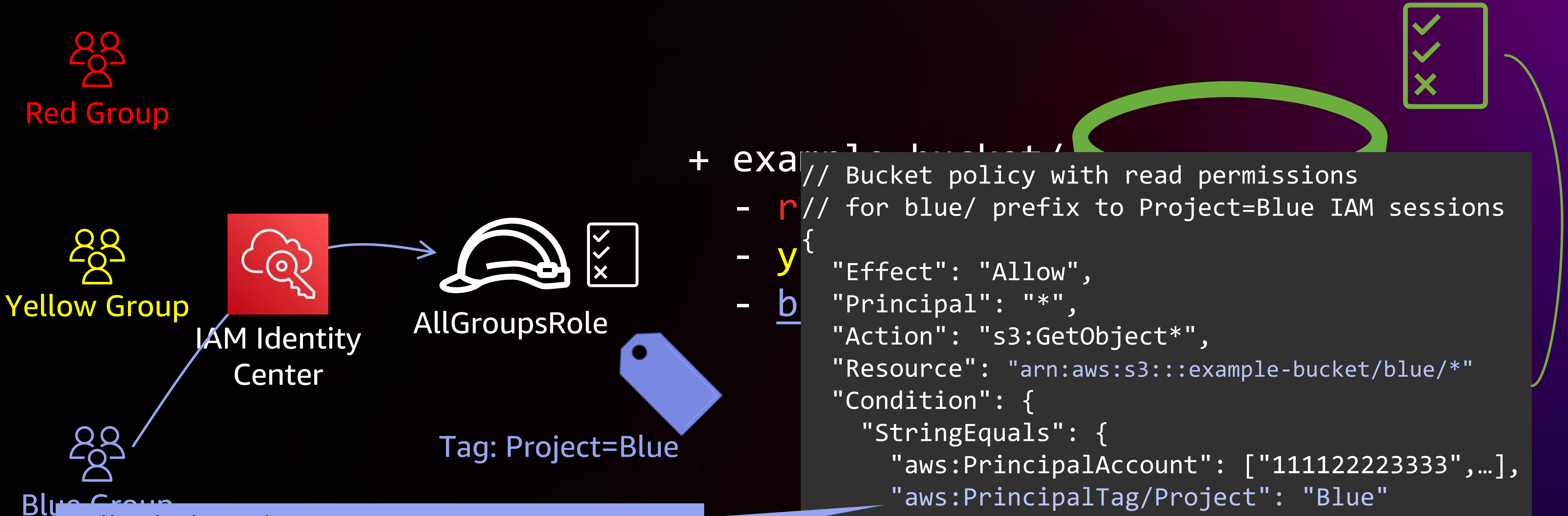


Important: You need to trust the account doing the tagging

```
// Bucket policy with read permissions
// for blue/ prefix to Project=Blue IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…]
      "aws:PrincipalTag/Project": "Blue"
    }
  }
}
```

Tagging of IAM principals

[SAME-ACCOUNT USE CASES]

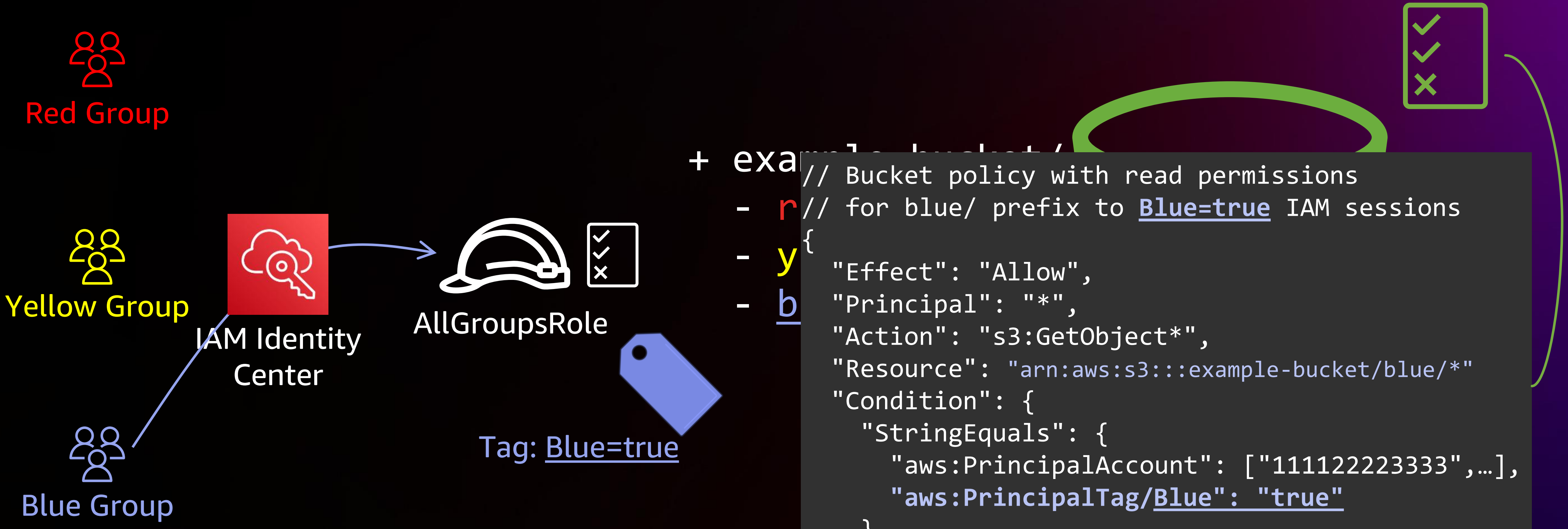


```
// Bucket policy with read permissions
// for blue/ prefix to Project=Blue IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…],
      "aws:PrincipalTag/Project": "Blue"
    }
  }
}
```

Still role-based access pattern: No easy way to represent two colors (e.g., yellow AND blue)

Tagging of IAM principals

[SAME-ACCOUNT USE CASES]



```
// Bucket policy with read permissions
// for blue/ prefix to Blue=true IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…],
      "aws:PrincipalTag/Blue": "true"
    }
  }
}
```

Comparing tagging schemes

```
// Bucket policy with read permissions
// for blue/ prefix to Project=Blue IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…],
      "aws:PrincipalTag/Project": "Blue"
    }
  }
}
```

Can have as many values for this as you want . . . but only one at a time

```
// Bucket policy with read permissions
// for blue/ prefix to Blue=true IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…],
      "aws:PrincipalTag/Blue": "true"
    }
  }
}
```

Project=Blue

Comparing tagging schemes

```
// Bucket policy with read permissions
// for blue/ prefix to Project=Blue IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…],
      "aws:PrincipalTag/Project": "Blue"
    }
  }
}
```

```
// Bucket policy with read permissions
// for blue/ prefix to Blue=true IAM sessions
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject*",
  "Resource": "arn:aws:s3:::example-bucket/blue/*"
  "Condition": {
    "StringEquals": {
      "aws:PrincipalAccount": ["111122223333",…],
      "aws:PrincipalTag/Blue": "true"
    }
  }
}
```

Allows for multiple values on a single session . . . up to a limit (50)

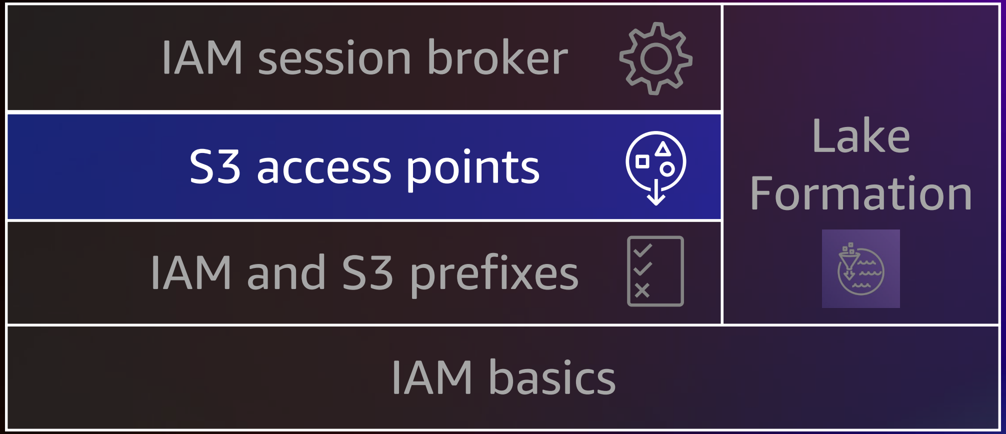
Blue=true
Yellow=true

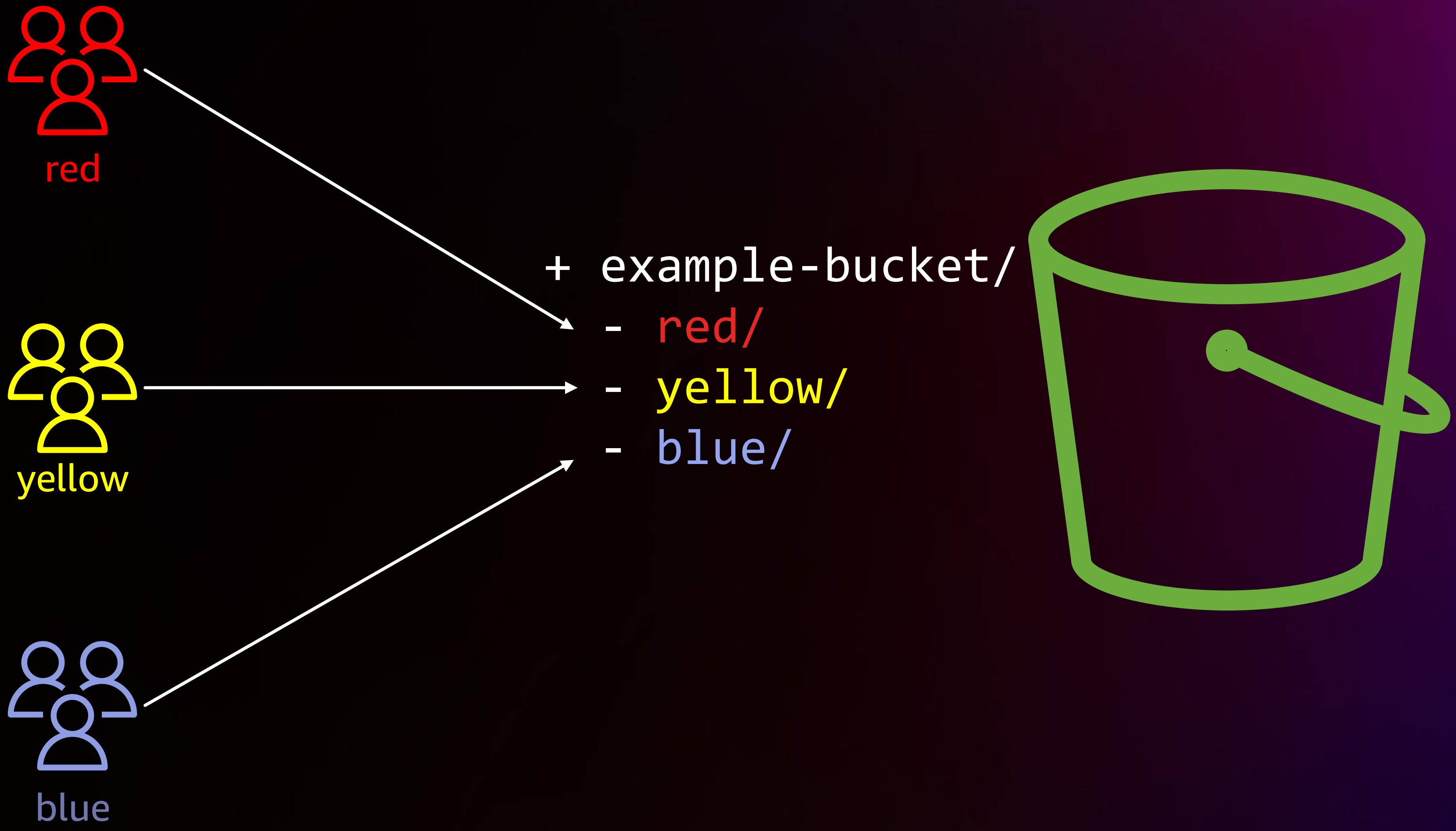
Doing the math

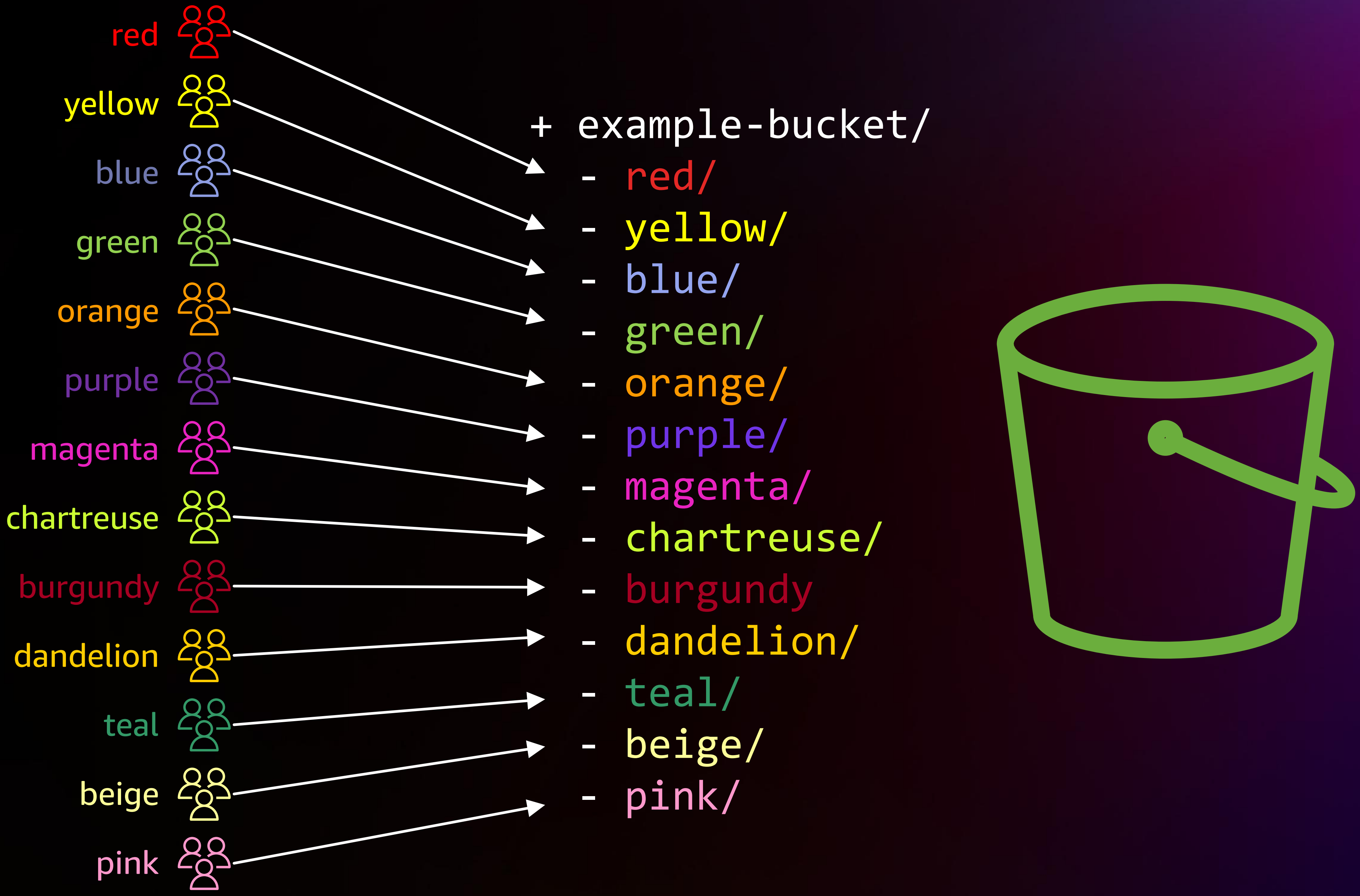
- Bucket policy grows with number of prefixes
  - Bucket policy max = 20 KB ➔ Best for <30 prefixes in bucket
- Each IAM session can have up to 50 tags

Primary constraints:

- Number of prefixes ➔ bucket policy size
- Number of groups for a single session ➔ IAM session tags

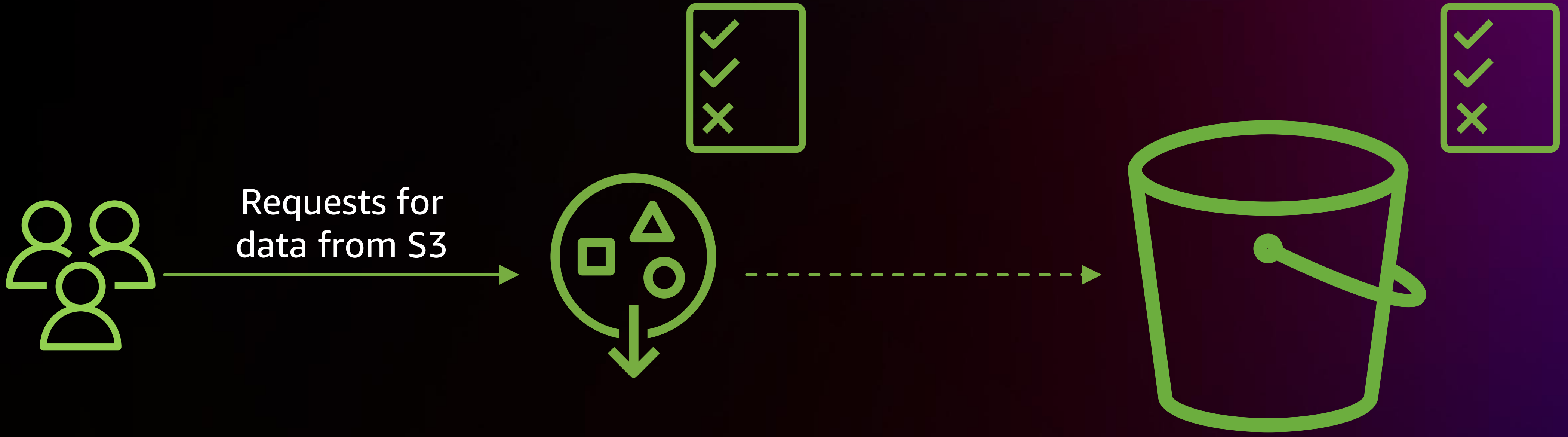

IAM session broker
S3 access points
IAM and S3 prefixes
Lake Formation
IAM basics


red
yellow
blue
+ example-bucket/
- red/
- yellow/
- blue/



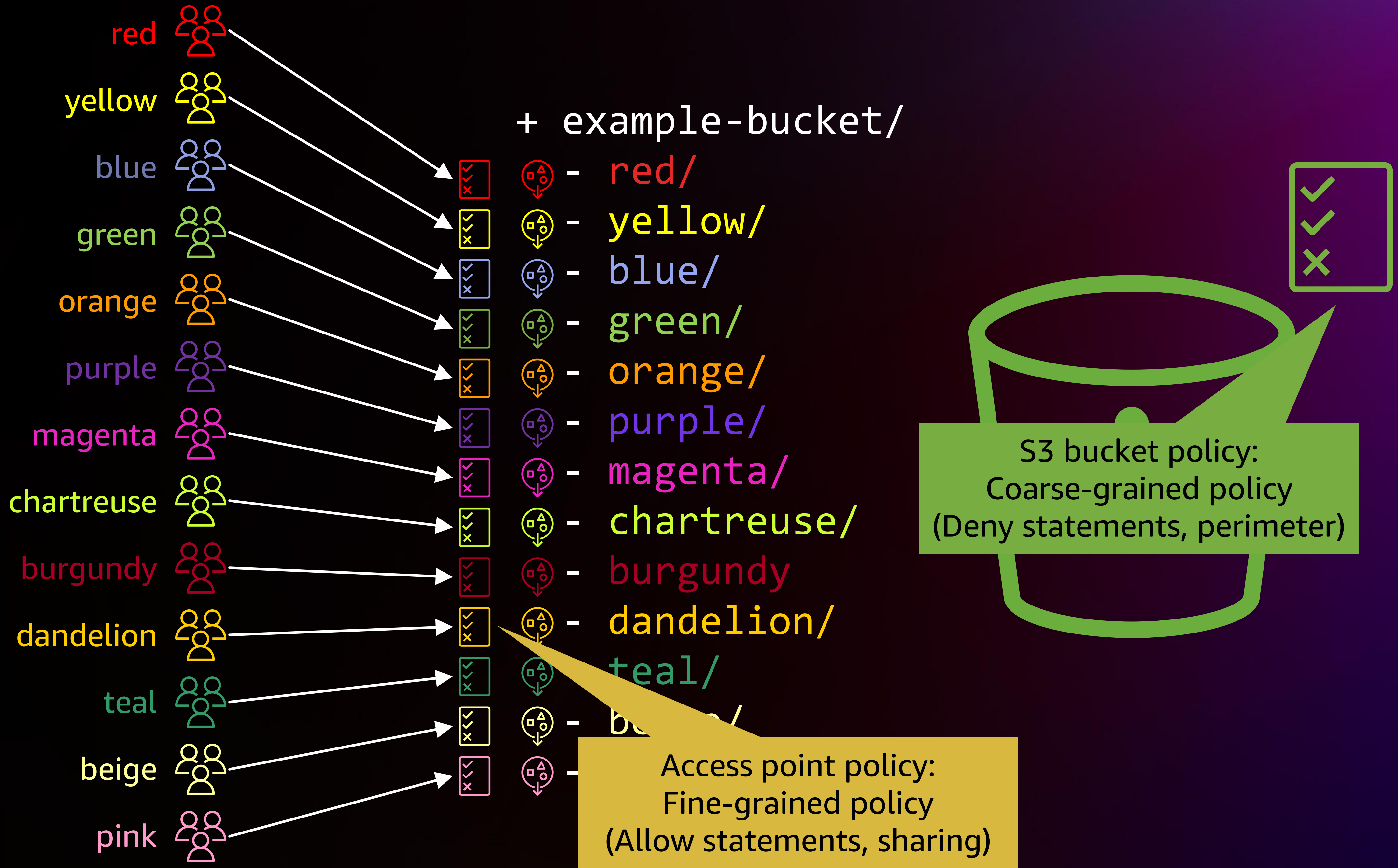
```
+ example-bucket/
  - red/
  - yellow/
  - blue/
  - green/
  - orange/
  - purple/
  - magenta/
  - chartreuse/
  - burgundy
  - dandelion/
  - teal/
  - beige/
  - pink/
```

