AWS re:Inforce

JUNE 10 - 12, 2024 | PHILADELPHIA, PA

IAM431-R

The life of an IAM policy

Dan Peebles
Principal Security Engineer
AWS

Matt Luttrell
Principal Solutions Architect
AWS

Disclaimers so they let us give this talk

- We will try, but may not be able, to answer all of your questions
- This is how things work today
- Services may have minor nuance or variation

Agenda

- Control plane and data plane
- Authentication
- Authorization
- Putting it all together

Internal architecture of IAM

Internal architecture of AWS Identity and Access Management (IAM) overview

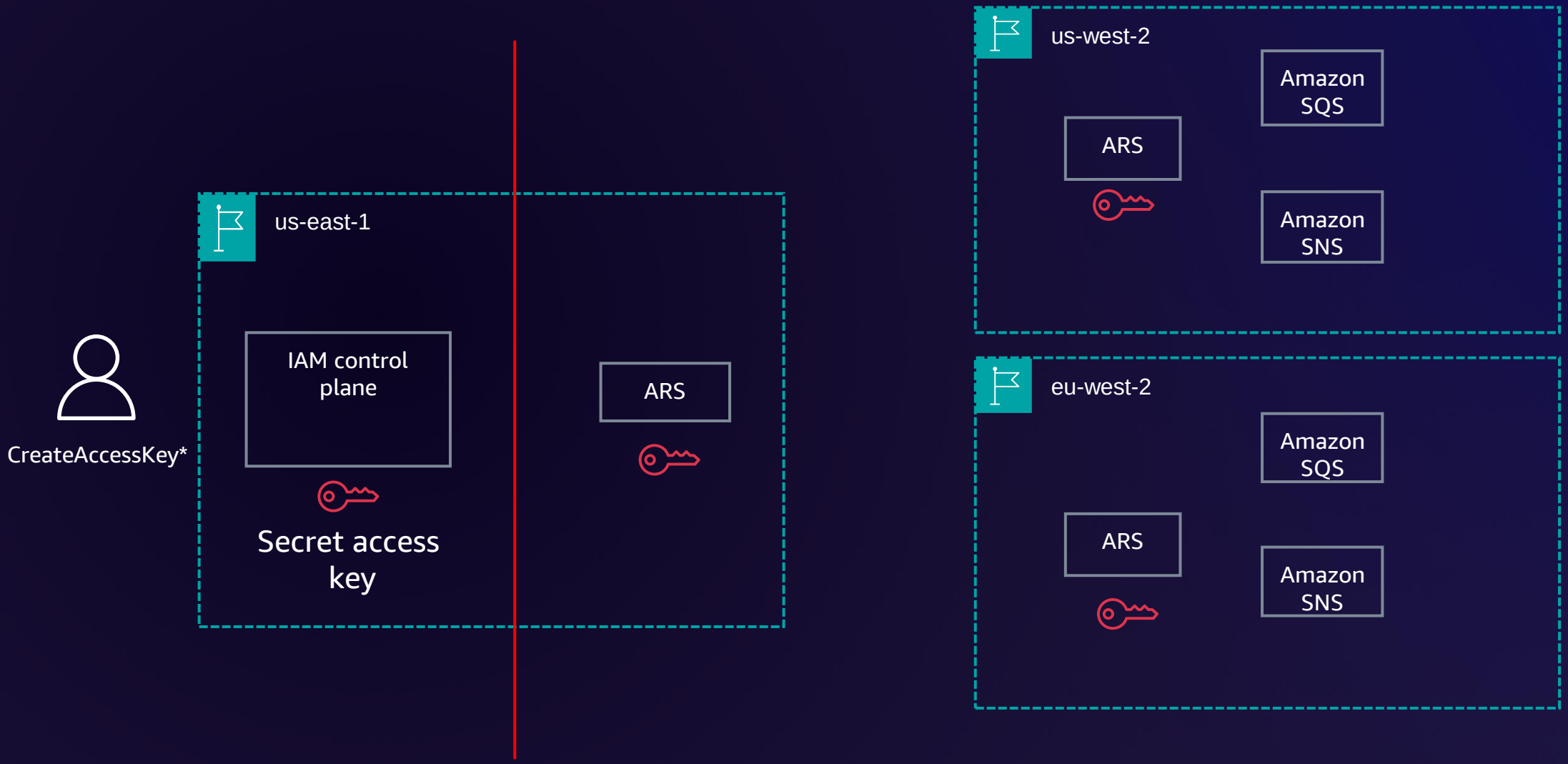
- Intentional separation of control plane and data plane
  - IAM control plane
  - Auth runtime service (ARS)

- Signature version 4's (Sigv4) hierarchical design that limits blast radius

Internal architecture of IAM

*for illustrative purposes, prefer temporary credentials instead

Internal architecture of IAM

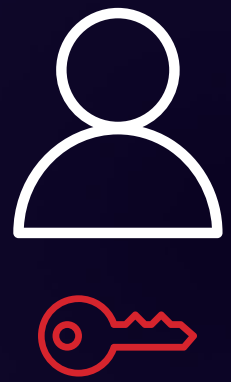


*for illustrative purposes, prefer temporary credentials instead

Internal architecture of IAM

*for illustrative purposes, prefer temporary credentials instead

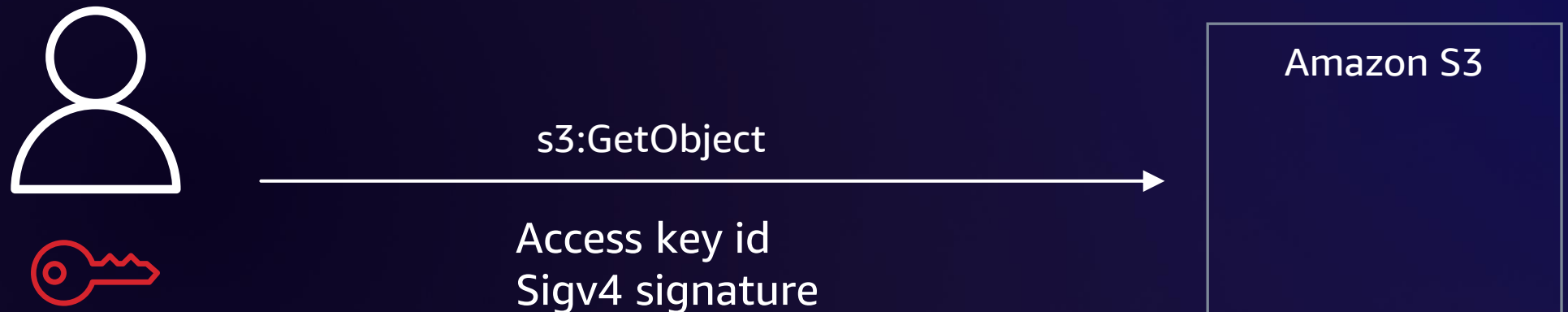
Authentication – Sigv4



Access key id
Secret access key

Amazon S3

Authentication – Sigv4



Authentication – Signature version 4 (Sigv4)

String to sign

Algorithm	AWS4-HMAC-SHA256
Request date time	20220830
Credential scope	20220830/us-east-1/ec2/aws4_request
Hashed canonical request	Hash(HTTP verb + URI + query string + headers + payload)

Authentication – Sigv4 signing key

Secret access key

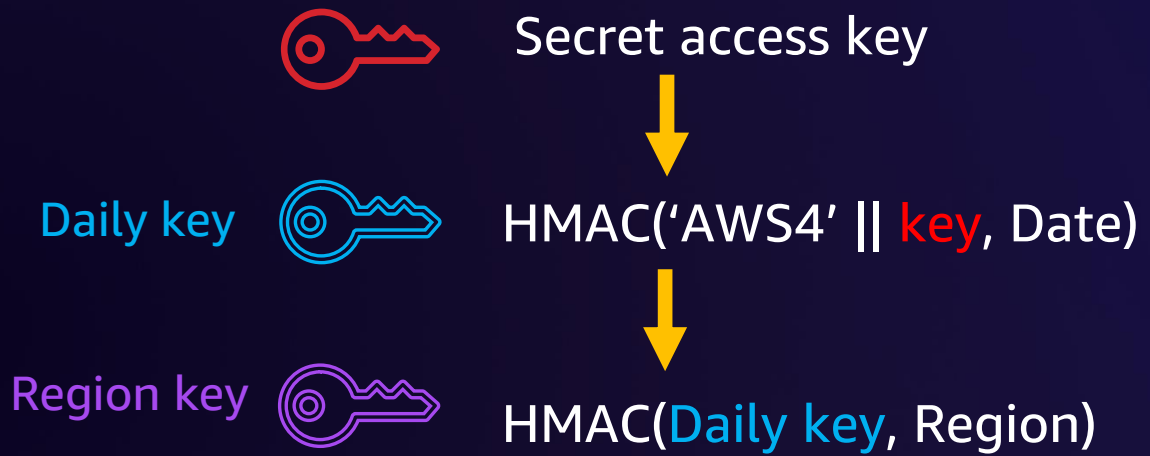
Authentication – Sigv4 signing key

Secret access key

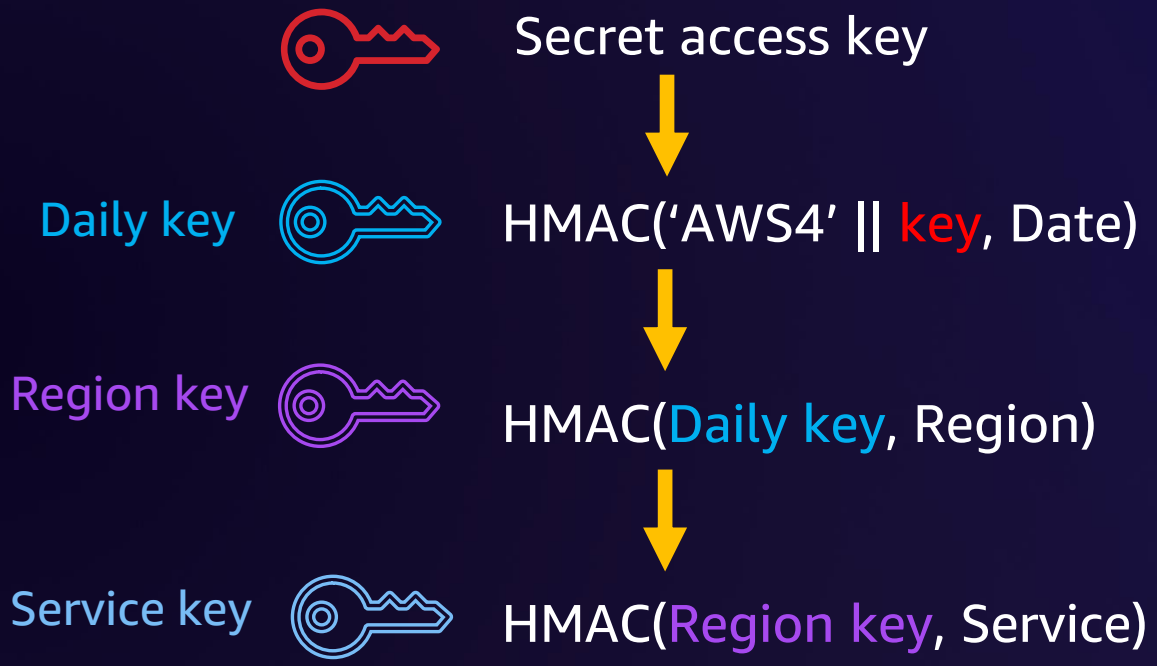
Daily key

HMAC(‘AWS4’ || key, Date)

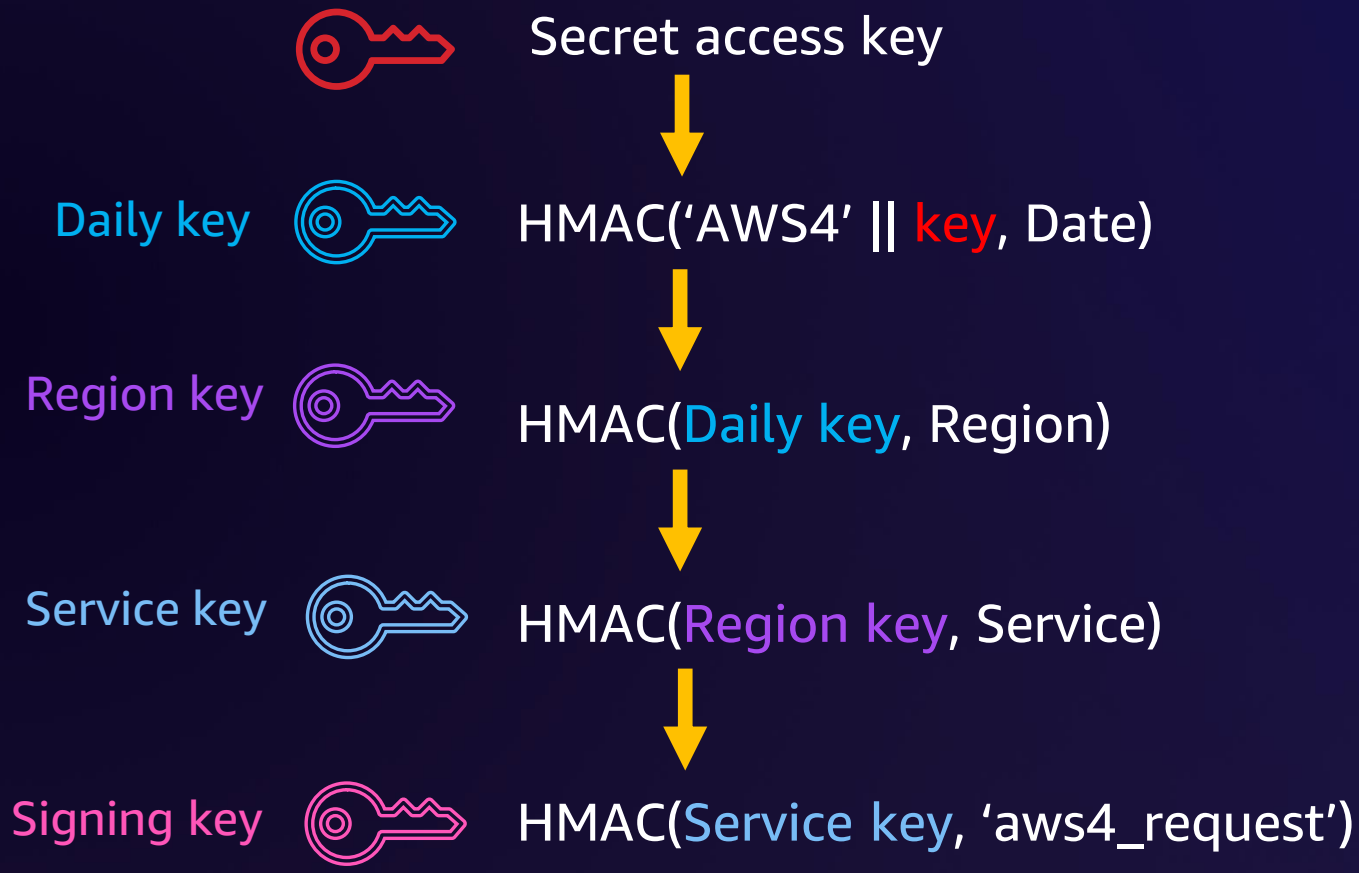
Authentication – Sigv4 signing key



Authentication – Sigv4 signing key



Authentication – Sigv4 signing key



Authentication – Signature version 4 (Sigv4)

HMAC(Signing key, String to sign) = Signature

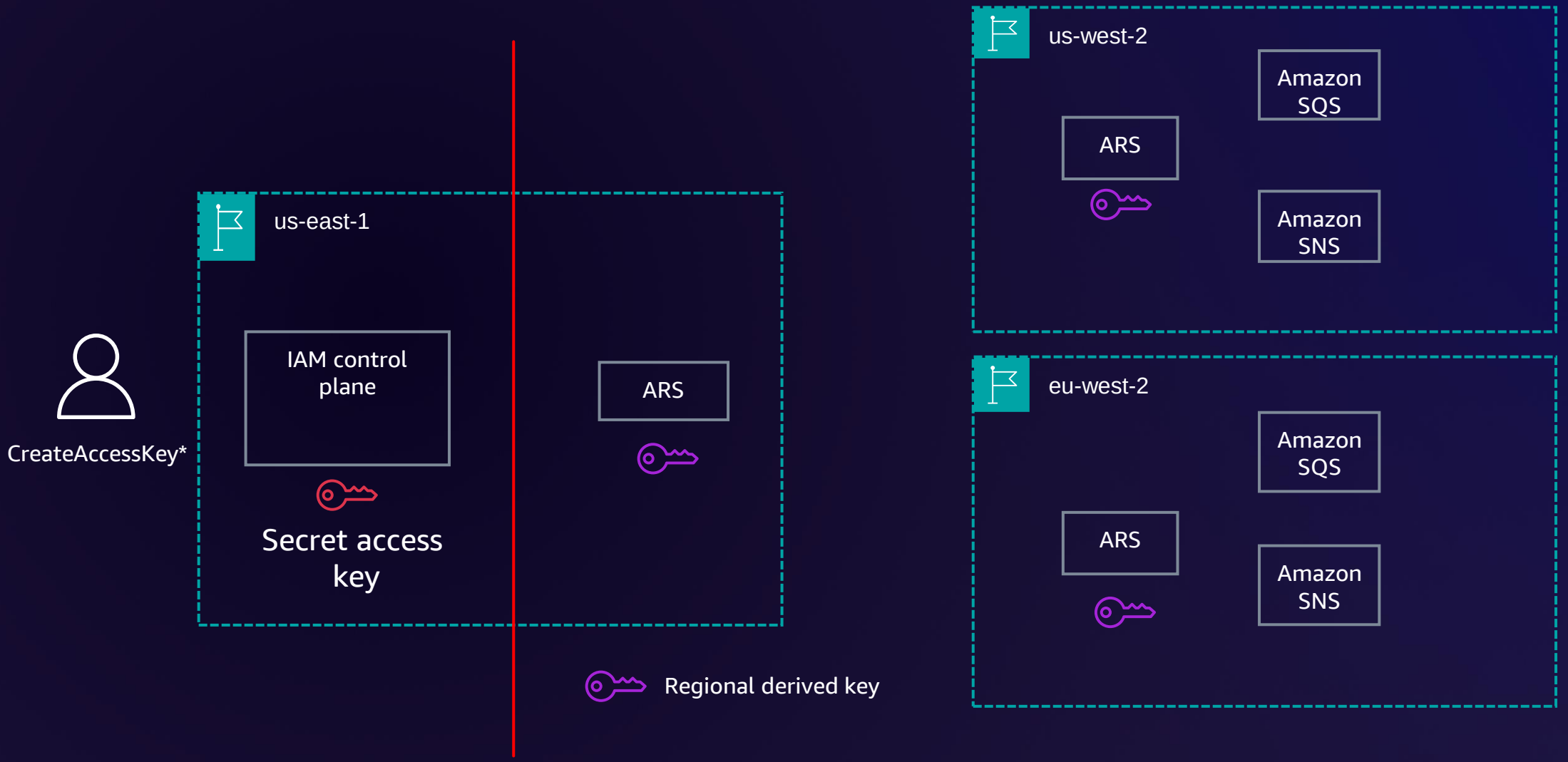
Authorization header:

AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543

Internal architecture of IAM – derived keys

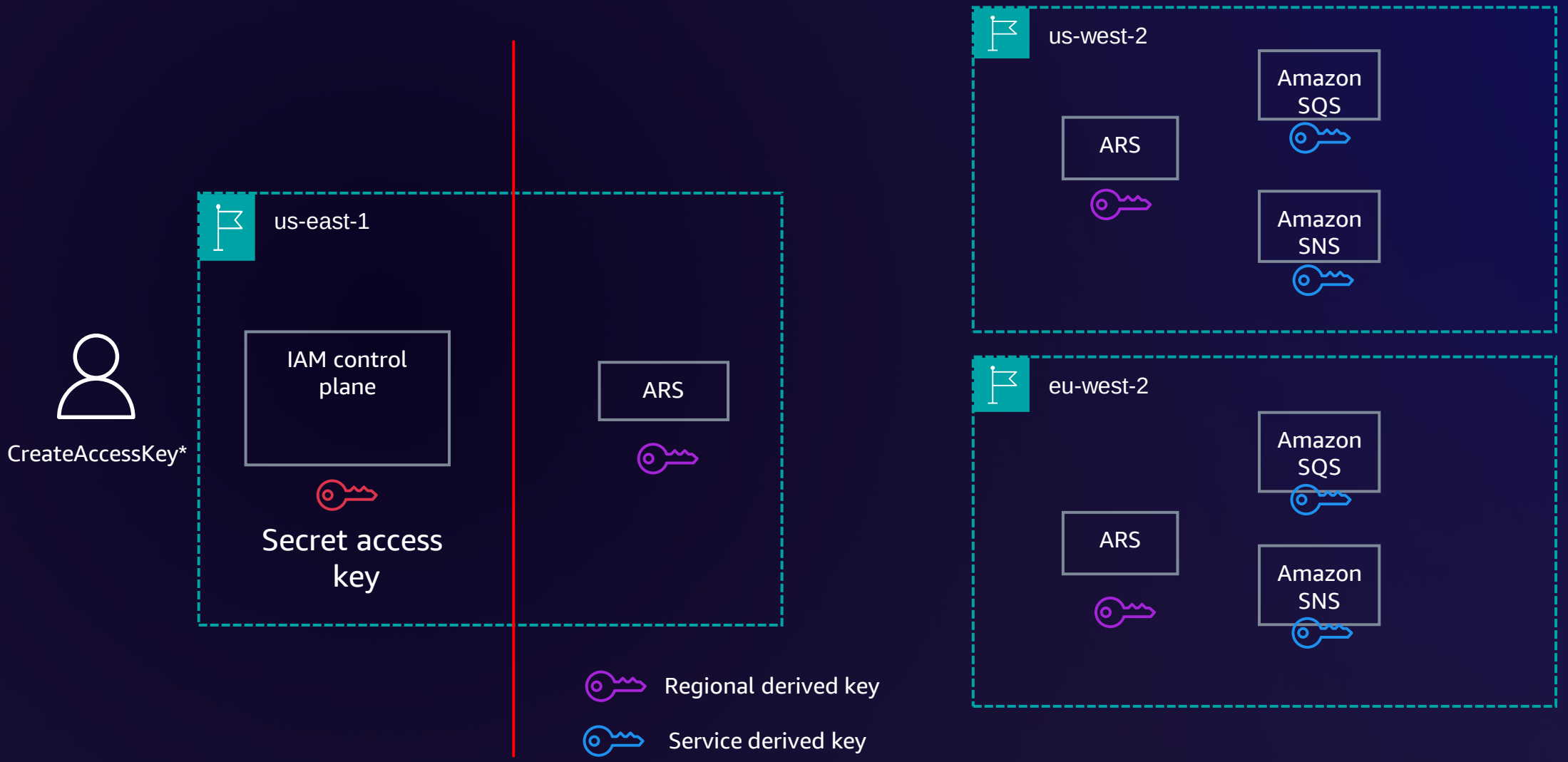
*for illustrative purposes, prefer temporary credentials instead

Internal architecture of IAM – derived keys



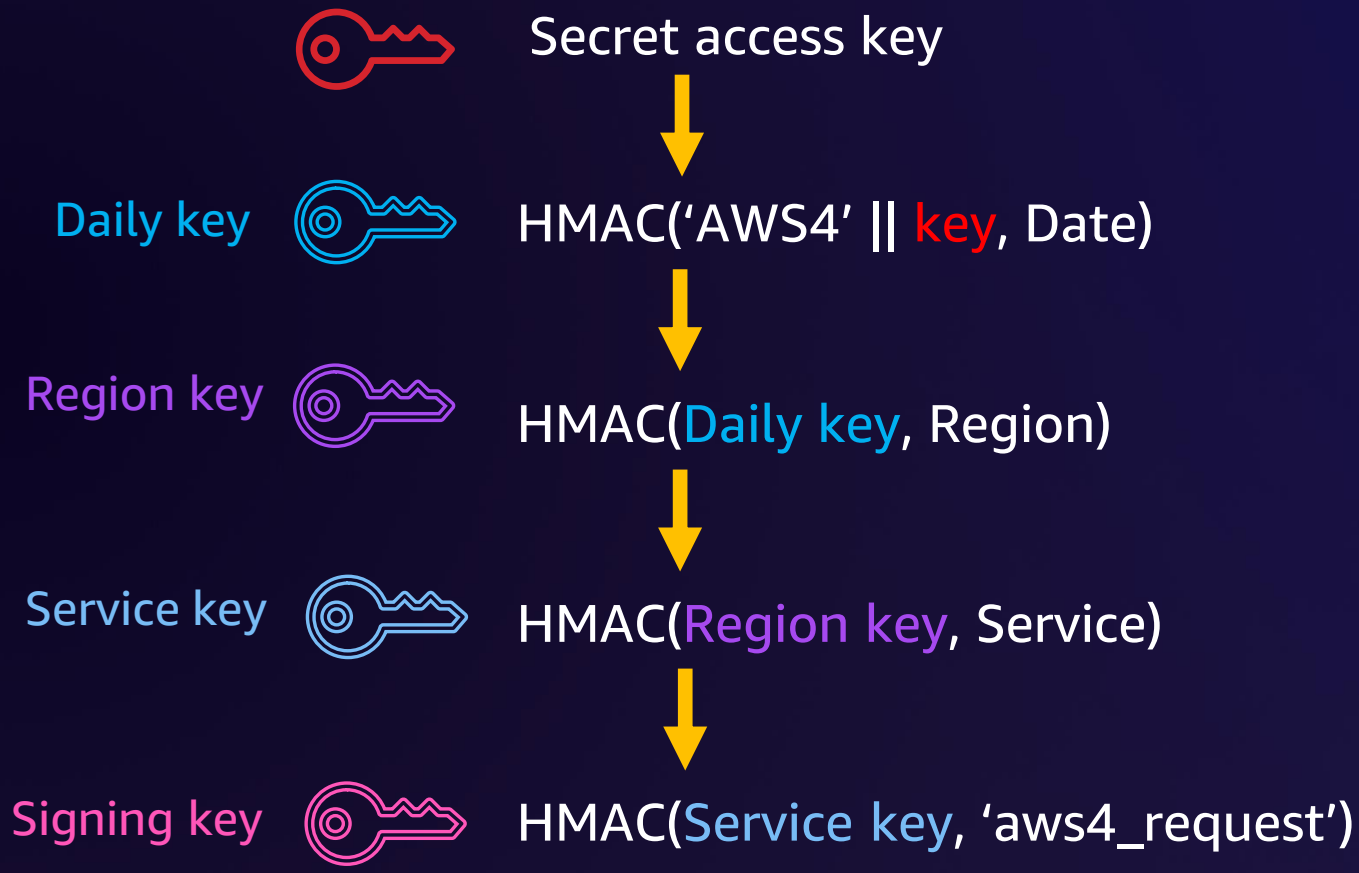
*for illustrative purposes, prefer temporary credentials instead

Internal architecture of IAM – derived keys

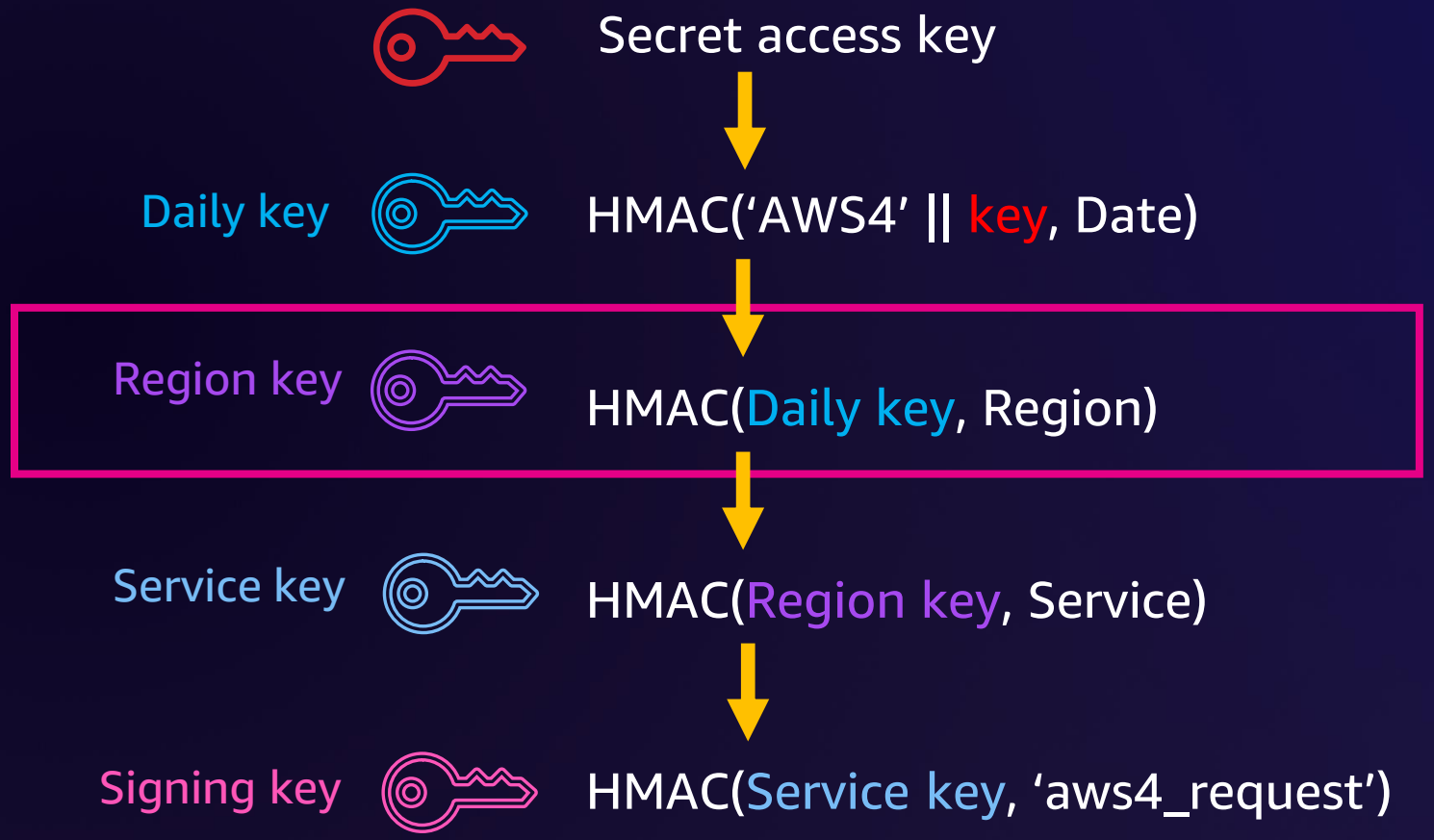


*for illustrative purposes, prefer temporary credentials instead

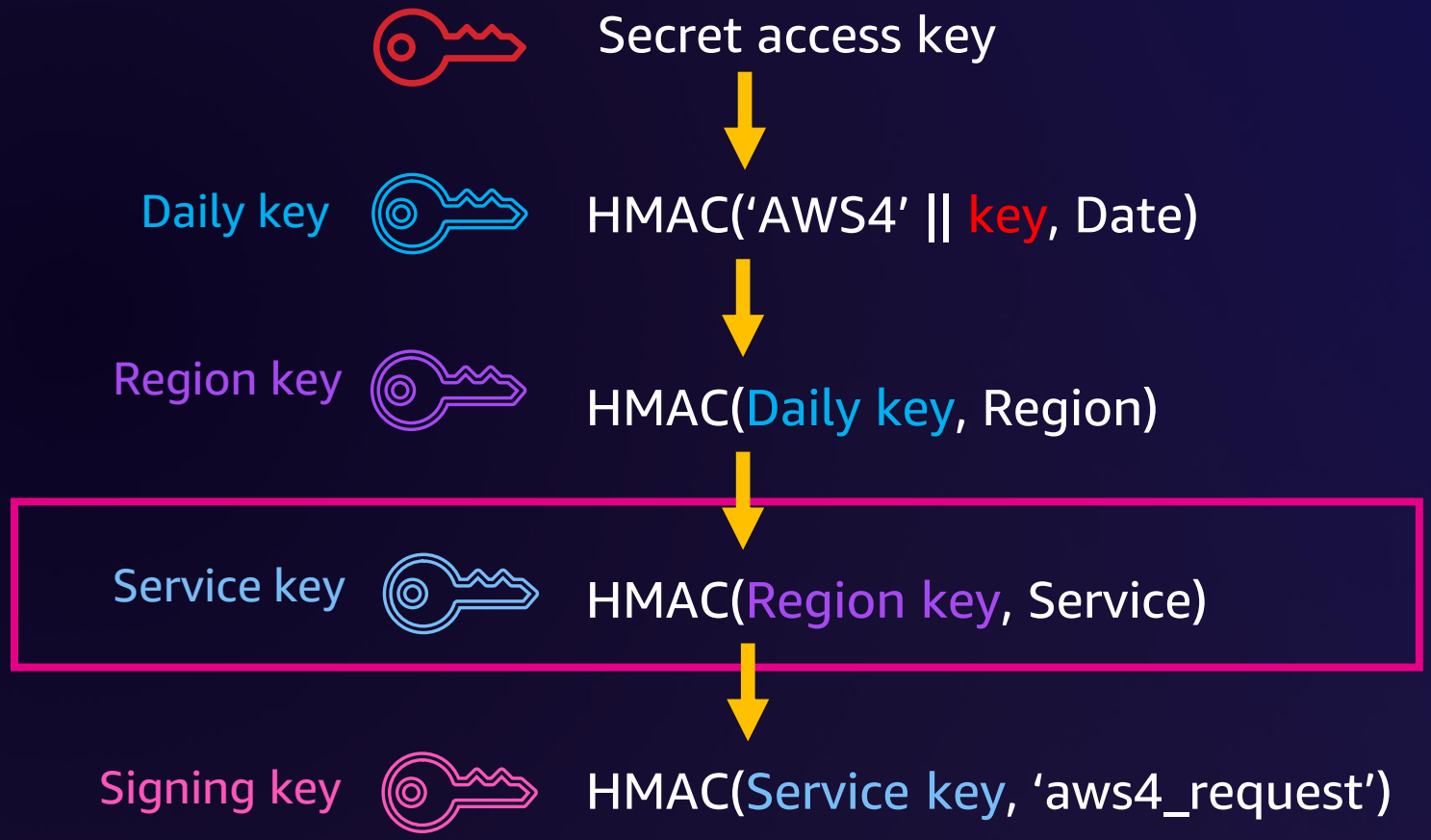
Authentication – Sigv4 signing key



Authentication – Sigv4 signing key



Authentication – Sigv4 signing key



Internal architecture of IAM takeaways

- Control plane vs. data plane
  - IAM control plane
  - Auth runtime service (ARS)

- Authentication and the magic of Sigv4

But where are the policies?

