Demystifying Identity @ AWS

Amit Jha , Developer Advocate

@amitkjha_rjn

Agenda

Customer – Migration – Modernization

Basics of Auth-Auth

Identity Basics on AWS

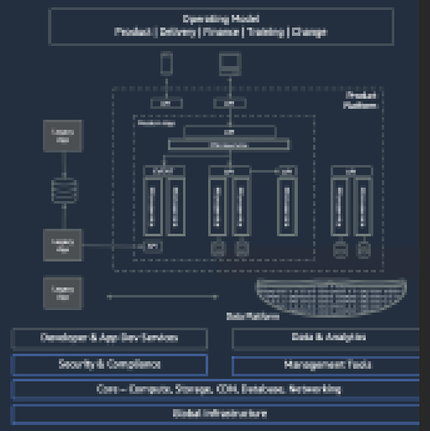
Federation/SAML/oAuth

Users/Custom Applications Use cases

Q&A

What are the key characteristics of successful customer migration-modernization look like?

Cloud Platform & Services



Make underline infrastructure a self-service mechanism and apply best practices for resiliency, availability, performance, security, cost

Automate for updates, upgrades, provisioning, scaling

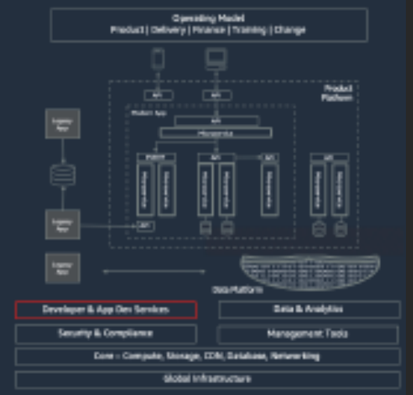
Everything is Code following SDLC best practices for infrastructure, configuration, settings

Infrastructure Automation
…freeing-up developers to focus on business value

Capital One

Reduce their data center footprint by 60% while enabling 10k+ developers (internal + external) globally

Developer-first Workflow



Consistent, reliable experience
with tools and frameworks

Standardization and self service
for productivity gains

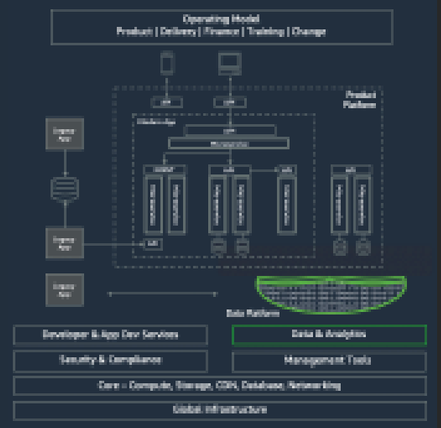
Focus on building newer
functionality over ongoing one
off code promotion heroics

Builder Experience
…accelerating time
from idea to production code

The Washington Post

Ability to release over 50+ deployments
per hour

Self-service Data Strategy



Data democratization for various self service use

Data portability and interoperability for feeding data to multi-source, multi-dimensional analysis

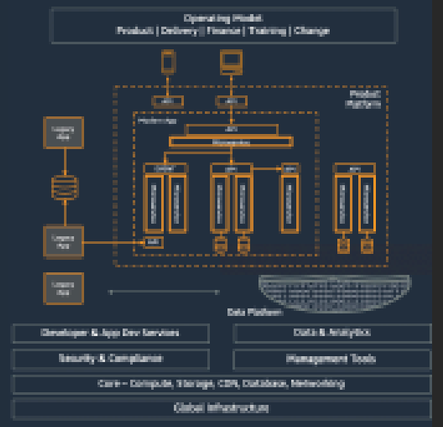
Intelligent insights with AI/ML for preventive and predictive purposes

Ubiquitous Access to Data …making intelligence assets easily consumable

FINRA

Ingest 75B+ financial records daily providing access to 1000x more market events, while decreasing query time from mins to secs

Decoupled Product Services



Functional Domains with autonomous services and independent datastores

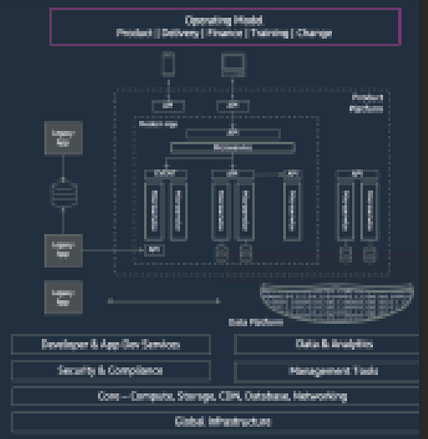
SLA bound communication among services with built-in failover safety

Built-in observability for SLAs and KPIs

Architecture Evolution
…enabling rapid composition for innovation

Supporting 8600 transactions per second via 500k+ POS devices, McDonald's was able to build a Global Home Delivery service in 4 months that is scalable to 1M orders / hour

Modern Operating Model



Small, focused, teams accountable for addressing specific, identifiable customer or business needs

End to end responsibility for validating design, architecture, operations, and support

Business value KPIs for continuous validation of business value

Organizing for Value
…aligned to discrete business outcomes & value

GUARDIAN

Retrained 2500 employees to agile practices and saw a significant increase in their ability to Invest, Test, Learn, & Fail Quickly (aka. Innovate)

Why is on-premises security traditionally challenging?