S3 access points: Additional endpoints to a bucket



S3 access point with access point policy

S3 bucket with bucket policy


red
yellow
blue
green
orange
purple
magenta
chartreuse
burgundy
dandelion
teal
beige
pink
+ example-bucket/
- red/
- yellow/
- blue/
- green/
- orange/
- purple/
- magenta/
- chartreuse/
- burgundy
- dandelion/
teal/
- be e/
S3 bucket policy:
Coarse-grained policy
(Deny statements, perimeter)
Access point policy:
Fine-grained policy
(Allow statements, sharing)

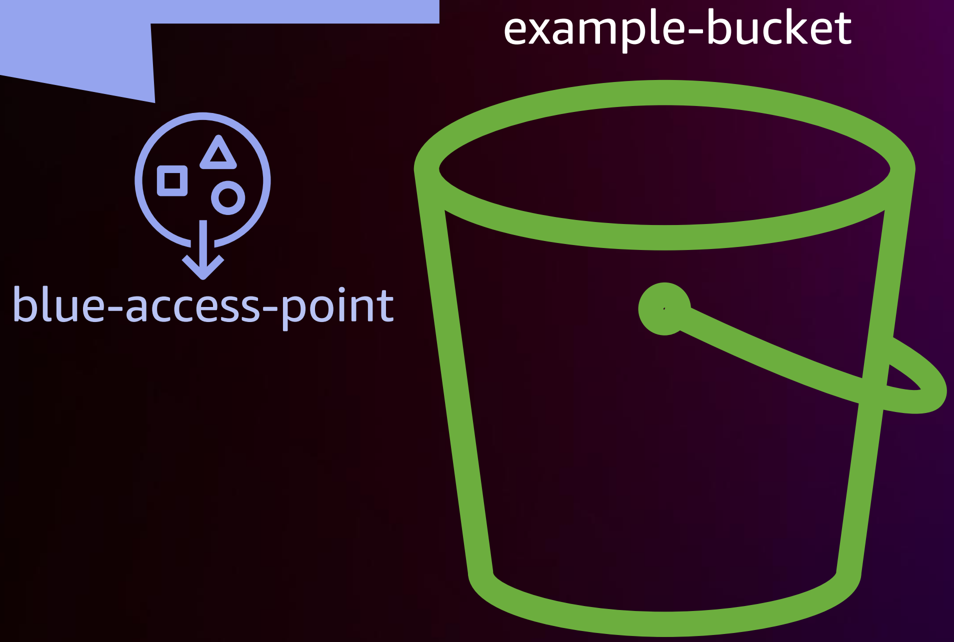
S3 access points: Additional endpoints to a bucket

Typical request to an S3 bucket (without access point)

```
$ aws s3api get-object \
  --bucket example-bucket
  --key blue/ocean-data.csv

# … makes a request to …
GET https://example-bucket.s3.us-east-2.amazonaws.com/blue/ocean-data.csv
```

S3 access points: Additional endpoints to a bucket

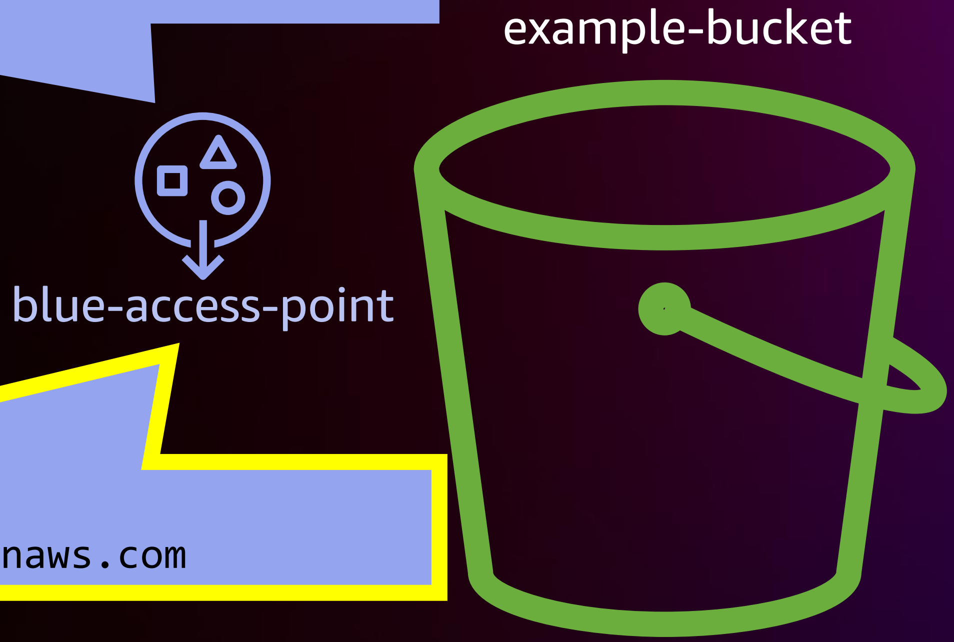
DNS name of the access point
`blue-access-point-111122223333.s3-accesspoint.us-east-2.amazonaws.com`



DNS name of the bucket
`example-bucket.s3.us-east-2.amazonaws.com`

S3 access points: Additional endpoints to a bucket

DNS name of the access point
`blue-access-point-111122223333.s3-accesspoint.us-east-2.amazonaws.com`

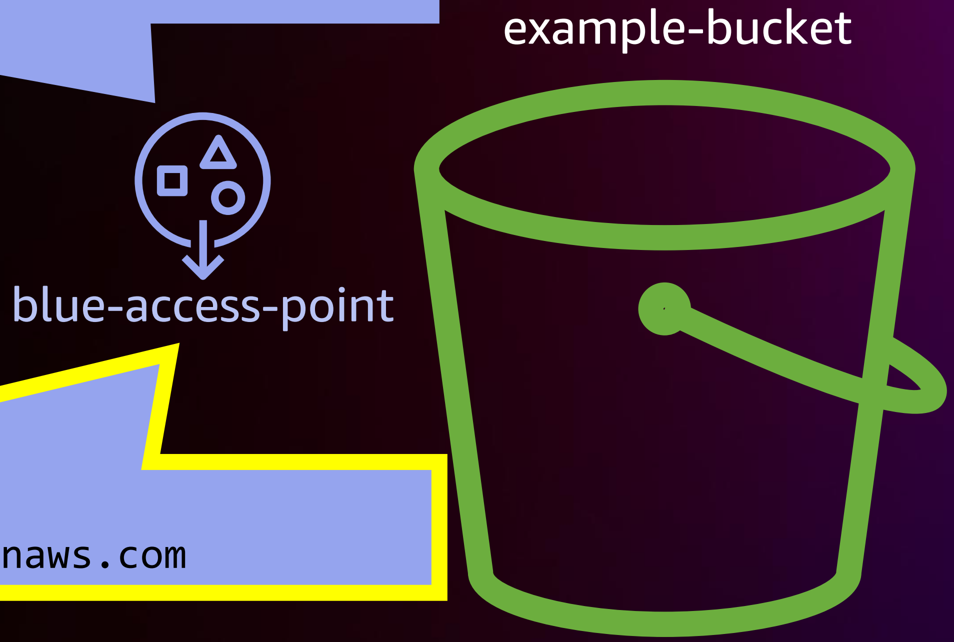


DNS name of the access point, using bucket alias
`blue-access-point-razthp3ehn-s3alias.s3.us-east-2.amazonaws.com`

DNS name of the bucket
`example-bucket.s3.us-east-2.amazonaws.com`

S3 access points: Additional endpoints to a bucket

DNS name of the access point
`blue-access-point-111122223333.s3-accesspoint.us-east-2.amazonaws.com`

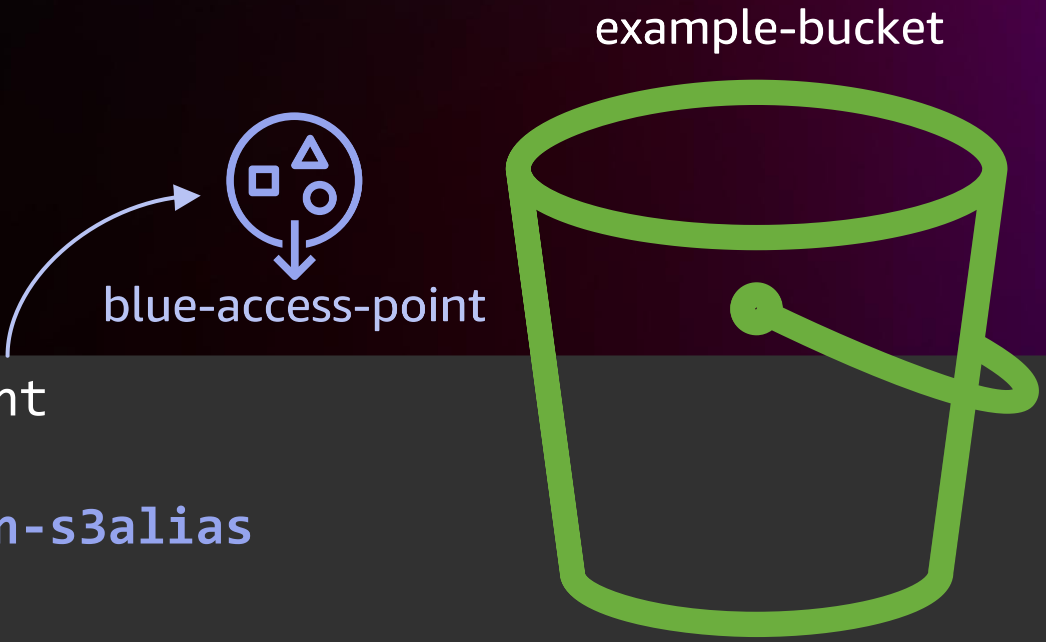


DNS name of the access point, using bucket alias
`blue-access-point-razthp3ehn-s3alias.s3.us-east-2.amazonaws.com`

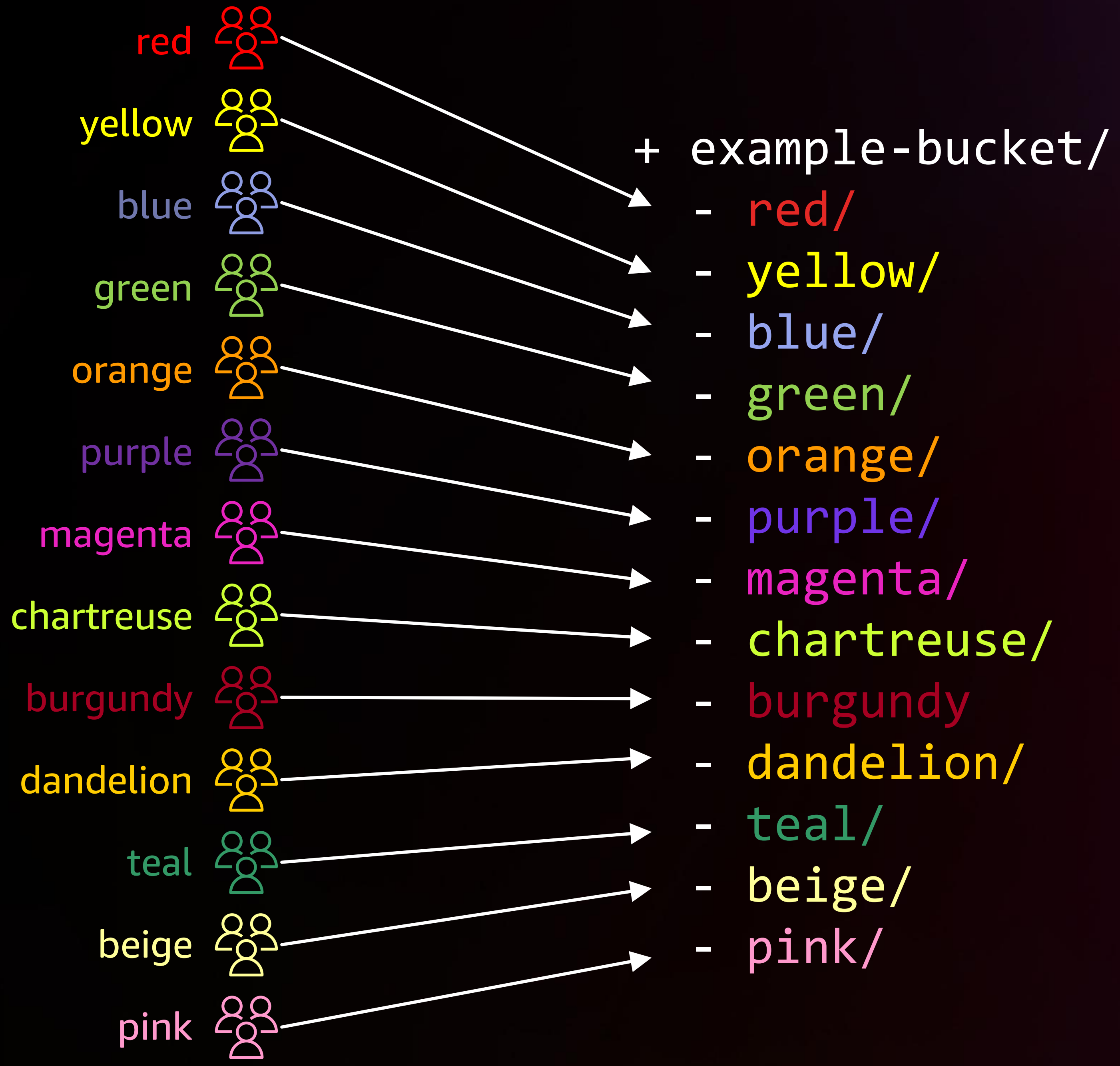
Use access point alias like a bucket name

DNS name of the bucket
`example-bucket.s3.us-east-2.amazonaws.com`

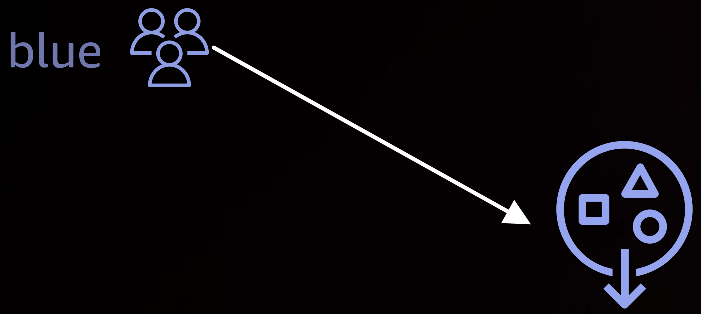
S3 access points: Additional endpoints to a bucket



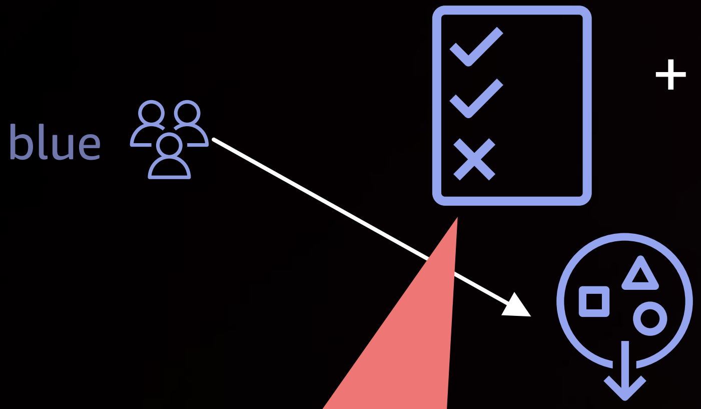
```
# Requesting data via an S3 access point
$ aws s3api get-object \
  --bucket blue-access-point-razthp3ehn-s3alias
  --key blue/ocean-data.csv

# … makes a request to …
GET https://blue-access-point-razthp3ehn-s3alias.s3.us-east-2.amazonaws.com/
blue/ocean-data.csv
```



```
+ example-bucket/
  - red/
  - yellow/
  - blue/
  - green/
  - orange/
  - purple/
  - magenta/
  - chartreuse/
  - burgundy
  - dandelion/
  - teal/
  - beige/
  - pink/
```

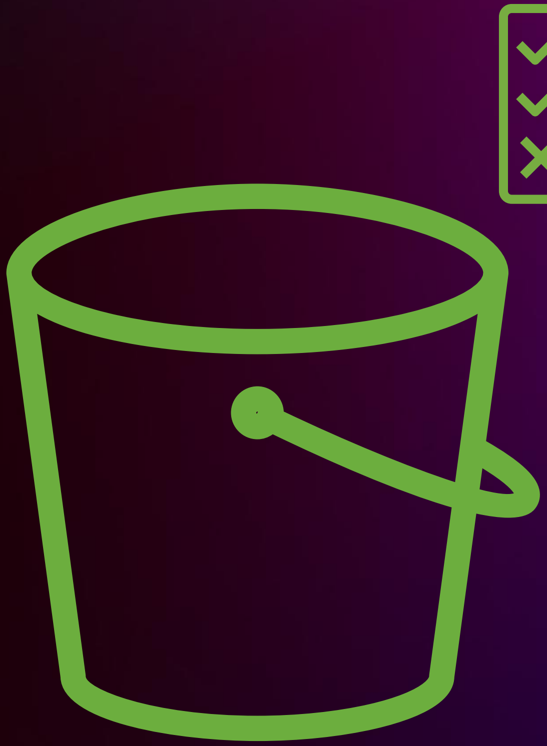


```
+ example-bucket/
  - red/
  - yellow/
  - blue/
  - green/
  - orange/
  - purple/
  - magenta/
  - chartreuse/
  - burgundy
  - dandelion/
  - teal/
  - beige/
  - pink/
```