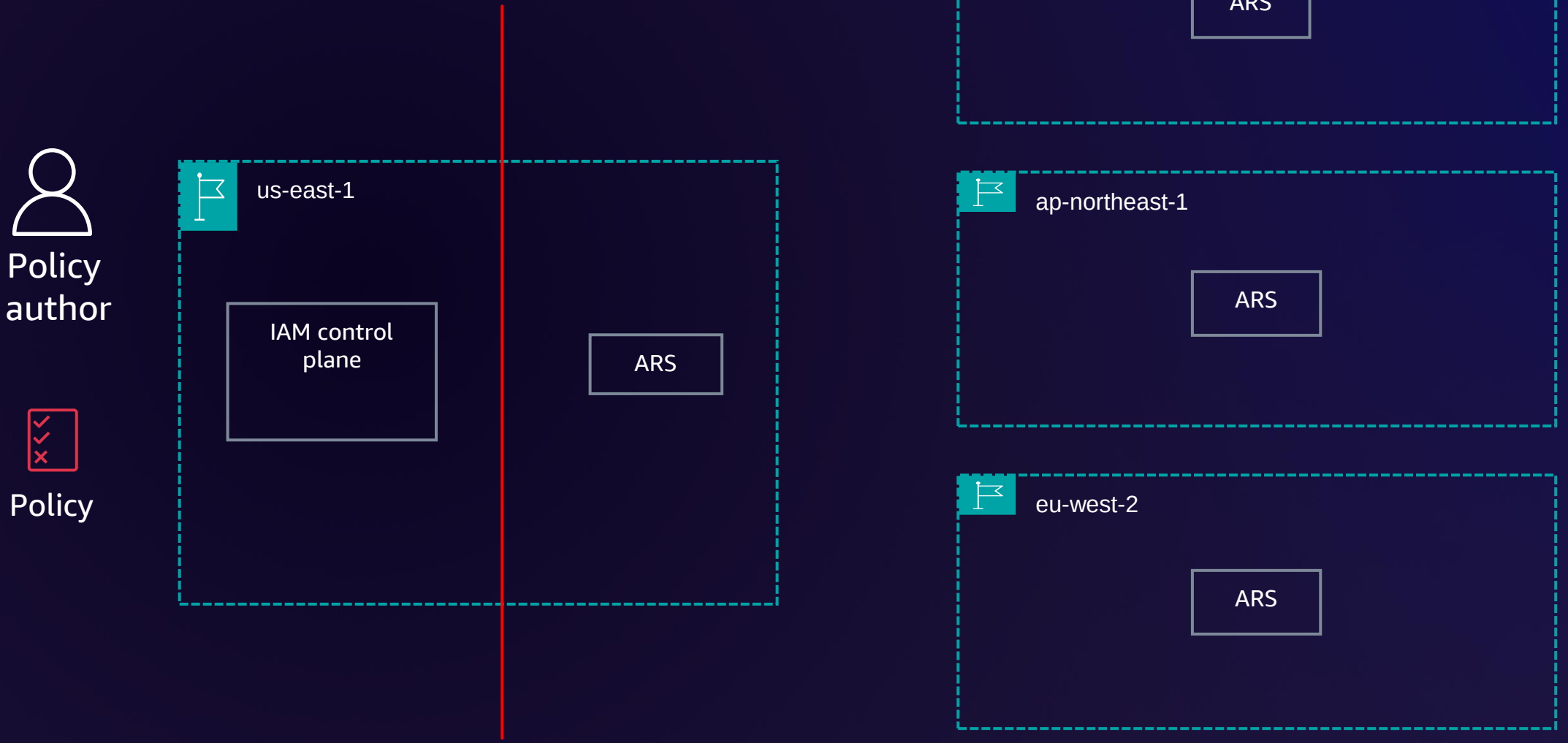
Policy type primer

- Identity-based policies
  - Managed and inline
  - Permissions boundaries
- Service control policies
- Resource-based policies
- Session policies
- VPC endpoint policies
- Forward access session (FAS) policies

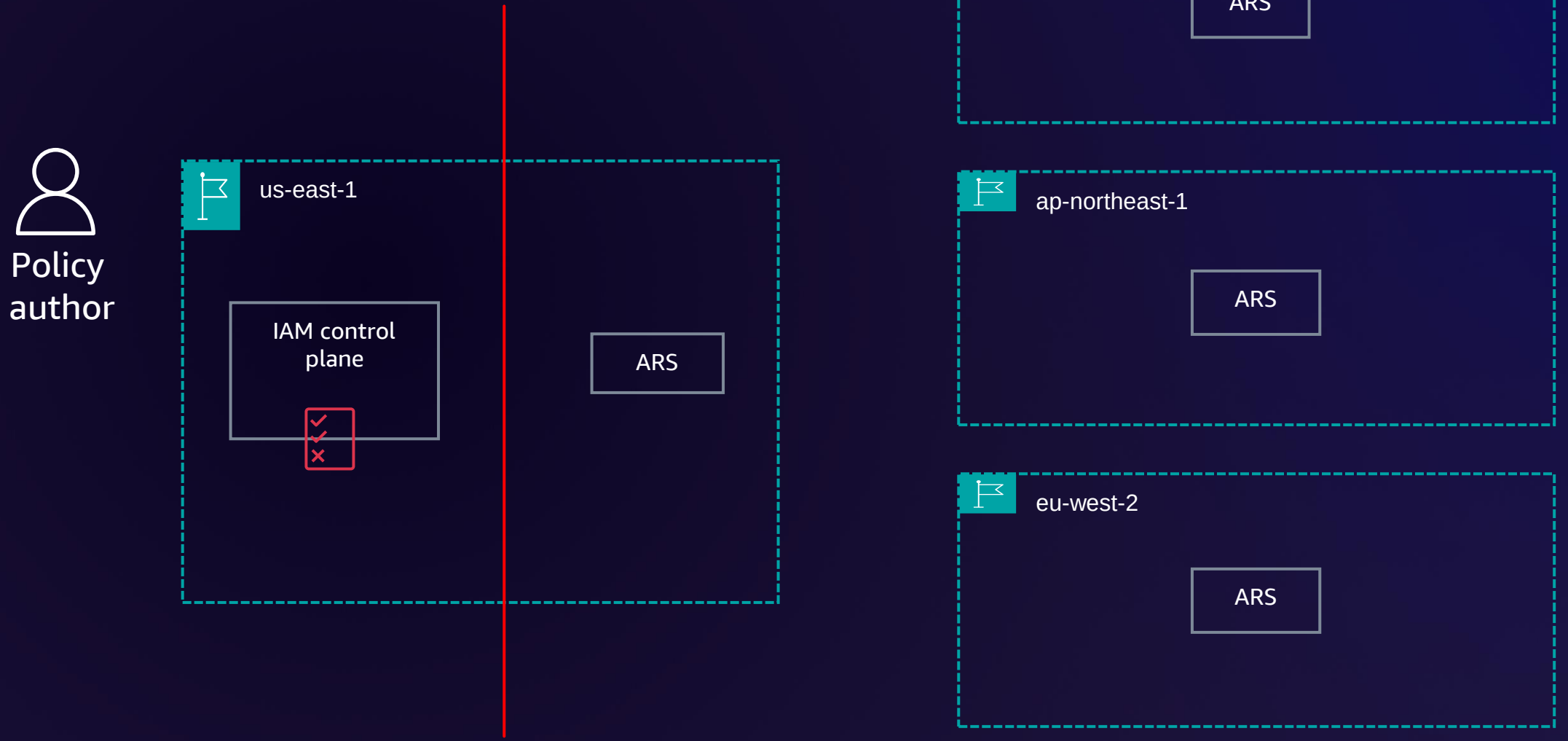
Identity-based policies overview

- Propagated from the IAM control plane to a regional IAM data plane (ARS)
- AWS managed policies go through a review process before they touch the IAM control plane

Identity-based policies



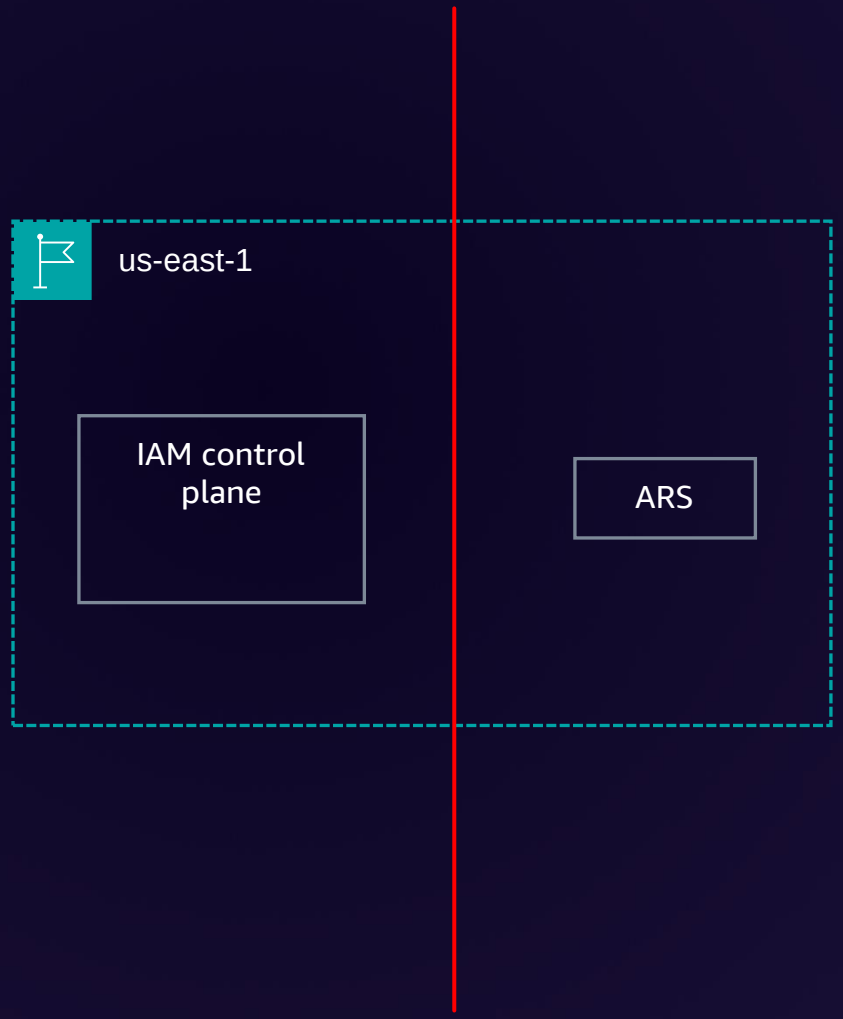
Identity-based policies

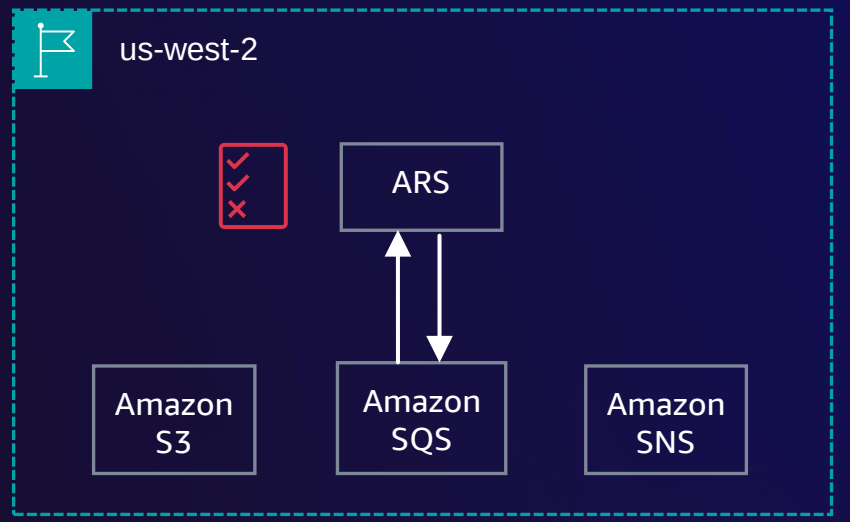


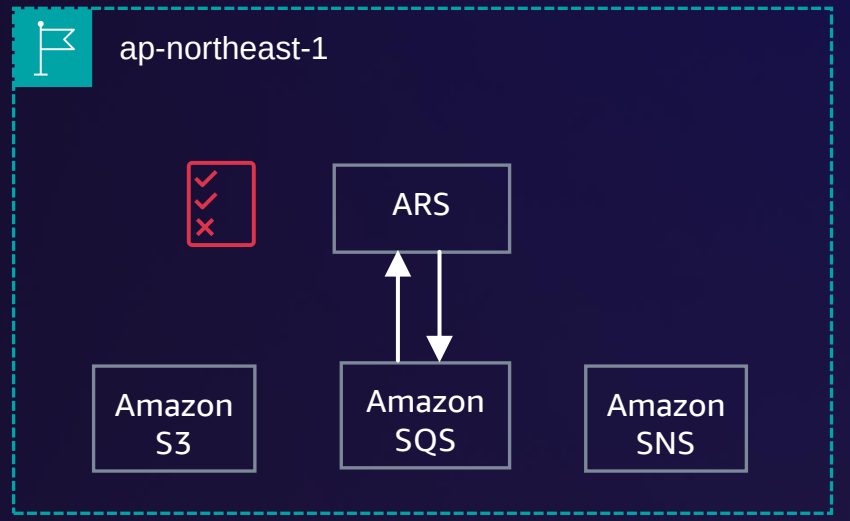
Identity-based policies

Identity-based policies

Policy

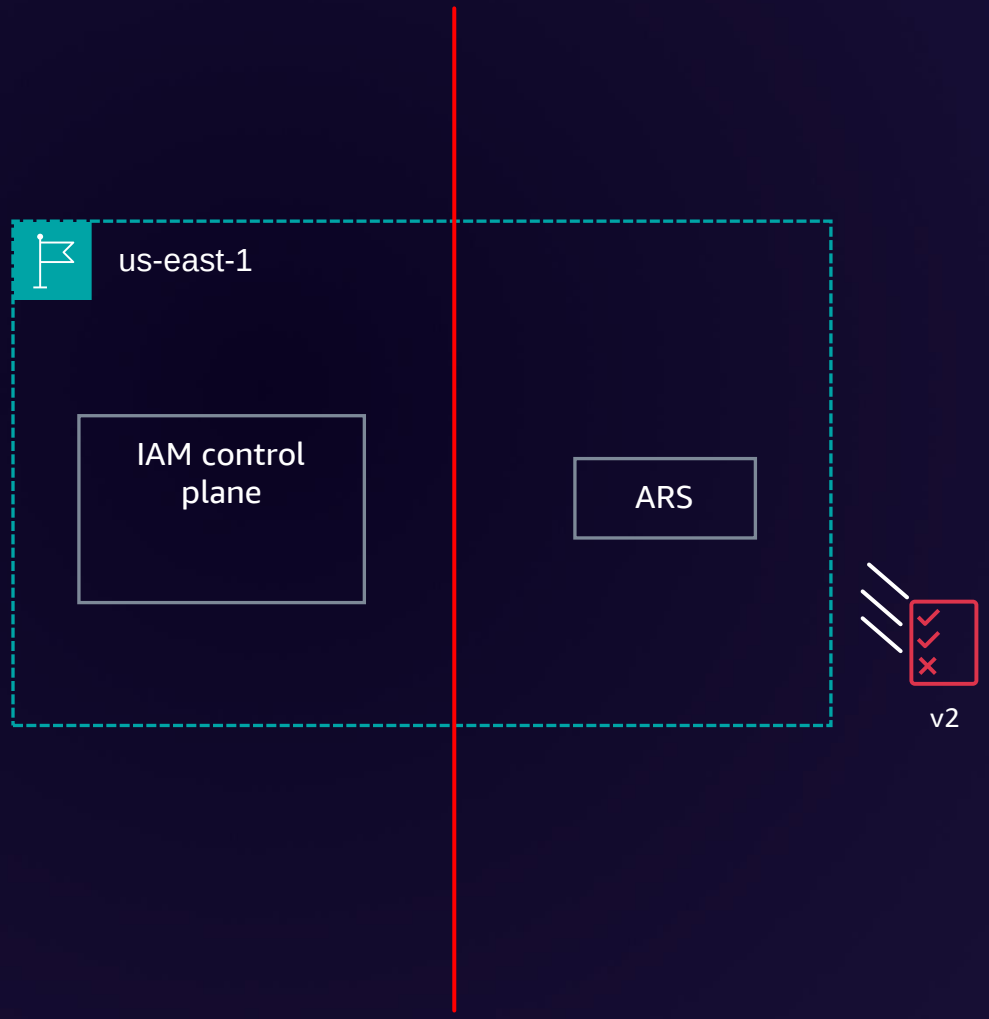


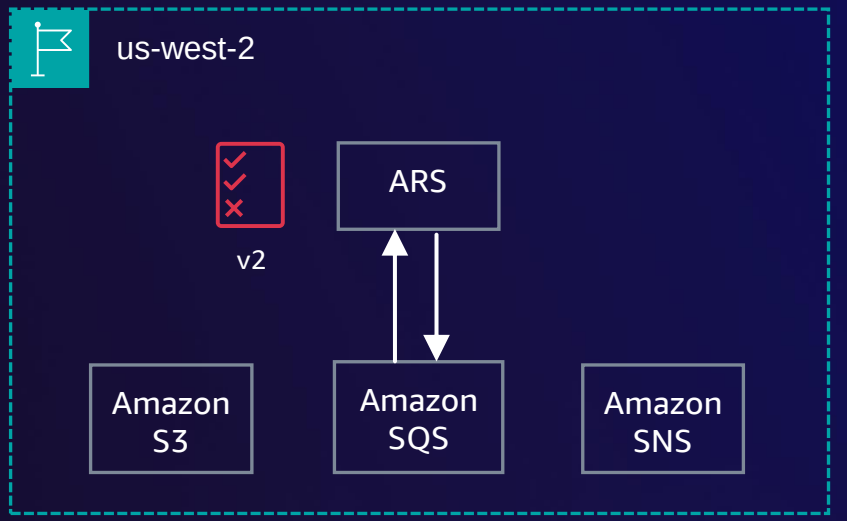


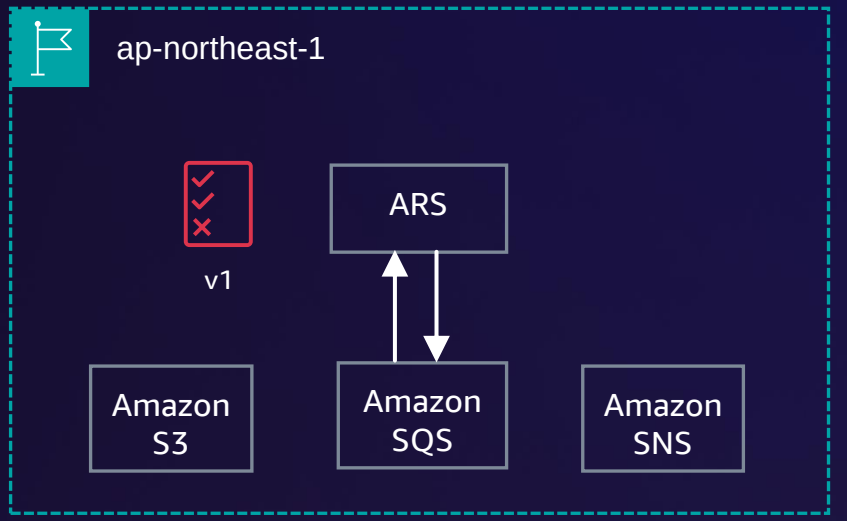


Identity-based policies

Policy



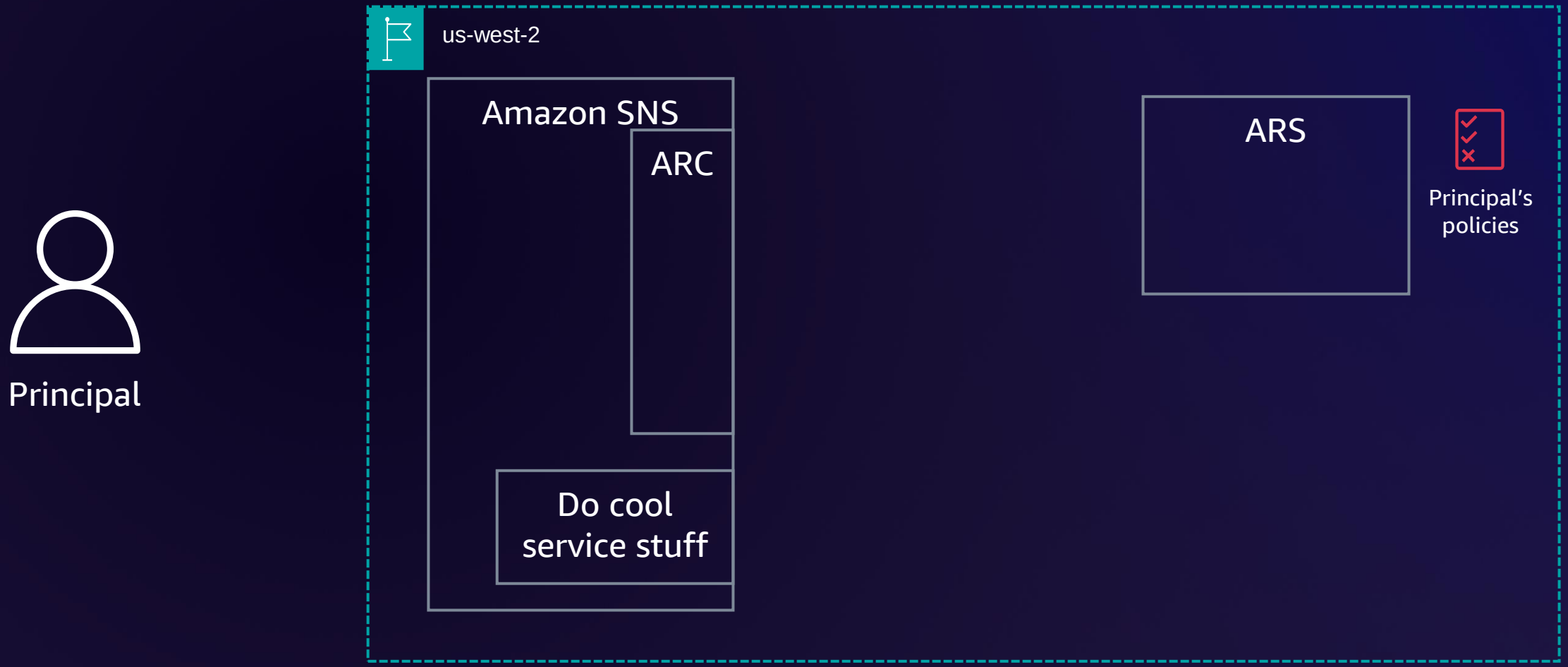




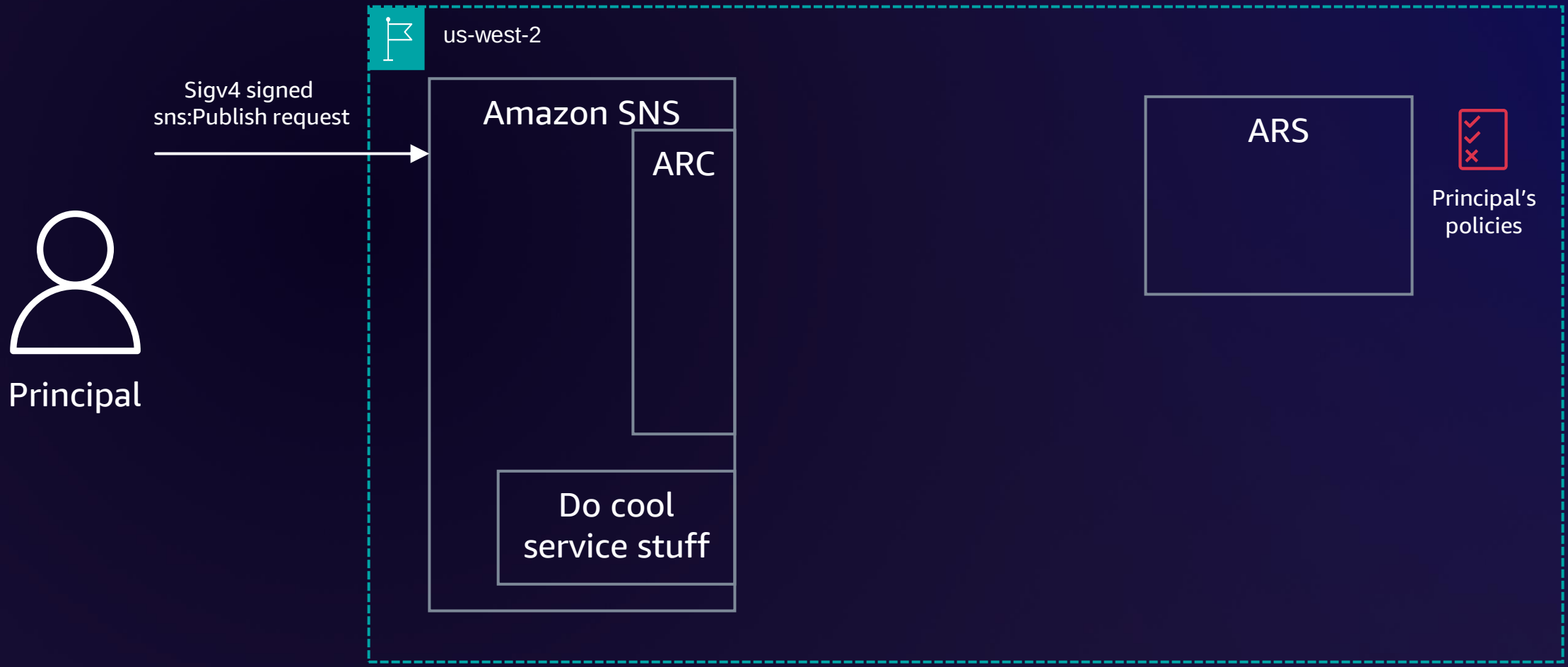
Where are policies evaluated?