Lack of visibility

Low degree of automation

Before...

Move fast **OR** Stay secure

Now…

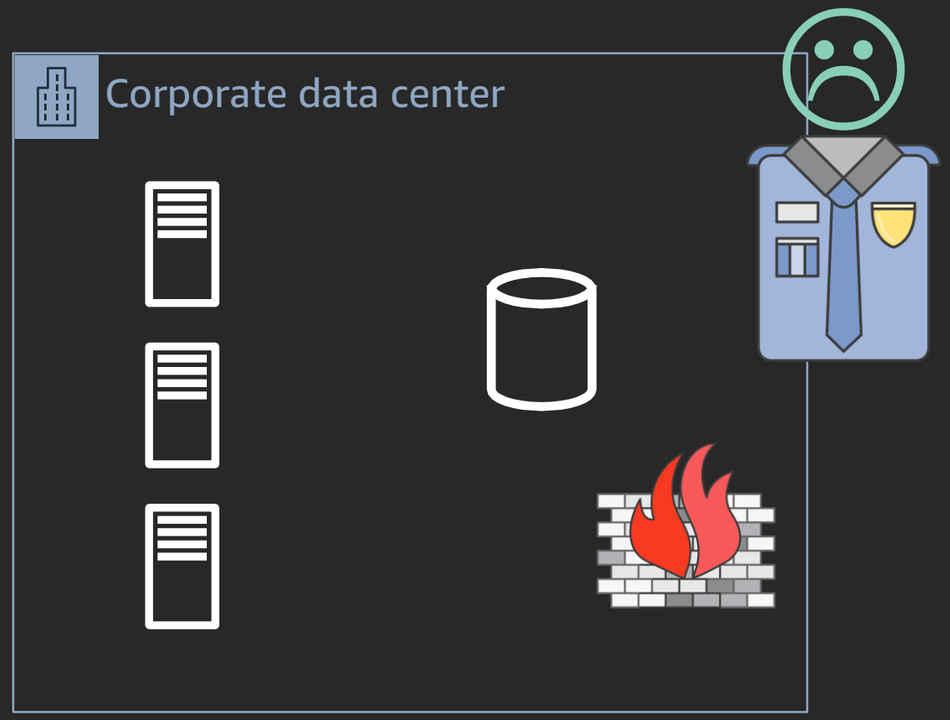
Move fast AND Stay secure

AWS security, identity, and compliance solutions

Identity and access management	Detective controls	Infrastructure protection	Data protection	Incident response	Compliance
AWS Identity and Access Management (IAM)	AWS Security Hub	AWS Firewall Manager	Amazon Macie	Amazon Detective	AWS Artifact
AWS Single Sign-On	Amazon GuardDuty	AWS Network Firewall	AWS Key Management Service (KMS)	Amazon EventBridge	AWS Audit Manager
AWS Organizations	Amazon Inspector	AWS Shield	AWS CloudHSM	AWS Backup	
AWS Directory Service	Amazon CloudWatch	AWS WAF – Web application firewall	AWS Certificate Manager	AWS Security Hub	
Amazon Cognito	AWS Config	Amazon Virtual Private Cloud	AWS Secrets Manager	CloudEndure Disaster Recovery	
AWS Resource Access Manager	AWS CloudTrail	AWS PrivateLink	AWS VPN		
	VPC Flow Logs	AWS Systems Manager	Server-Side Encryption		
	AWS IoT Device Defender				

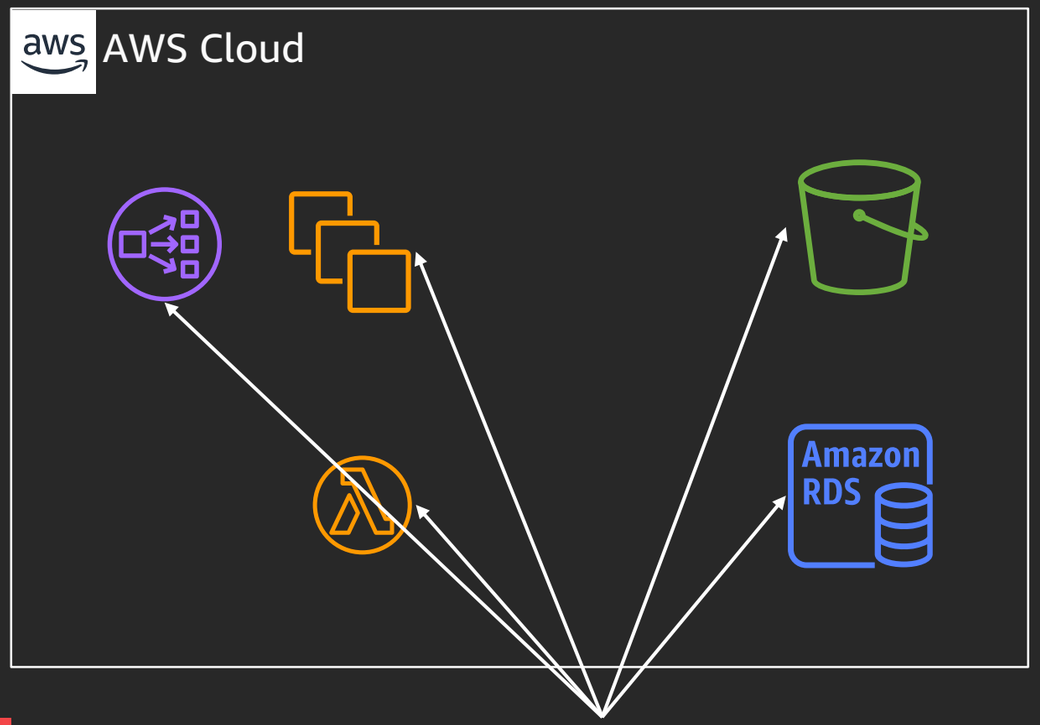
Why learn more deeply about Identity (Auth-Auth) ?