Each S3 access point has its own policy, in addition to the S3 bucket policy

```
+ example-bucket/
  - red/
  - yellow/
  - blue/
  - green/
  - orange/
  - purple/
  - magenta/
  - chartreuse/
  - burgundy
  - dandelion/
  - teal/
  - beige/
  - pink/
```



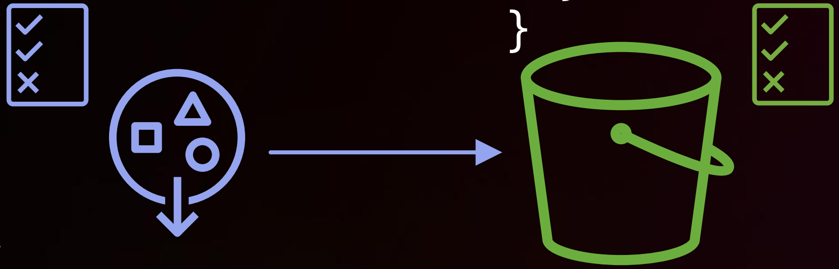
Using S3 access point and bucket policies together

Access point policy:
Grants access to account 444455556666

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "444455556666"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:us-east-
2:111122223333:accesspoint/blue-access-
point/object/blue/*"
}
```

S3 bucket policy:
Grants access via access points in this account

```
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/*",
  "Condition": {
    "StringEquals": {
      "s3:DataAccessPointAccount":
                                  "111122223333",
      "aws:PrincipalOrgId": "o-a1b2c3"
    }
  }
}
```



Using S3 access point and bucket policies together

Access point policy:

Grants access to account 444455556666

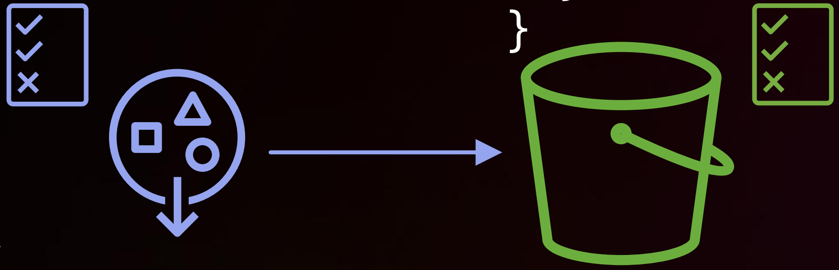
```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "444455556666"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:us-east-
2:111122223333:accesspoint/blue-access-
point/object/blue/*"
}
```

ARN for an object when accessed via an access point

S3 bucket policy:

Grants access via access points in this account

```
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/*",
  "Condition": {
    "StringEquals": {
      "s3:DataAccessPointAccount":
                                 "111122223333",
      "aws:PrincipalOrgId": "o-a1b2c3"
    }
  }
}
```



Using S3 access point and bucket policies together

Access point policy:

Grants access to account 444455556666

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "444455556666"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:us-east-
2:111122223333:accesspoint/blue-access-
point/object/blue/*"
}
```

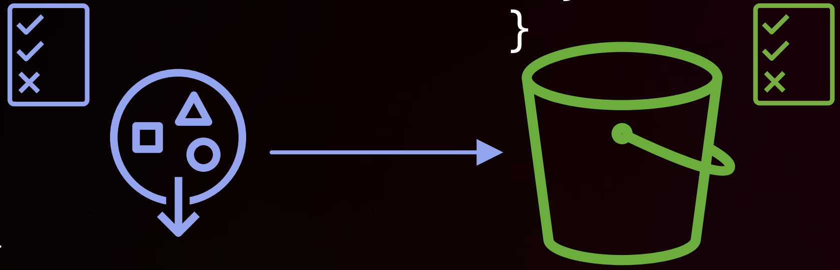
ARN for an object when accessed via an access point

S3 bucket policy:

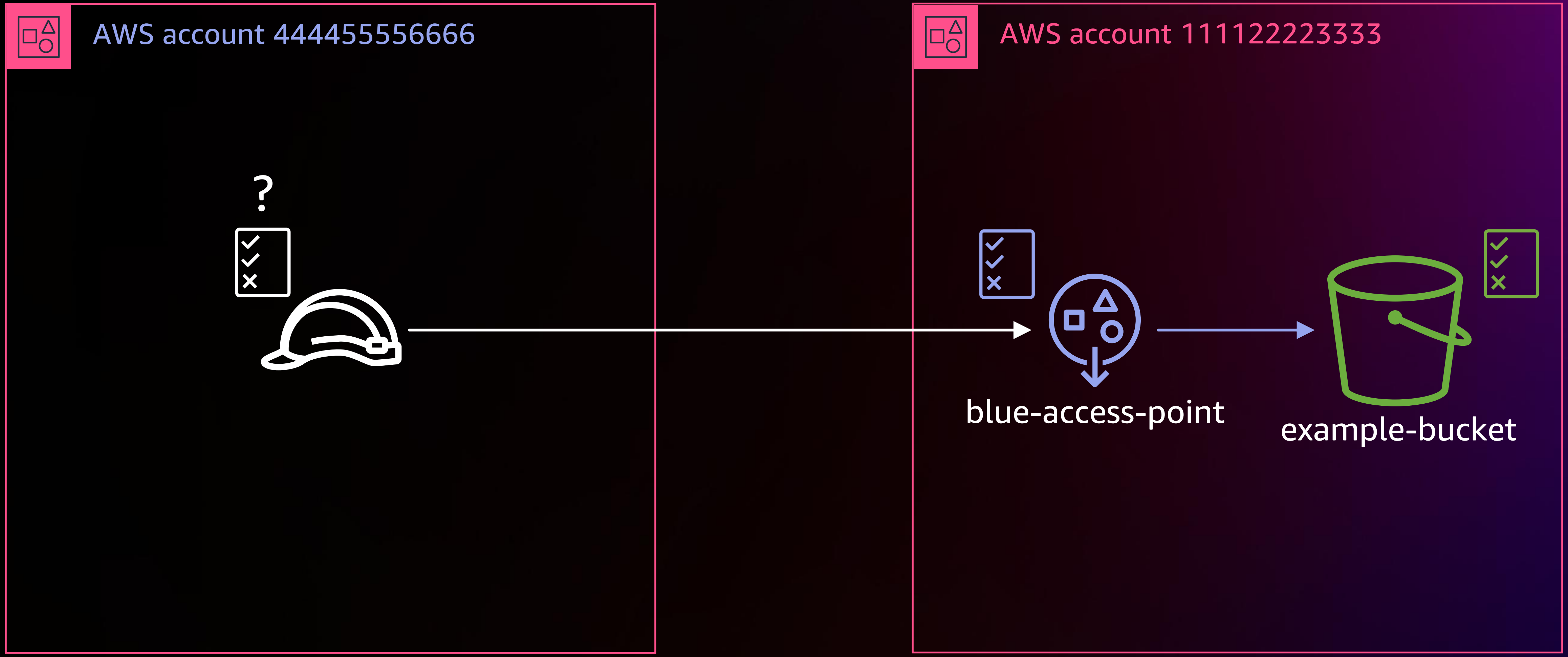
Grants access via access points in this account

```
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/*",
  "Condition": {
    "StringEquals": {
      "s3:DataAccessPointAccount":
                                  "111122223333",
      "aws:PrincipalOrgId": "o-a1b2c3"
    }
  }
}
```

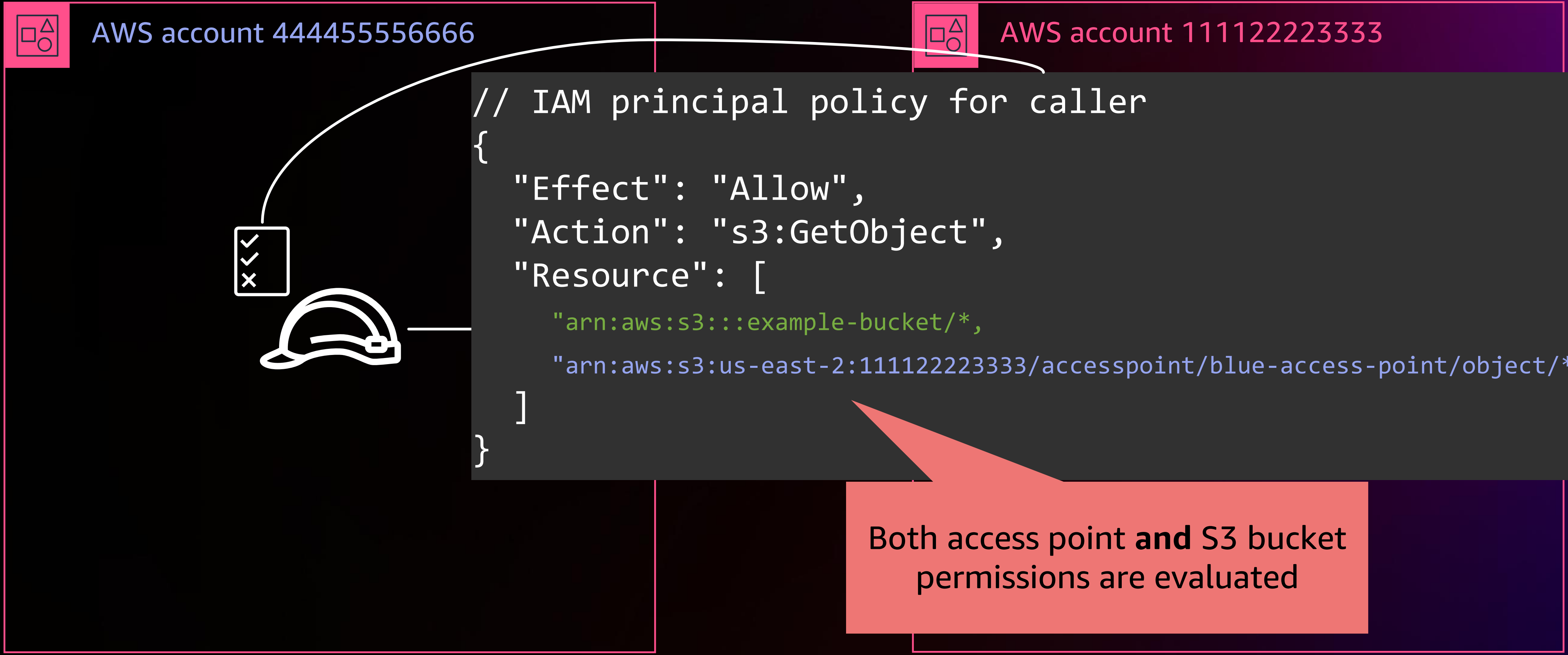
Will allow any access from this AWS Organization that is allowed by an access point



Using S3 access points: IAM principal policy



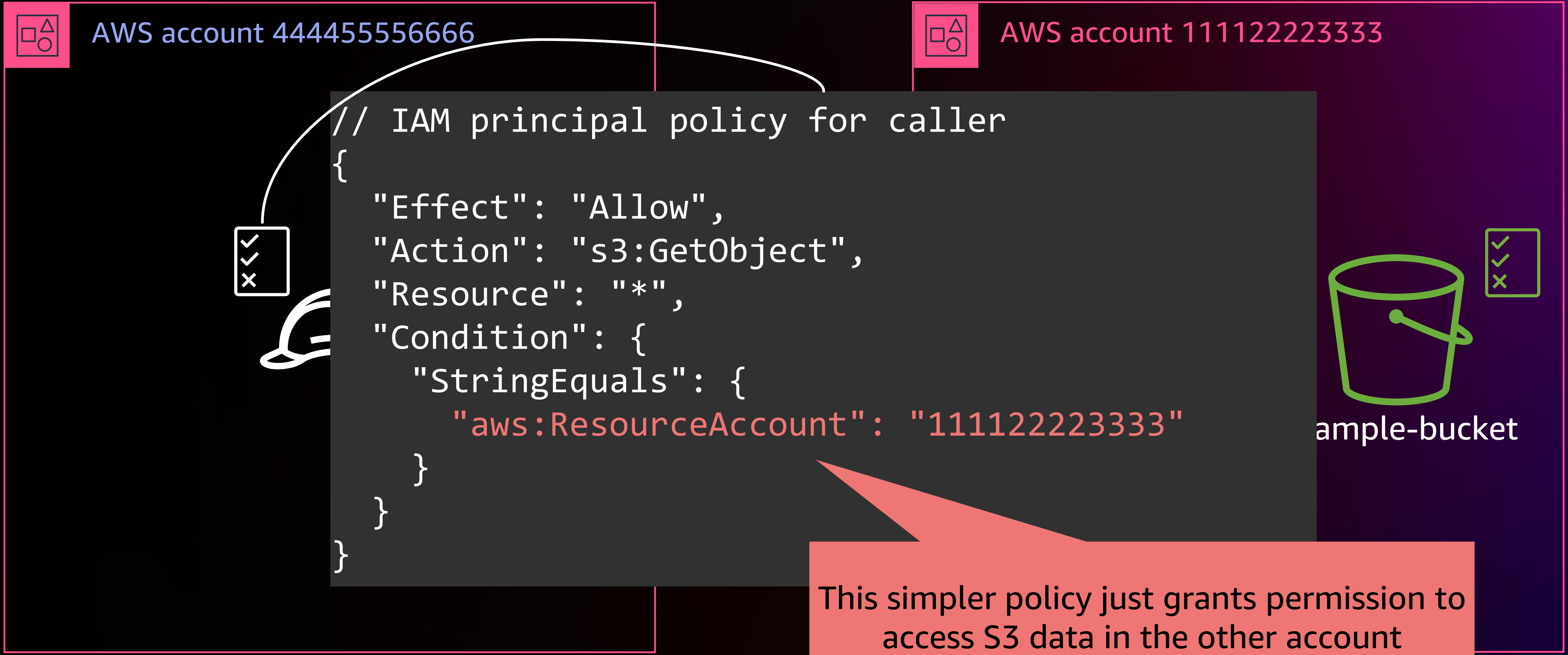
Using S3 access points: IAM principal policy



```
// IAM principal policy for caller
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": [
    "arn:aws:s3:::example-bucket/*,
    "arn:aws:s3:us-east-2:111122223333/accesspoint/blue-access-point/object/*"
  ]
}
```

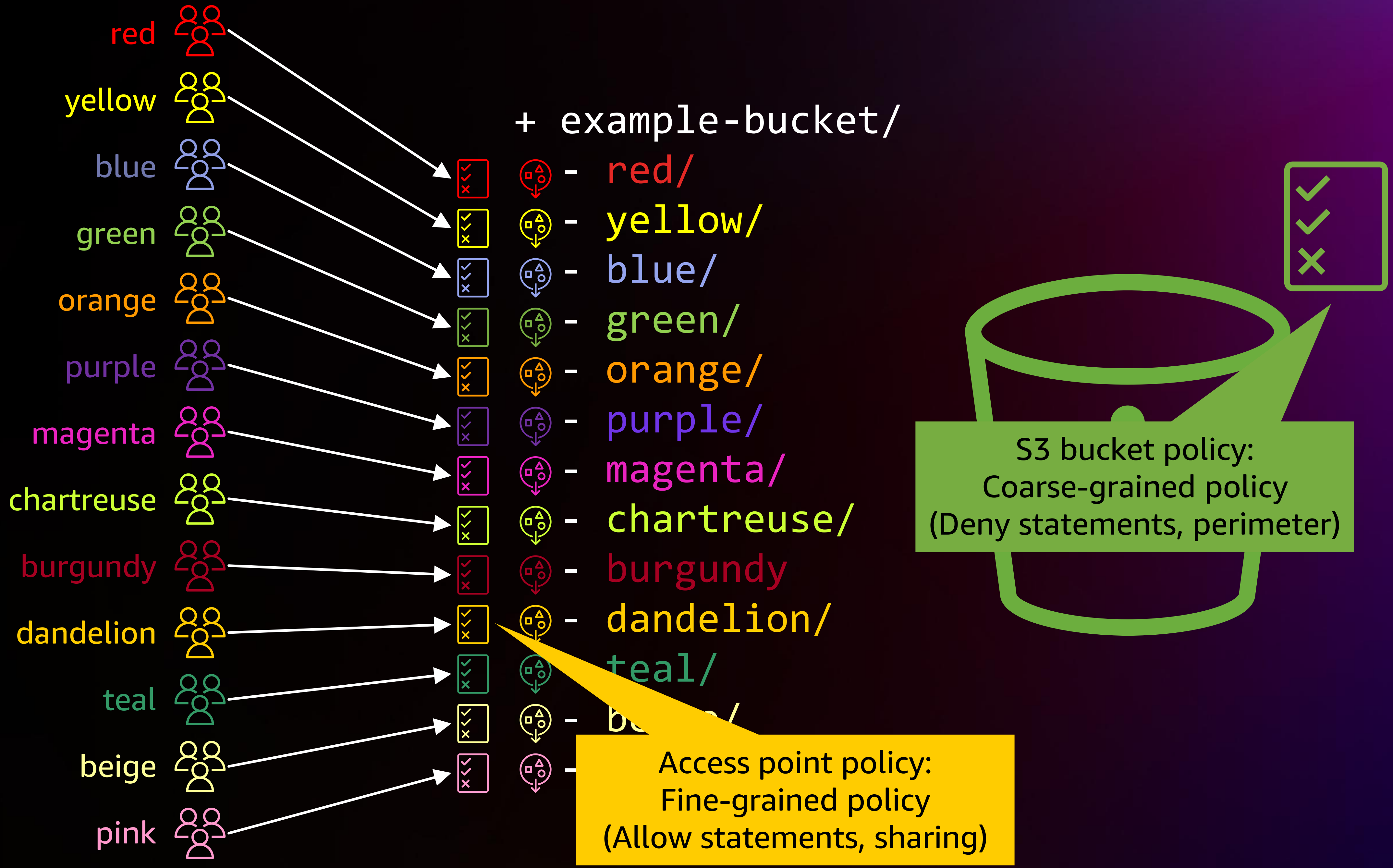
Both access point **and** S3 bucket permissions are evaluated

Using S3 access points: IAM principal policy



```
// IAM principal policy for caller
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "111122223333"
    }
  }
}
```

This simpler policy just grants permission to access S3 data in the other account


red
yellow
blue
green
orange
purple
magenta
chartreuse
burgundy
dandelion
teal
beige
pink
+ example-bucket/
- red/
- yellow/
- blue/
- green/
- orange/
- purple/
- magenta/
- chartreuse/
- burgundy
- dandelion/
- teal/
- be
-
S3 bucket policy:
Coarse-grained policy
(Deny statements, perimeter)
Access point policy:
Fine-grained policy
(Allow statements, sharing)

Do S3 access points fit your use case?

Best-fit use cases:

- Large number of static access patterns (up to 10,000 by default)
- Delegation-with-guardrails: Separate policies of access points and S3 buckets

Considerations:

- Discovery mechanism: How to find the right access point?
- Rate of change required for access patterns

Do S3 access points fit your use case?