- AWS services authorize their own requests
- Services use the auth runtime client (ARC) to consistently evaluate policies
- ARS performs authentication, ARC performs authorization

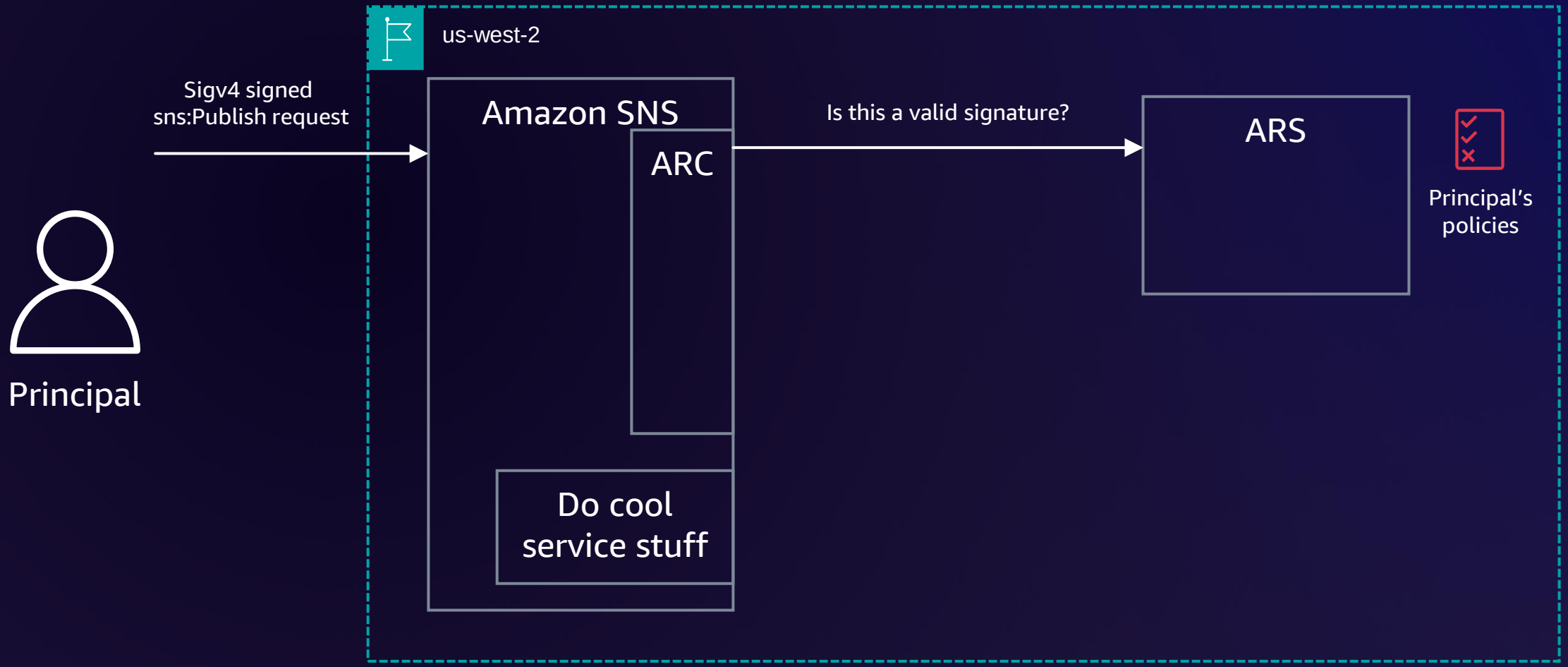
Identity-based policies – a sample request



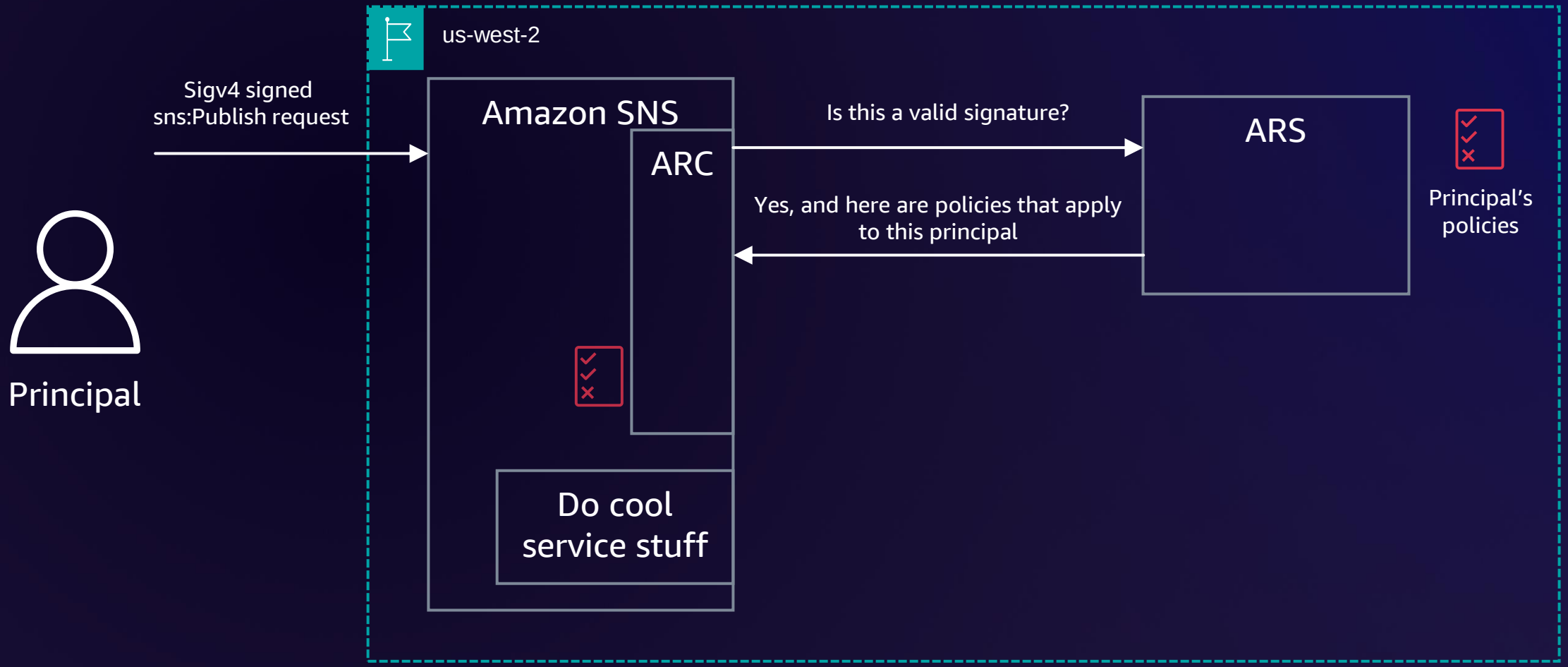
Identity-based policies – a sample request



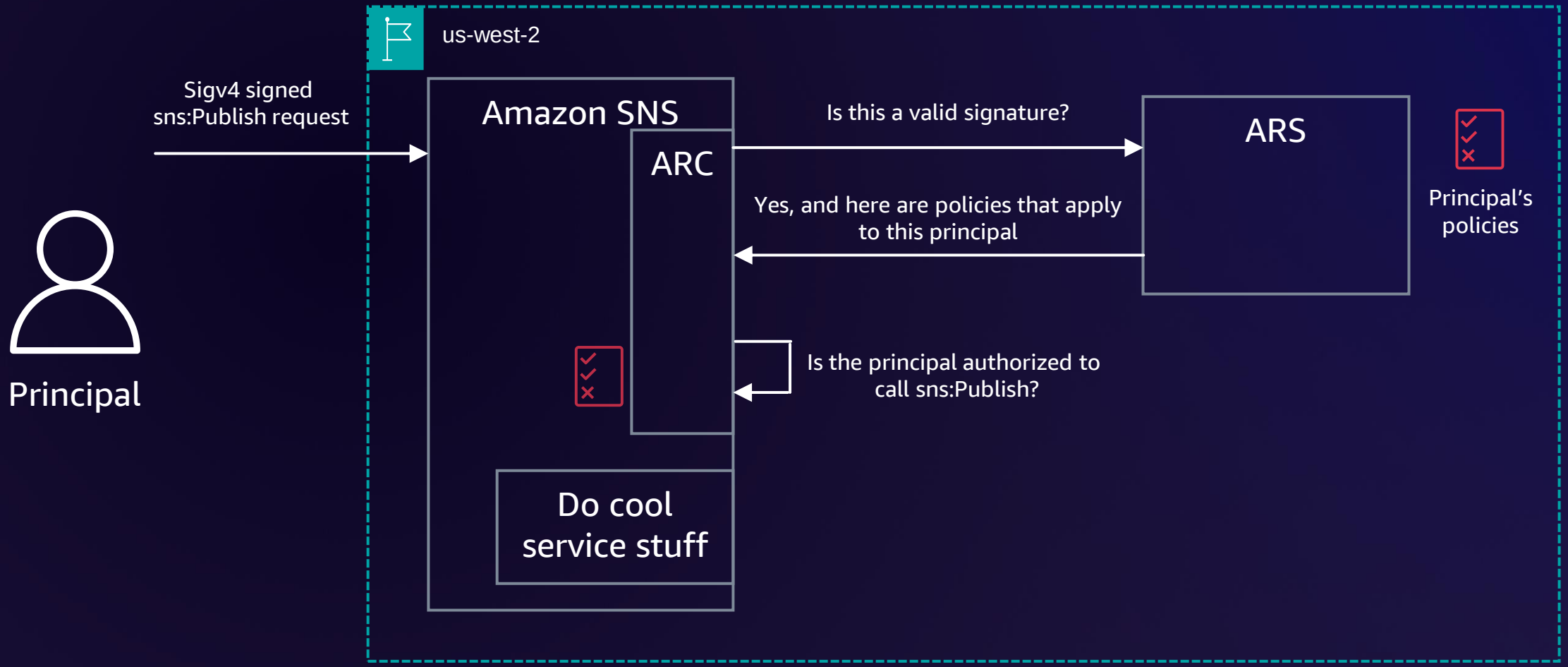
Identity-based policies – a sample request



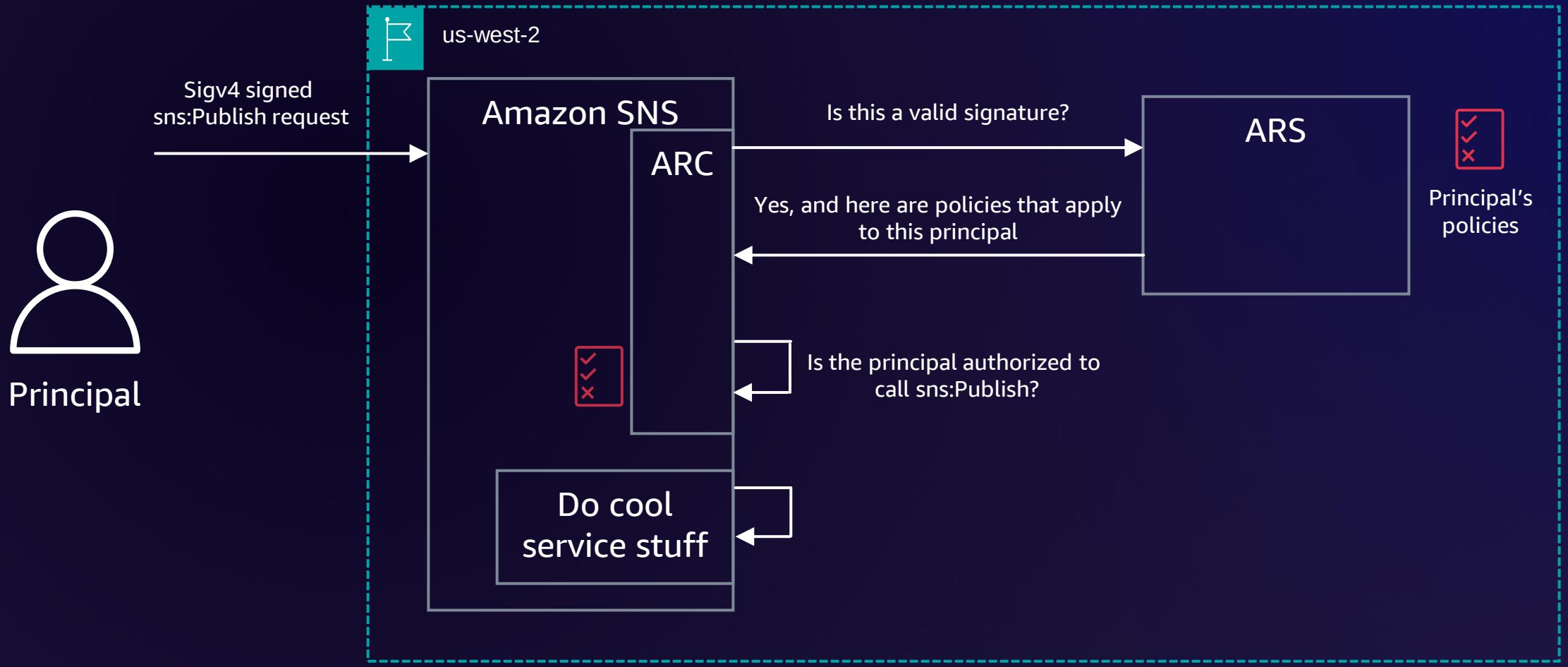
Identity-based policies – a sample request



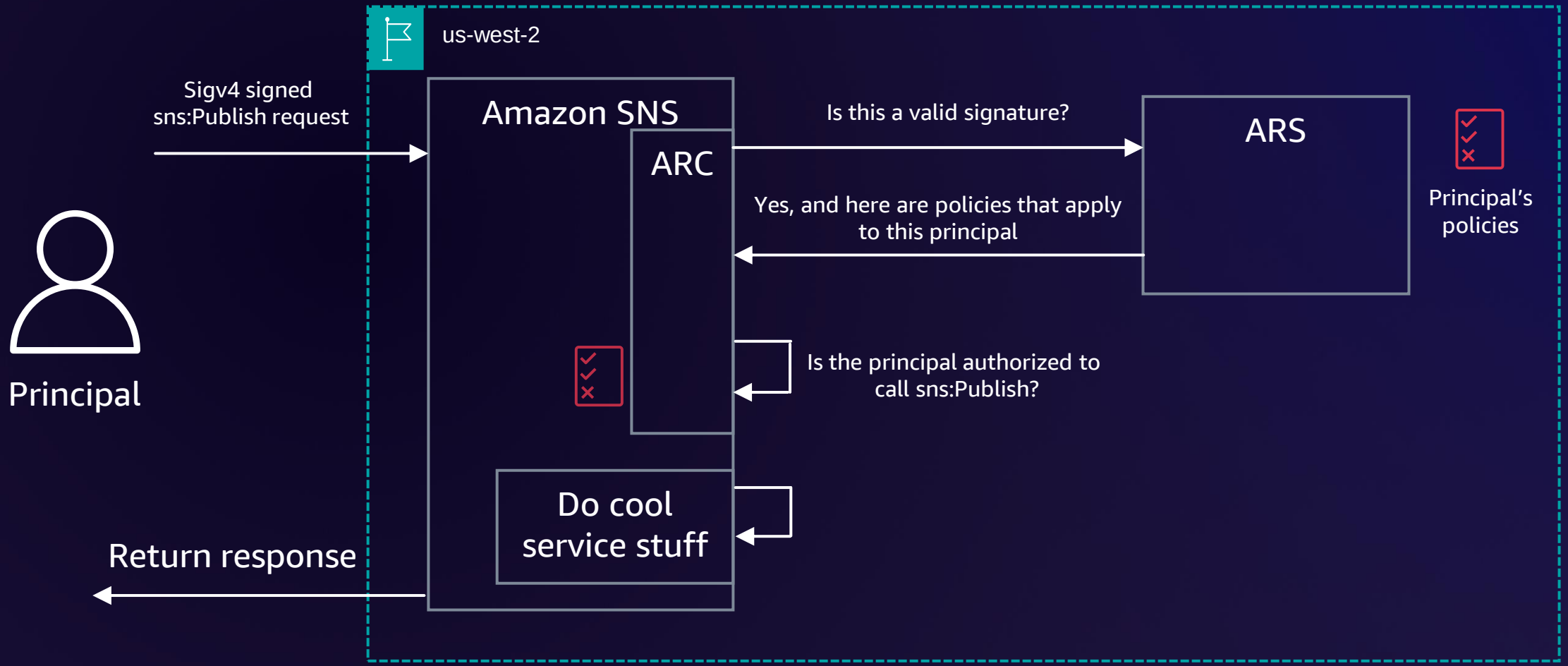
Identity-based policies – a sample request