Security before the cloud



Security implemented at perimeter

Security in the cloud



IAM authorization at every resource:

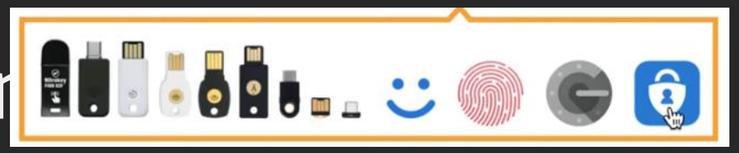
Pervasive security that's part of your applications

BASICS

AuthN = Authentication

AuthZ = Authorization

MFA = Multi Factor Authentication



WHO

Identity Management

CAN ACCESS

Access Management

WHAT

Resource Management

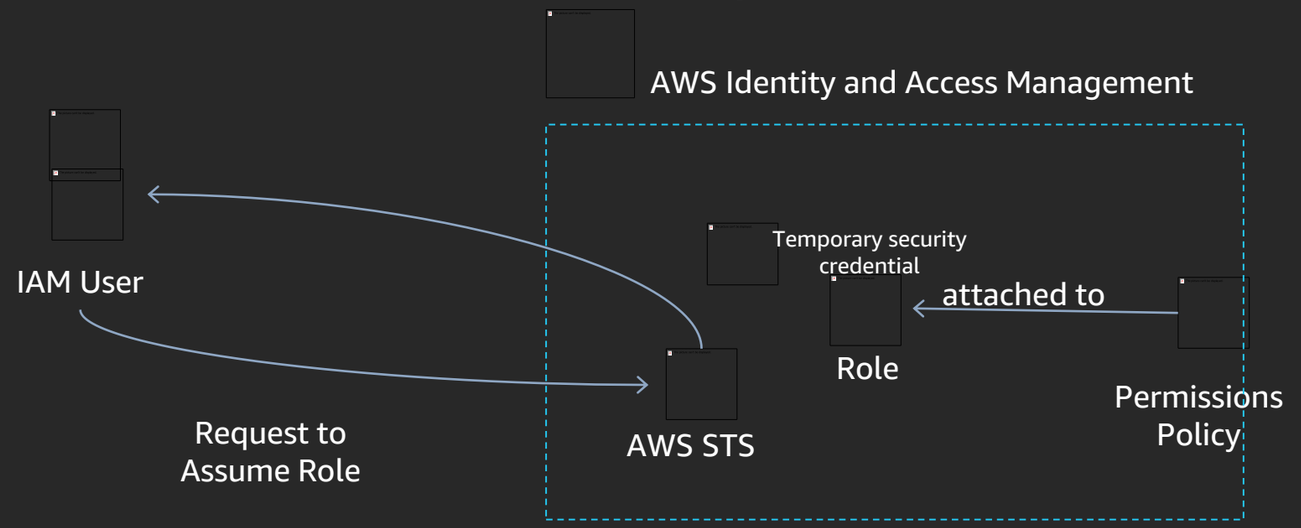
AWS Identity – Brief History

Year	Launch	Brief	Detail
2006	AWS	Root User	One account, One user
2011	IAM	IAM Users	One account, Many users
2013	SAML Federation	Corp Directory users	One account, Corporate users
2015	Switch Role	Ability to switch role	Same user switching roles
2017	AWS Organization SSO Service	SSO users	Many account, Many users
2019	SSO External Directory	SSO + Corporate directory users	Many accounts, Corporate users
2020	AWS IAM Access Analyzer	helps you identify the resources in your organization and accounts, such as Amazon	

AWS IAM Basics

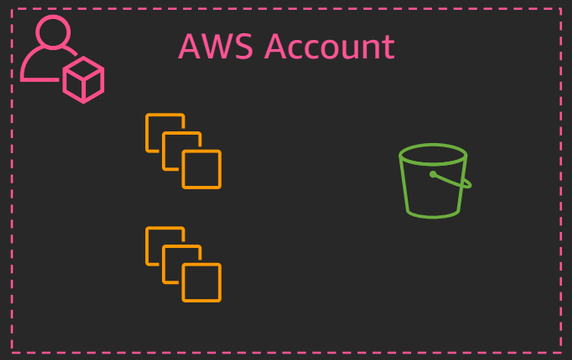
- IAM User
  - Entity that you create in AWS, representing the person or service who uses the IAM user to interact with AWS
- IAM Group
  - Collection of IAM users (A management convenience)
- IAM Role
  - Similar to a user but does not have standard long-term credentials (e.g. password or access keys) associated with it