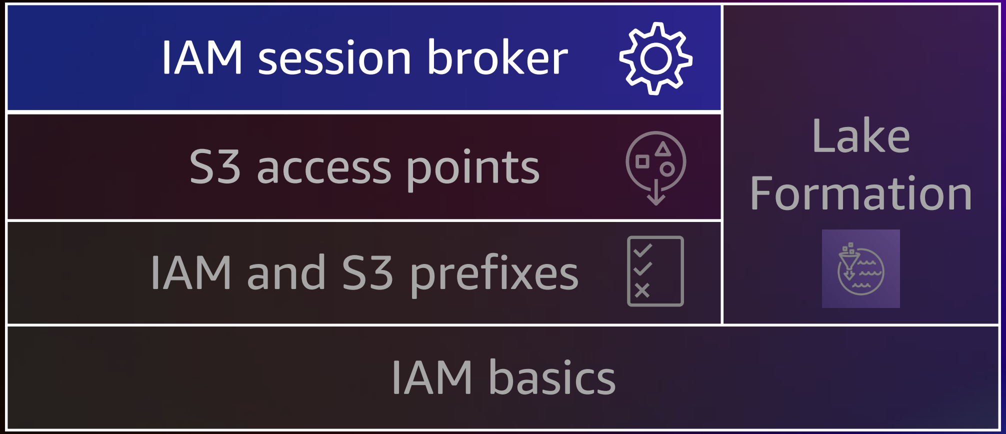
Best-fit use cases:

- Large number of static access patterns (up to 10,000 by default)
- Delegation-with-guardrails: Separate policies of access points and S3 buckets

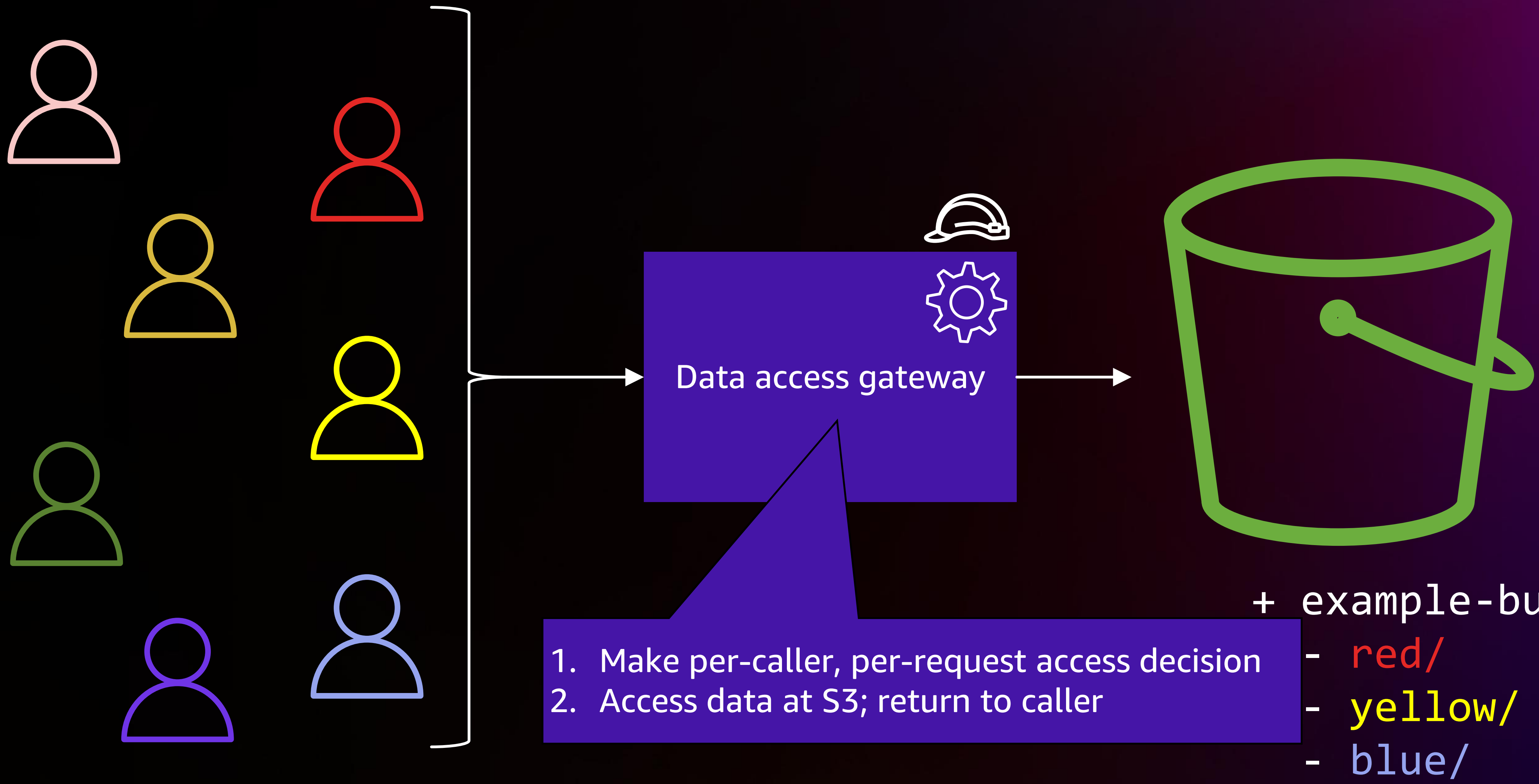
Considerations:

- Discovery mechanism: How to find the right access point?
- Rate of change required for access patterns

. . . What to do if they're dynamic?


IAM session broker
S3 access points
IAM and S3 prefixes
Lake Formation
IAM basics

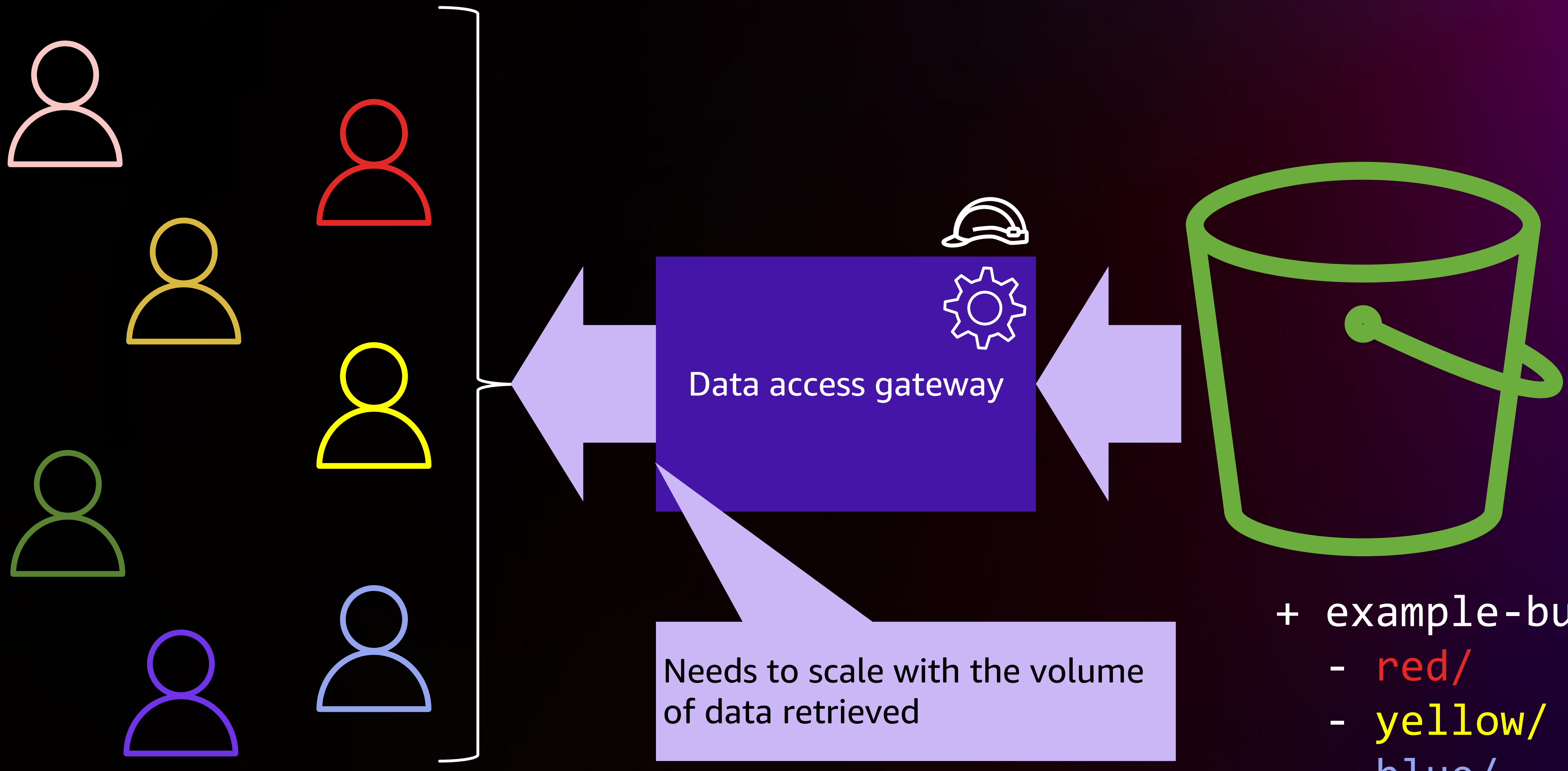
One solution: Build a gateway



1. Make per-caller, per-request access decision
2. Access data at S3; return to caller

```
+ example-bucket/
- red/
- yellow/
- blue/
```

One solution: Build a gateway



Needs to scale with the volume of data retrieved

```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

Sidebar: IAM sessions and scoping policy

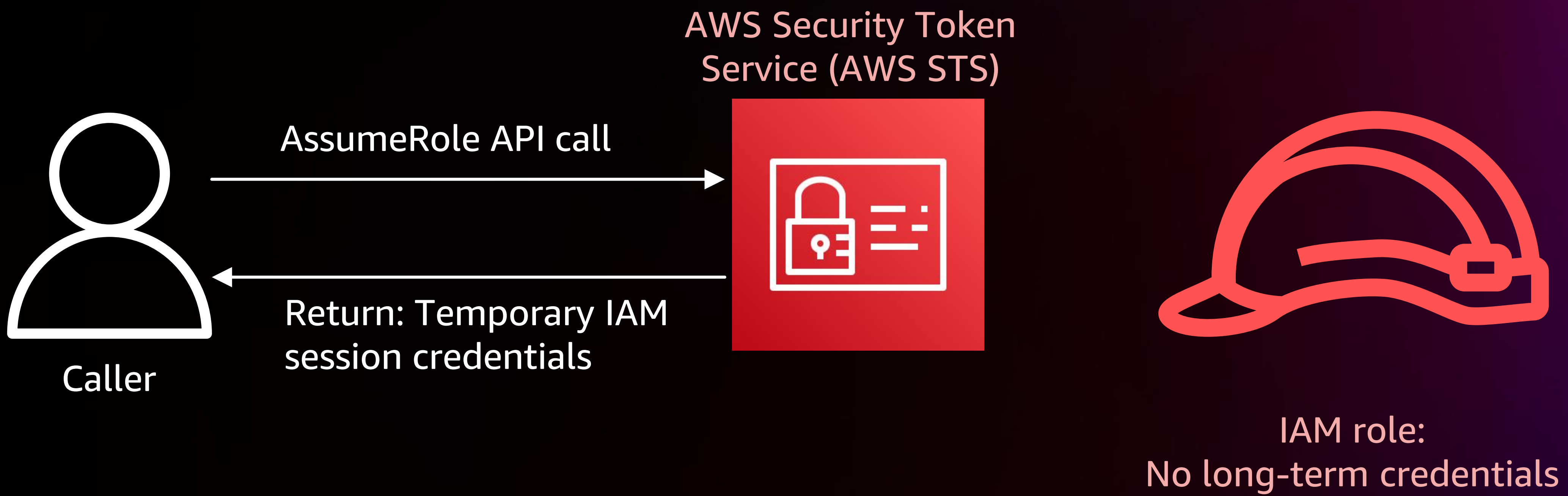
IAM role:
No long-term credentials

Sidebar: IAM sessions and scoping policy

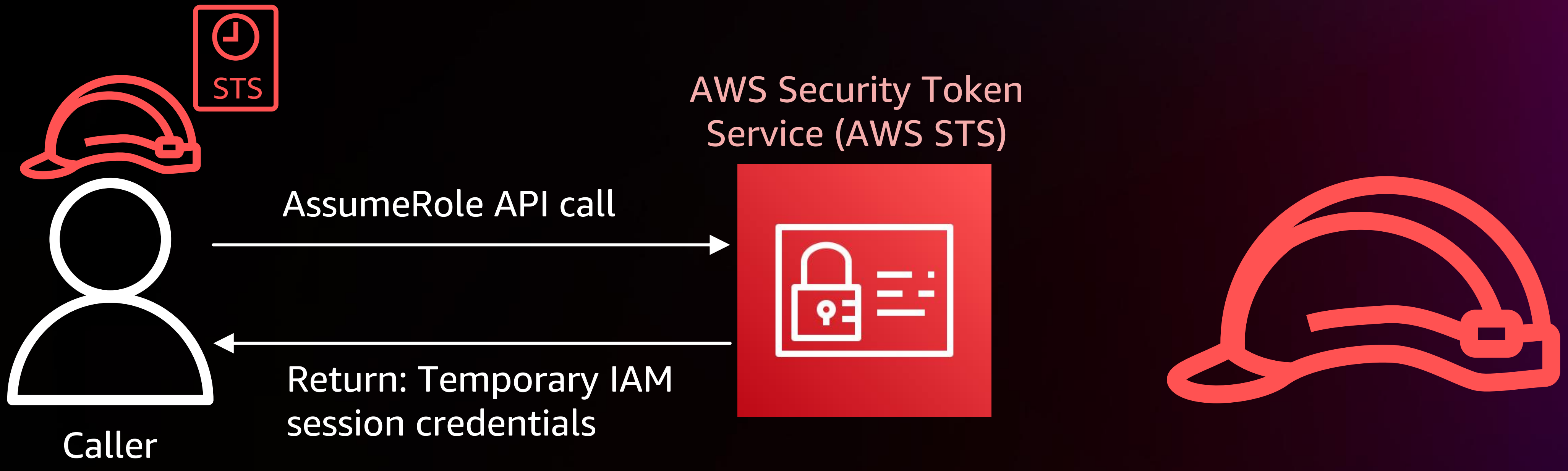
Caller

IAM role:
No long-term credentials

Sidebar: IAM sessions and scoping policy


AWS Security Token Service (AWS STS)
AssumeRole API call
Return: Temporary IAM session credentials
Caller
IAM role:
No long-term credentials

Sidebar: IAM sessions and scoping policy


STS
AWS Security Token Service (AWS STS)
AssumeRole API call
Return: Temporary IAM session credentials
Caller

IAM role:
No long-term credentials

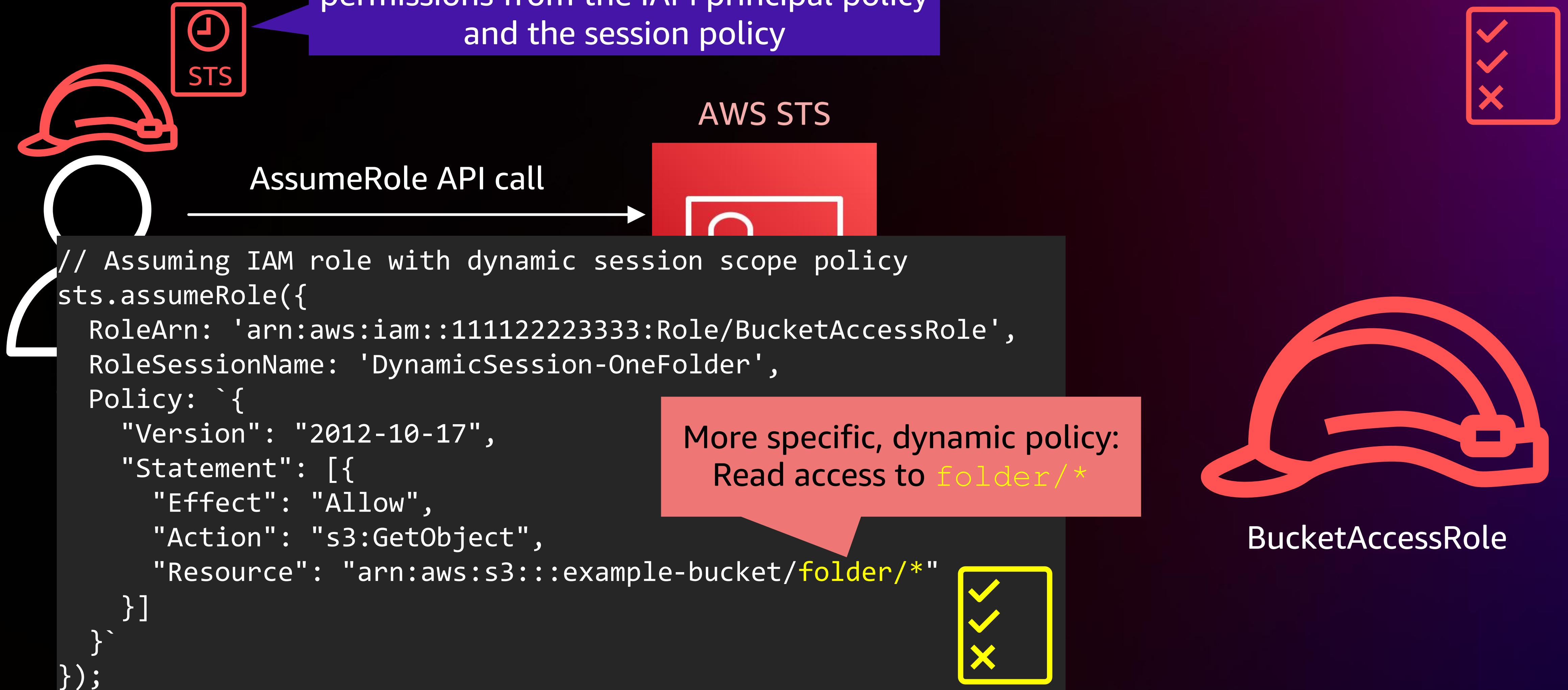
Sidebar: IAM sessions and scoping policy

```
// IAM principal policy (static)
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/*"
}
```

General read access to all objects in the bucket

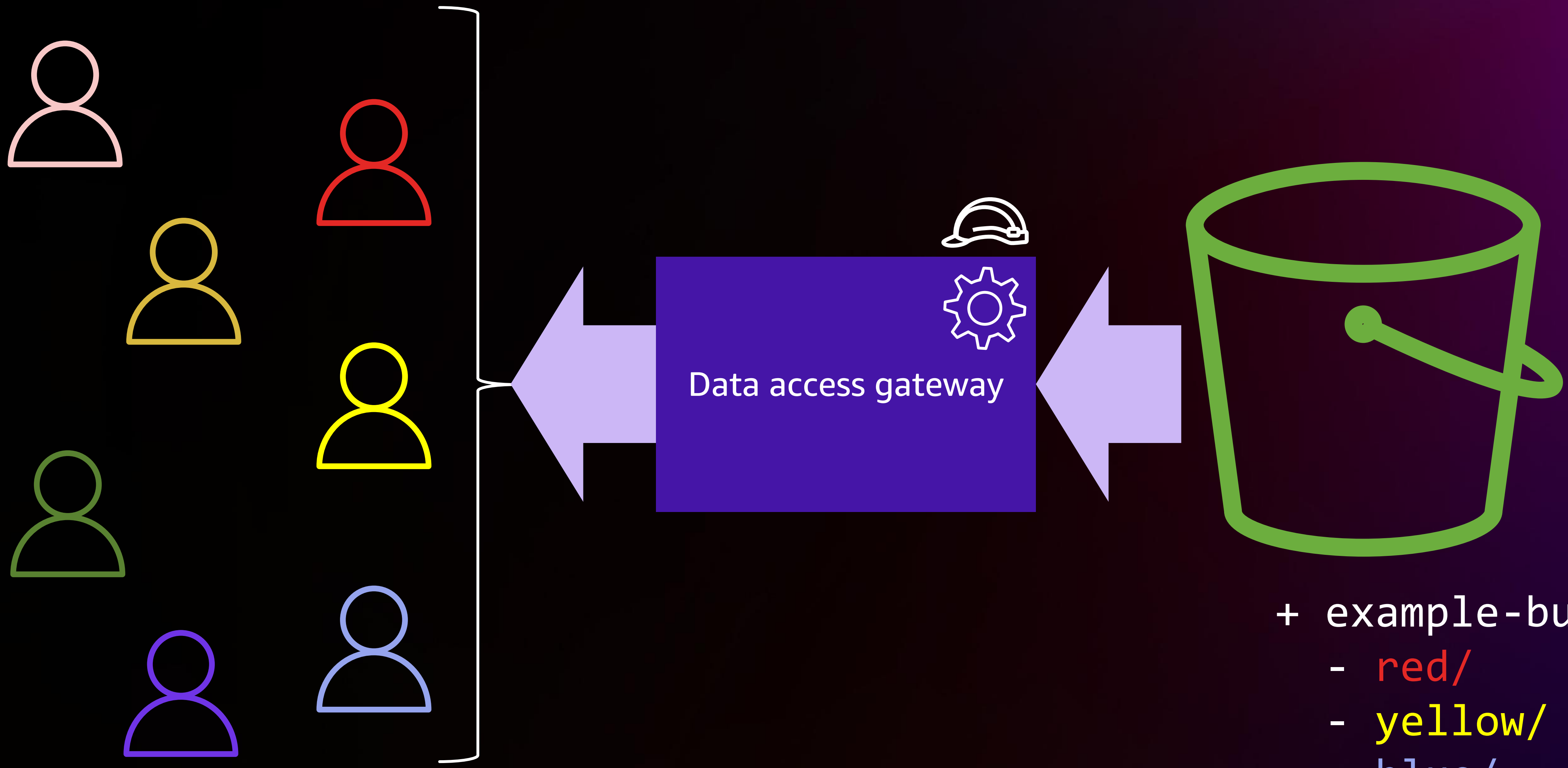
BucketAccessRole

Sidebar: IAM sessions and scoping policy


Resulting session has **intersection** of permissions from the IAM principal policy and the session policy
STS
AWS STS
AssumeRole API call
More specific, dynamic policy: Read access to folder/*
BucketAccessRole