Identity-based policies – a sample request

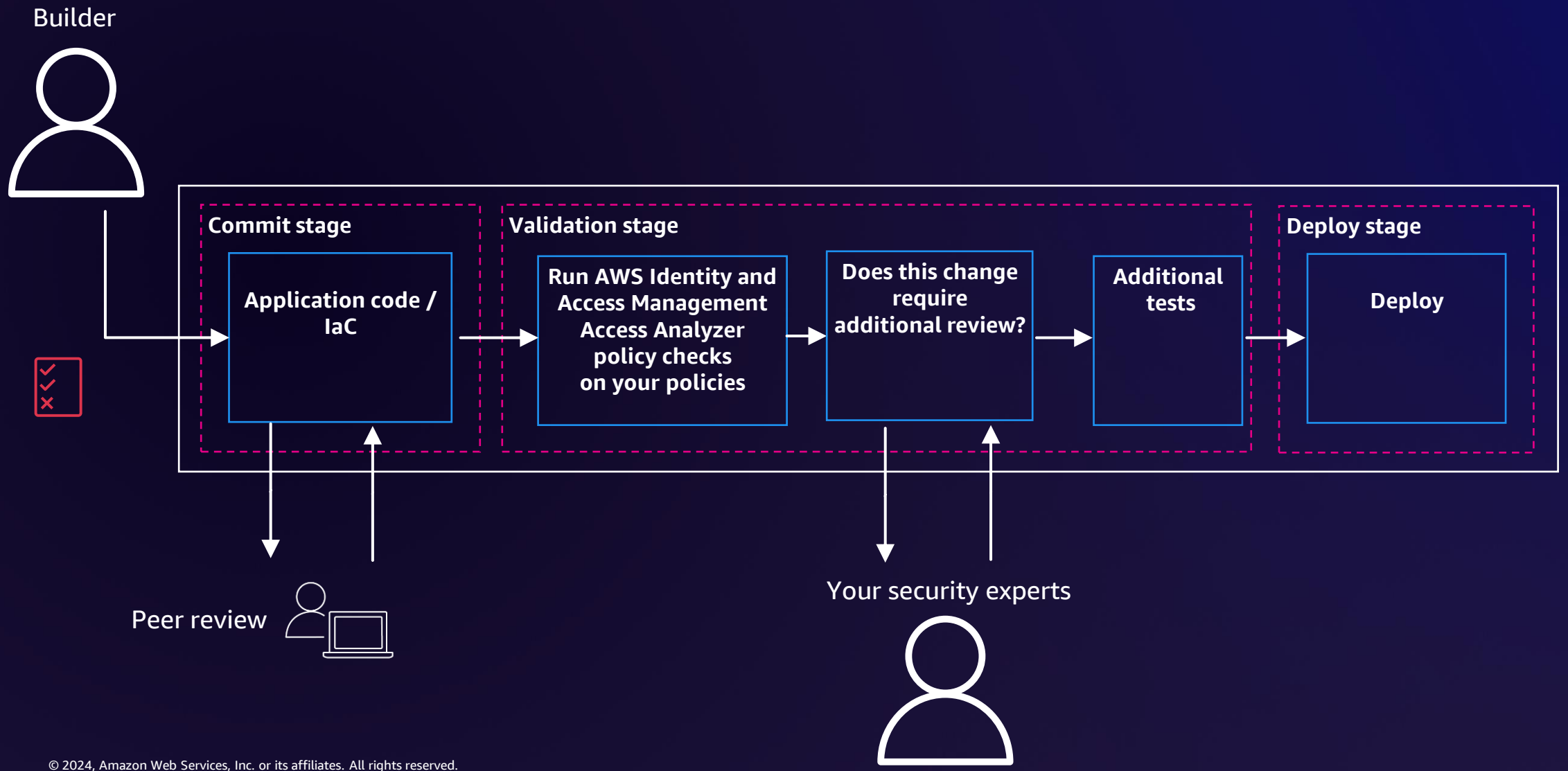


Identity-based policies – a sample request

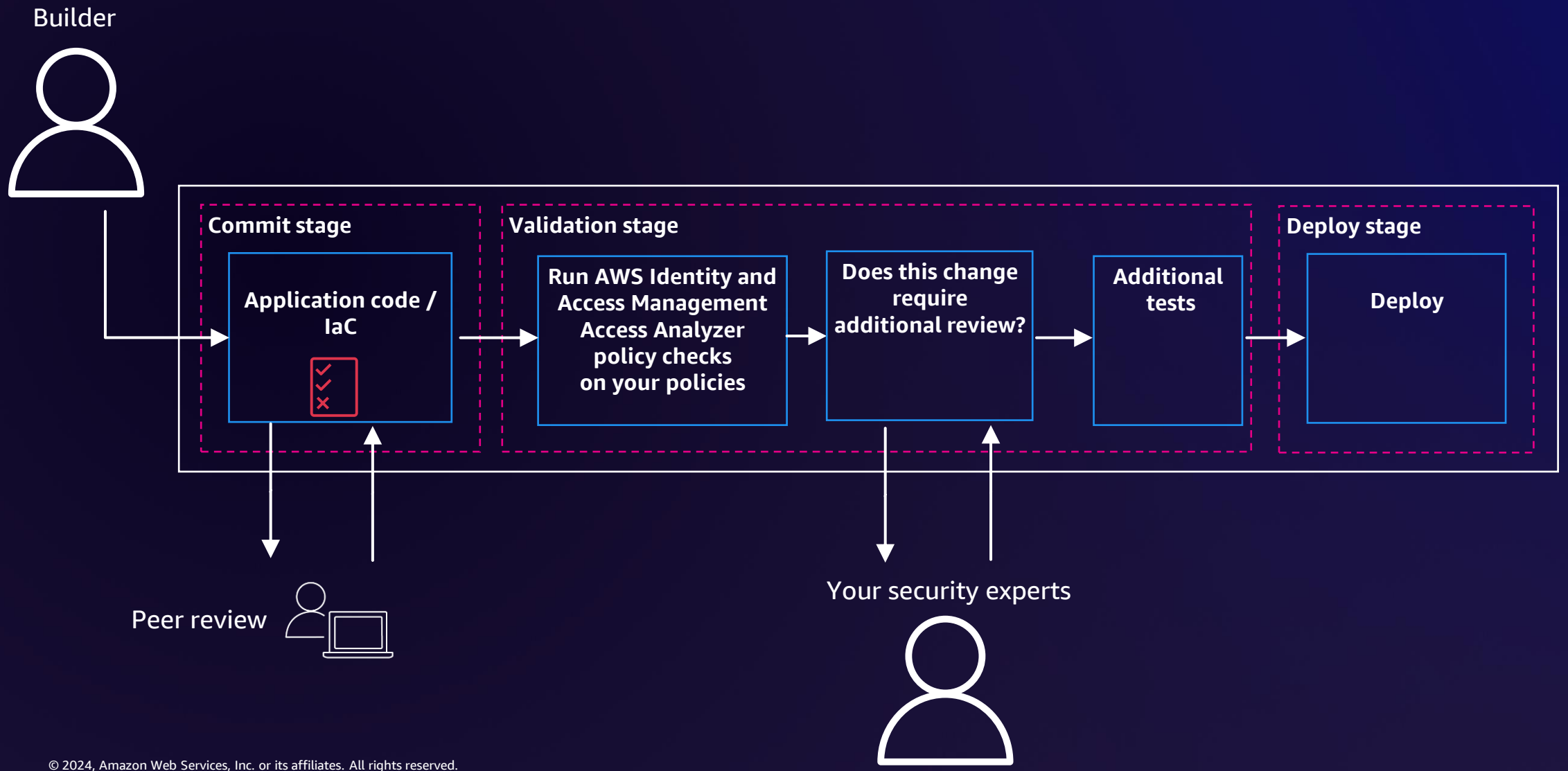


AWS managed policy review process

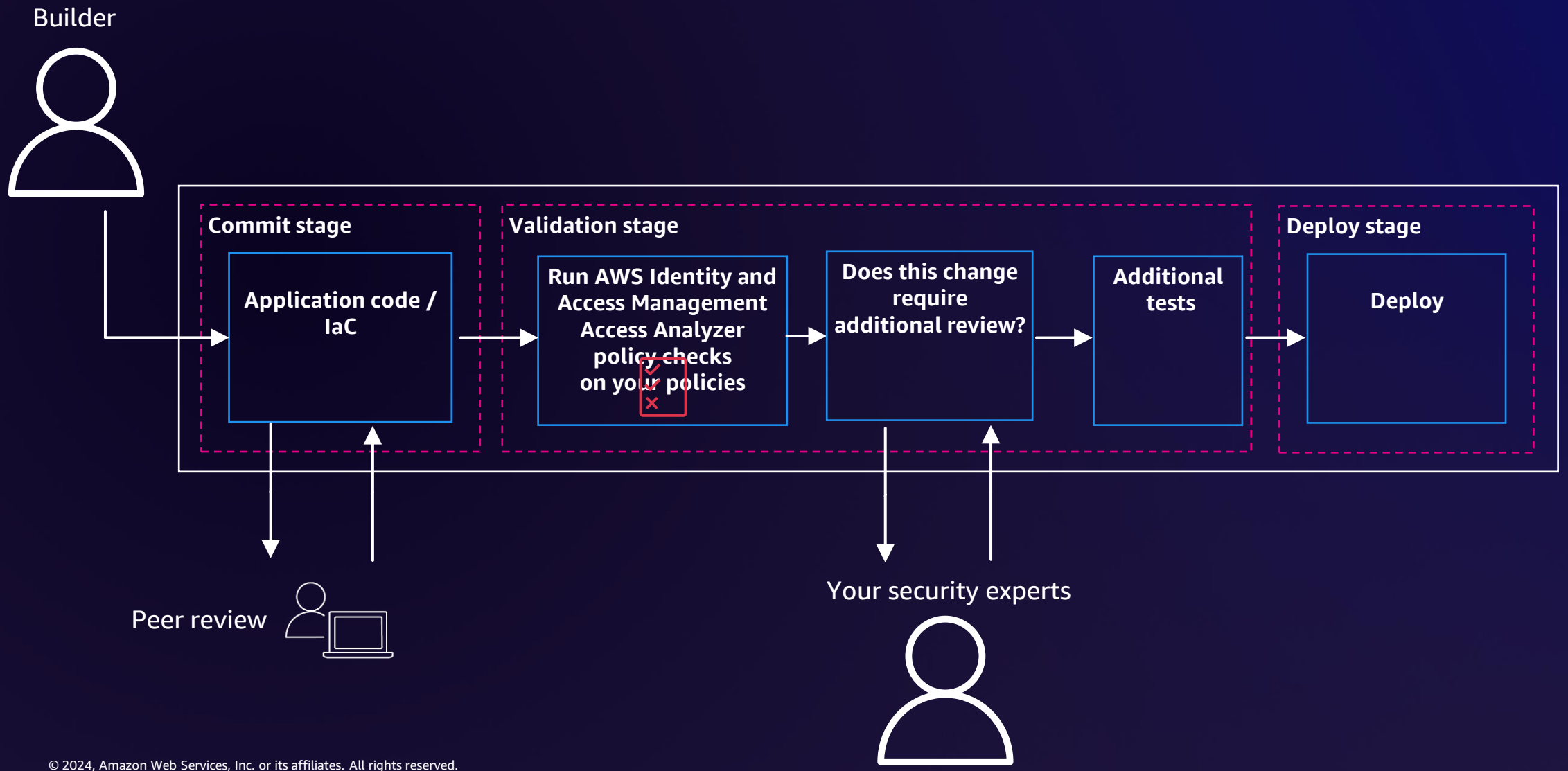
You can do this too



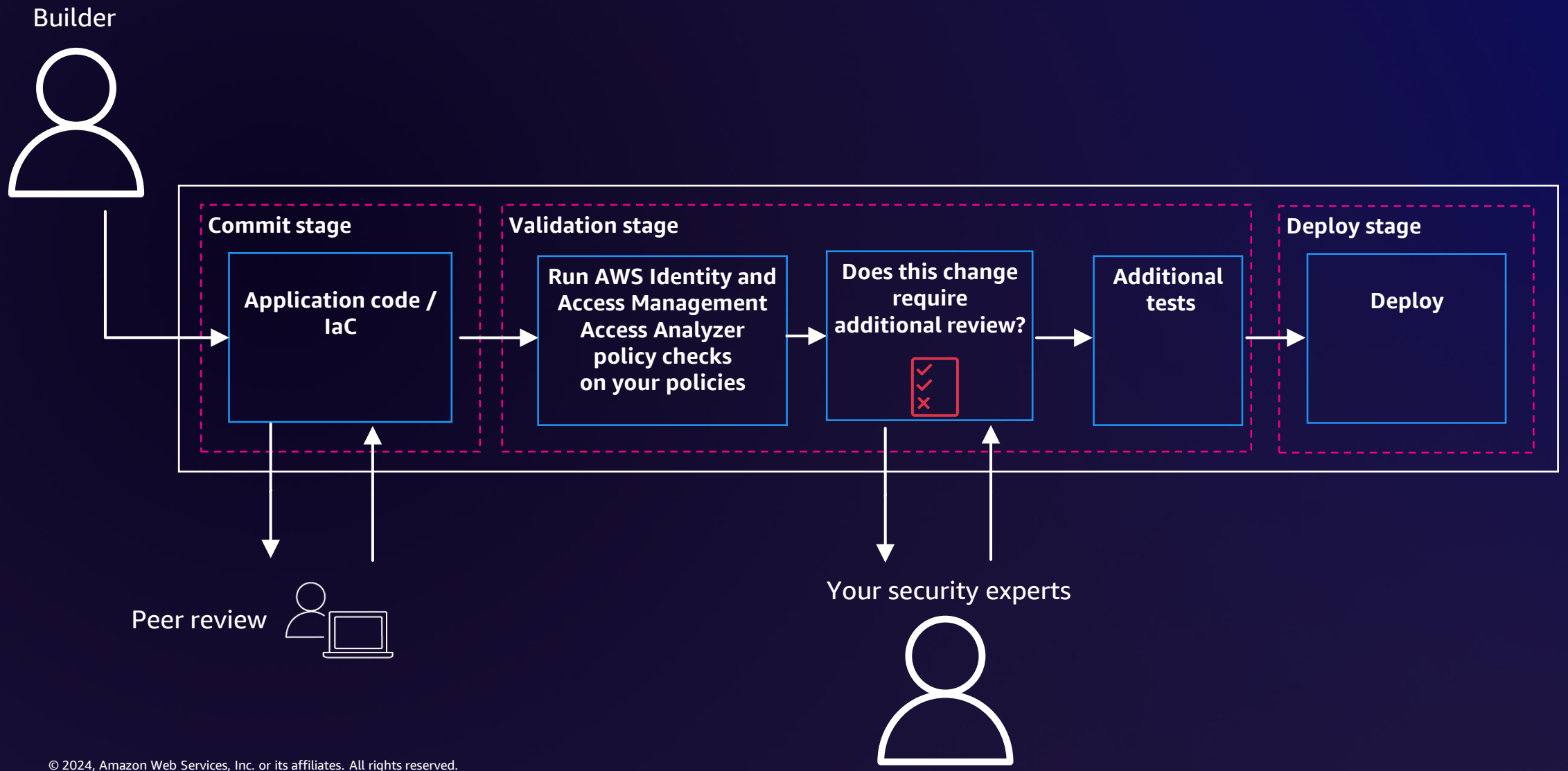
You can do this too



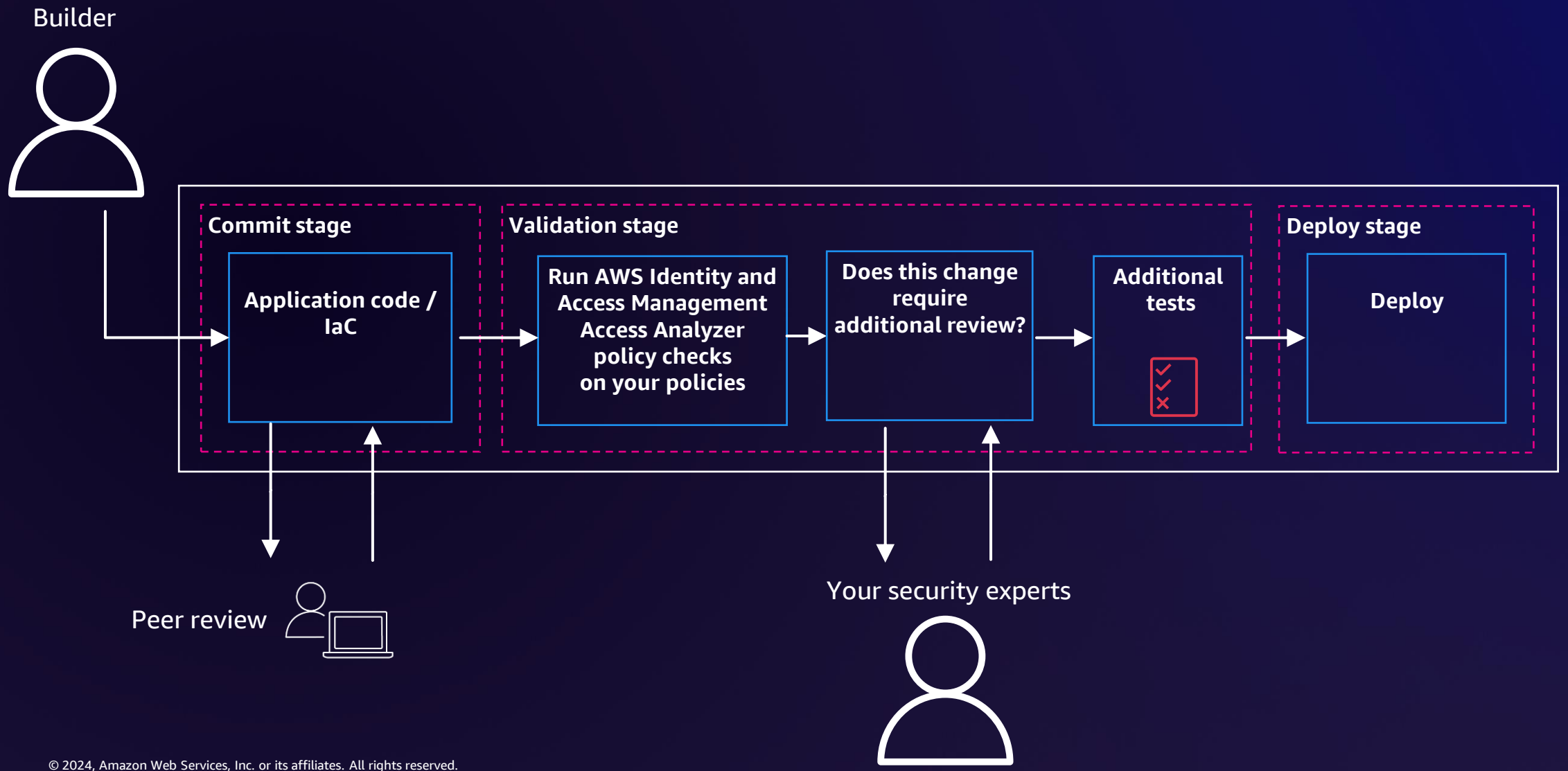
You can do this too



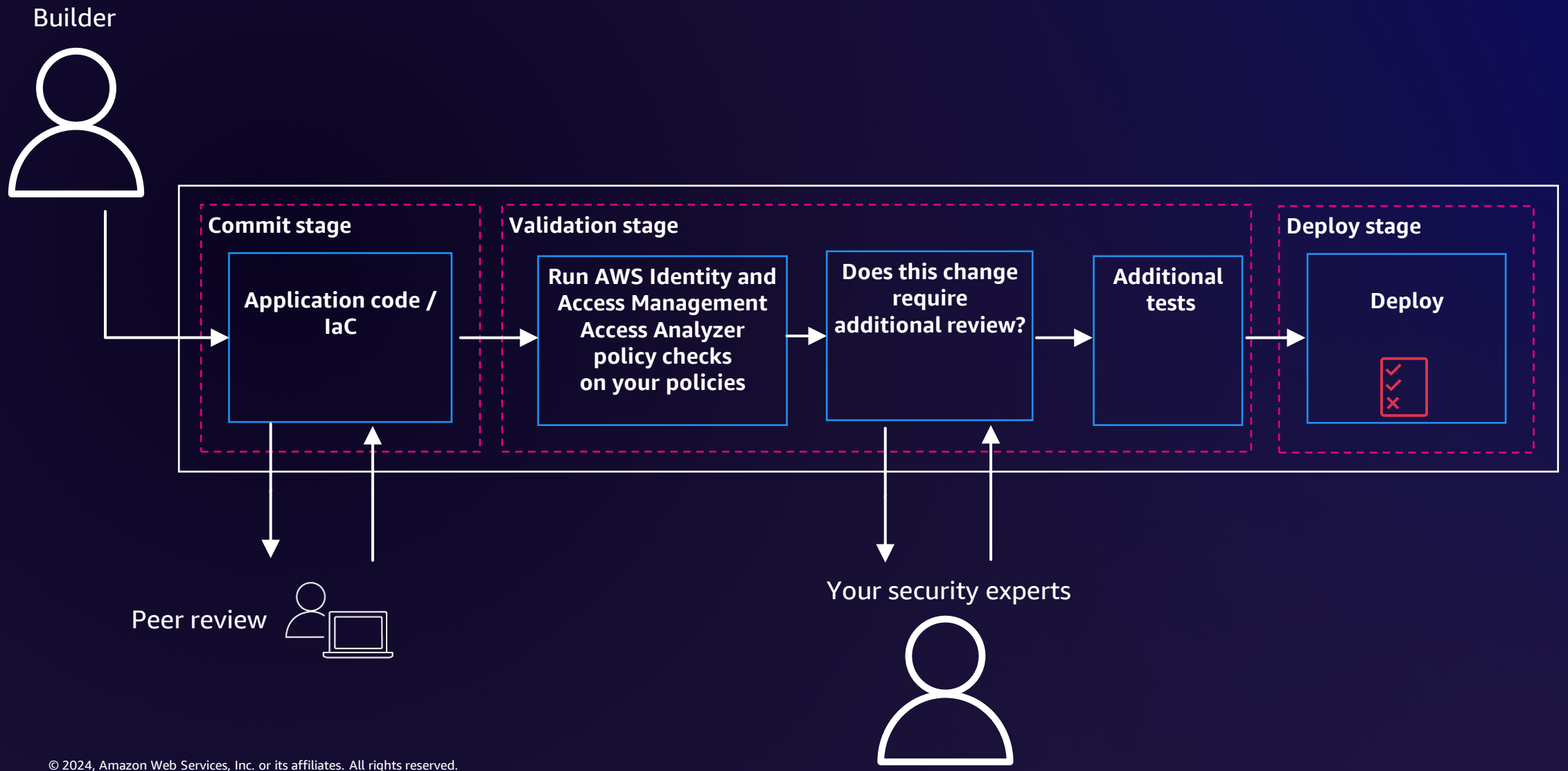
You can do this too



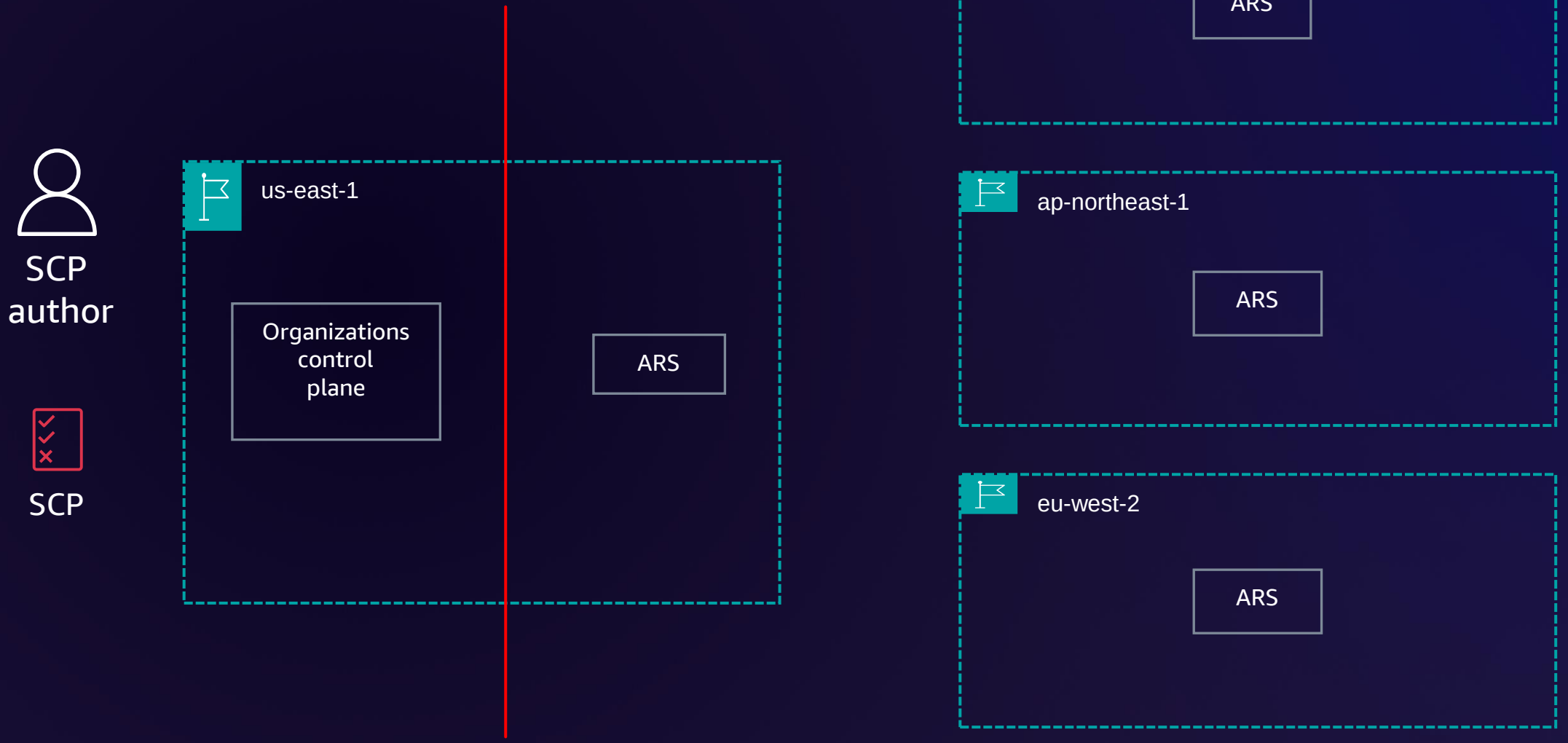
You can do this too



You can do this too



Service control policies



Service control policies

Service control policies

Identity-based policy takeaways

- Policy updates are eventually consistent for good reason
- Policy evaluation is consistently performed by a service

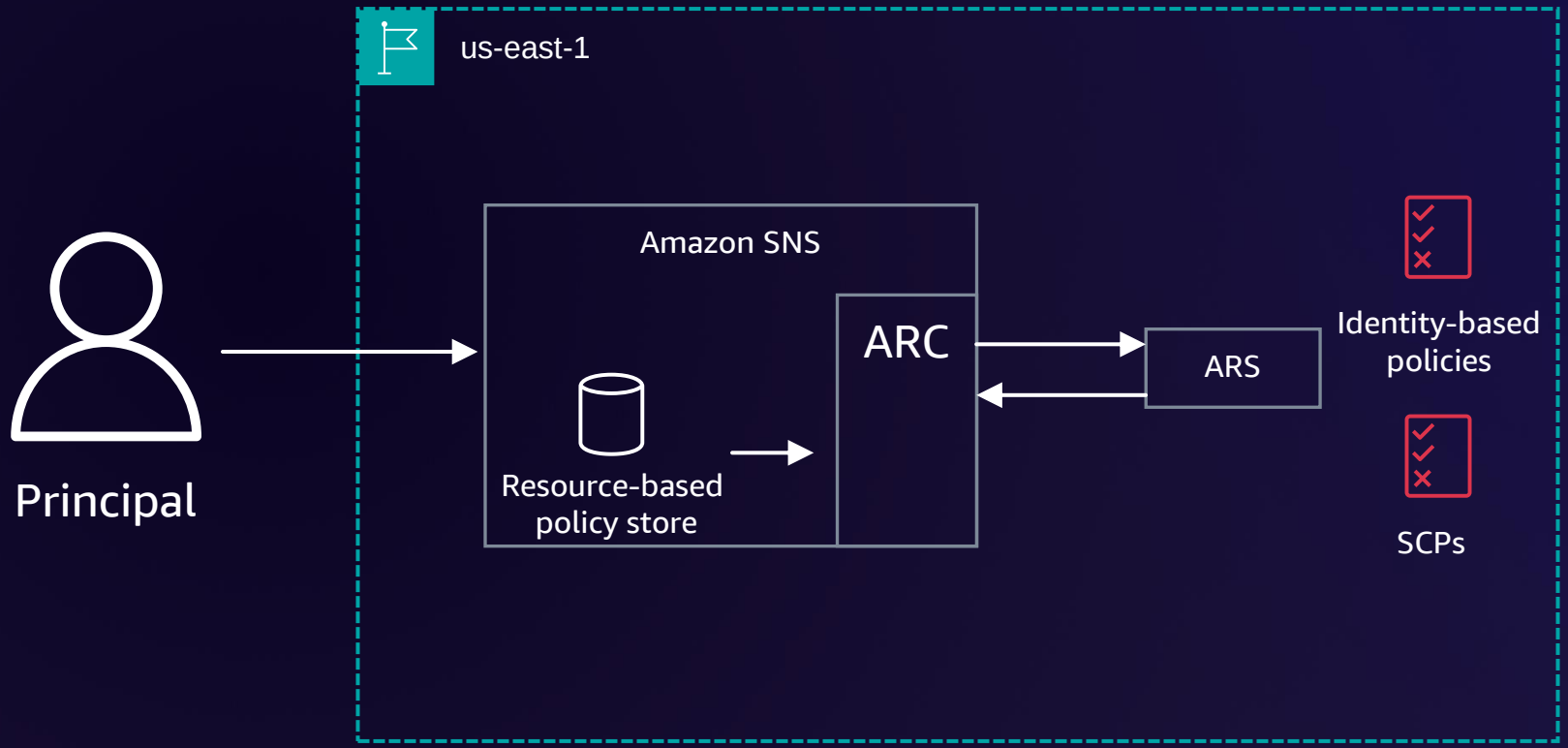
Identity-based policy takeaways

- Policy updates are eventually consistent for good reason
- Policy evaluation is consistently performed by a service
- You can build your own policy review process
- You may not need the level of review that we have

Resource-based policies overview

- Owned by an individual service
- Adhere to the IAM policy specification, but services can customize available policy elements

Resource-based policies



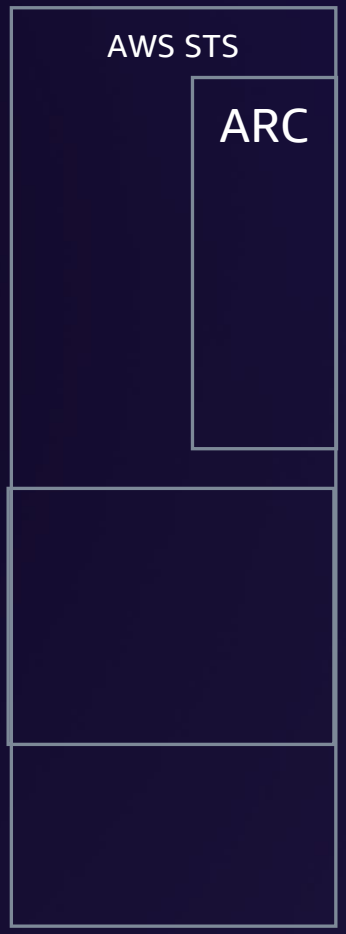
Session policies overview

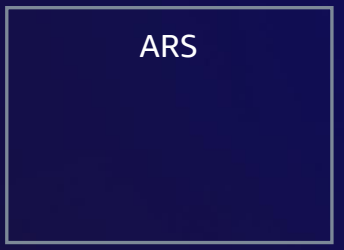
- Injected into the session token for an AssumeRole* request
- Only copy of session policy is in the token itself

* or AssumeRoleWithSAML, AssumeRoleWithWebIdentity

Session policies – creating a session with AWS STS

Principal

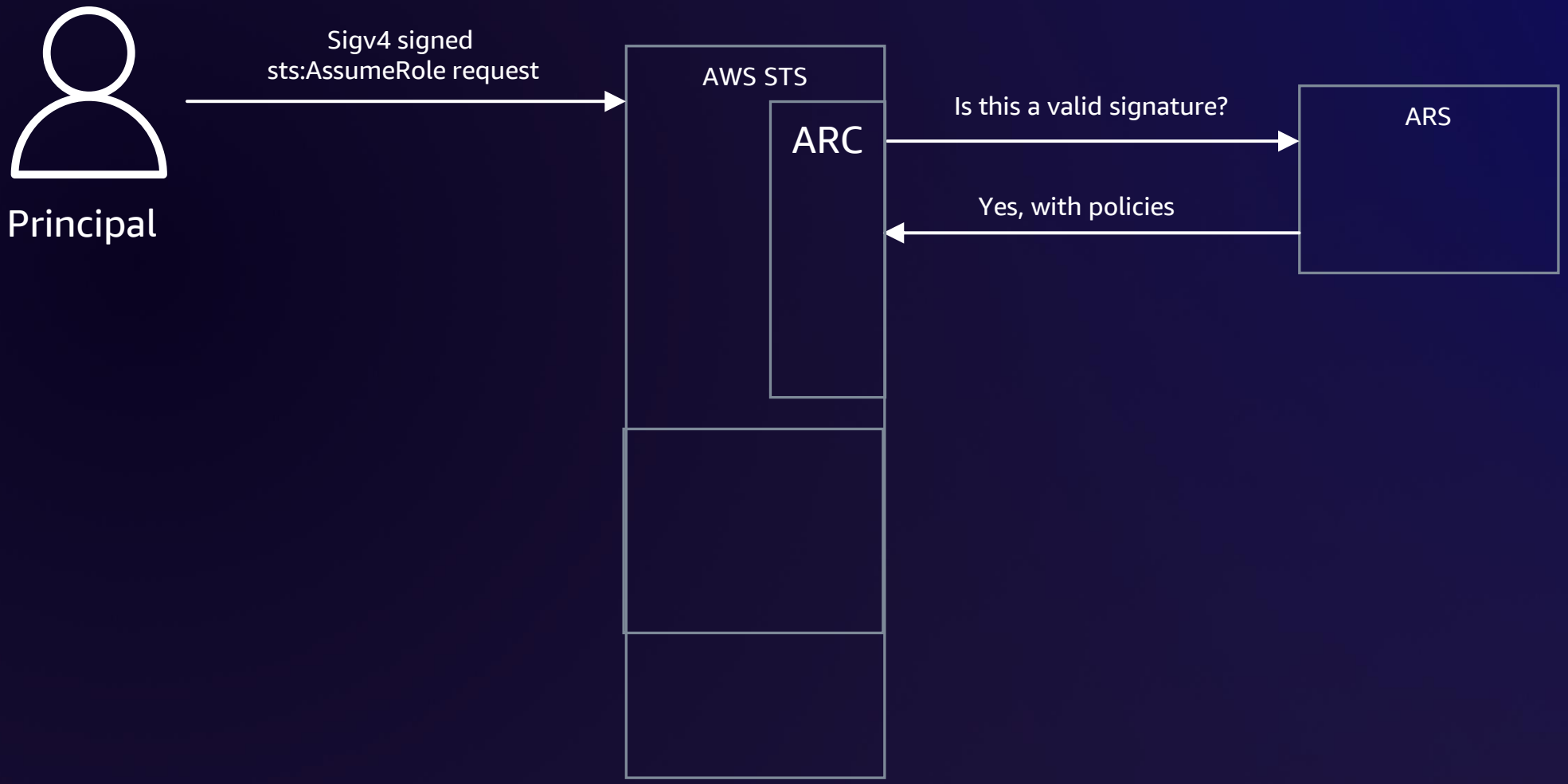




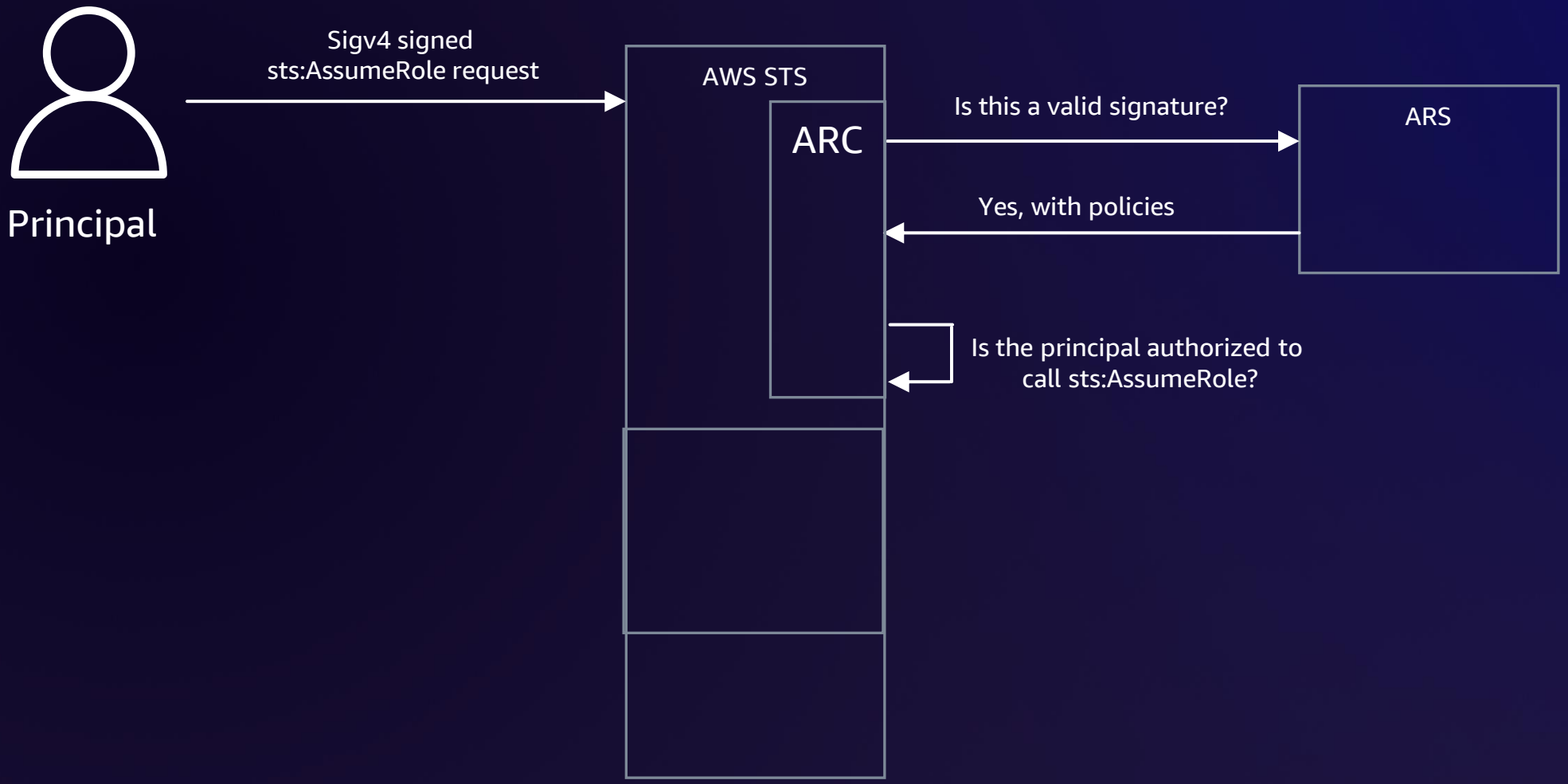
Session policies – creating a session with AWS STS

Session policies – creating a session with AWS STS

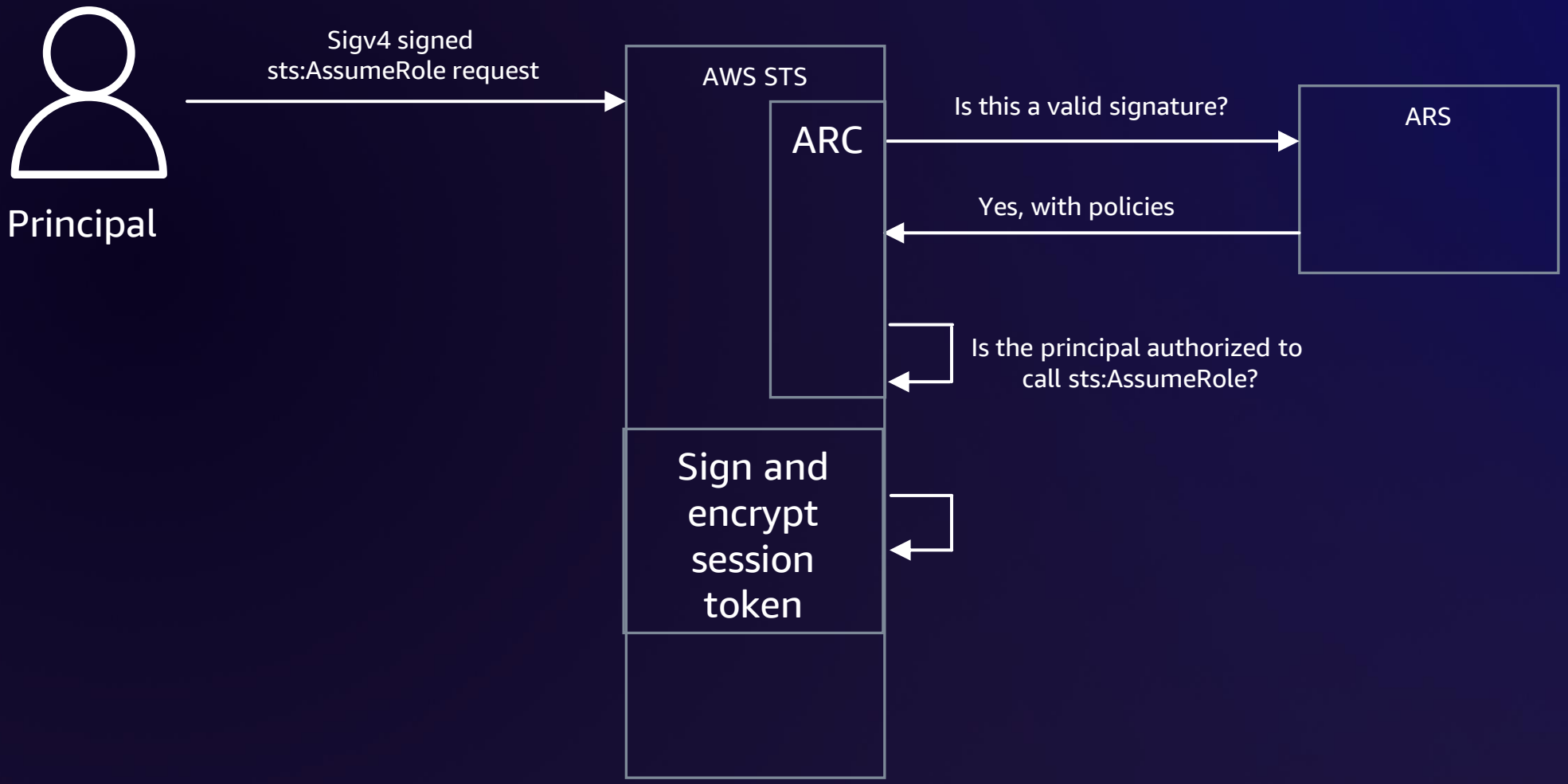
Session policies – creating a session with AWS STS