  - An IAM User can assume a Role to take on the permissions of the role

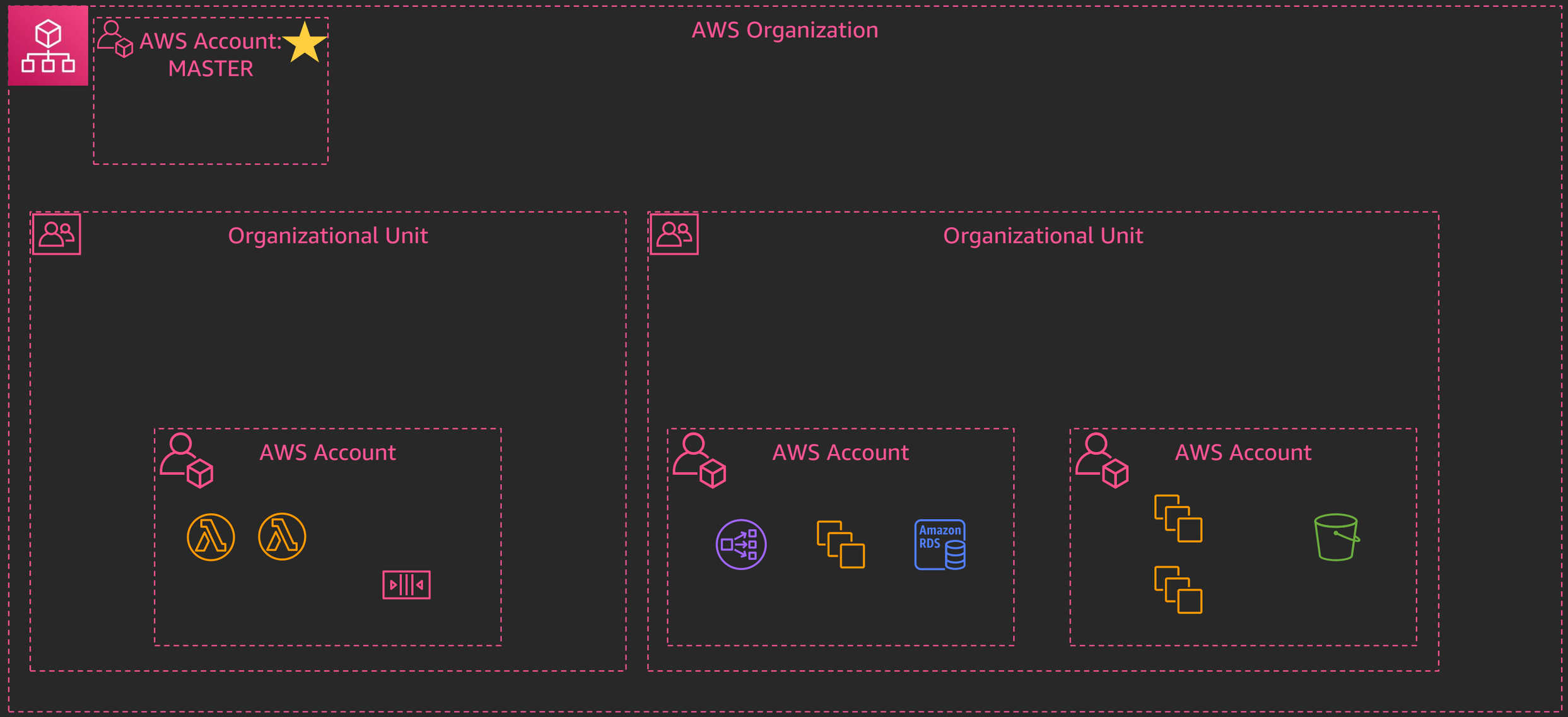


Human Access

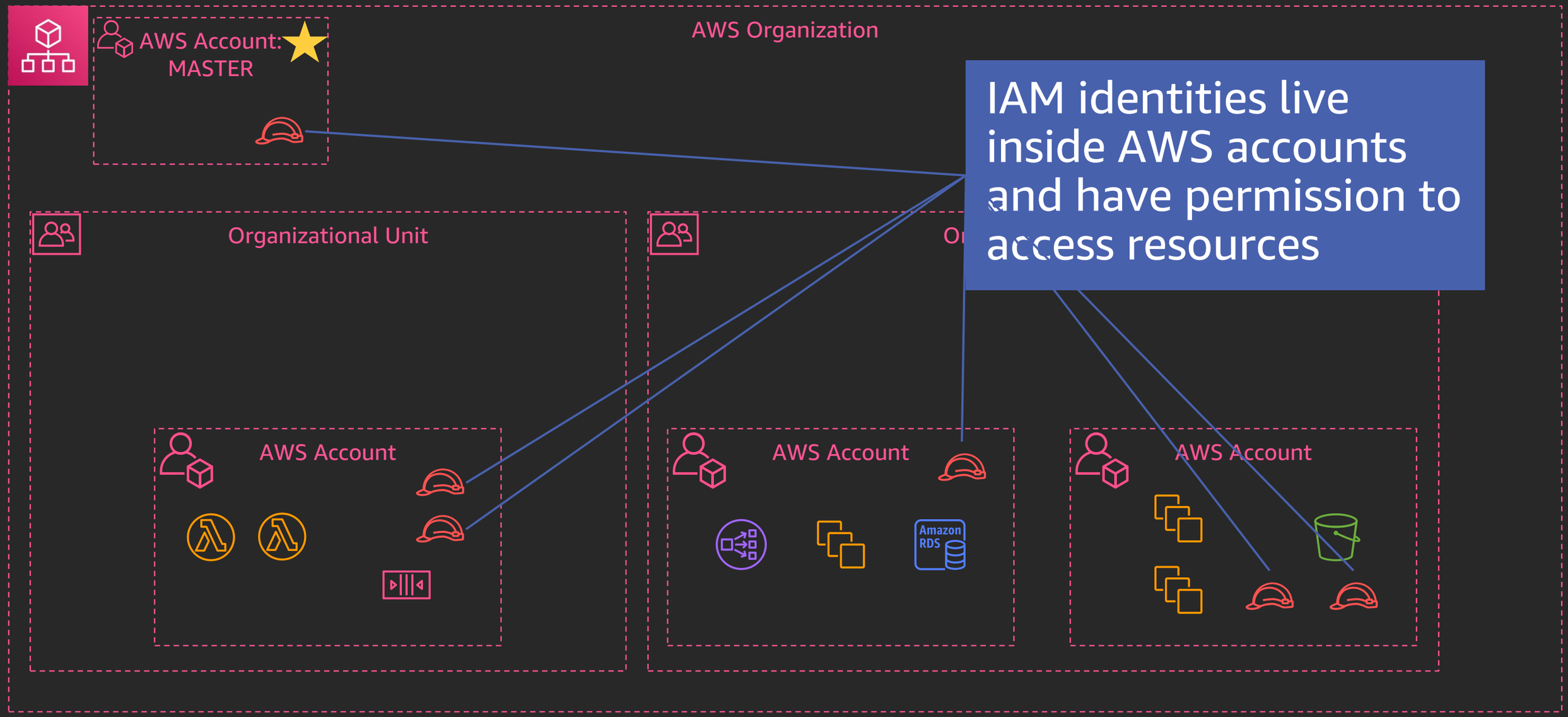
Accounts in AWS



Accounts in AWS