```
// Assuming IAM role with dynamic session scope policy
sts.assumeRole({
  RoleArn: 'arn:aws:iam::111122223333:Role/BucketAccessRole',
  RoleSessionName: 'DynamicSession-OneFolder',
  Policy: `{
    "Version": "2012-10-17",
    "Statement": [{
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::example-bucket/folder/*"
    }]
  }`
});
```

Instead of building a gateway...

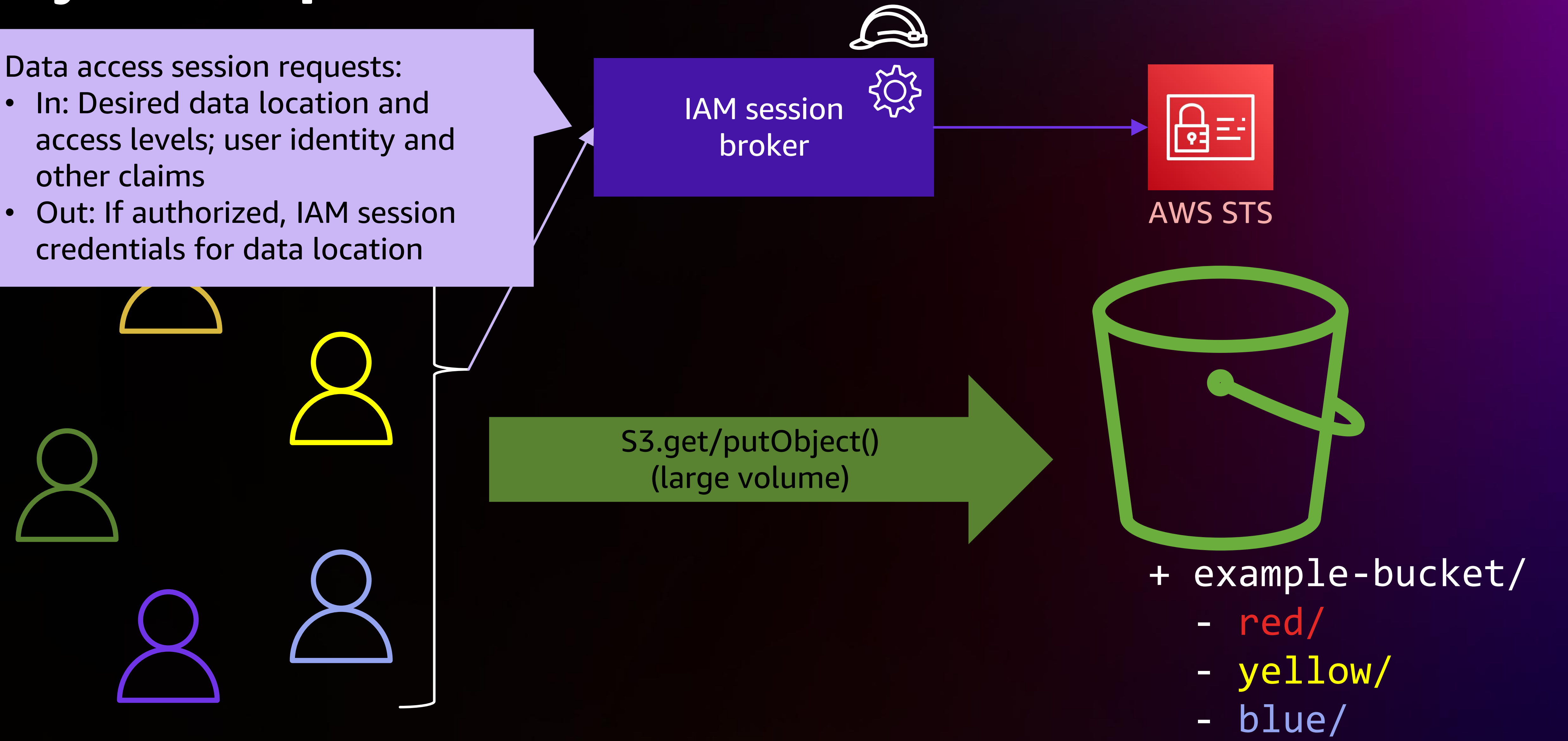


```
+ example-bucket/
  - red/
  - yellow/
  - blue/
```

Dynamic permissions at scale: Session broker

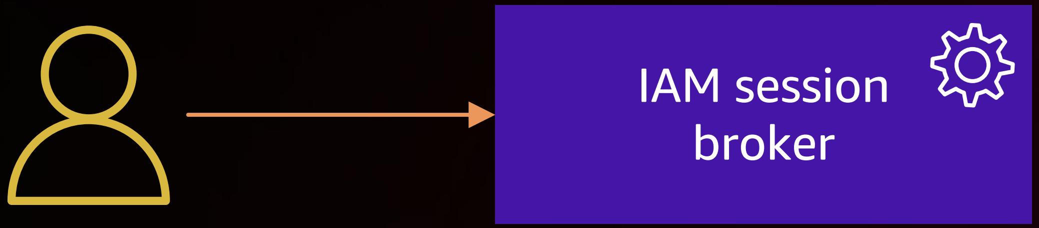
Data access session requests:

- In: Desired data location and access levels; user identity and other claims
- Out: If authorized, IAM session credentials for data location



How to implement an IAM session broker

1. Authenticate caller
2. Make access decision: Should the caller be able to access the S3 location requested (e.g., `s3://example-bucket/folder/`)?
3. If yes, call STS AssumeRole API to create a session from a central IAM role with a superset access policy. Specify a session policy that grants access to the requested location.
4. Return STS credentials to caller, who will use them to sign requests to S3

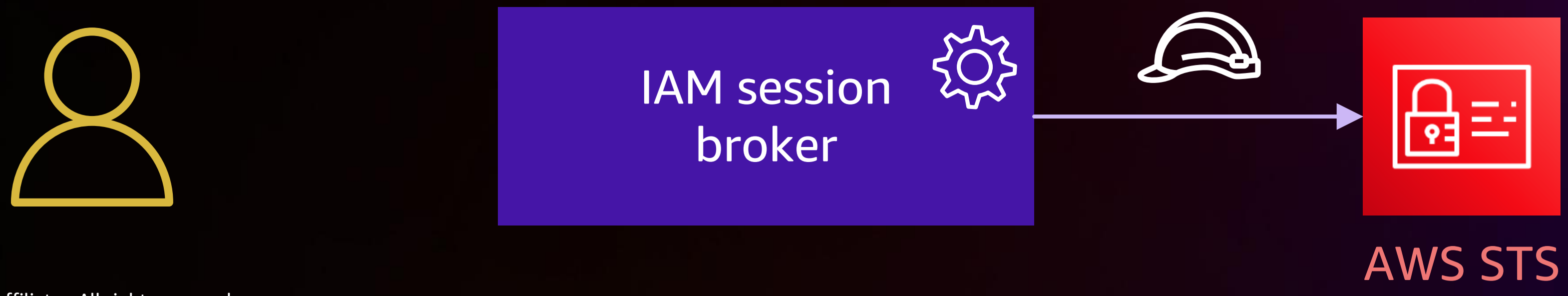


How to implement an IAM session broker

1. Authenticate caller
2. Make access decision: Should the caller be able to access the S3 location requested (e.g., `s3://example-bucket/folder/`)?
3. If yes, call STS AssumeRole API to create a session from a central IAM role with a superset access policy. Specify a session policy that grants access to the requested location.
4. Return STS credentials to caller, who will use them to sign requests to S3

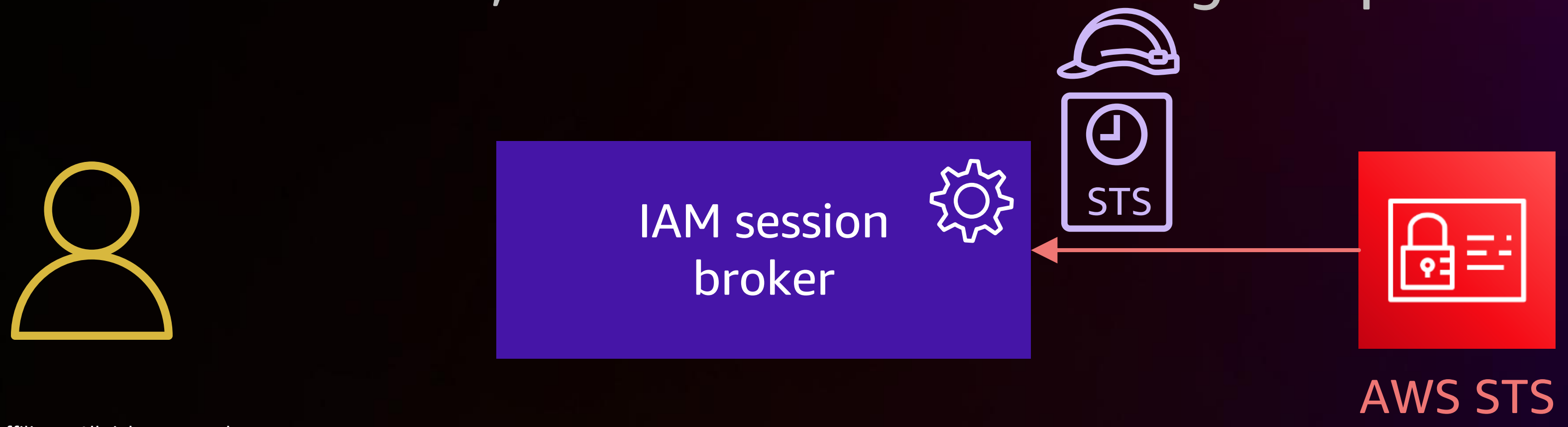
How to implement an IAM session broker

1. Authenticate caller
2. Make access decision: Should the caller be able to access the S3 location requested (e.g., `s3://example-bucket/folder/`)?
3. If yes, call STS AssumeRole API to create a session from a central IAM role with a superset access policy. Specify a session policy that grants access to the requested location.
4. Return STS credentials to caller, who will use them to sign requests to S3



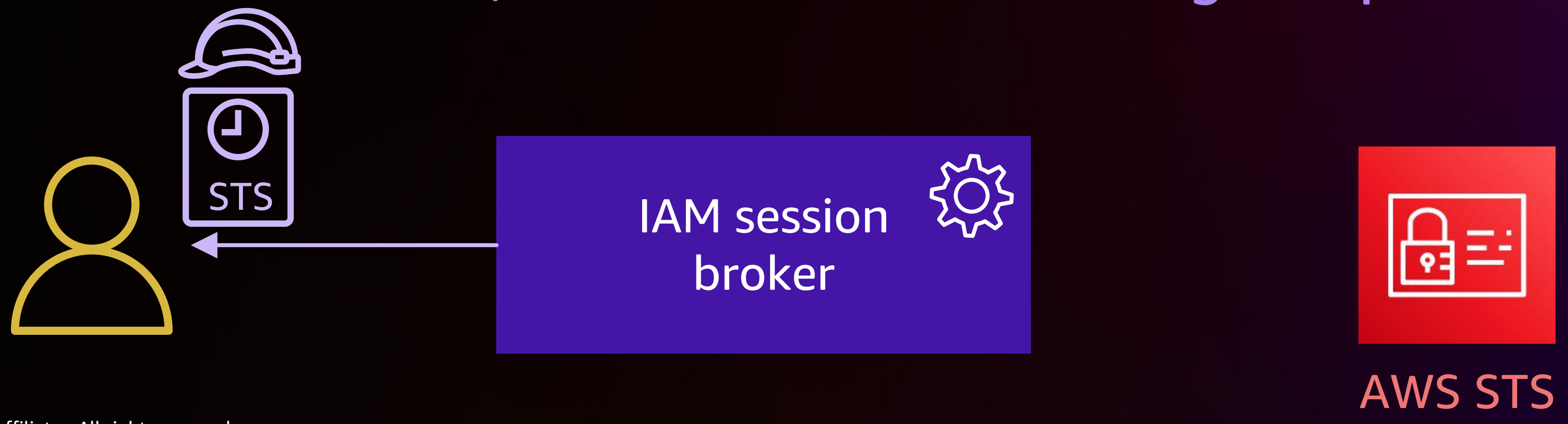
How to implement an IAM session broker

1. Authenticate caller
2. Make access decision: Should the caller be able to access the S3 location requested (e.g., `s3://example-bucket/folder/`)?
3. If yes, call STS AssumeRole API to create a session from a central IAM role with a superset access policy. Specify a session policy that grants access to the requested location.
4. Return STS credentials to caller, who will use them to sign requests to S3



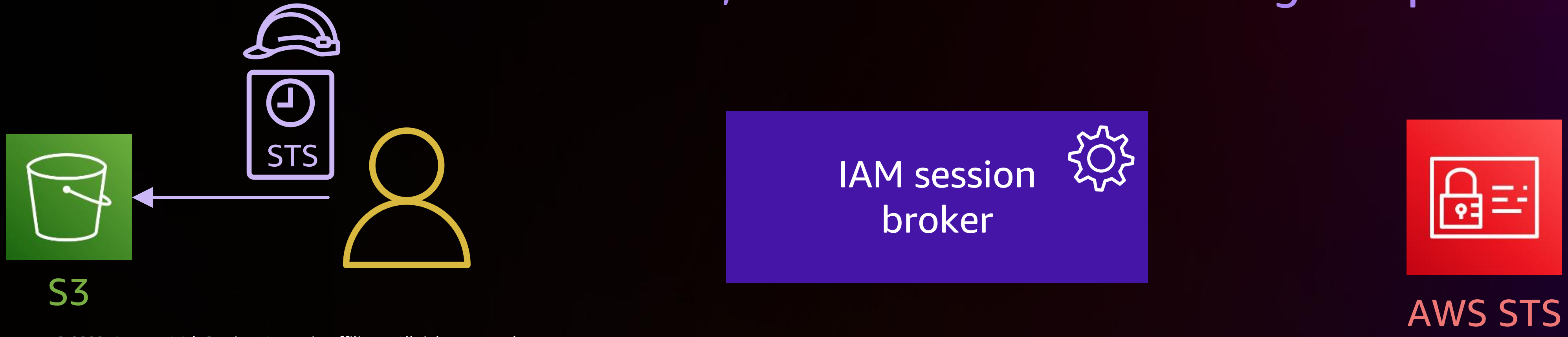
How to implement an IAM session broker

1. Authenticate caller
2. Make access decision: Should the caller be able to access the S3 location requested (e.g., `s3://example-bucket/folder/`)?
3. If yes, call STS AssumeRole API to create a session from a central IAM role with a superset access policy. Specify a session policy that grants access to the requested location.
4. Return STS credentials to caller, who will use them to sign requests to S3

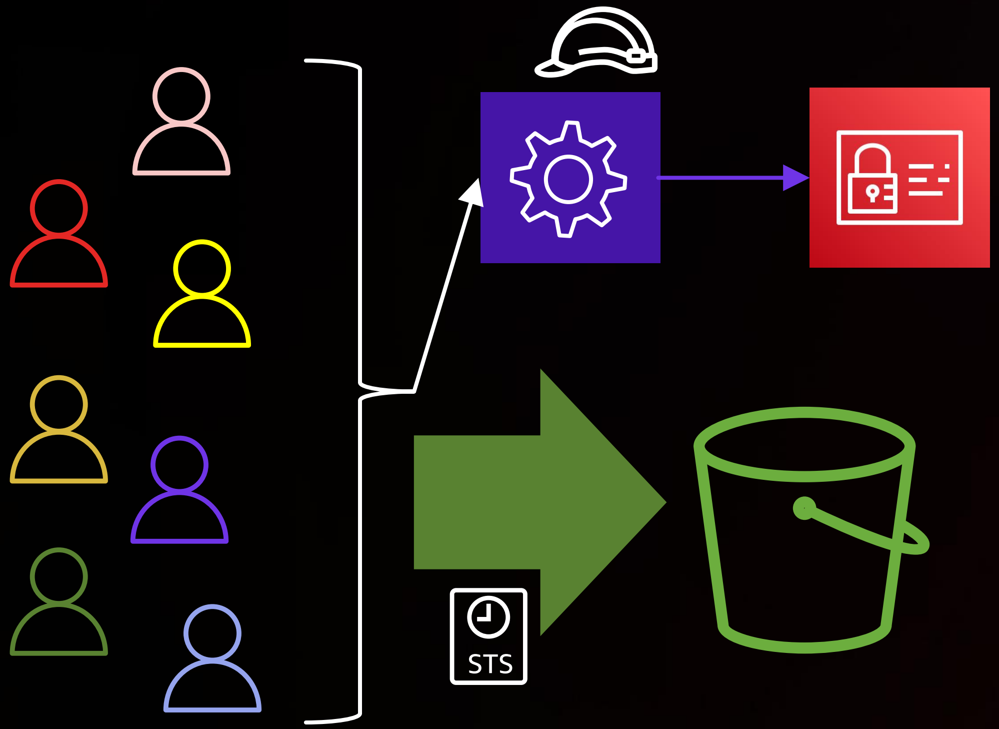


How to implement an IAM session broker

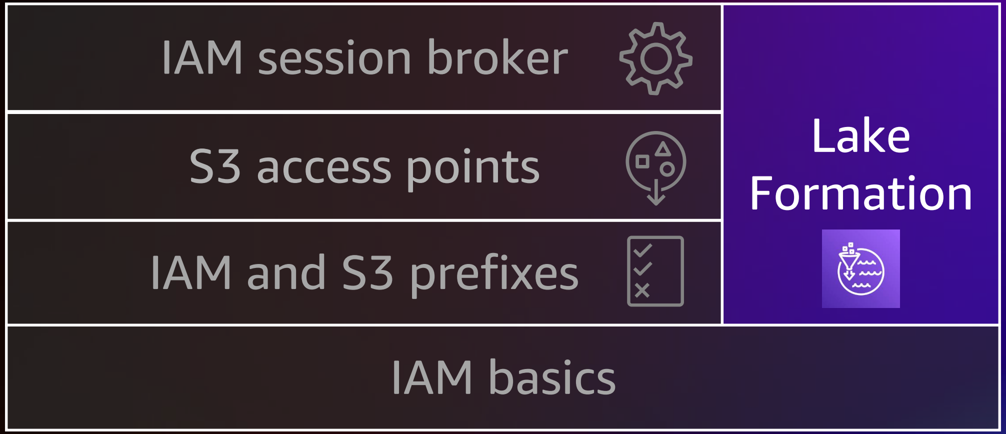
1. Authenticate caller
2. Make access decision: Should the caller be able to access the S3 location requested (e.g., `s3://example-bucket/folder/`)?
3. If yes, call STS AssumeRole API to create a session from a central IAM role with a superset access policy. Specify a session policy that grants access to the requested location.
4. Return STS credentials to caller, who will use them to sign requests to S3



Why the session broker pattern scales well



- Credential-vending is built on AWS STS, a high-volume, regional data plane
- Uses S3 for what it does best: Scale and data throughput
- Number of static IAM resources (IAM roles, policy) is minimal
- Session broker component scales with sessions, not data or even object requests


IAM session broker
S3 access points
IAM and S3 prefixes
Lake Formation
IAM basics

In S3, an object is an object