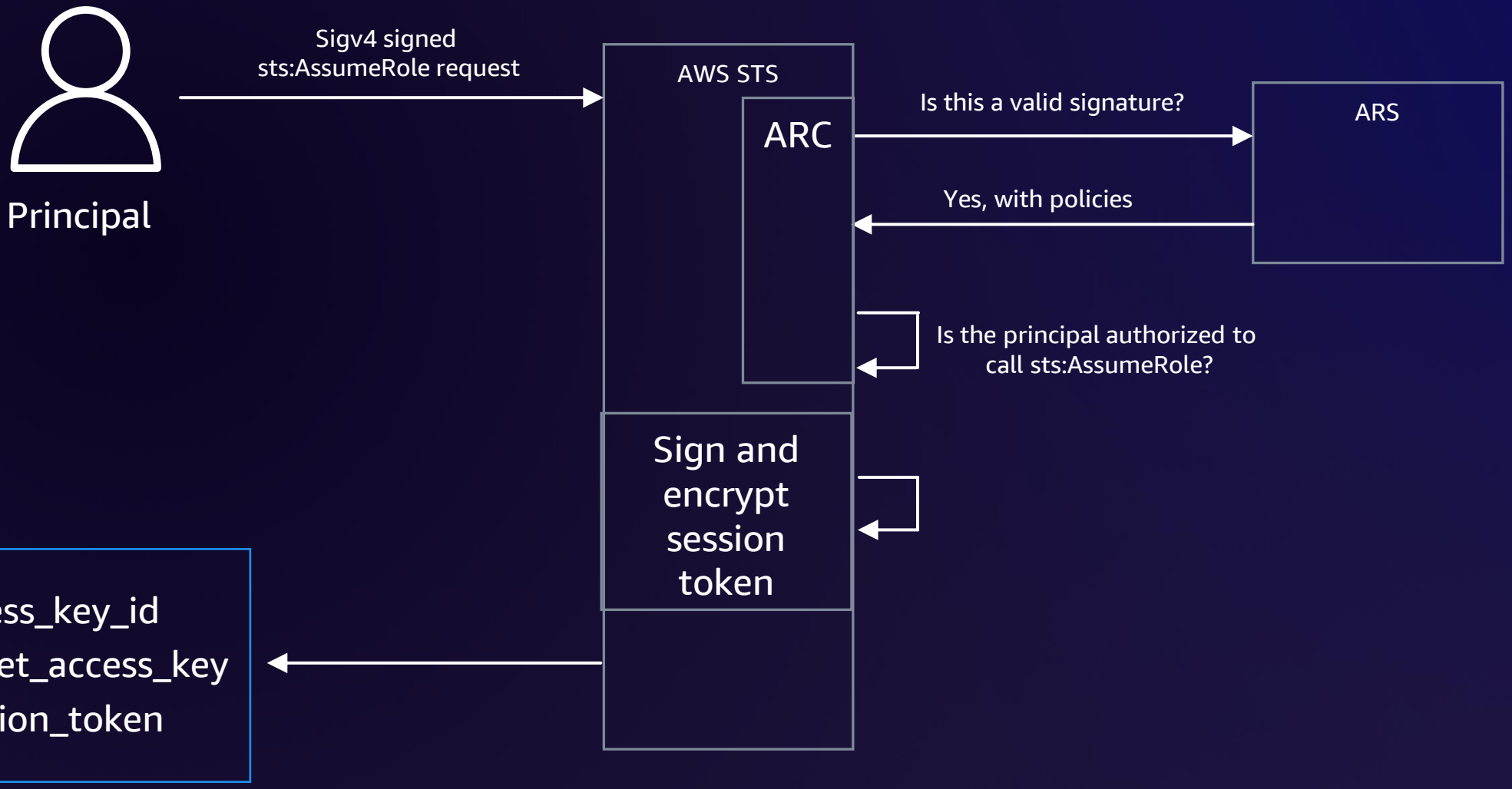
Session policies – creating a session with AWS STS



Session policies – creating a session with AWS STS



Session policies – creating a session with AWS STS

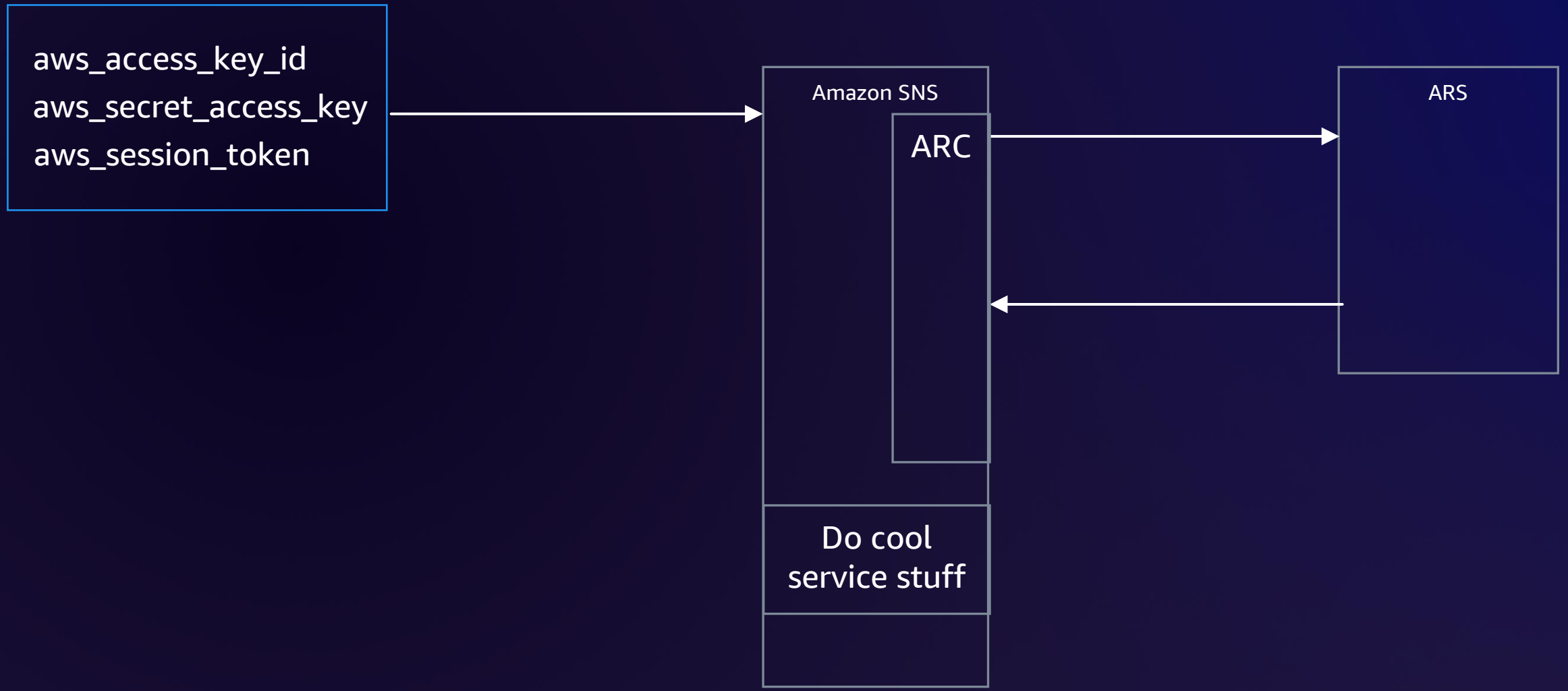


The session token

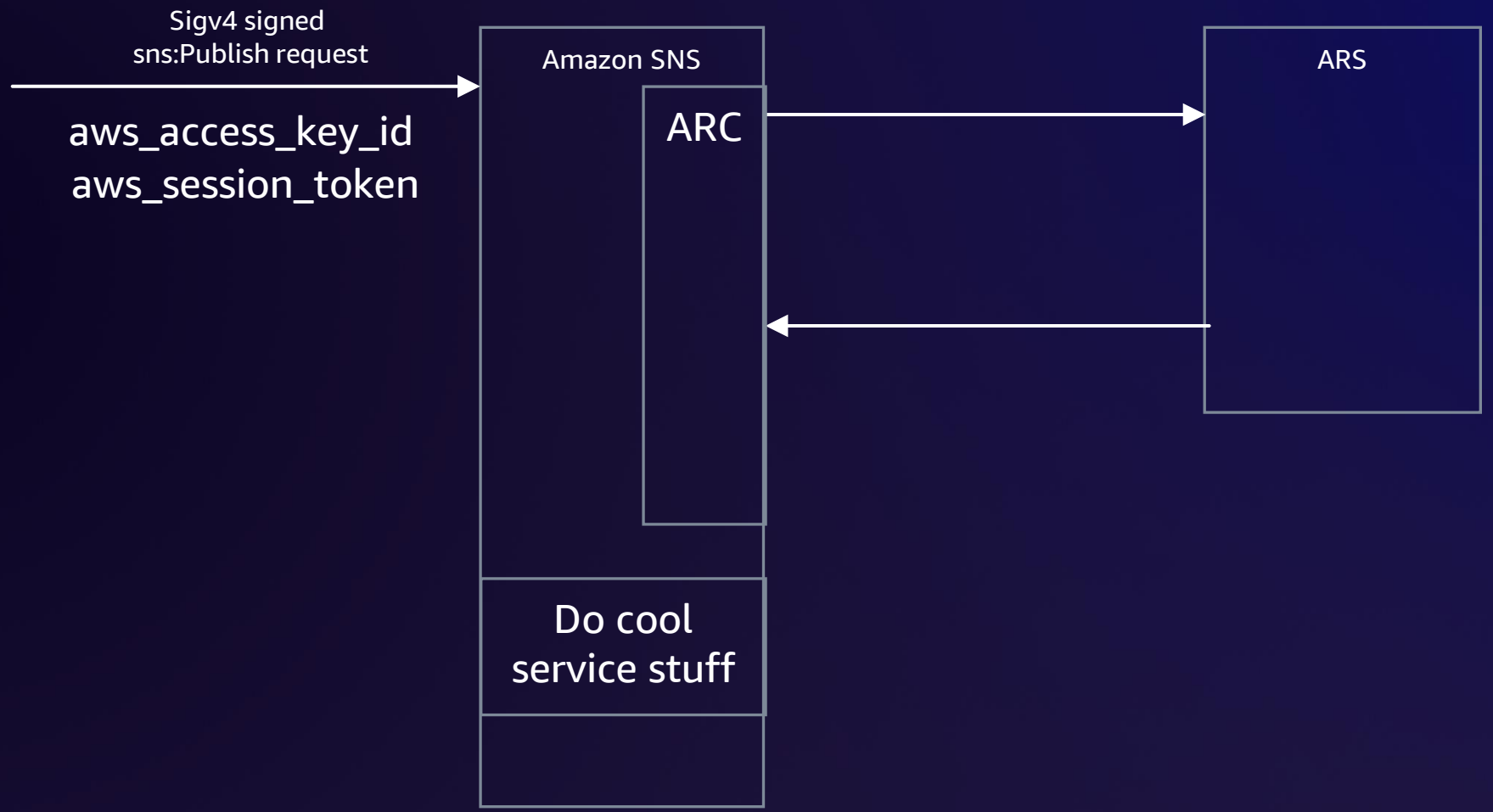
aws_session_token

```
Issuer:              AROABCDEFGHIEXAMPLE
Name:                MySession
AccessKeyId:         ASIAJKLMNOPQEXAMPLE
SecretAccessKey:     x+0Bra63Fr+cER48CUtkHpCxLk8gFV8MawMS0RRF

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

ManagedPolicyArns: arn:aws:iam::111111111111:policy/MyPolicy

Context:
    ec2:RoleDelivery=2.0
    ec2:SourceInstanceArn=arn:aws:ec2:us-west-2:111111111111:instance/i-123
    aws:TokenIssueTime=2022-06-01T23:47:00Z
```

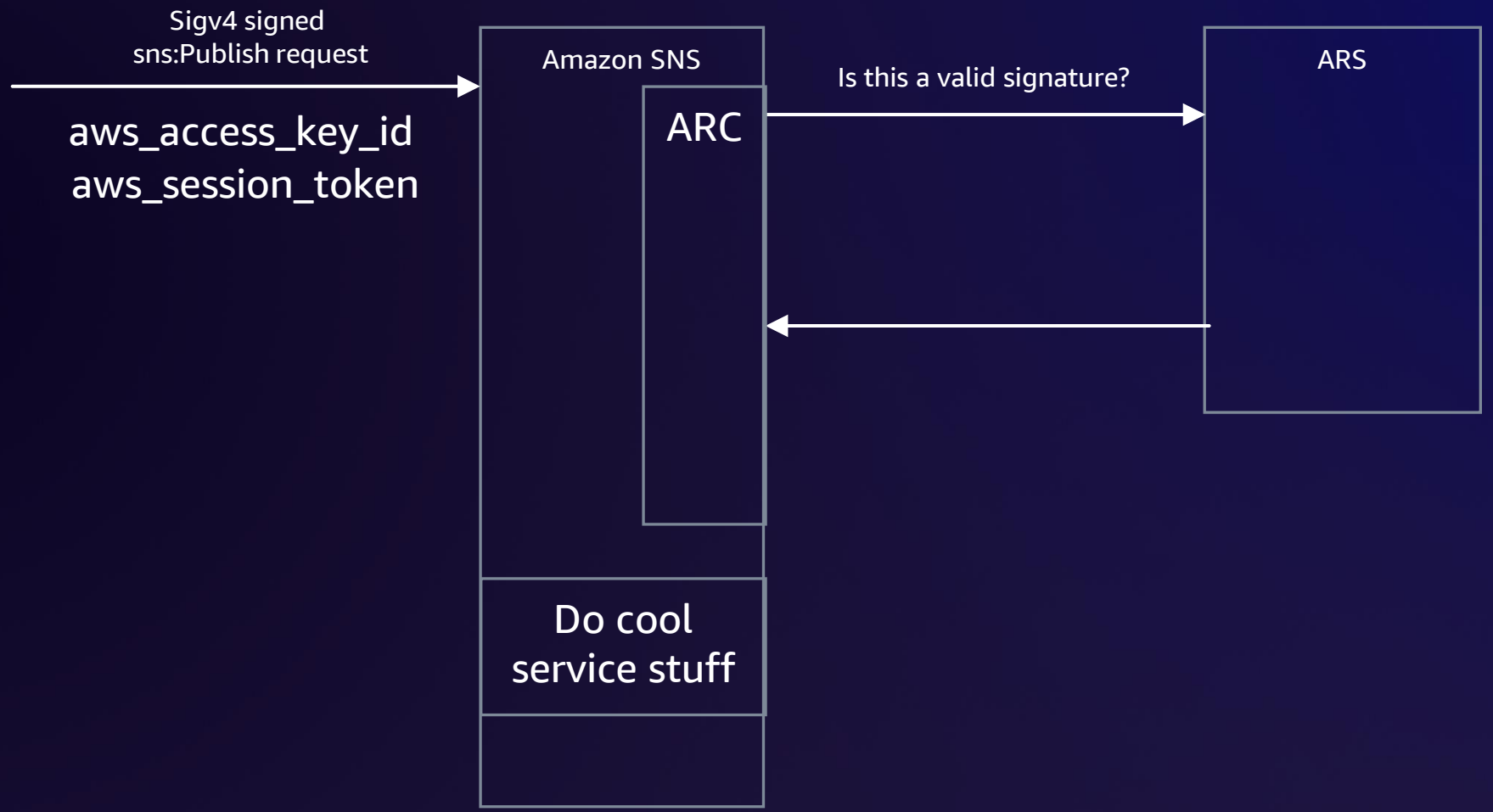
A request with temporary credentials



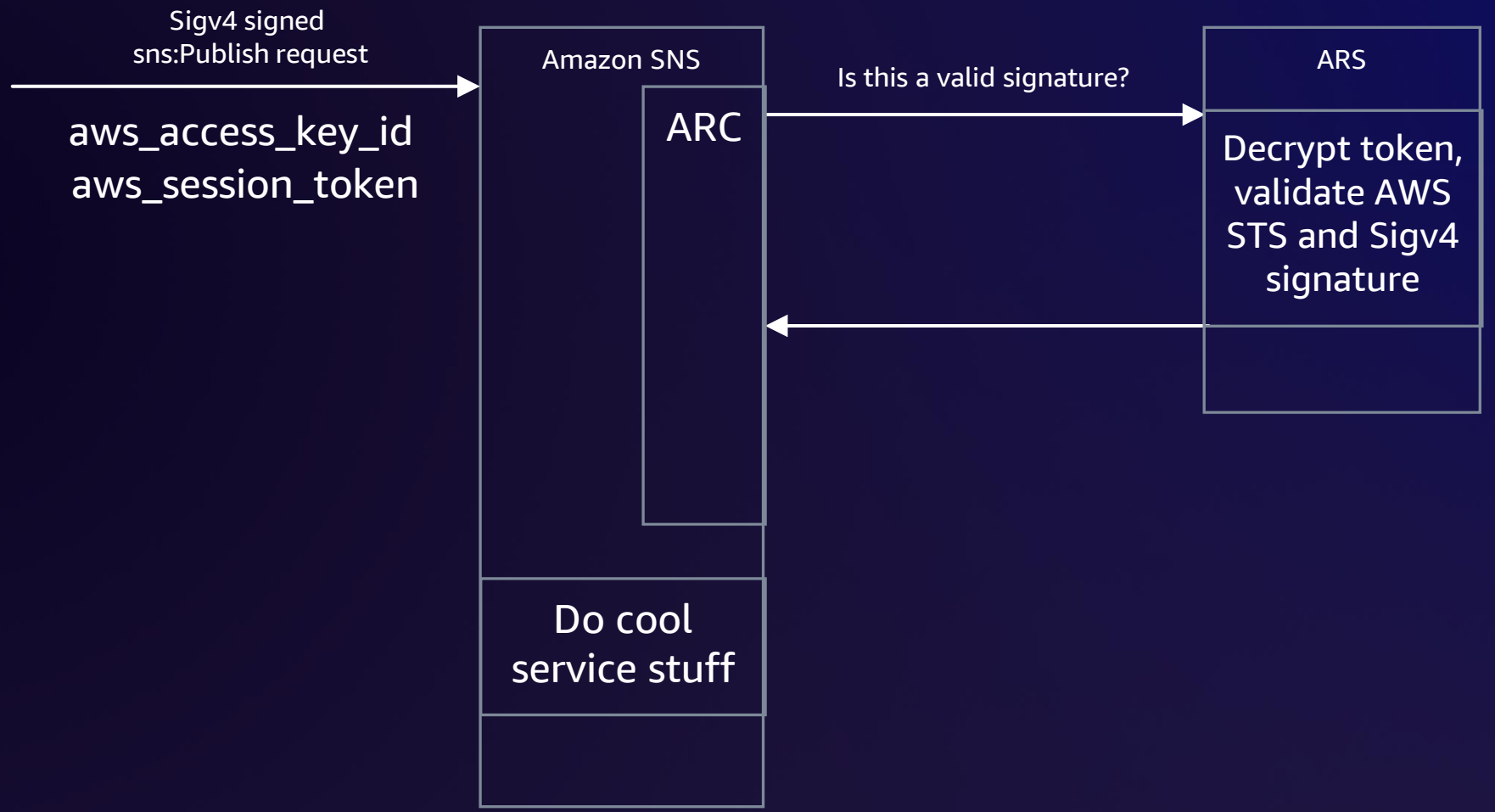
A request with temporary credentials



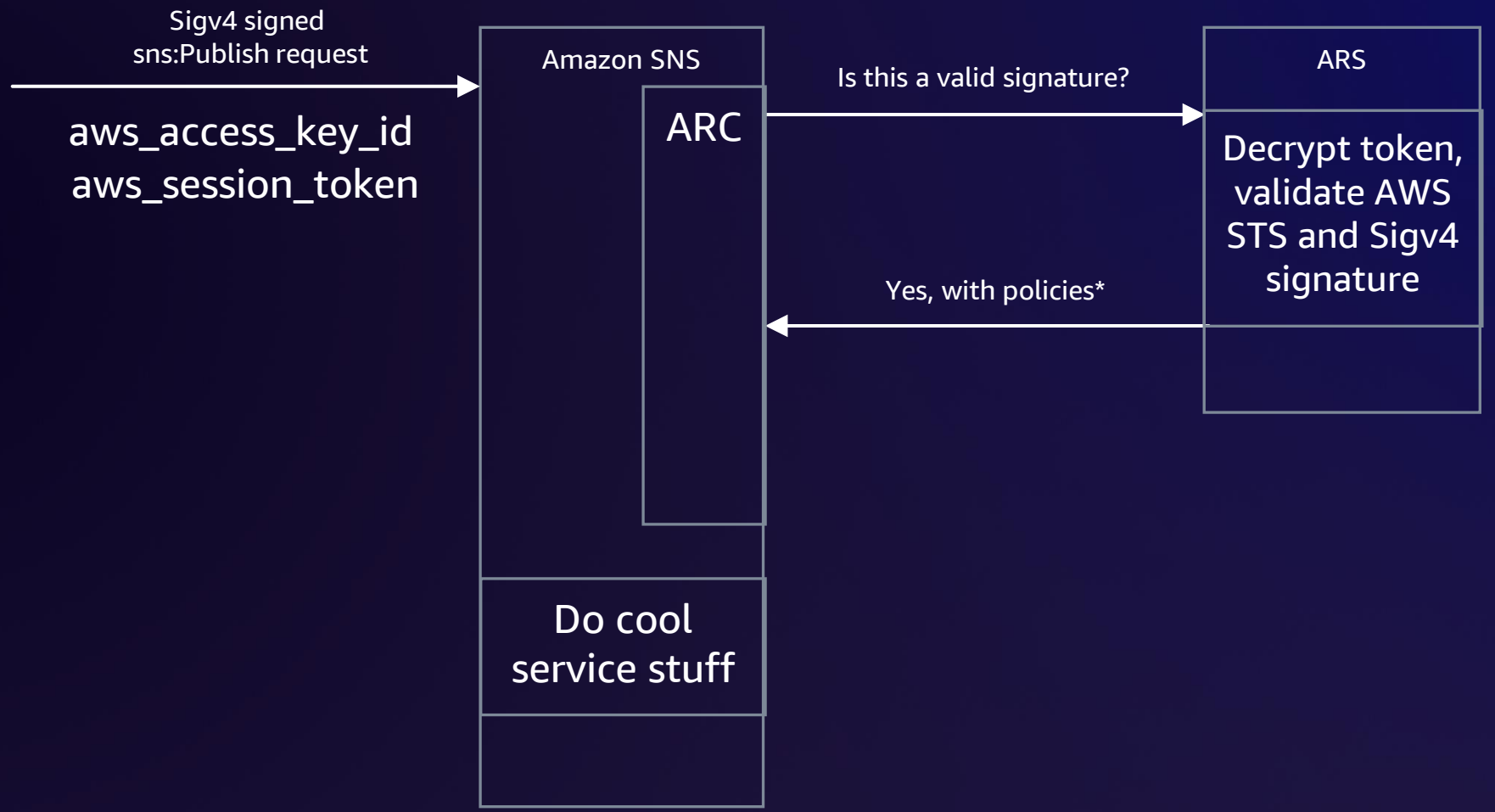
A request with temporary credentials



A request with temporary credentials

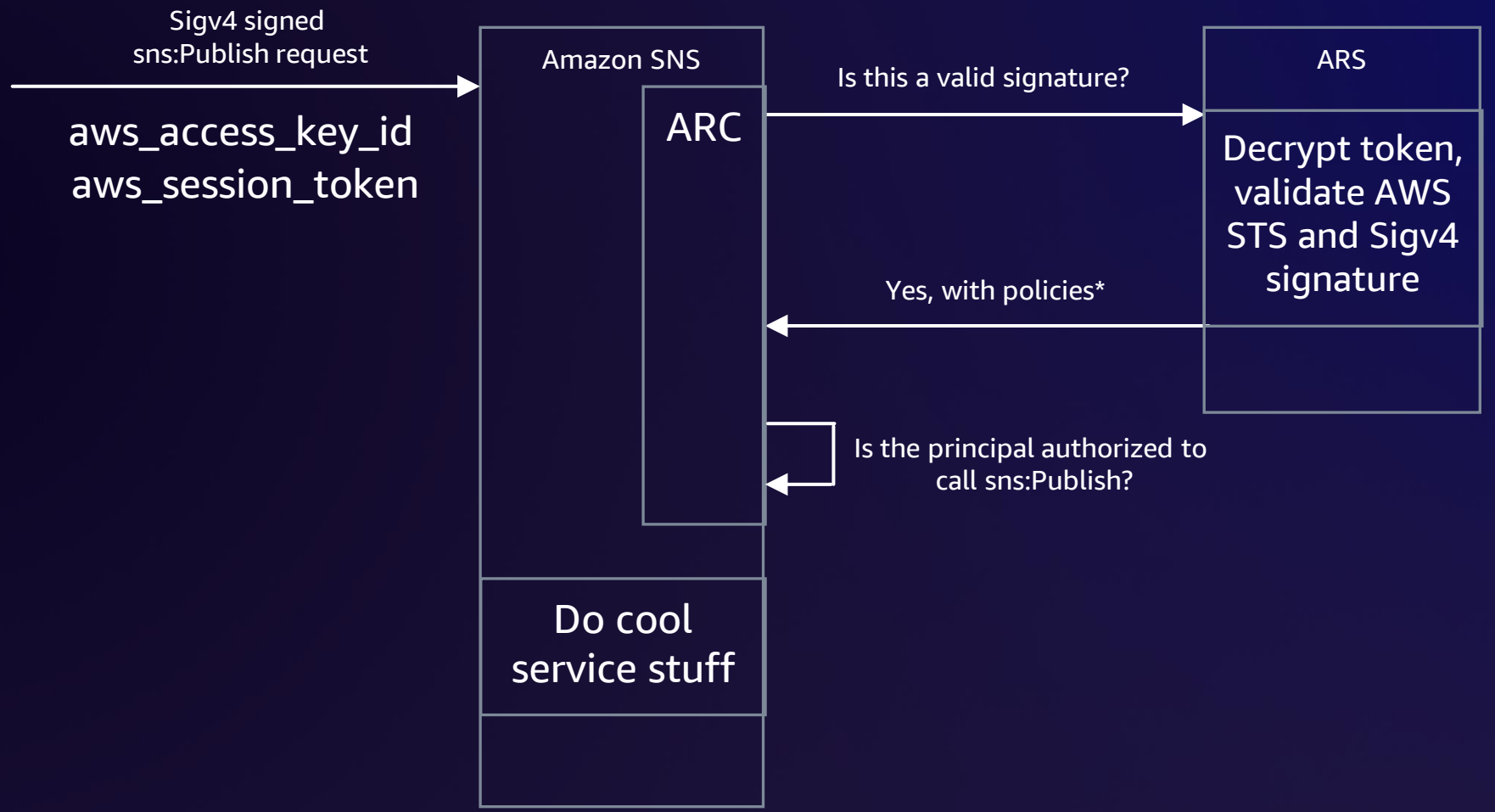


A request with temporary credentials



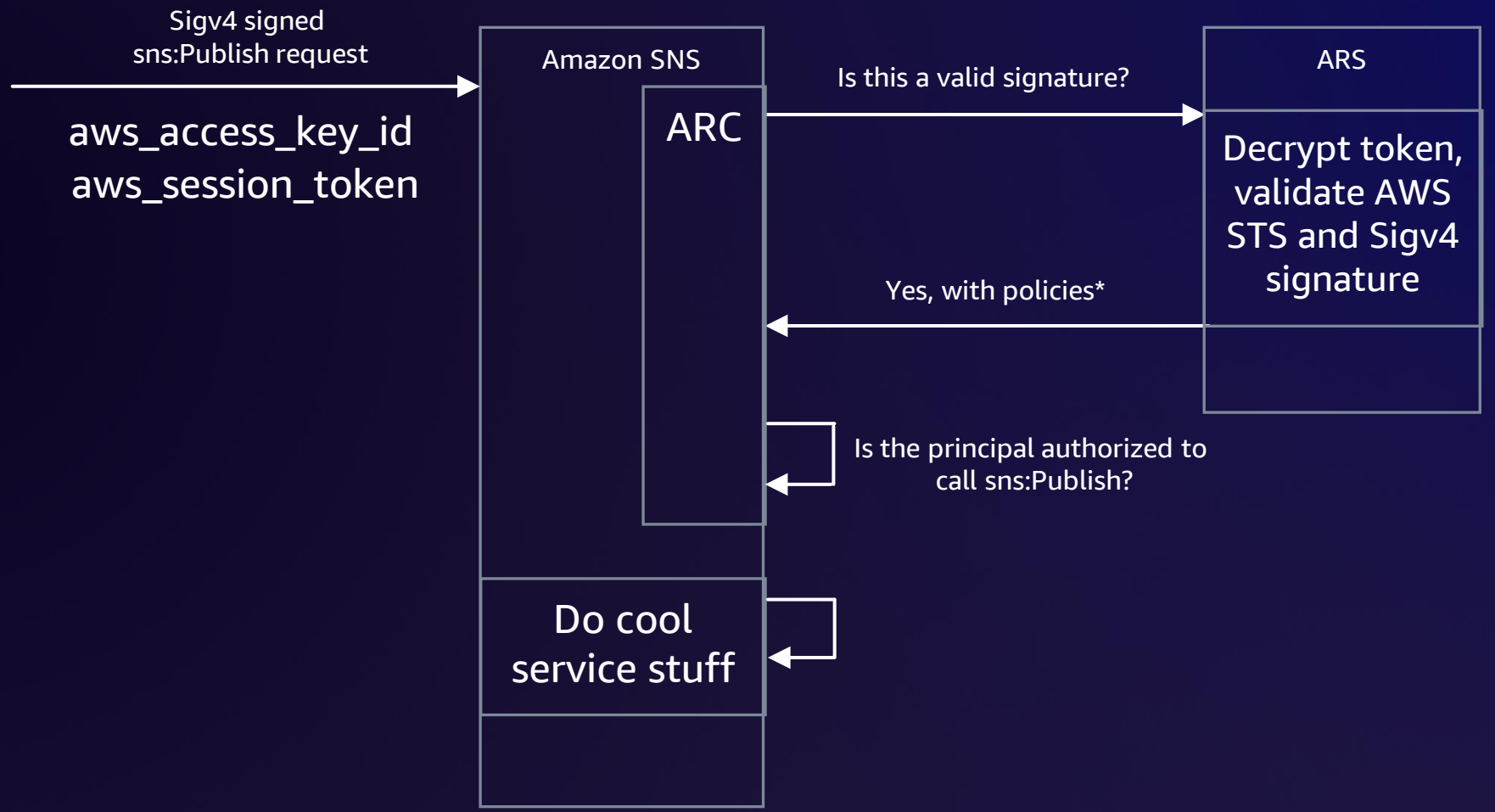
* Includes session policies

A request with temporary credentials



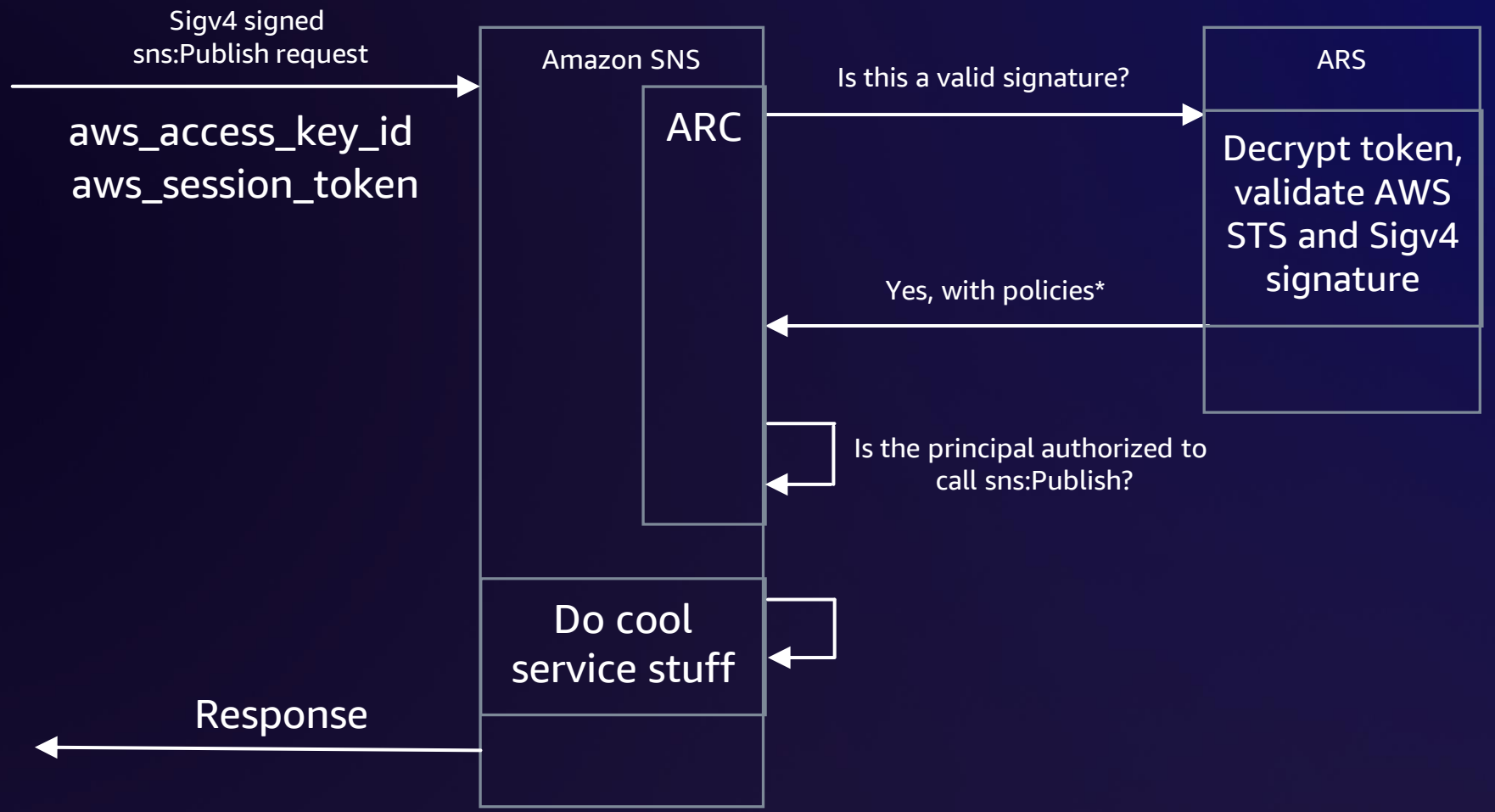
* Includes session policies

A request with temporary credentials



* Includes session policies

A request with temporary credentials



* Includes session policies

Testing with session policies

```
#!/bin/bash

export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_SESSION_TOKEN=

ROLE_ARN=arn:aws:iam::111111111111:role/RoleWithAdminPermissions

POLICY=$(cat session-policy-to-test.json)
SESSION_NAME="test"

eval $(aws sts assume-role --role-arn ${ROLE_ARN} --role-session-name ${SESSION_NAME} --policy "${POLICY}" | jq -r '.Credentials | "export AWS_ACCESS_KEY_ID=\(.AccessKeyId)\nexport AWS_SECRET_ACCESS_KEY=\(.SecretAccessKey)\nexport AWS_SESSION_TOKEN=\(.SessionToken)\n"')

aws sts get-caller-identity
```