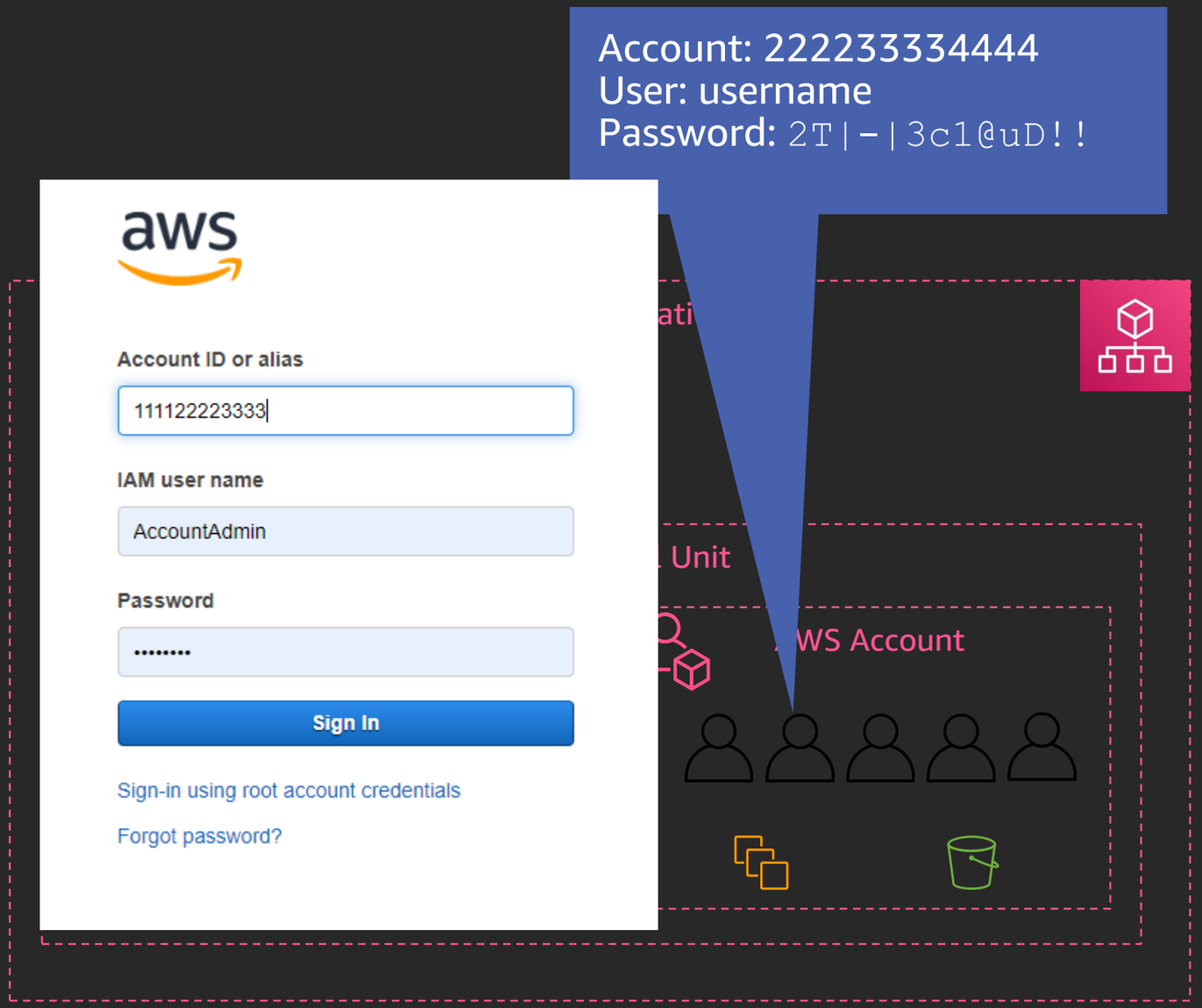


Accounts in AWS



IAM identities live inside AWS accounts and have permission to access resources

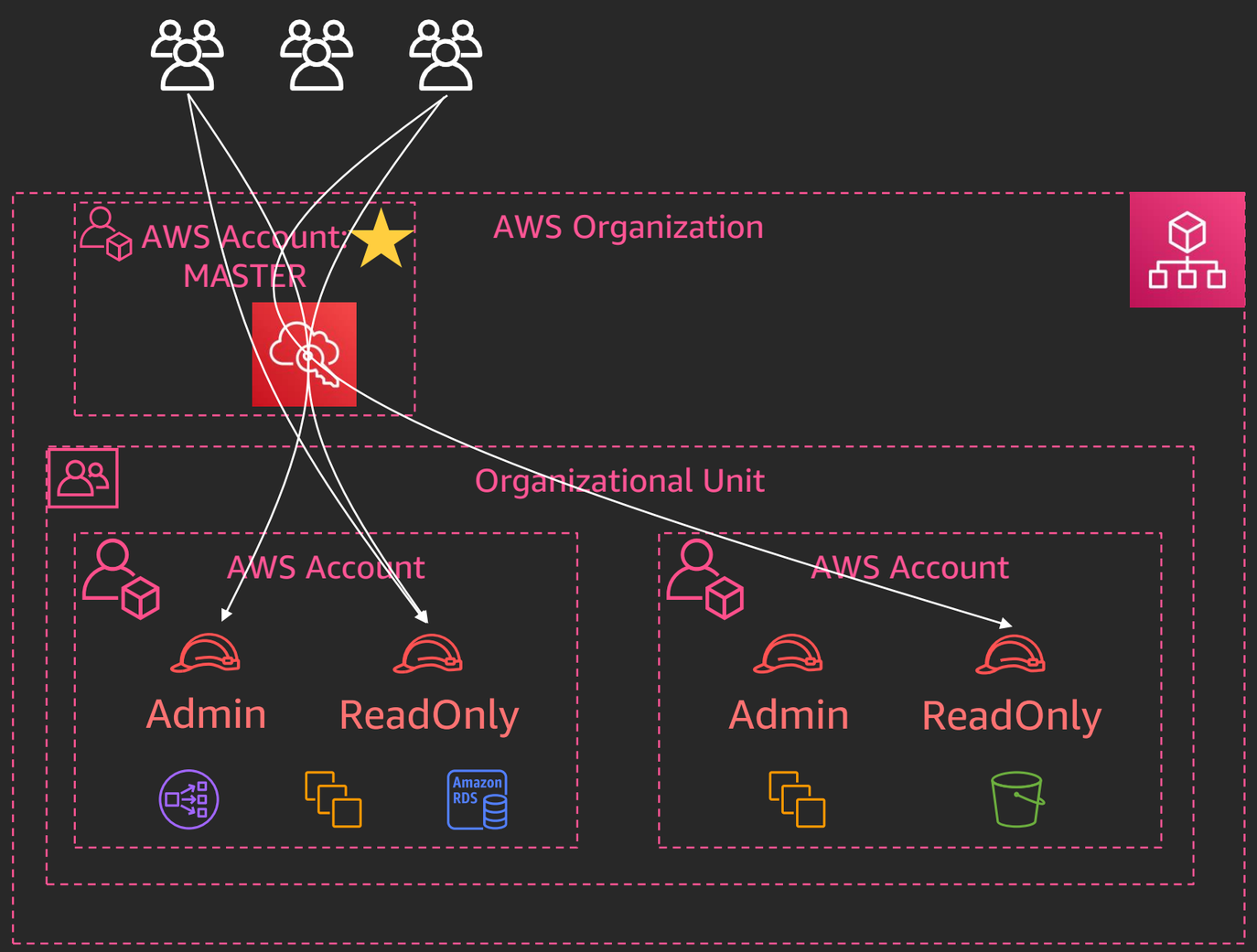
IAM users



Works best when you have:

- A relatively small number of users (limit is 5,000)
- One AWS account, or a relatively small number of them
- A need for long-term credentials
- No user directory, or no ability to connect your directory to AWS
- Your very first AWS account