```
+ example-bucket/
  - red/
  + yellow/
     part-00000-abcde.snappy.parquet
     part-00001-abcde.snappy.parquet
     part-00002-abcde.snappy.parquet
  - blue/
```

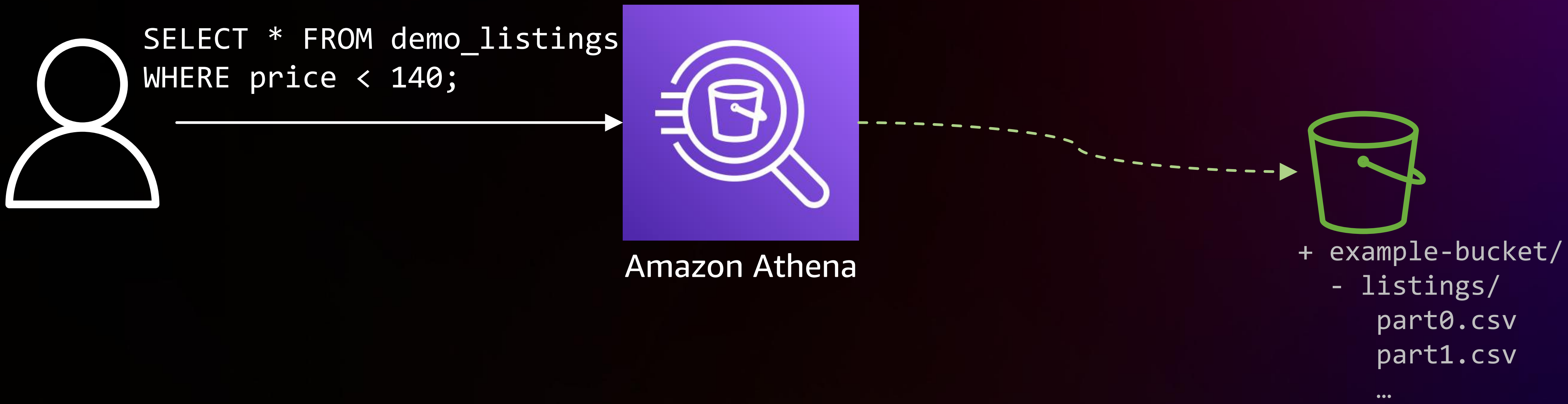
Structured data in S3

```
$ aws s3 cp s3://example-bucket/listings/part3.csv –
id,seller_name,latitude,longitude,price

3000,Alice,40.32,-73.59,150
3001,Bob,54.22,-75.18,110
3002,Carol,37.28,-78.20,115
...
```

id	seller_name	latitude	longitude	price
3000	Alice	40.32	-73.59	150
3001	Bob	54.22	-75.18	110
3002	Carol	37.28	-78.20	115

Structured data in S3

id	seller_name	latitude	longitude	price
3000	Alice	40.32	-73.59	150
3001	Bob	54.22	-75.18	110
3002	Carol	37.28	-78.20	115

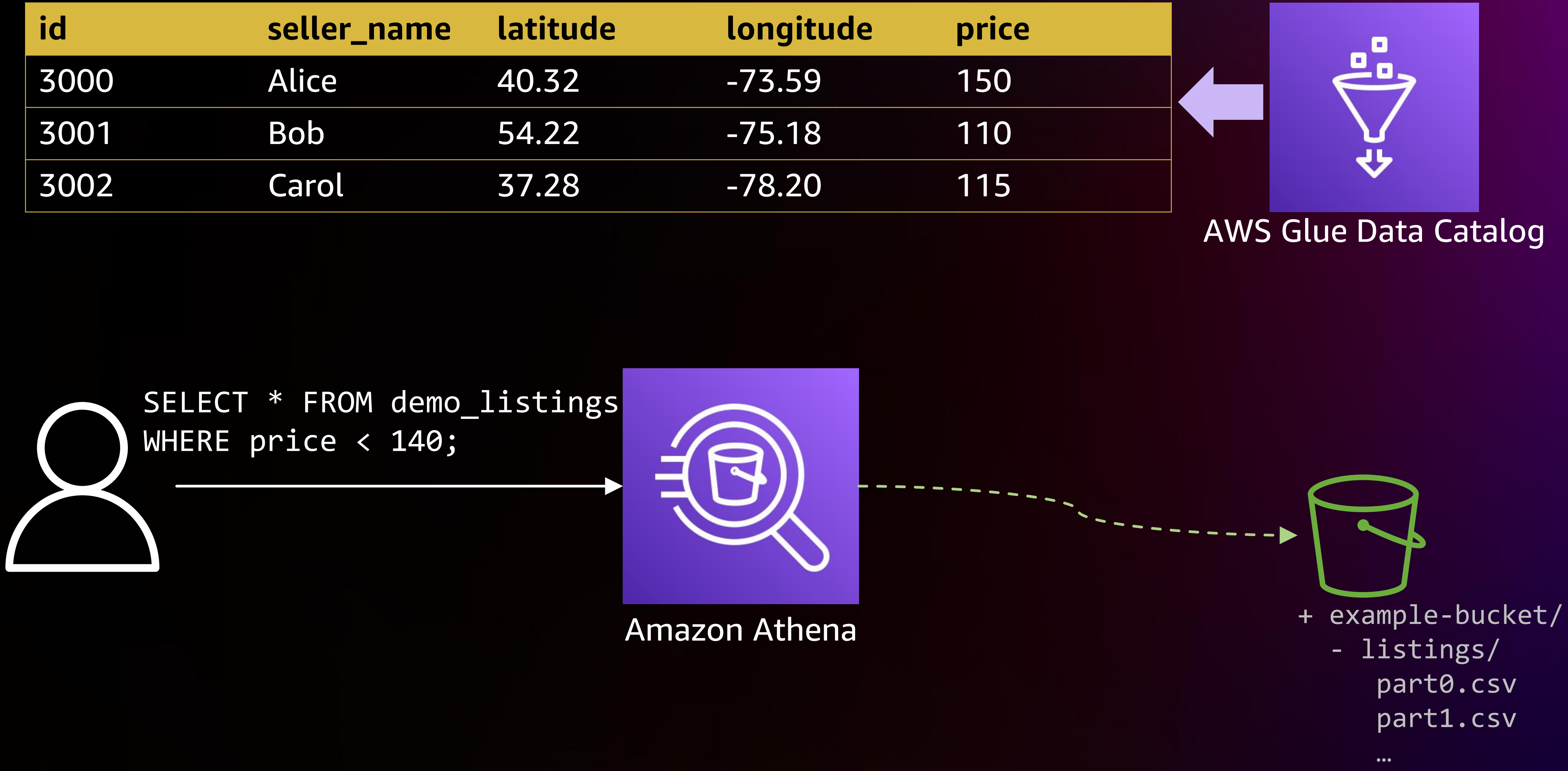


```
SELECT * FROM demo_listings
WHERE price < 140;
```

```
+ example-bucket/
  - listings/
     part0.csv
     part1.csv
     …
```

Data in S3, metadata in AWS Glue Data Catalog

id	seller_name	latitude	longitude	price
3000	Alice	40.32	-73.59	150
3001	Bob	54.22	-75.18	110
3002	Carol	37.28	-78.20	115



```
SELECT * FROM demo_listings
WHERE price < 140;
```

```
+ example-bucket/
  - listings/
     part0.csv
     part1.csv
     …
```

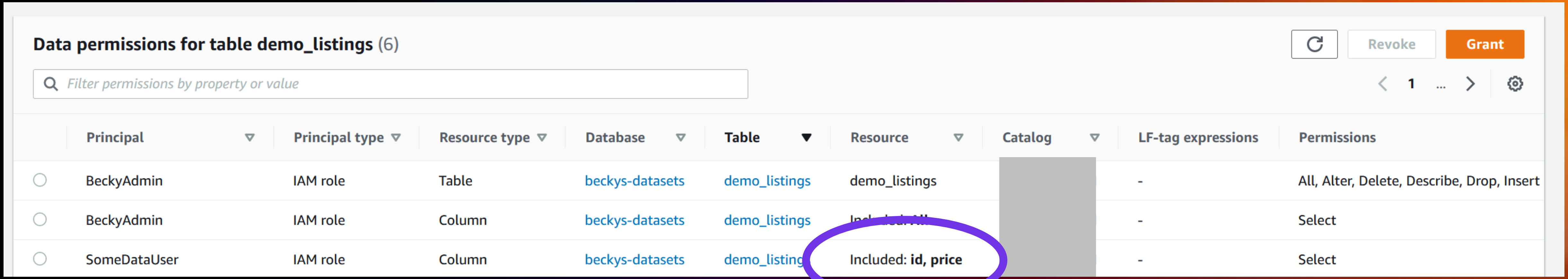
Data in S3, metadata in AWS Glue Data Catalog

id	seller_name	latitude	longitude	price
3000	Alice	40.32	-73.59	150
3001	Bob	54.22	-75.18	110
3002	Carol	37.28		

```
SELECT * FROM demo_listing
WHERE price < 140;
```

```
$ aws glue get-table --table-name demo_listings \
                     --database-name my-database
{
  "Name": "demo_listings",
  …
  "StorageDescriptor": {
    "Columns": [
      { "Name": "id", "Type": "bigint" },
      { "Name": "seller_name", "Type": "string" },
      …
    ],
    "Location": "s3://example-bucket/listings/"
    …
}
```

Permissions are often tied to schema



Data permissions for table demo_listings (6)

	Principal	Principal type	Resource type	Database	Table	Resource	Catalog	LF-tag expressions	Permissions
○	BeckyAdmin	IAM role	Table	beckys-datasets	demo_listings	demo_listings	[redacted]	-	All, Alter, Delete, Describe, Drop, Insert
○	BeckyAdmin	IAM role	Column	beckys-datasets	demo_listings	Included: All	[redacted]	-	Select
○	SomeDataUser	IAM role	Column	beckys-datasets	demo_listing	Included: **id, price**	[redacted]	-	Select

“SomeDataUser” can read only “id” and “price” columns

Permissions are often tied to schema

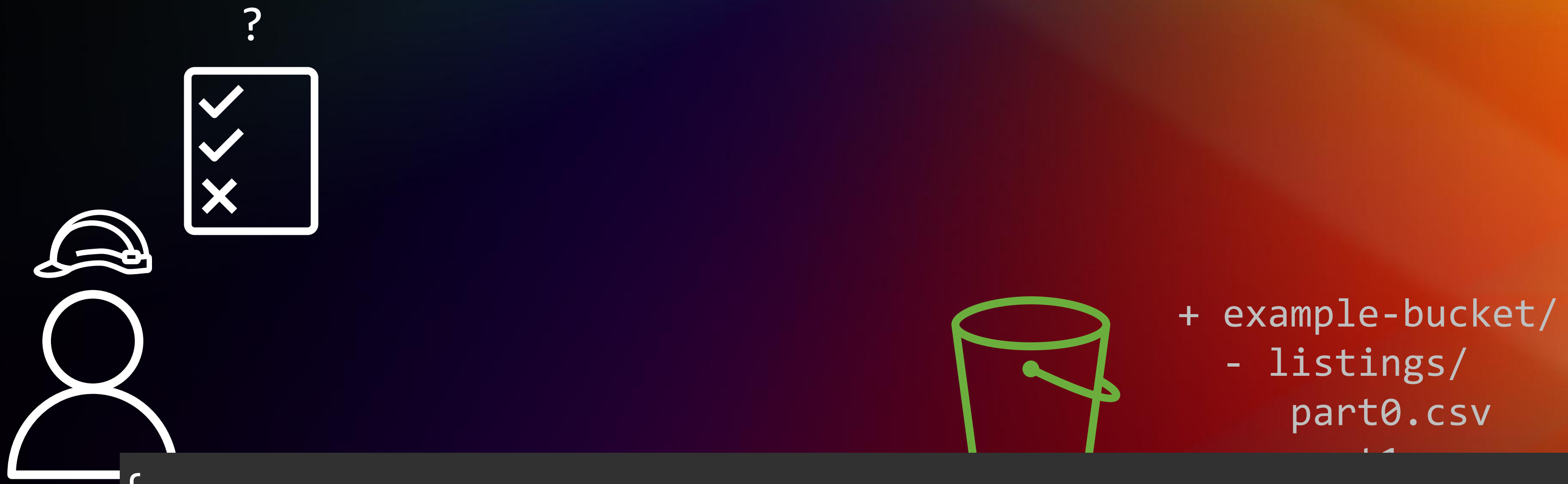
```
// Query by "Admin" IAM role
SELECT * FROM demo_listings
LIMIT 3;
```

id	seller_name	latitude	longitude	price
3000	Alice	40.32	-73.59	150
3001	Bob	54.22	-75.18	110
3002	Carol	37.28	-78.20	115

```
// Query by "SomeDataUser" IAM role
SELECT * FROM demo_listings
LIMIT 3;
```

id	price
3000	150
3001	110
3002	115

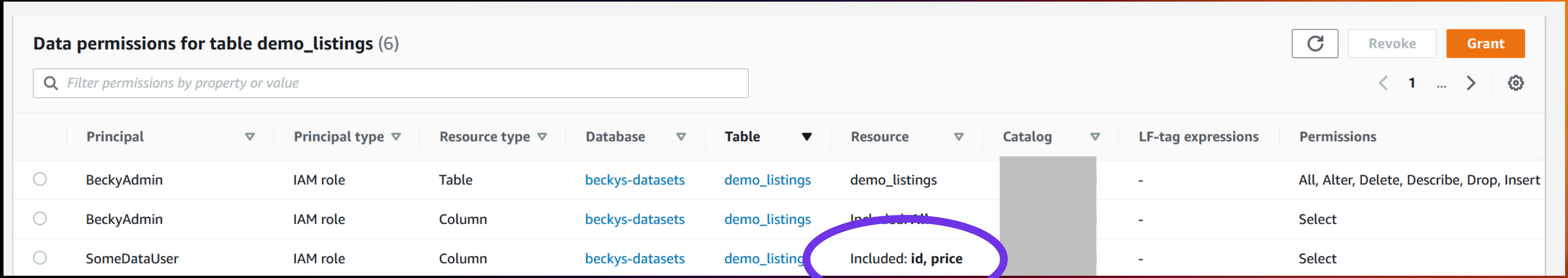
Schema-based permission requires filtering



IAM and S3 cannot filter the output of an S3 object