Reset env variables

Role to assume

Specify policy to test

Assume role and set env variables

Session policies – packed policy size

- Hard limit on the size of a session token

Session policies – packed policy size

- Hard limit on the size of a session token
- The token is compressed – the amount of characters you can use in a policy or number of tags is unpredictable

Session policies – packed policy size

- Hard limit on the size of a session token
- The token is compressed – the amount of characters you can use in a policy or number of tags is unpredictable
- Includes session policies, session tags, and other session context

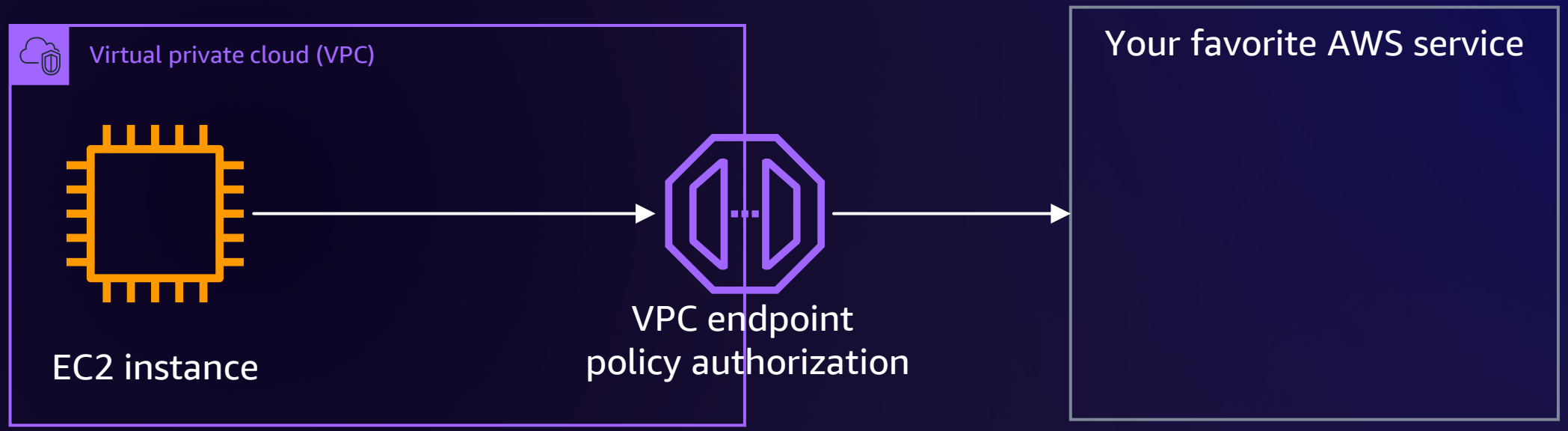
Session policies – packed policy size

- Hard limit on the size of a session token
- The token is compressed – the amount of characters you can use in a policy or number of tags is unpredictable
- Includes session policies, session tags, and other session context
- Think about what's unique to the session and only inject that into the token

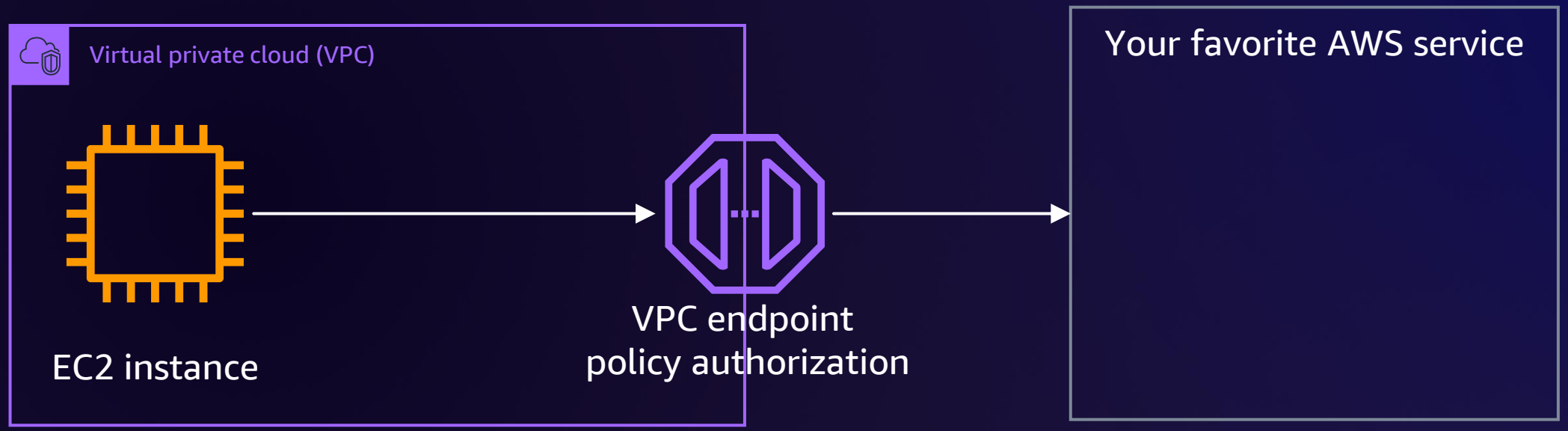
Session policies takeaways

- The policy and relevant context are embedded in the session token
- Most of the time you use the default policy
- Useful for token brokers, when a session needs unique permissions

VPC endpoint policies – logical view



VPC endpoint policies – physical view



VPC endpoint policies – physical view

VPC endpoint policies takeaways

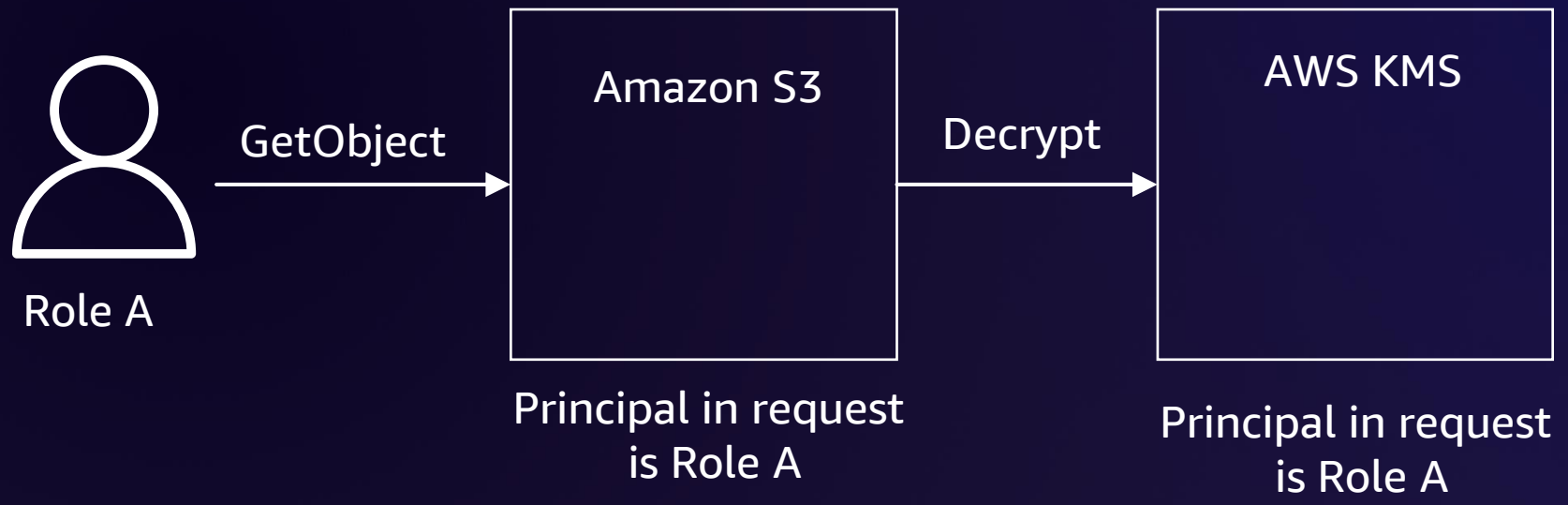
- They are the only place to prevent untrusted identities from making requests to untrusted resources

- Allow you to use the aws:SourceVpc condition key in resource and identity-based policies

FAS policies – What is FAS?

Forward access sessions (FAS) is an IAM technology that passes your identity, permissions, and session attributes when an AWS service makes a request on your behalf

FAS policies – What is FAS?



FAS policies – What is FAS?

ARS

I want to get an encrypted object

Role A

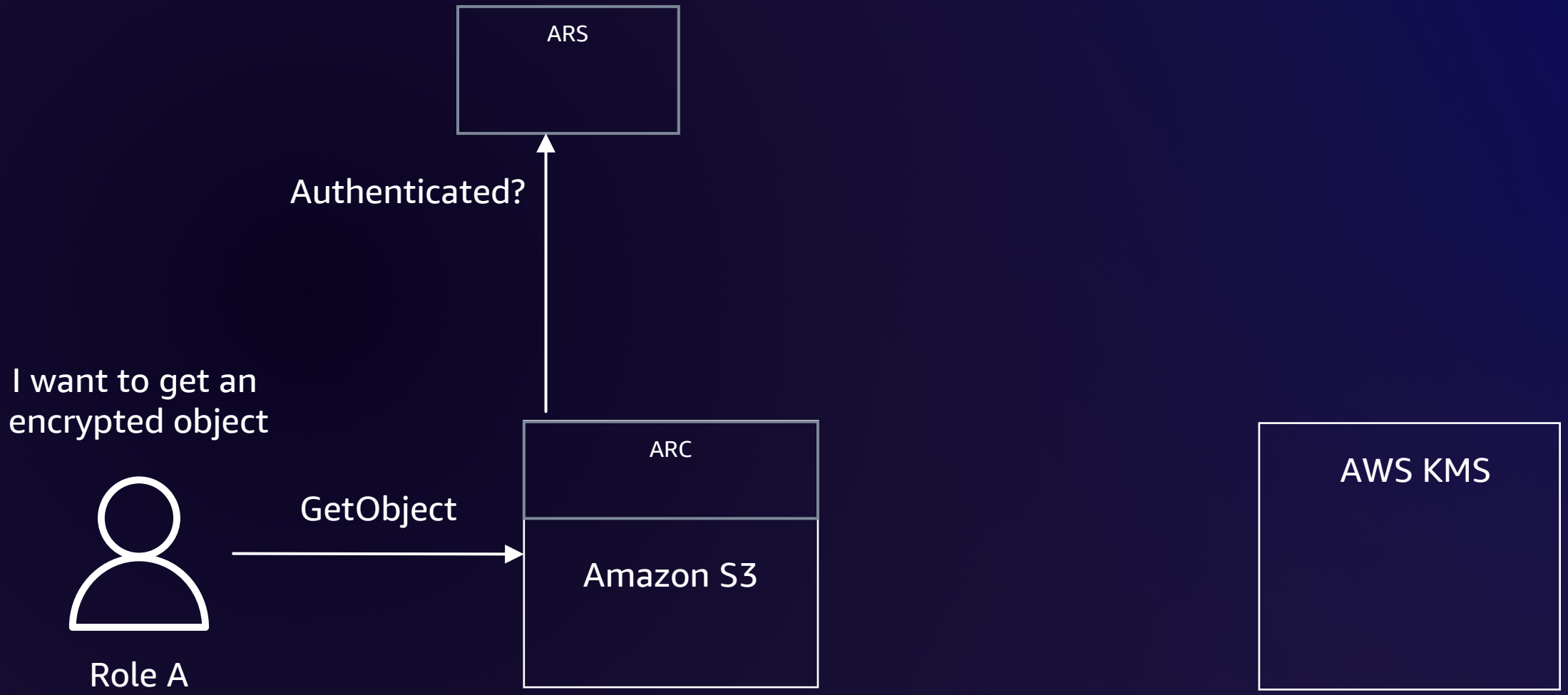
ARC

Amazon S3

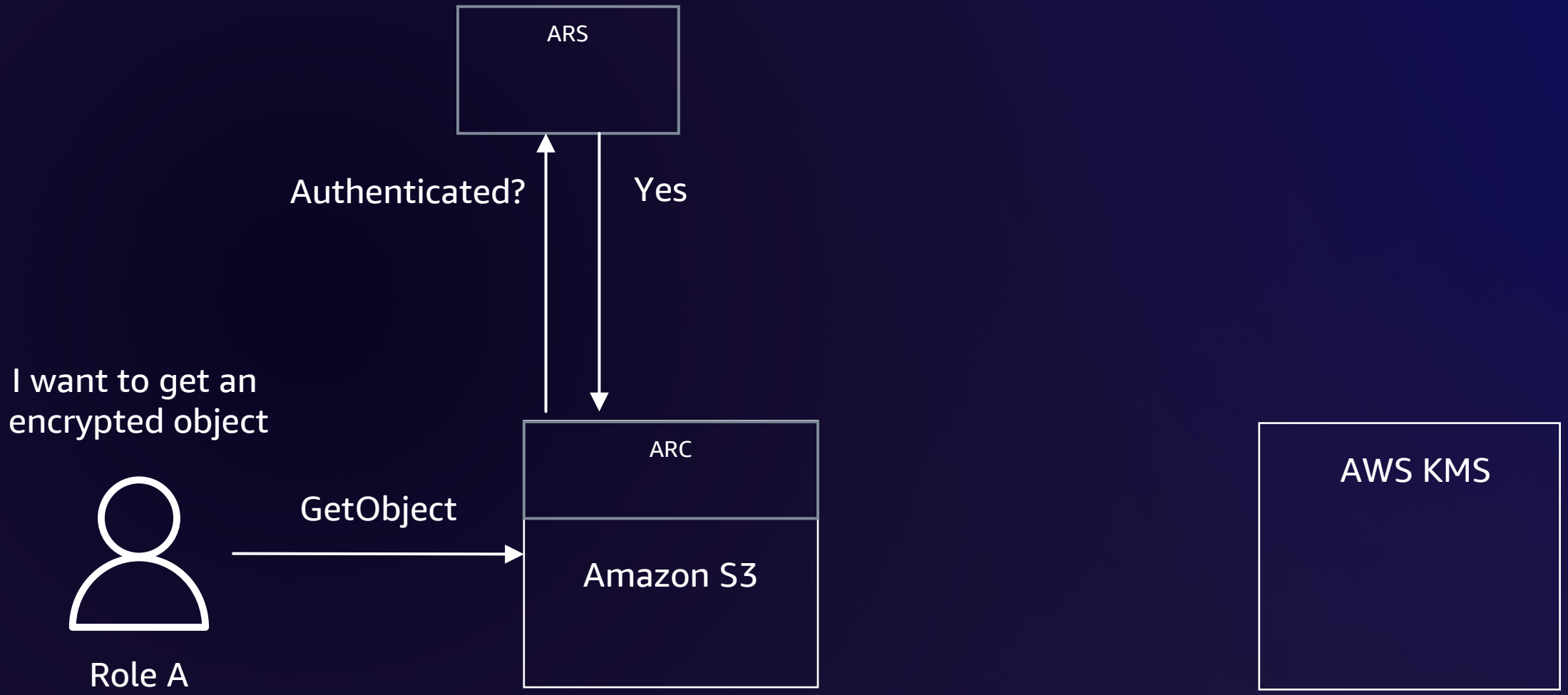
AWS KMS

FAS policies – What is FAS?

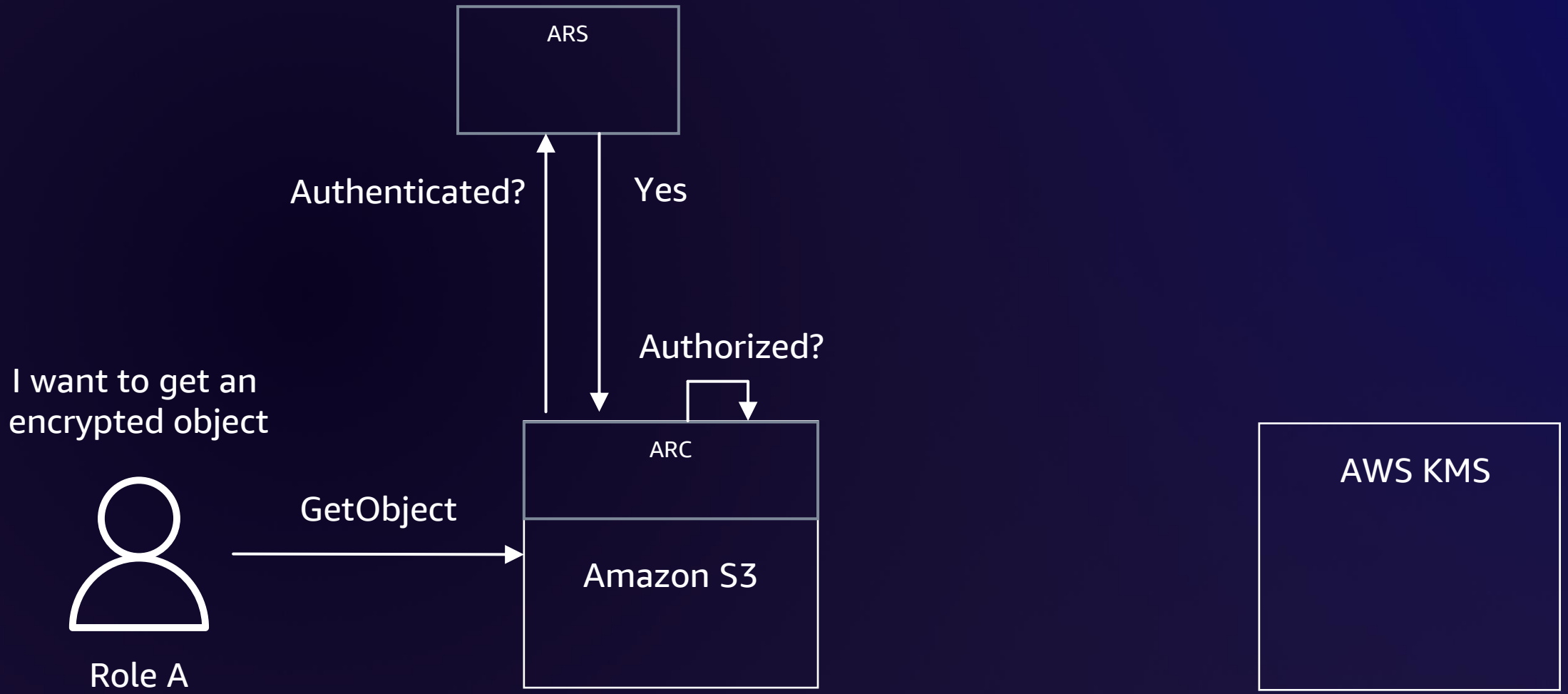
FAS policies – What is FAS?



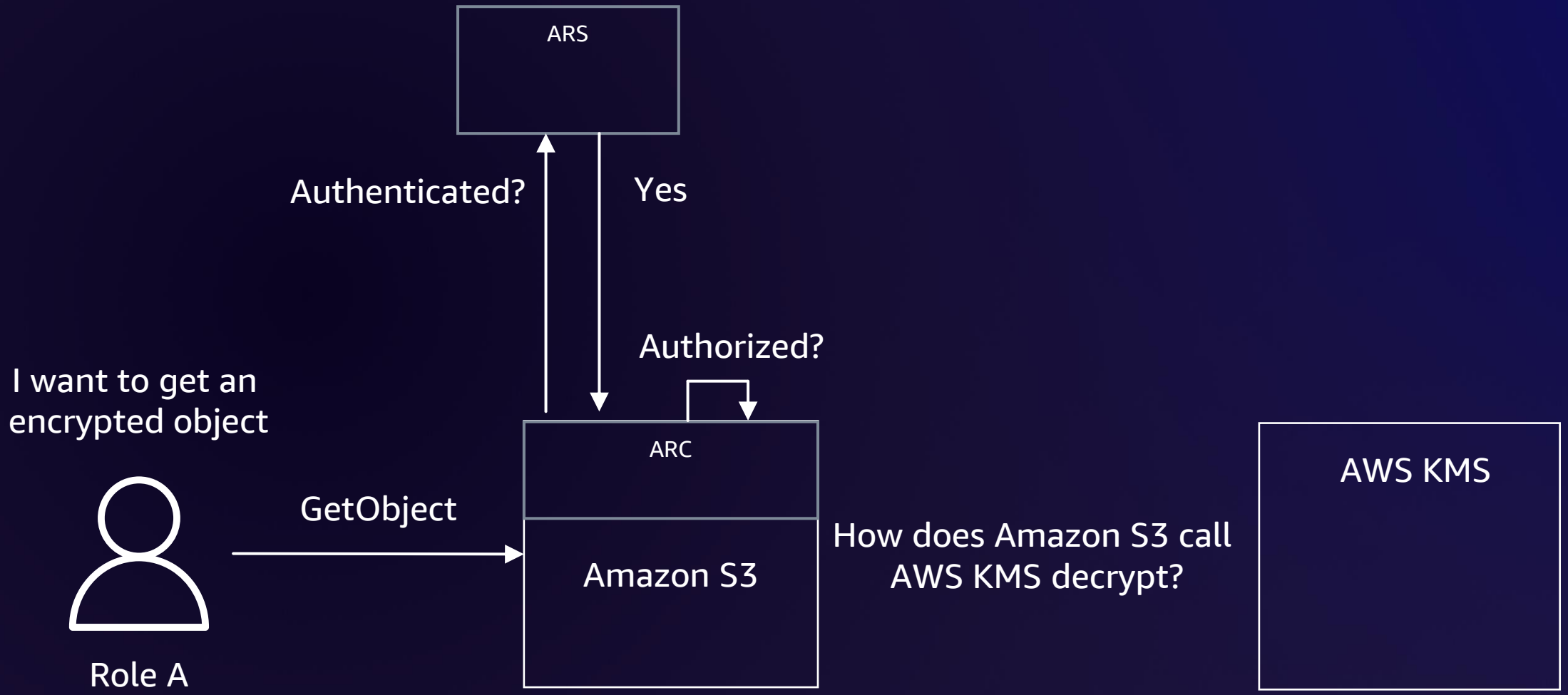
FAS policies – What is FAS?



FAS policies – What is FAS?



FAS policies – What is FAS?



FAS policies – What is FAS?

ARS

I want to get an encrypted object

Role A

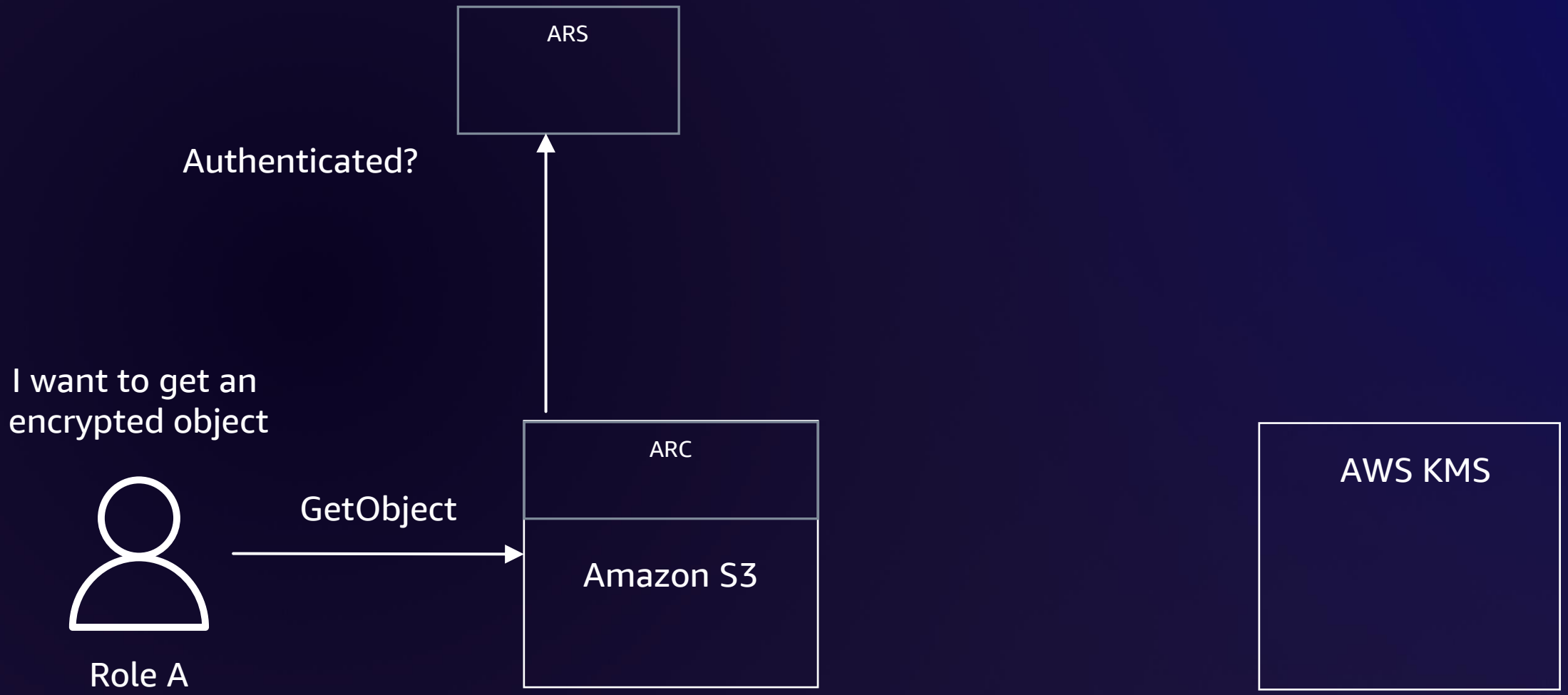
ARC

Amazon S3

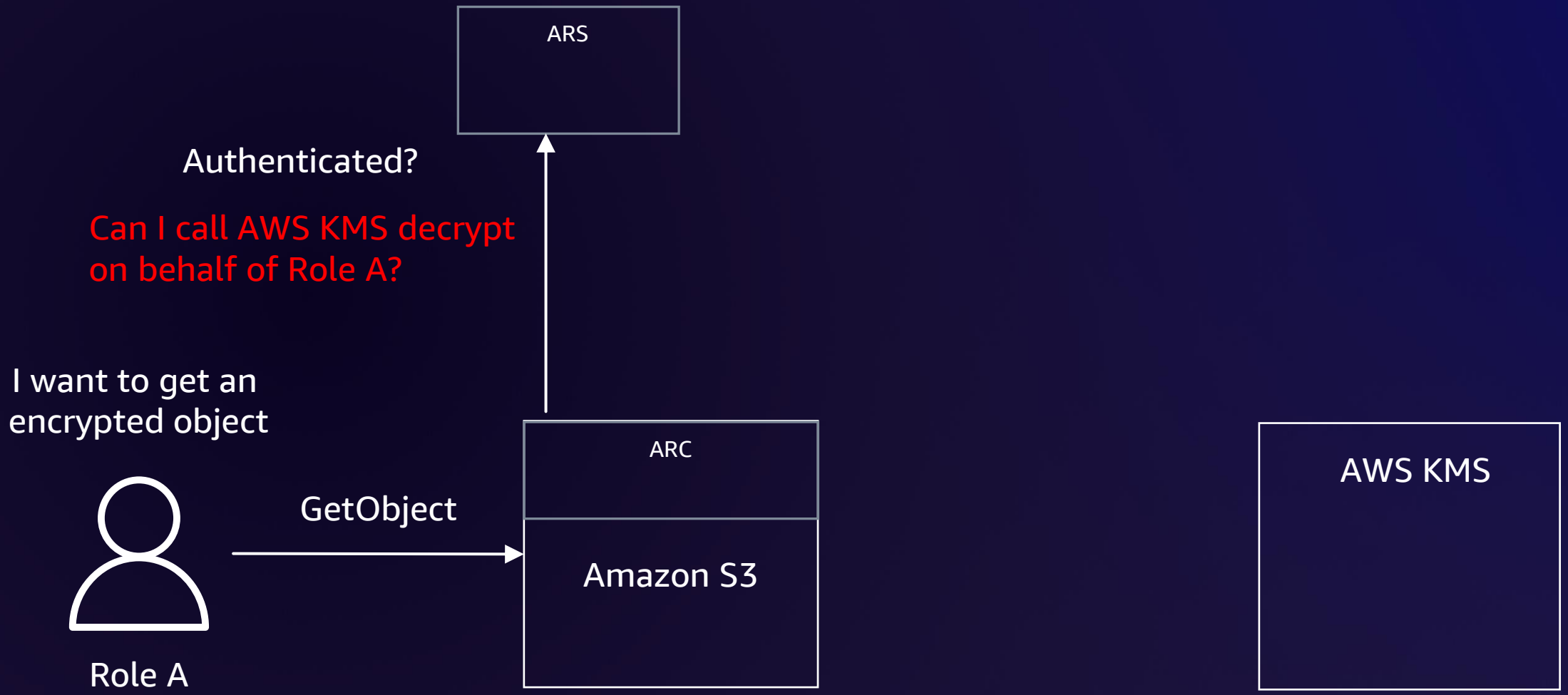
AWS KMS

FAS policies – What is FAS?