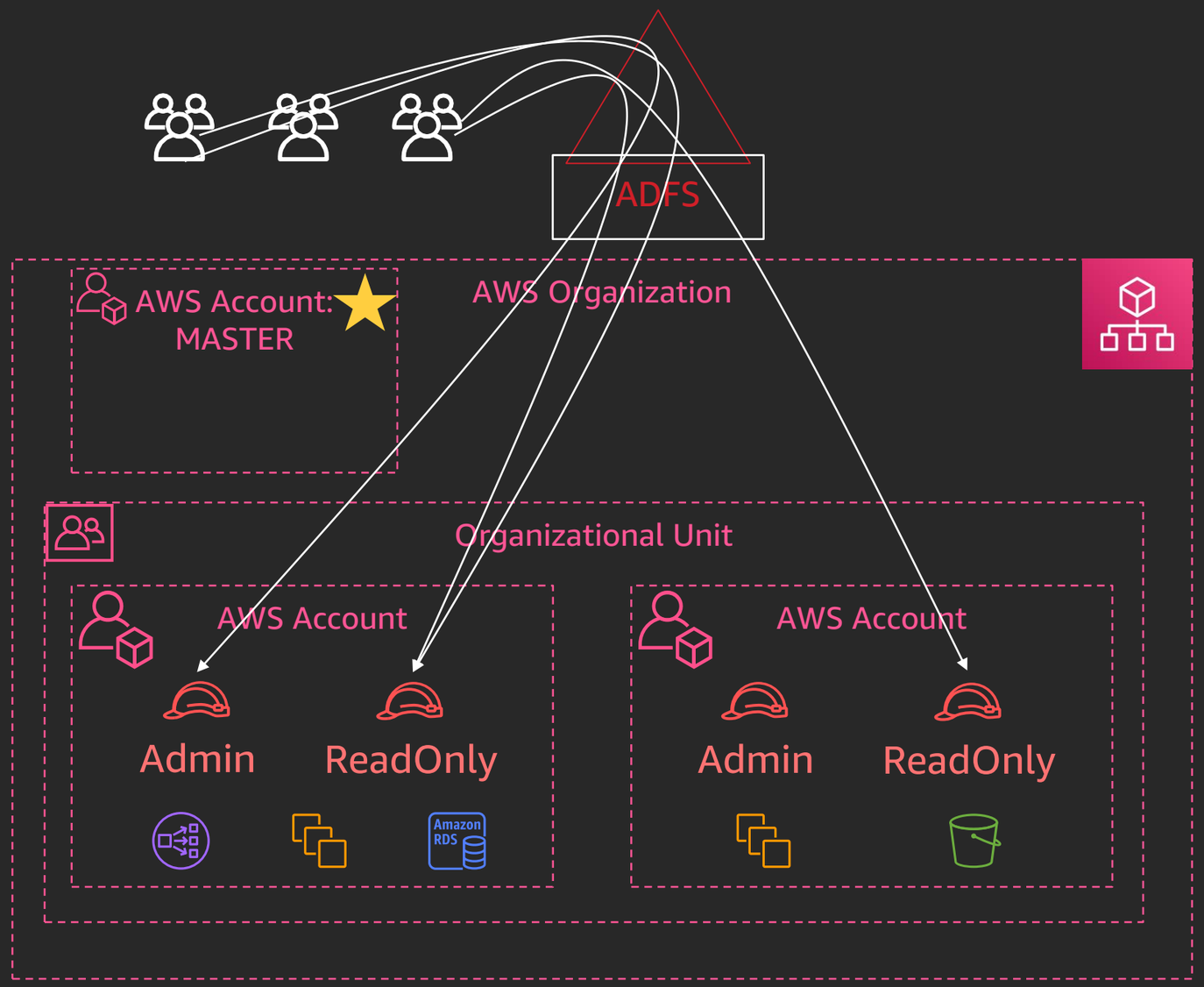
AWS Single Sign-On user pool



Works best when you have:

- A relatively small number of users (limit is 500)
- Simple authorization schemes of humans into AWS
- Rules to map groups of users to AWS environments
- No user directory, or no ability to connect your directory to AWS