```
{
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::example-bucket/listings/*",
  "Condition": {
    ????
```

Schema-based permission requires filtering



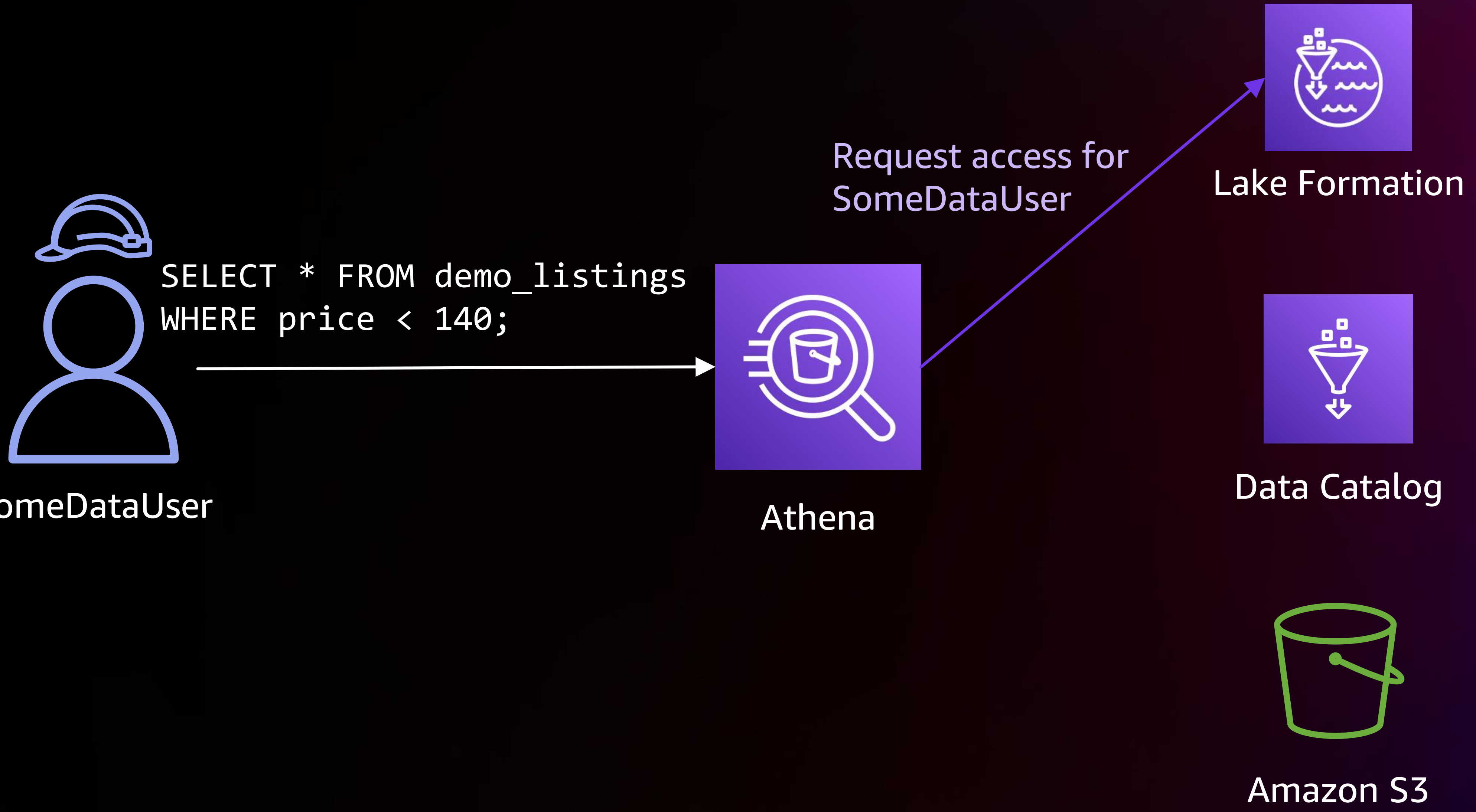
Data permissions for table demo_listings (6)

Filter permissions by property or value

	Principal	Principal type	Resource type	Database	Table	Resource	Catalog	LF-tag expressions	Permissions
○	BeckyAdmin	IAM role	Table	beckys-datasets	demo_listings	demo_listings	[redacted]	-	All, Alter, Delete, Describe, Drop, Insert
○	BeckyAdmin	IAM role	Column	beckys-datasets	demo_listings	Included: All	[redacted]	-	Select
○	SomeDataUser	IAM role	Column	beckys-datasets	demo_listings	Included: **id, price**	[redacted]	-	Select

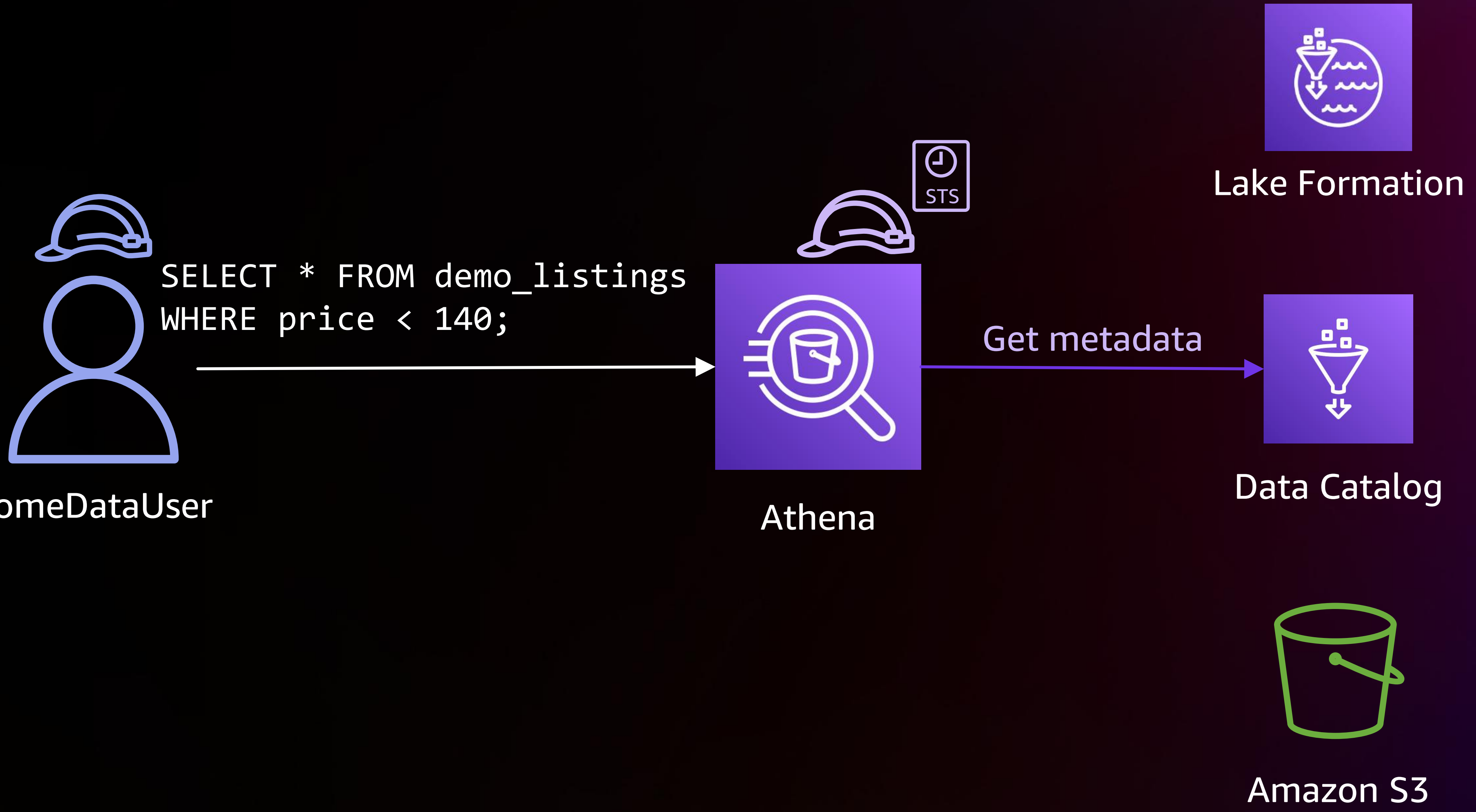
"SomeDataUser" can read only "id" and "price" columns

Lake Formation: How and why it works



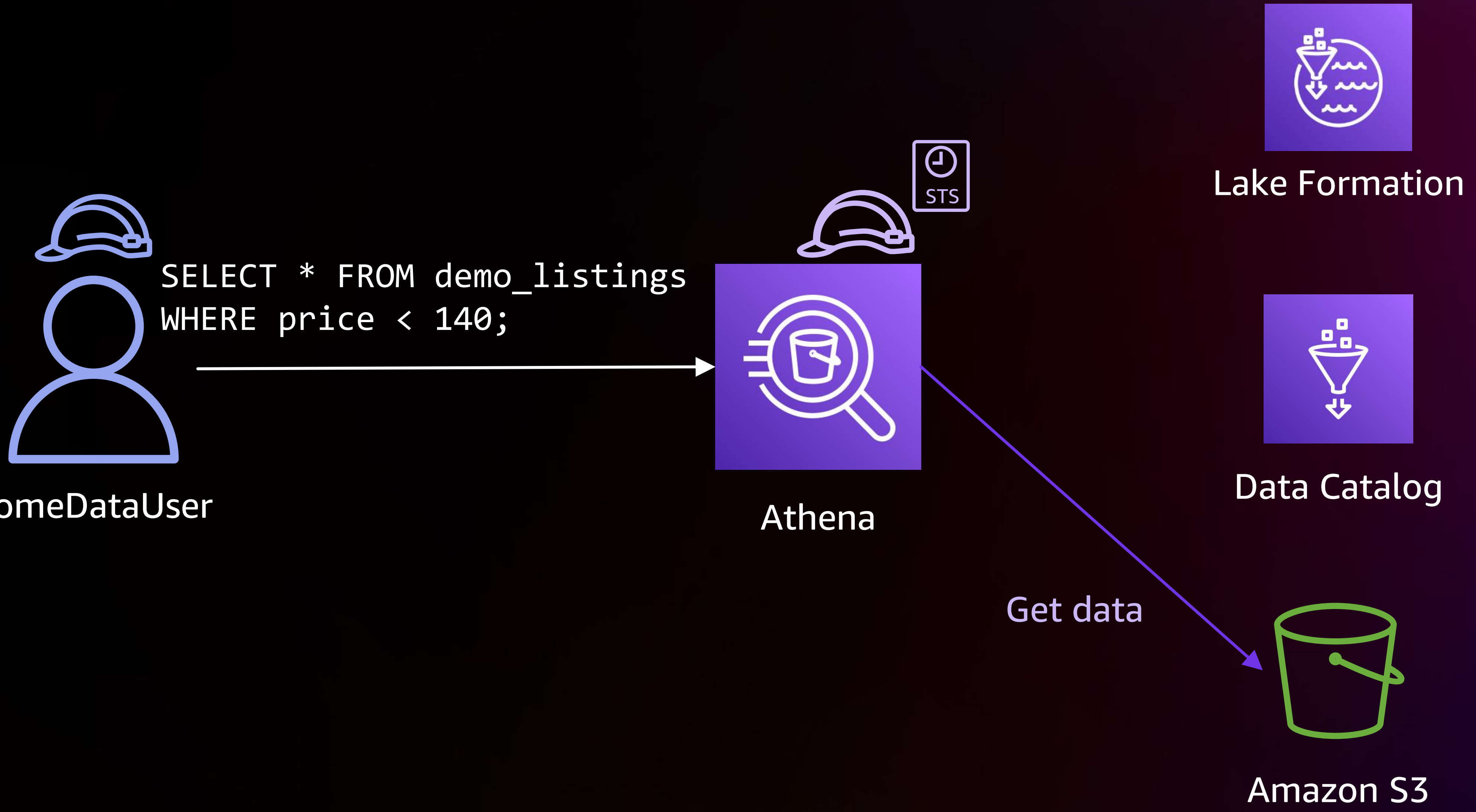
```
SELECT * FROM demo_listings
WHERE price < 140;
```

Lake Formation: How and why it works



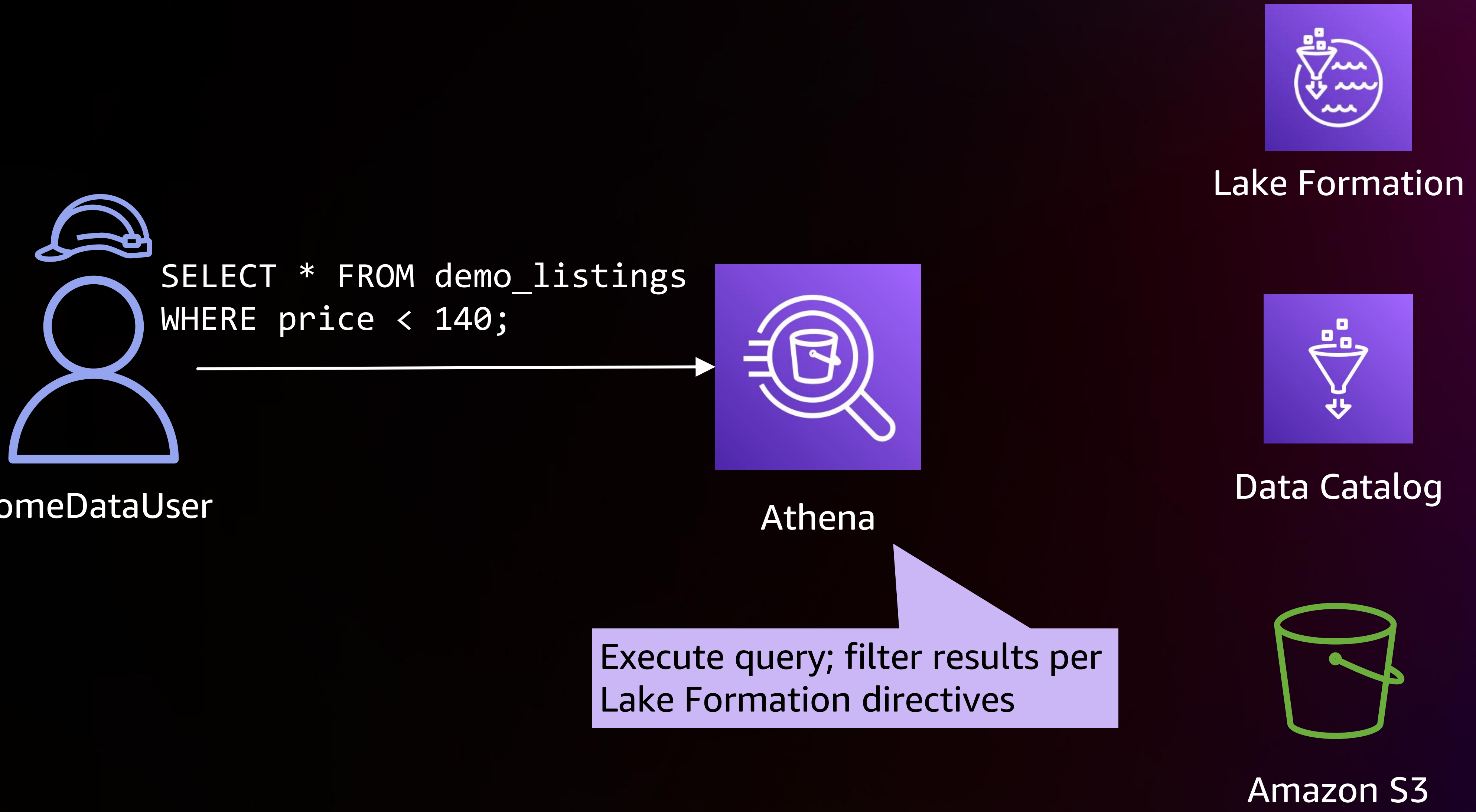
```
SELECT * FROM demo_listings
WHERE price < 140;
```

Lake Formation: How and why it works



```
SELECT * FROM demo_listings
WHERE price < 140;
```

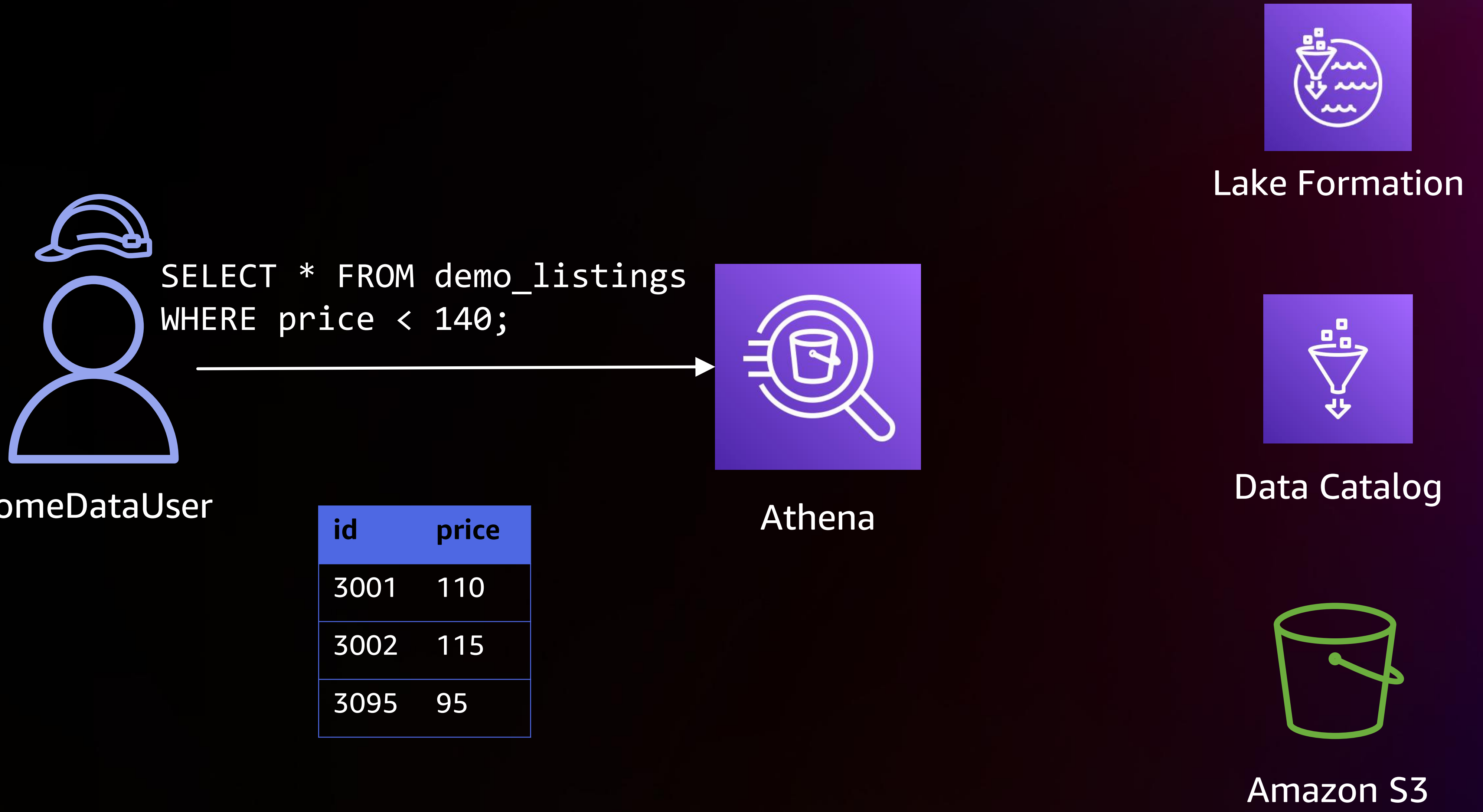
Lake Formation: How and why it works



```
SELECT * FROM demo_listings
WHERE price < 140;
```

Execute query; filter results per Lake Formation directives

Lake Formation: How and why it works

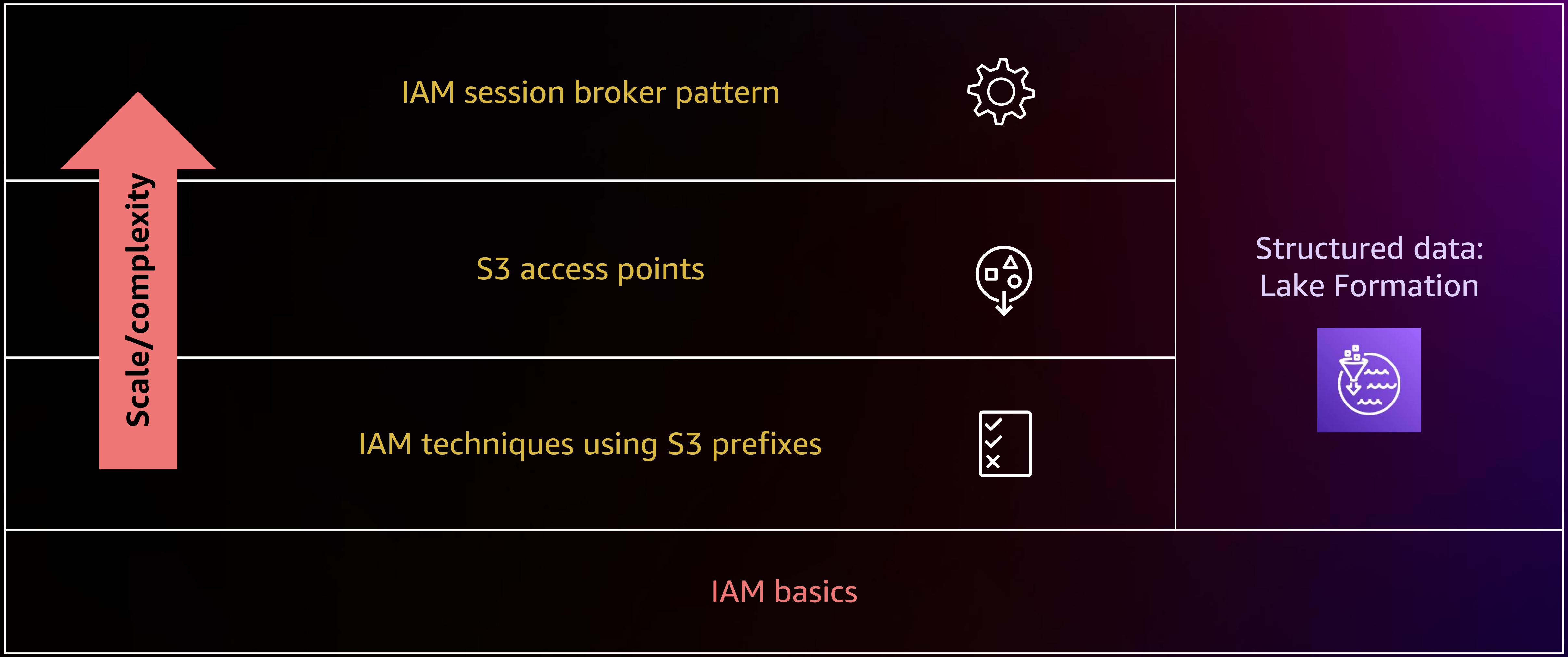