FAS policies – What is FAS?

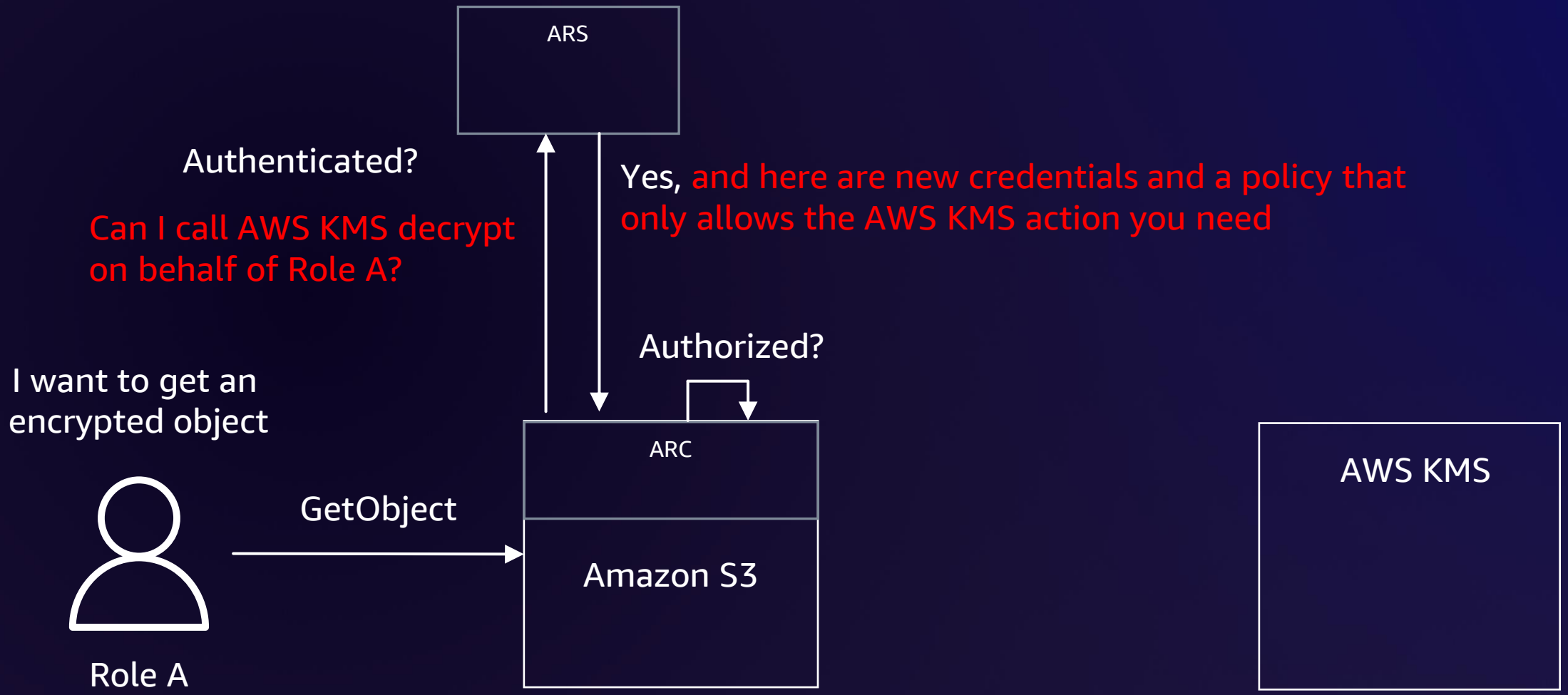


FAS policies – What is FAS?

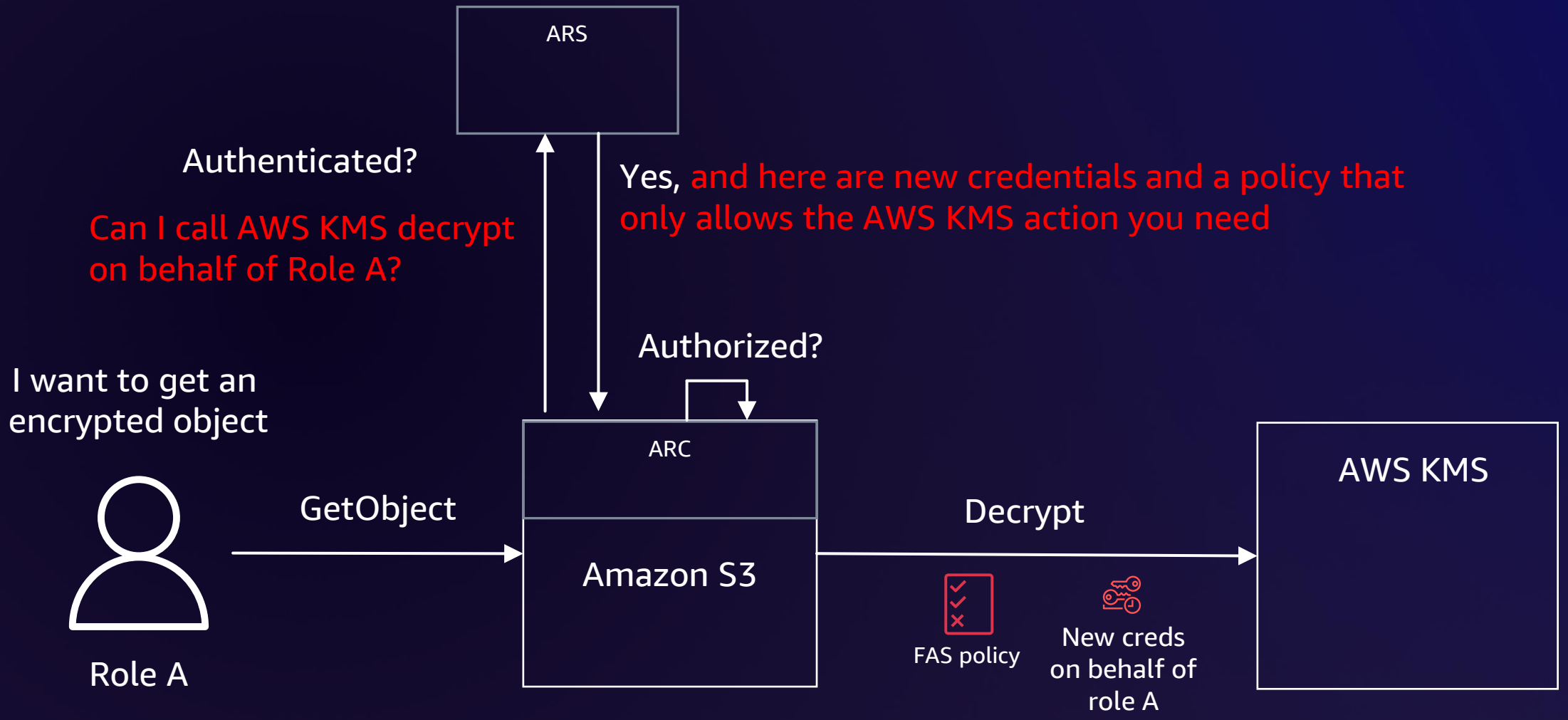


FAS policies – What is FAS?

FAS policies – What is FAS?



FAS policies – What is FAS?



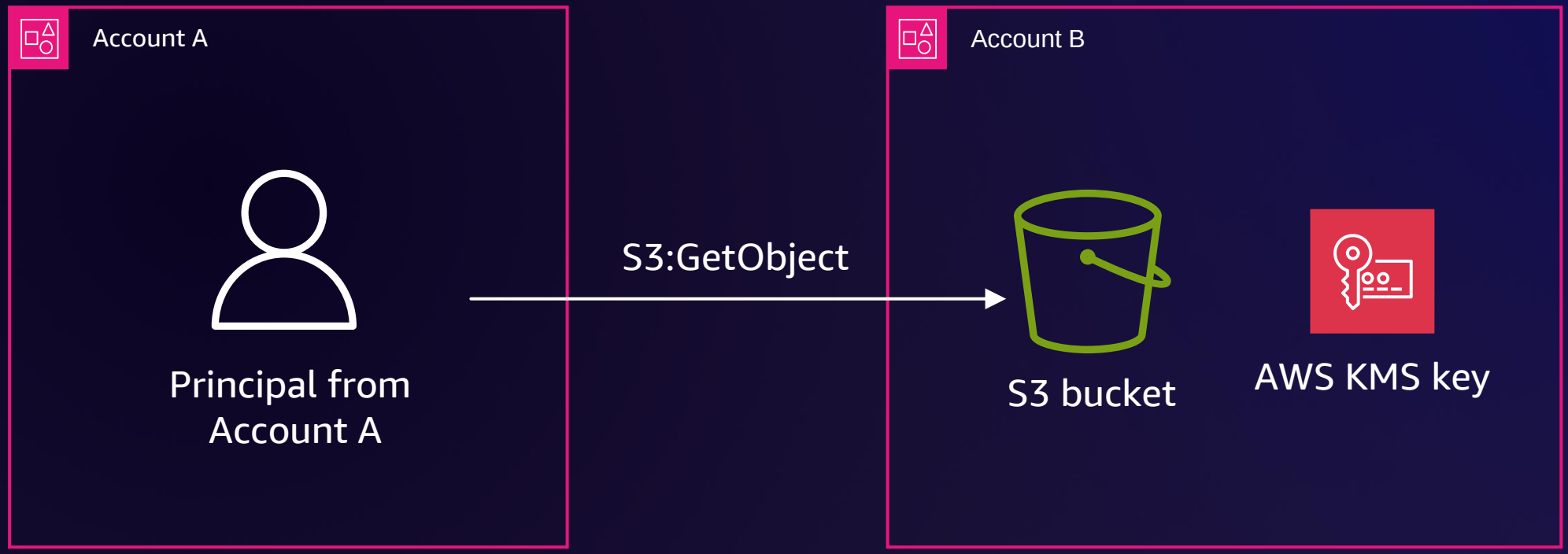
FAS policies - Takeaways

- FAS provides AWS services a scoped-down version of the calling principal’s permissions
  - Blast radius reduction
  - Avoidance of credential replay

- The aws:CalledVia and aws:ViaAWSService condition keys are set when a FAS call is made