```
SELECT * FROM demo_listings
WHERE price < 140;
```

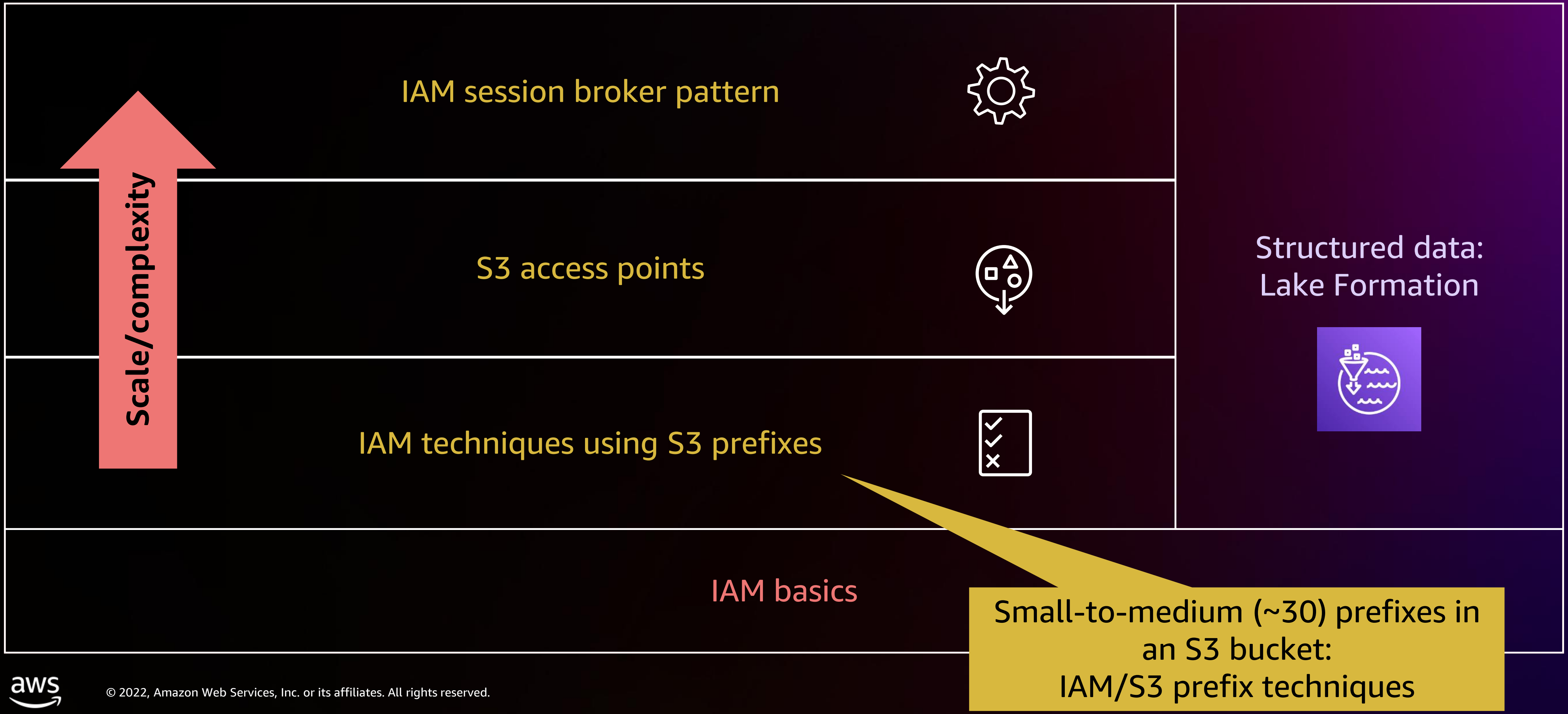
id	price
3001	110
3002	115
3095	95

How to choose?

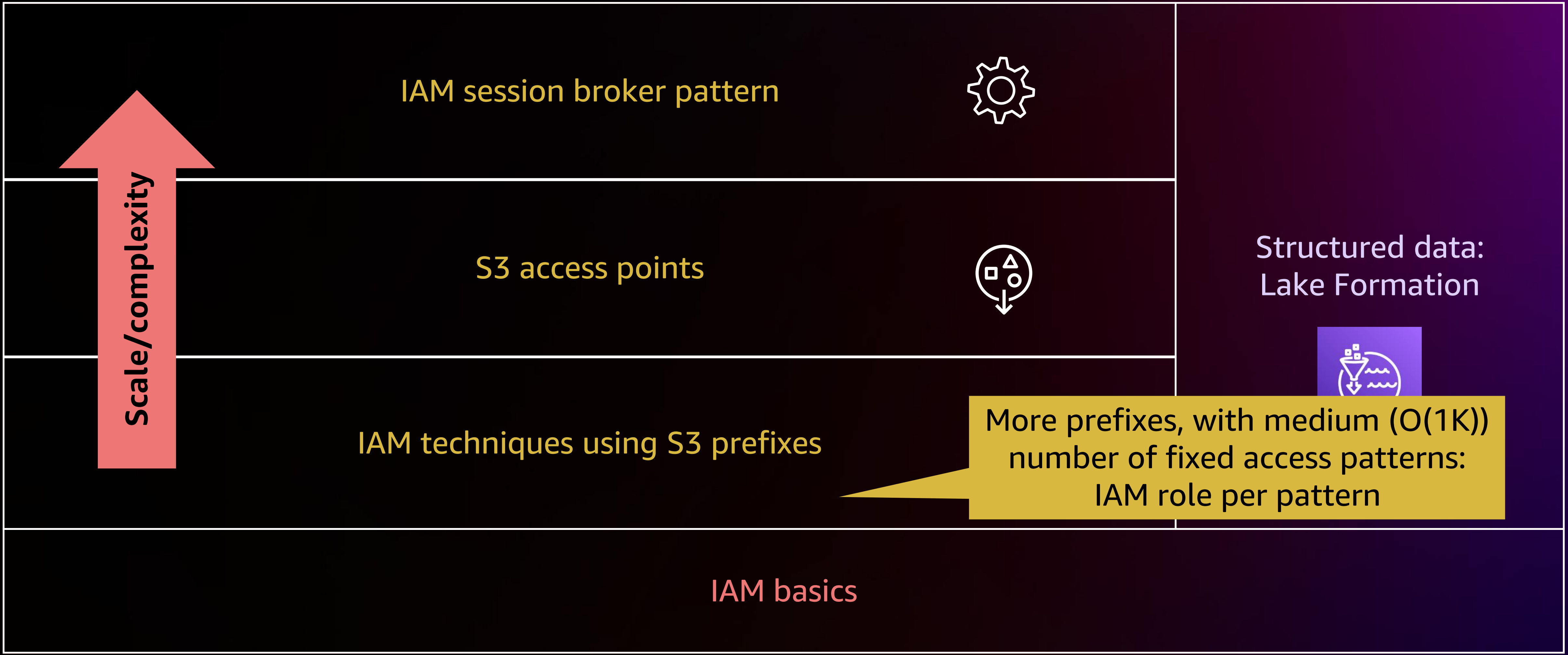
Access-control approaches discussed in this session


Scale/complexity
IAM session broker pattern
S3 access points
IAM techniques using S3 prefixes
Structured data: Lake Formation
IAM basics

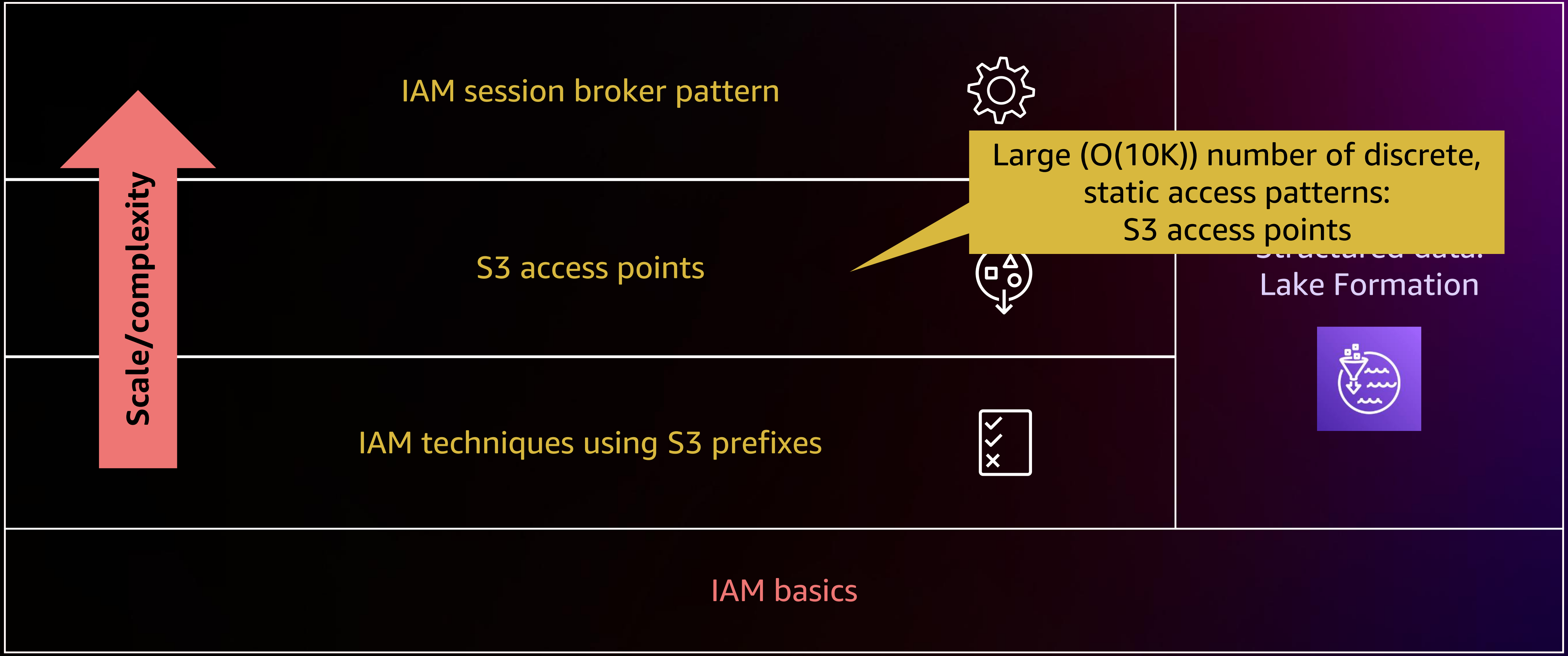
Access-control approaches discussed in this session


IAM session broker pattern
S3 access points
IAM techniques using S3 prefixes
IAM basics
Scale/complexity
Structured data: Lake Formation
Small-to-medium (~30) prefixes in an S3 bucket: IAM/S3 prefix techniques

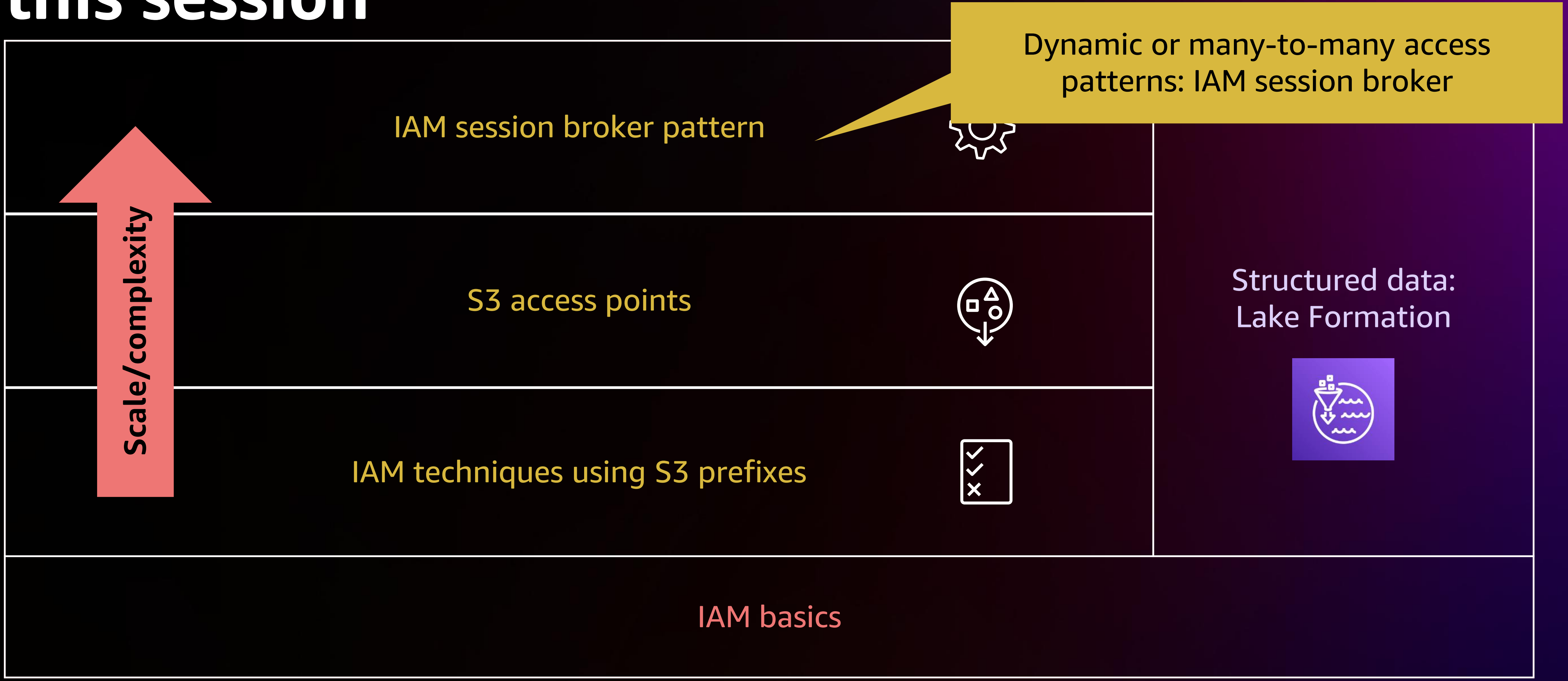
Access-control approaches discussed in this session


IAM session broker pattern
S3 access points
IAM techniques using S3 prefixes
IAM basics
Scale/complexity
Structured data: Lake Formation
More prefixes, with medium (O(1K)) number of fixed access patterns: IAM role per pattern

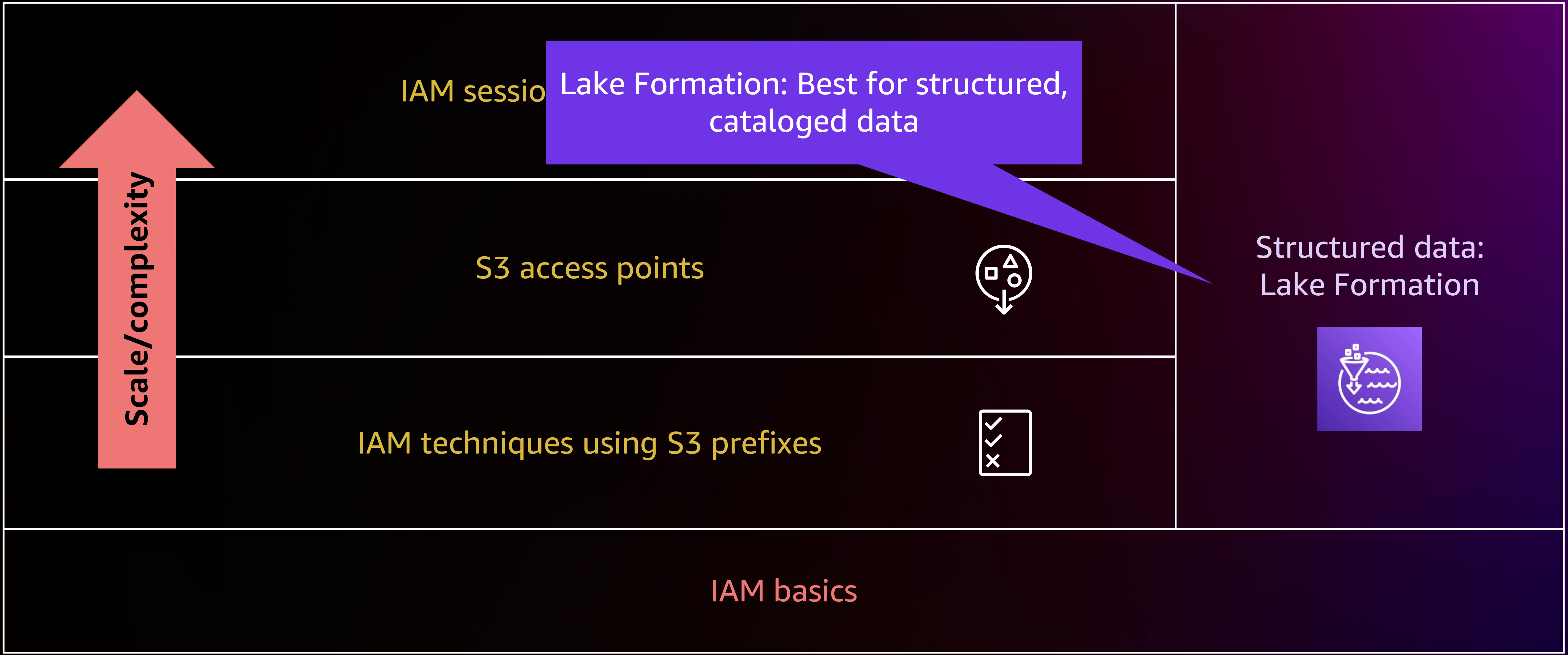
Access-control approaches discussed in this session


IAM session broker pattern
Large (O(10K)) number of discrete, static access patterns: S3 access points
S3 access points
Scale/complexity
IAM techniques using S3 prefixes
Lake Formation
IAM basics

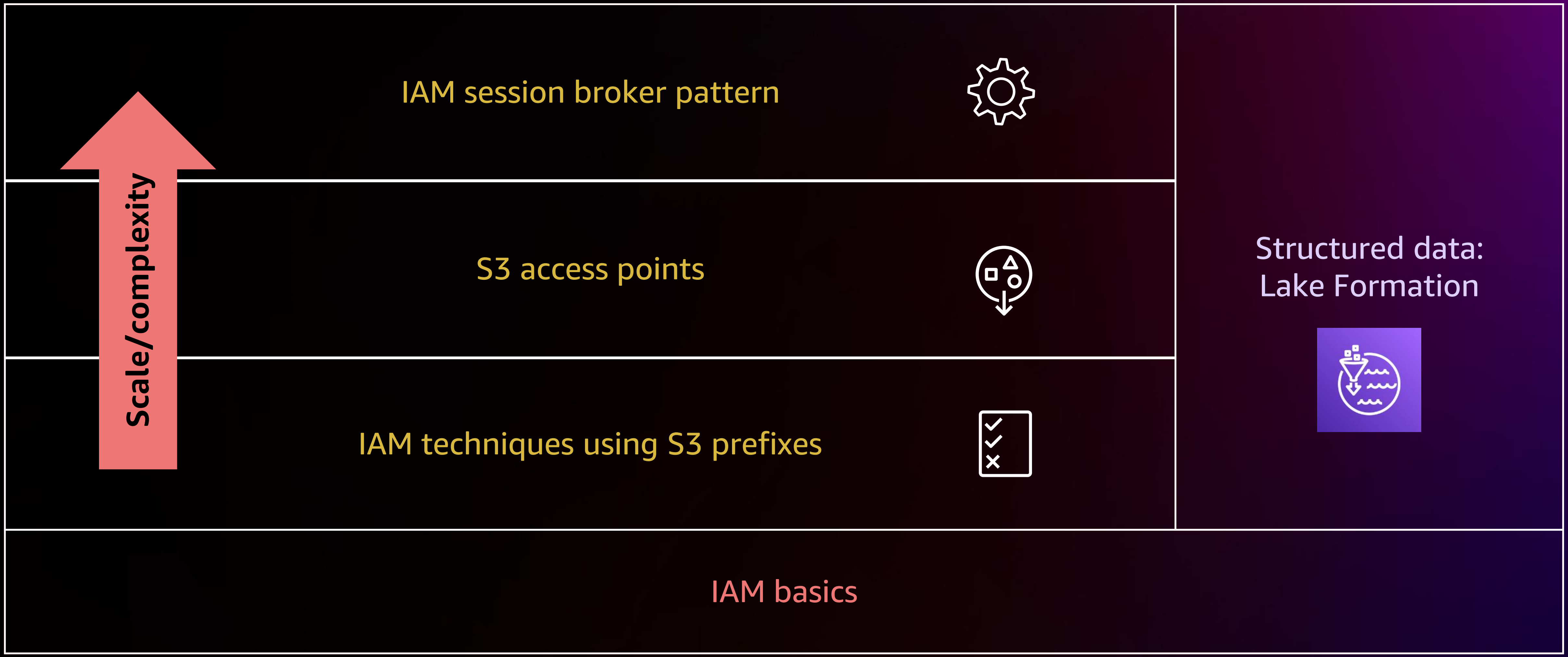
Access-control approaches discussed in this session


Dynamic or many-to-many access patterns: IAM session broker
IAM session broker pattern
S3 access points
IAM techniques using S3 prefixes
Scale/complexity
Structured data: Lake Formation
IAM basics

Access-control approaches discussed in this session


IAM sessio
Lake Formation: Best for structured, cataloged data
S3 access points
IAM techniques using S3 prefixes
Scale/complexity
Structured data: Lake Formation
IAM basics

Access-control approaches discussed in this session


Scale/complexity
IAM session broker pattern
S3 access points
IAM techniques using S3 prefixes
Structured data: Lake Formation
IAM basics

Continue your AWS Storage learning

Build a learning plan

Set your AWS Storage Learning Plans via **AWS Skill Builder**

Increase your knowledge

Use our **Ramp-Up Guides** to build your storage knowledge

Earn AWS Storage badges

Demonstrate your knowledge by achieving **digital badges**

aws.training/storage

Thank you!

Becky Weiss
becky@amazon.com

Please complete the session survey in the **mobile app**