Putting it all together

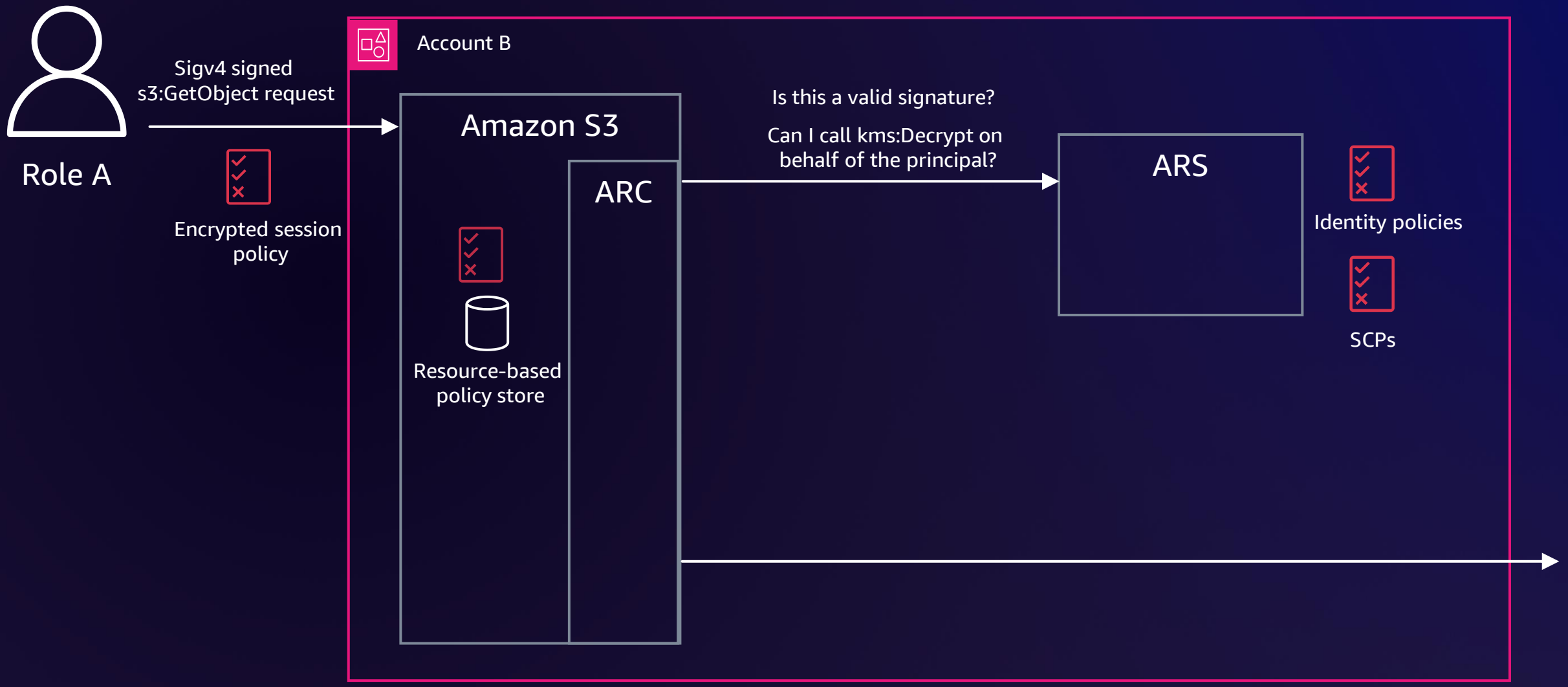
A sample cross-account request



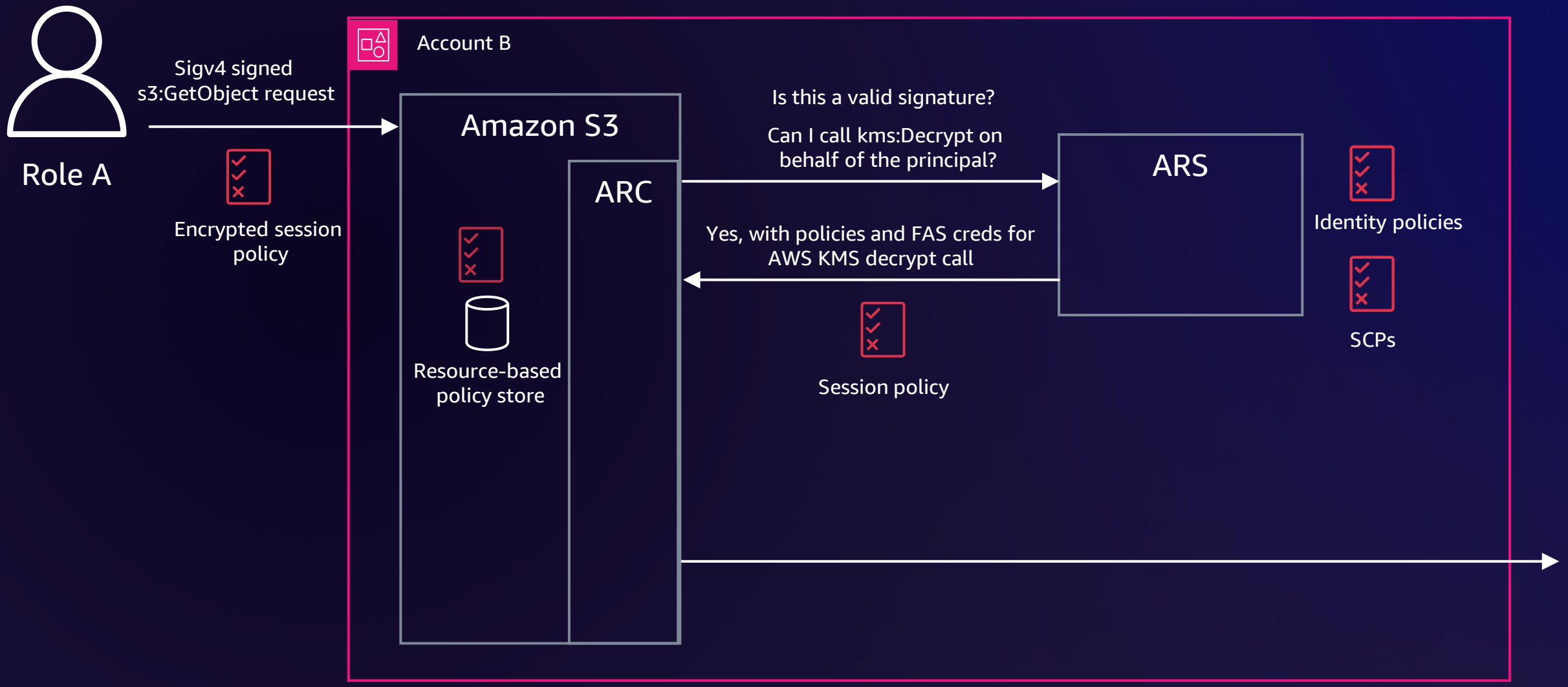
A sample cross-account request

A sample cross-account request

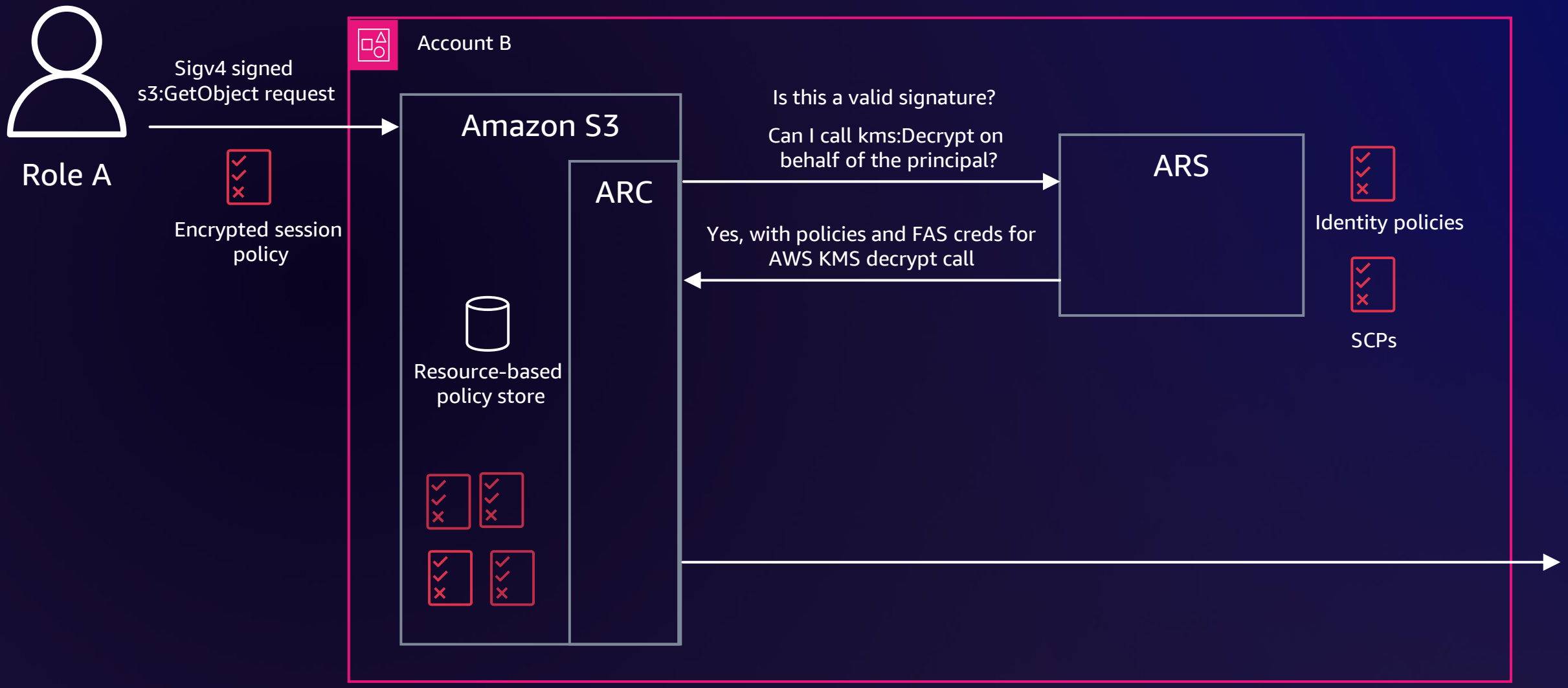
A sample cross-account request



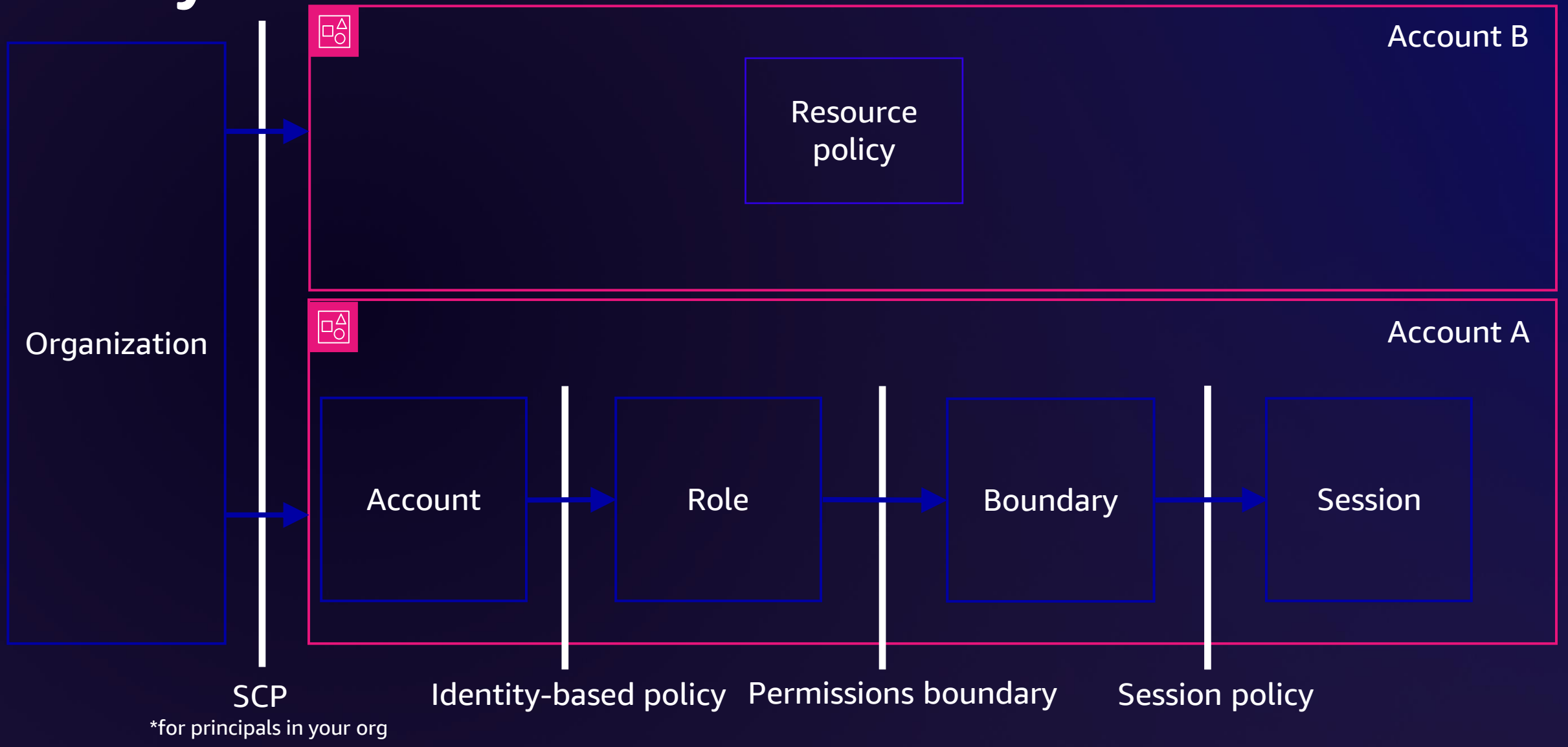
A sample cross-account request



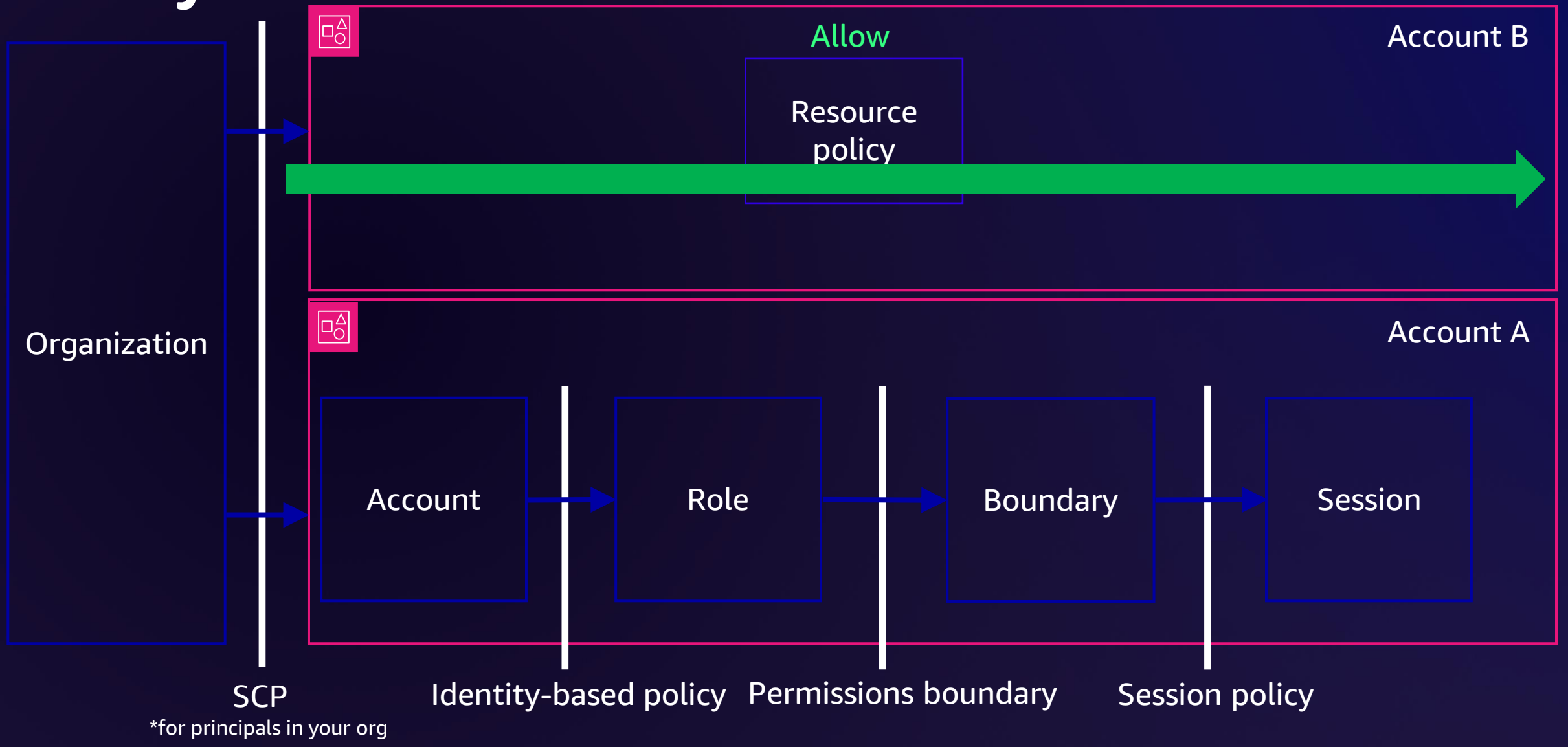
A sample cross-account request



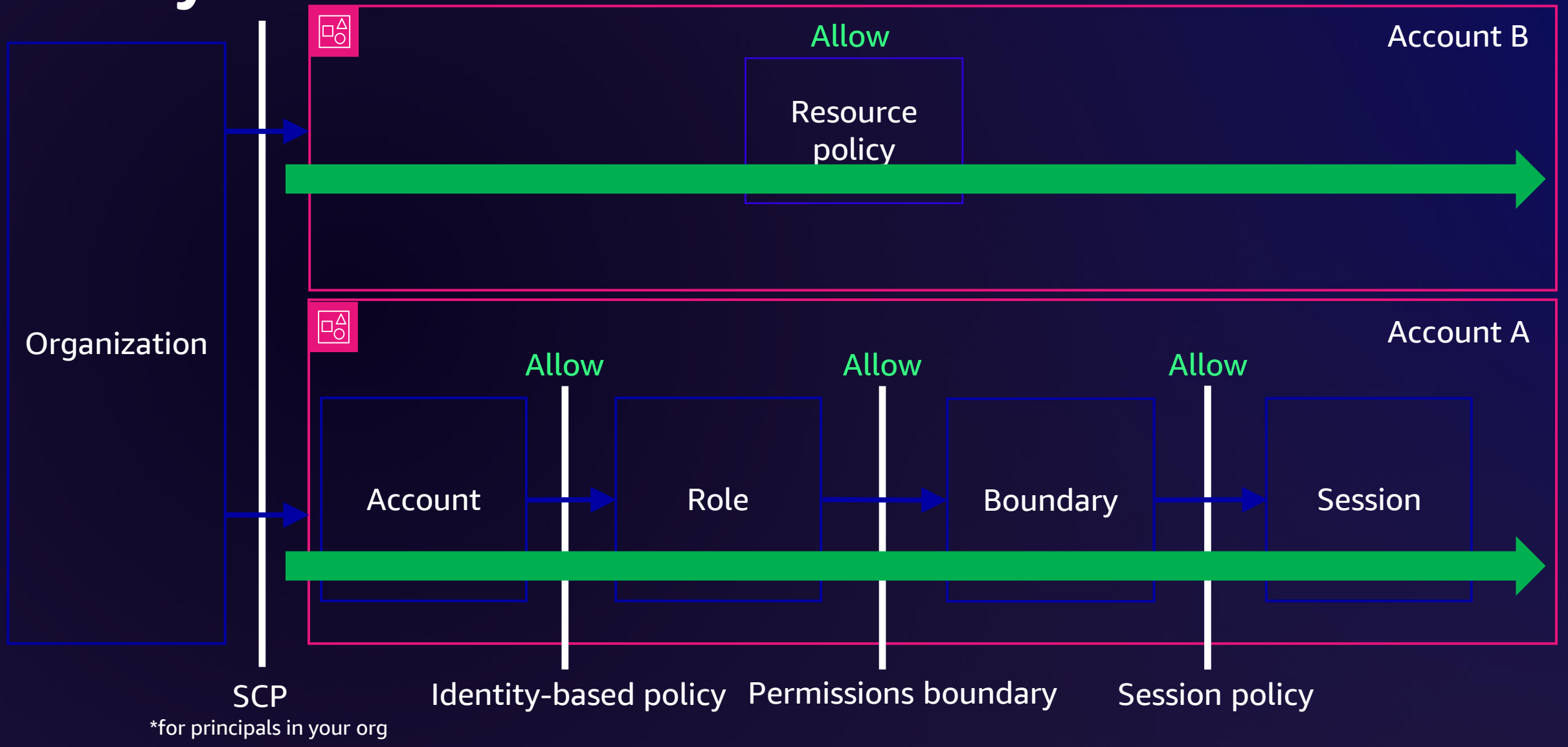
Policy evaluation: Cross-account



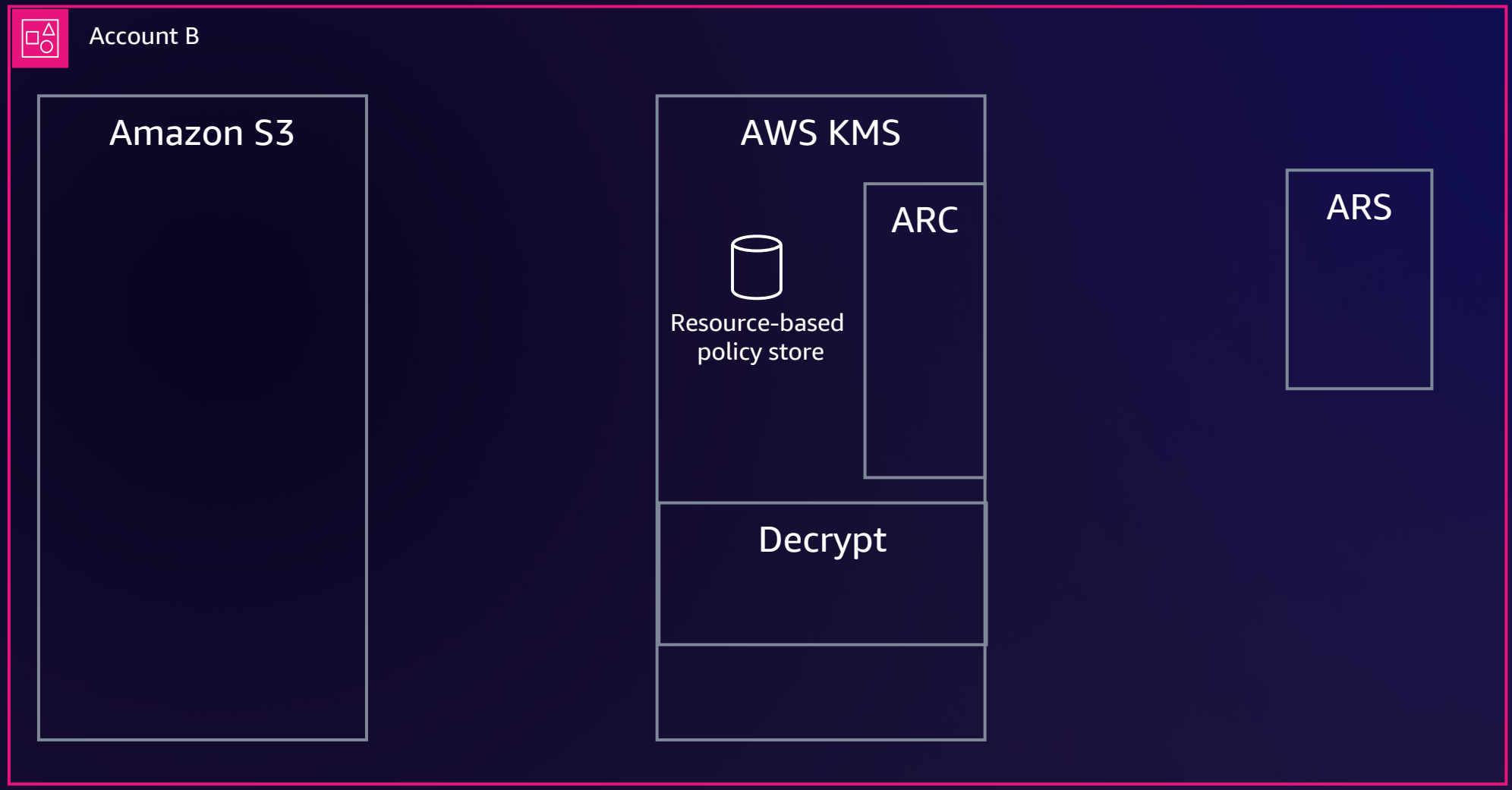
Policy evaluation: Cross-account



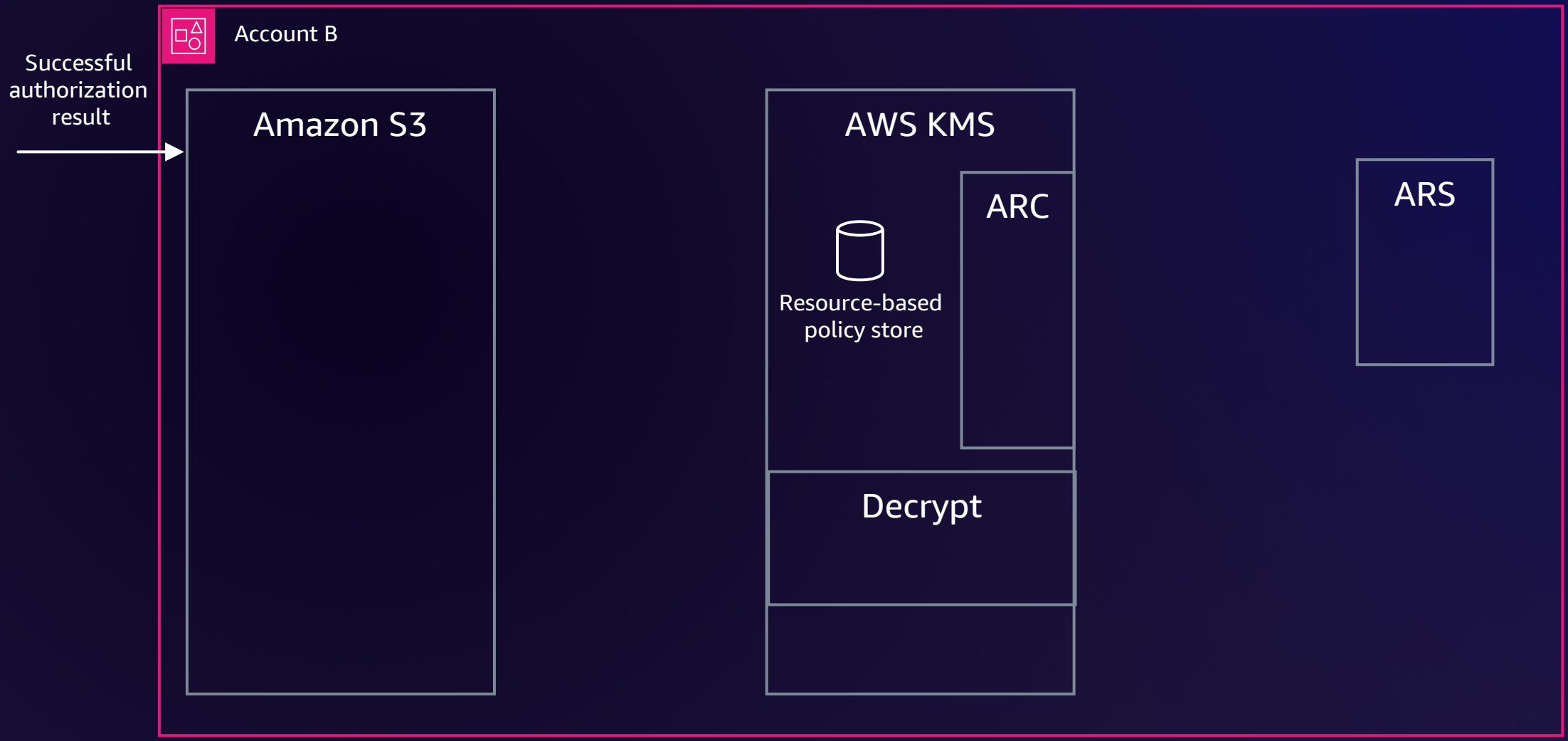
Policy evaluation: Cross-account



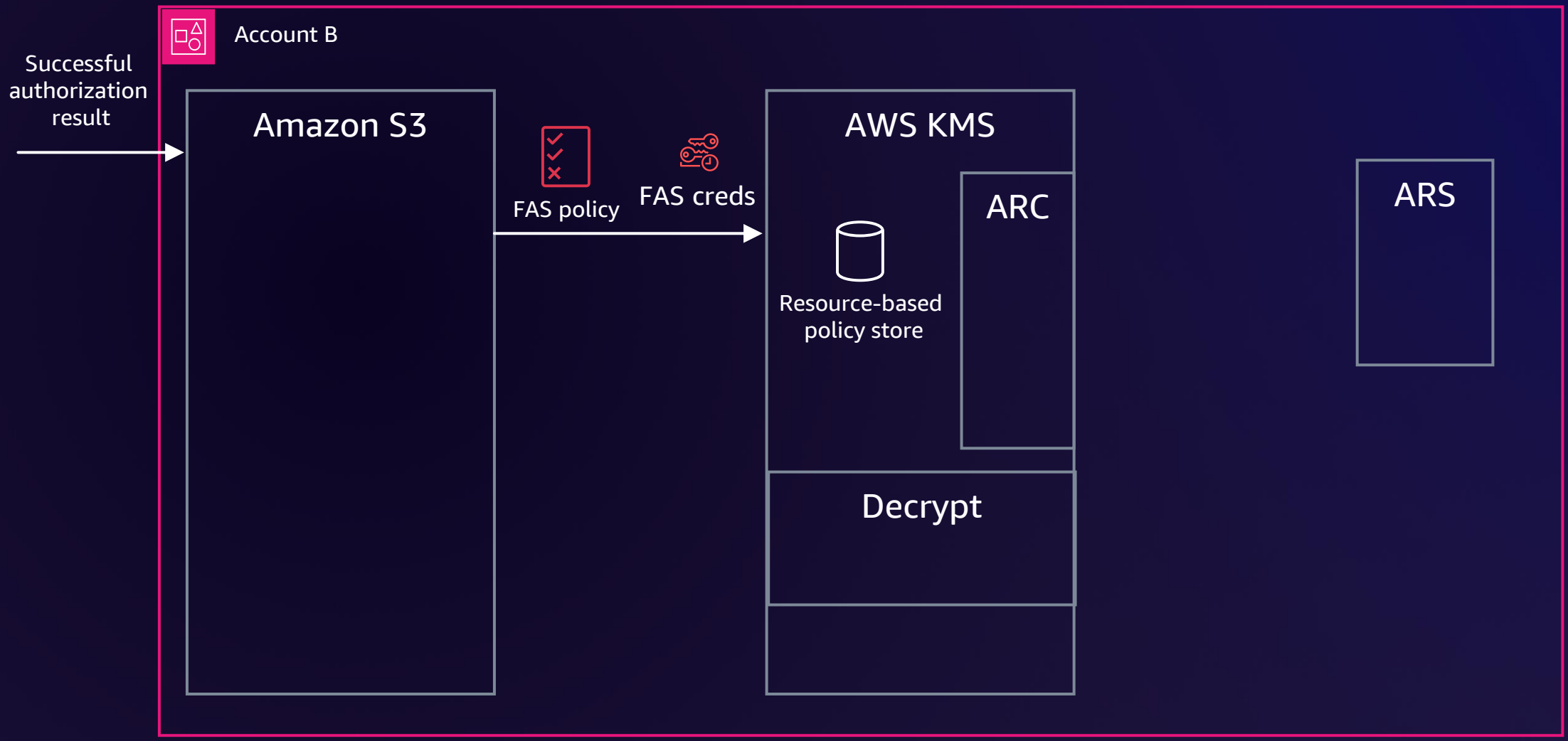
A sample cross-account request



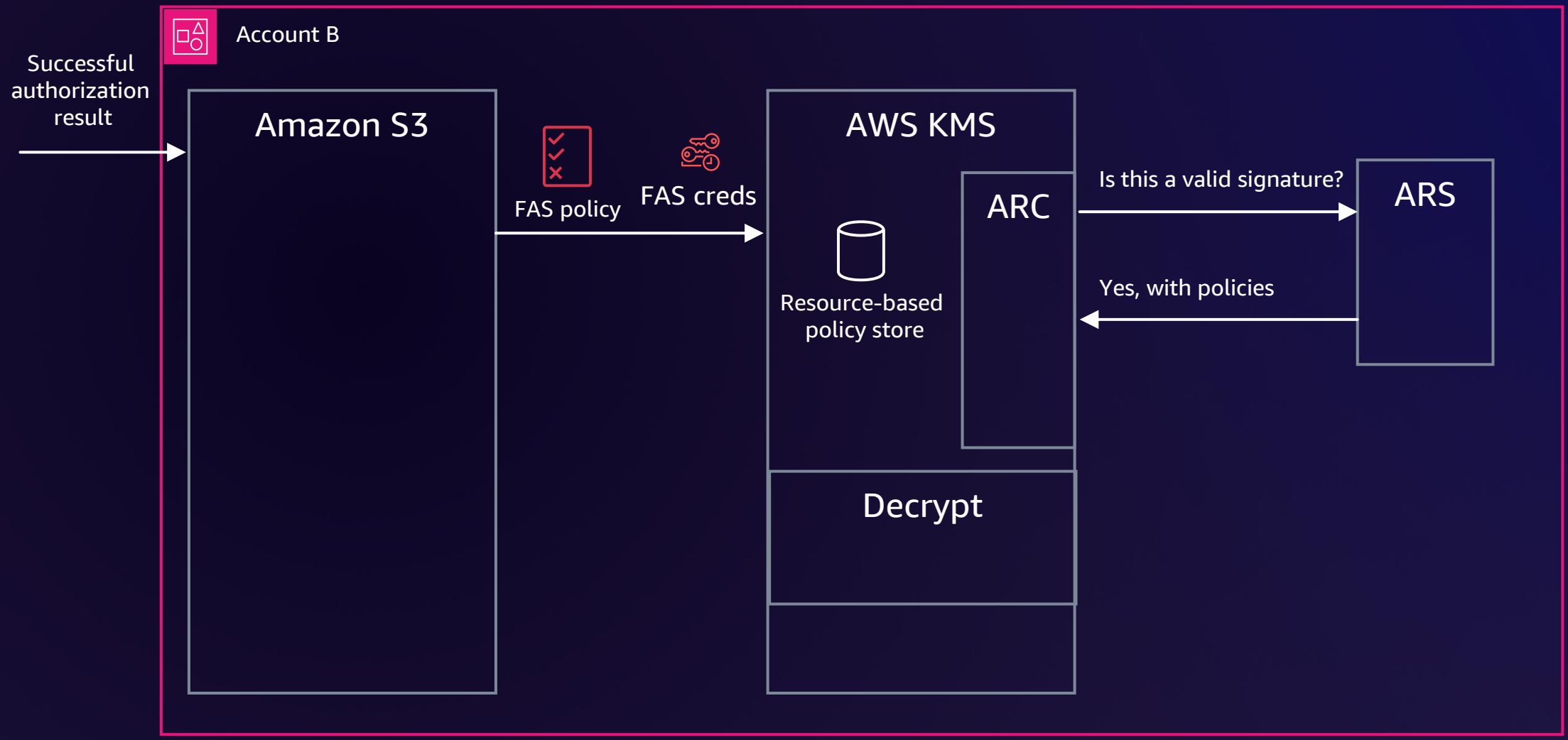
A sample cross-account request



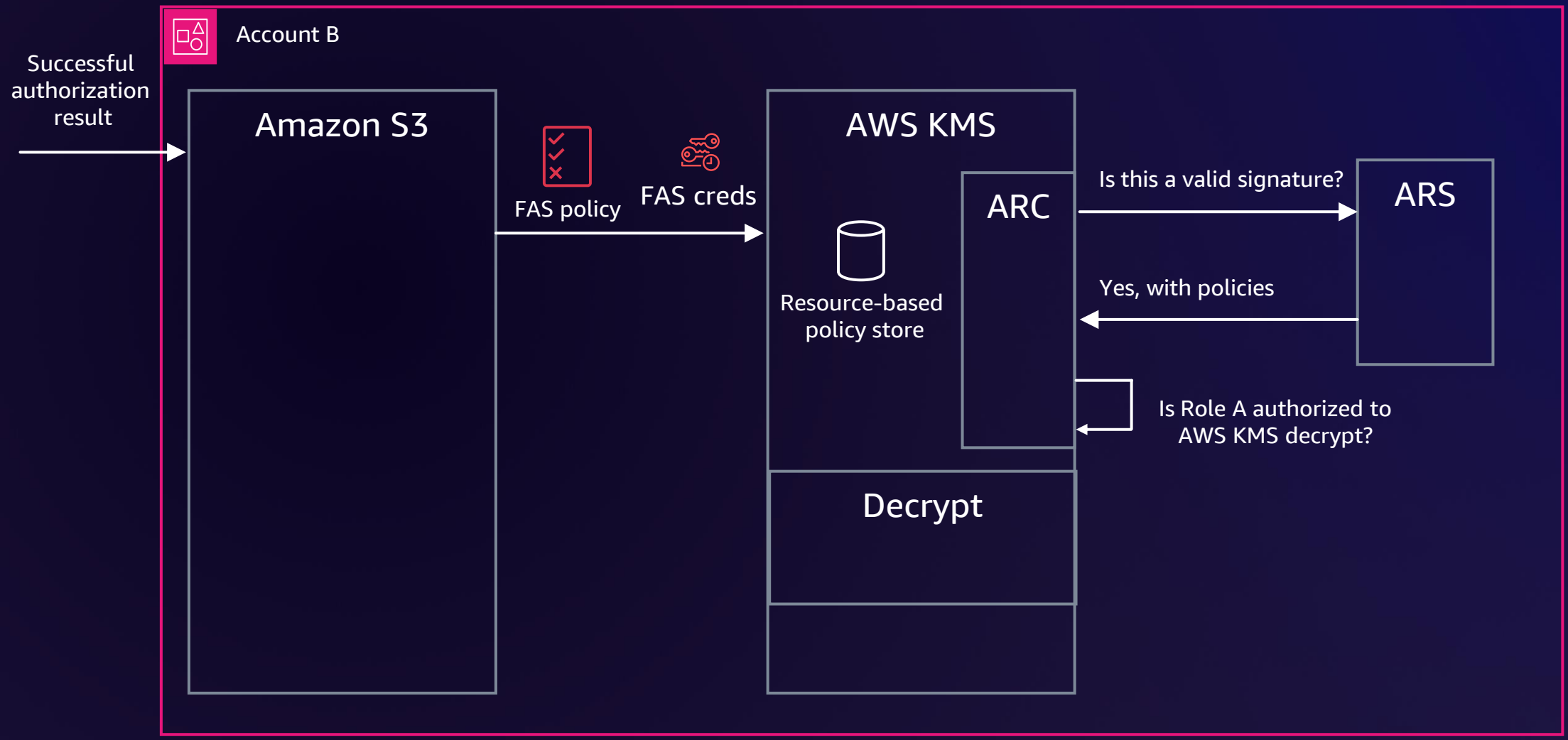
A sample cross-account request



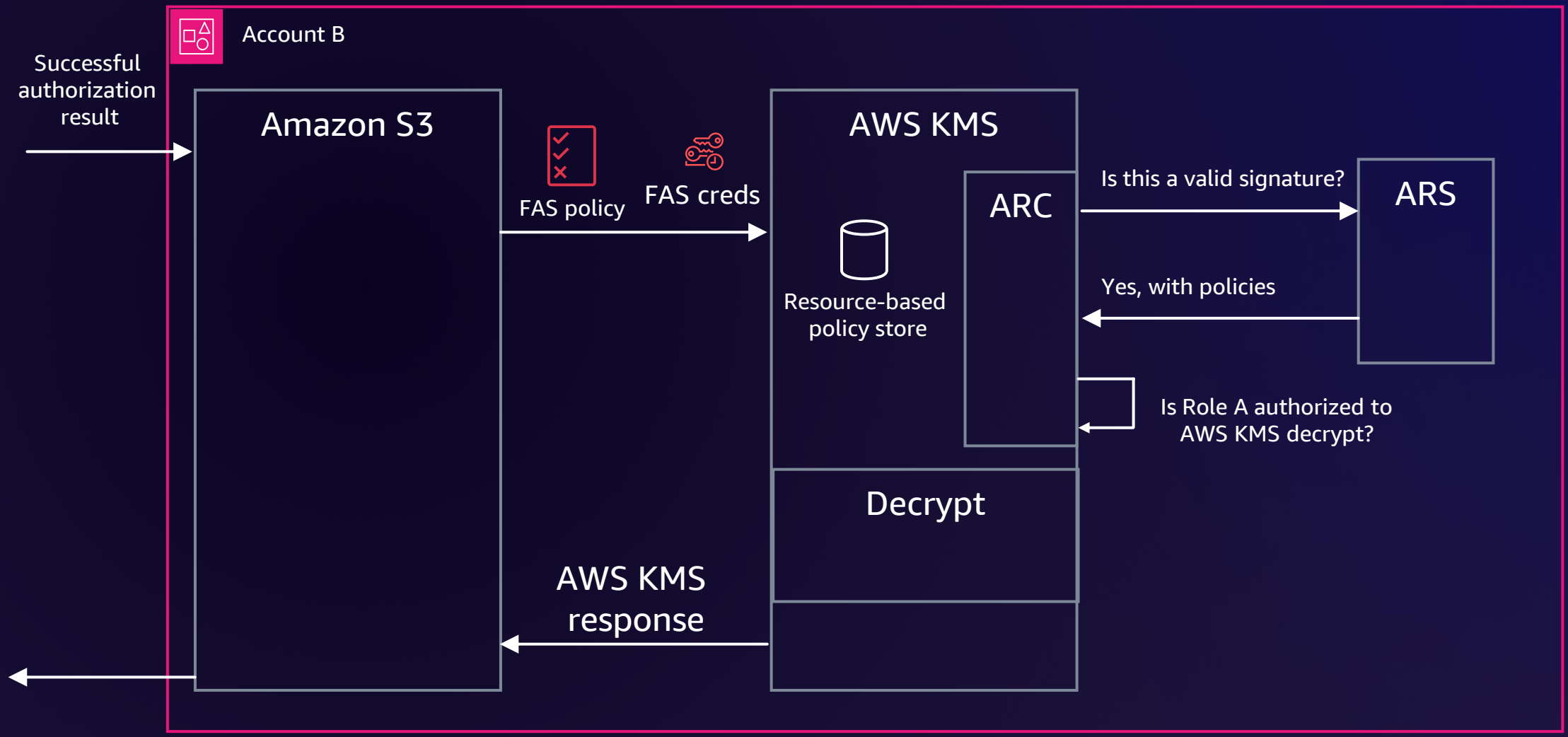
A sample cross-account request



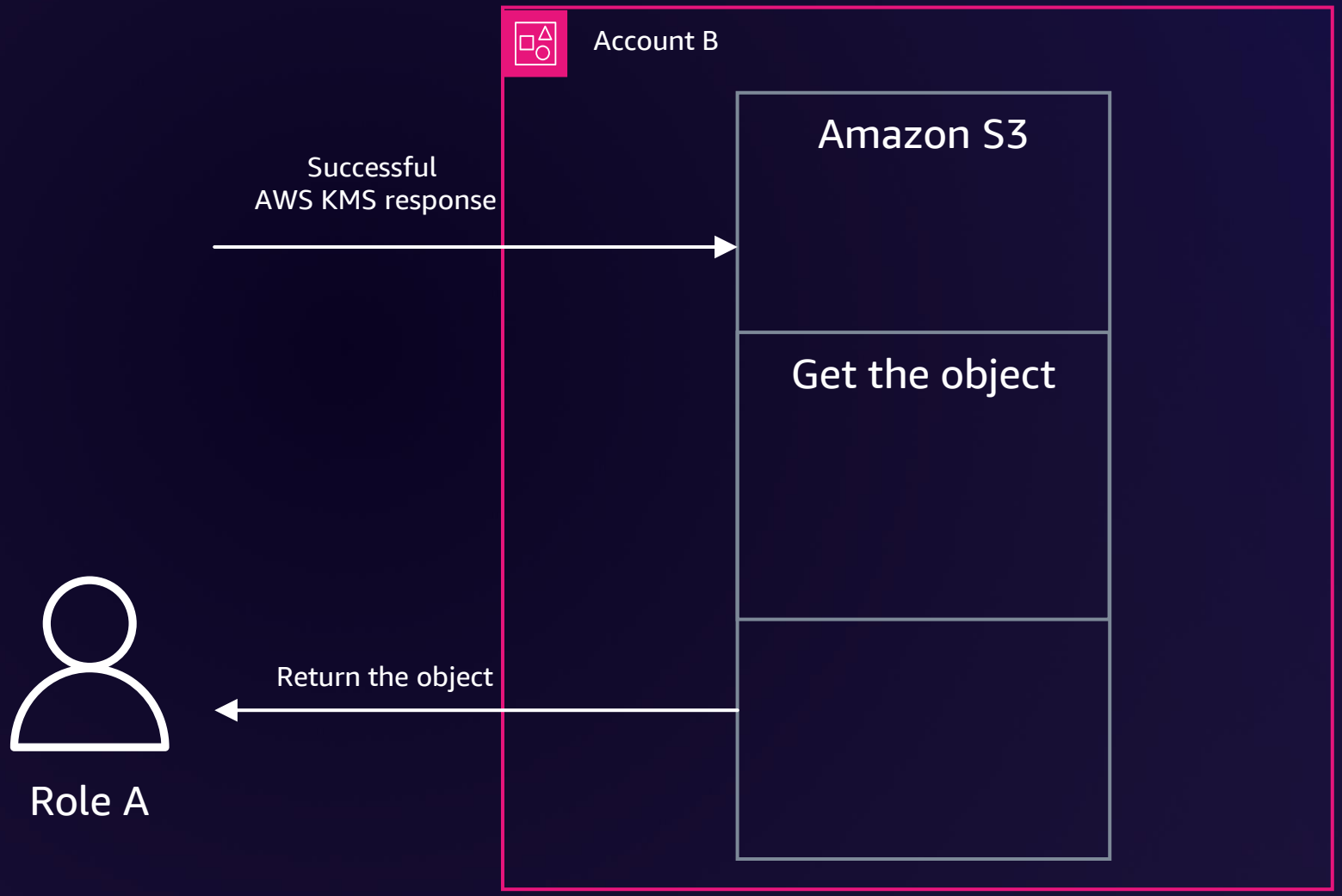
A sample cross-account request



A sample cross-account request

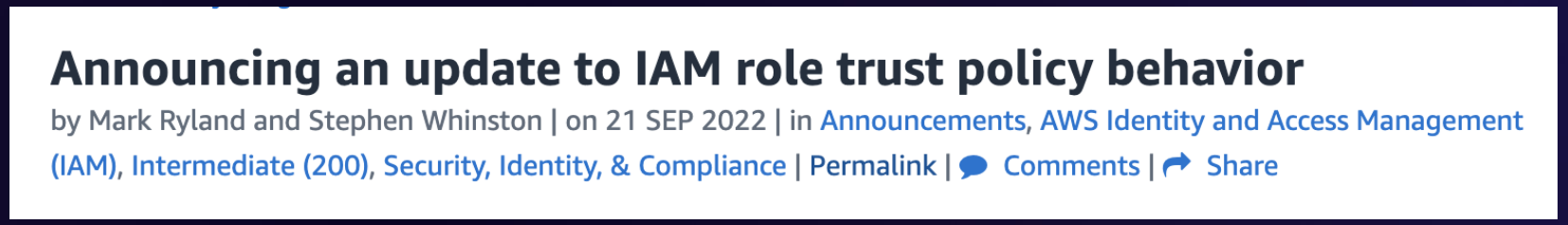


A sample request



What if we need to change authorization?

- Rare but does happen, e.g.:



  - STS decided to remove implicit self-trust from AssumeRole
  - Analysis of implications and discussions
  - Cutoff dates
  - Customer messaging

What if we need to change authorization?

- Many possible reasons!
- High bar to make the change
  - Analysis of historical request patterns
  - Background analysis of impact against ongoing requests
  - Zelkova or other static analysis where relevant
  - Allowlisting of impacted customers and reach-out

Summary

- IAM is eventually consistent for good reason
- Blast radius is a core design consideration
  - Sigv4
  - FAS
- ARC for authorization, ARS for authentication