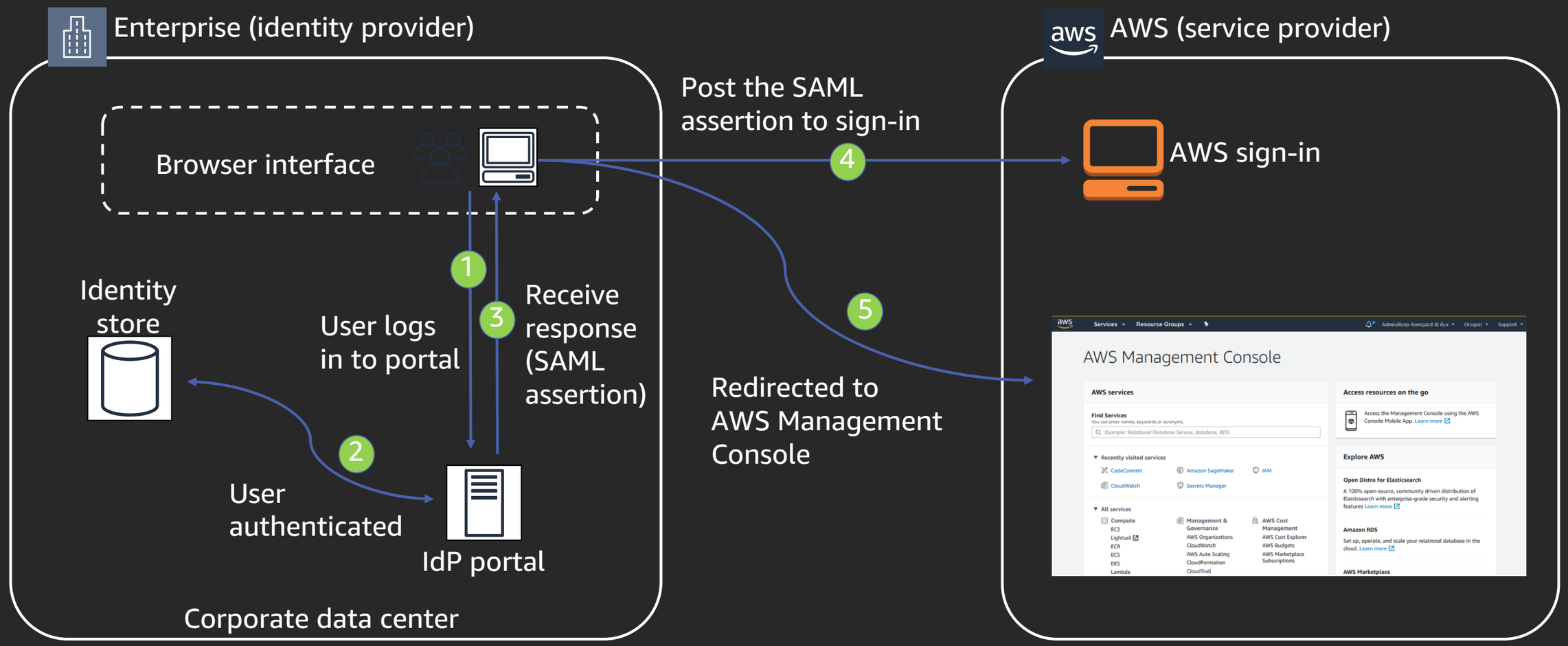
Active Directory Federation Services



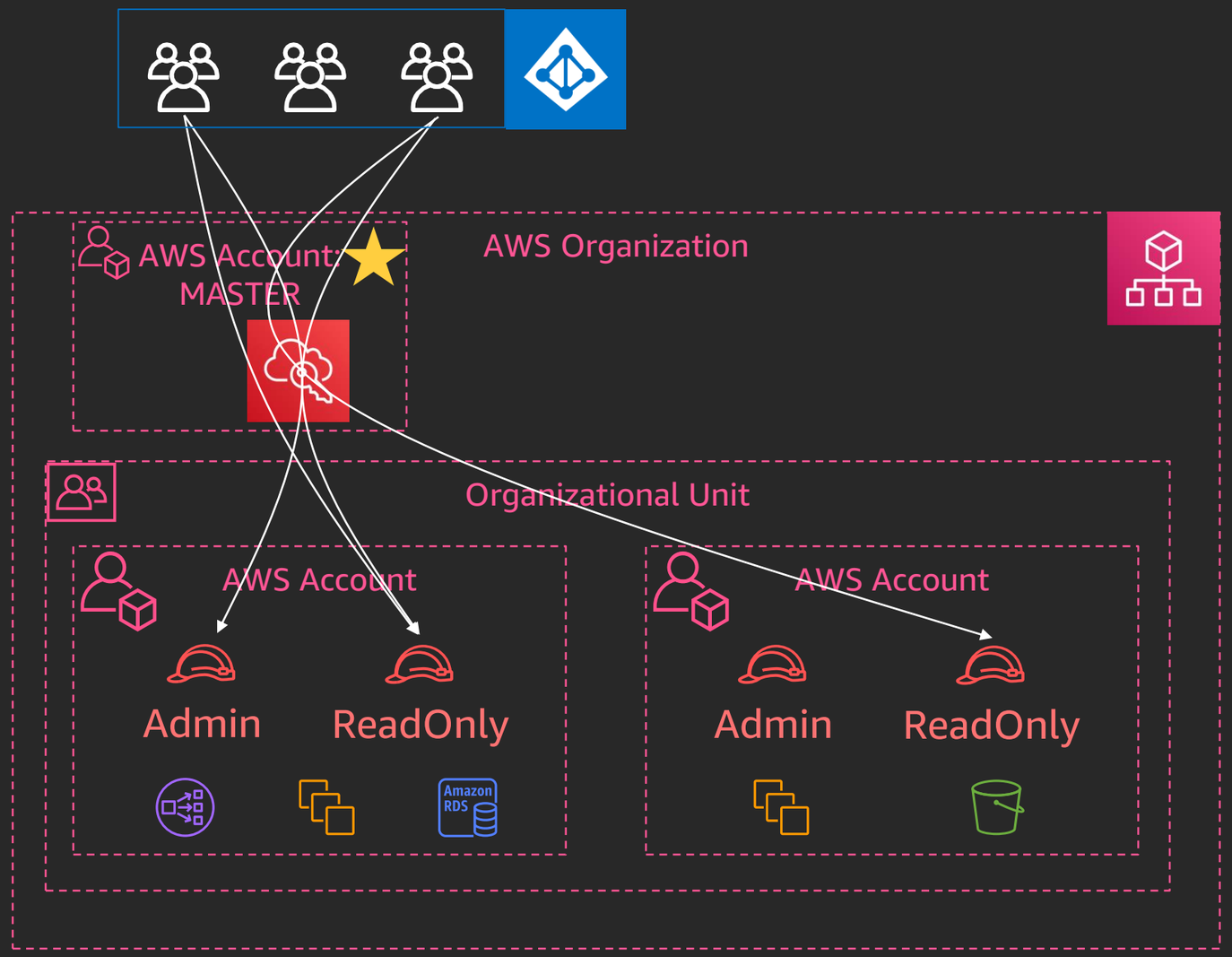
Works best when you have:

- Corporate users in a Microsoft Active Directory, either on-premises or managed in AWS
- An ADFS connected to your directory
- Control over ADFS claims
- A need for granular control over user permissions

Identity federation with SAML 2.



AWS Single Sign-On with Azure AD



Works best when you have:

- Azure Active Directory as identity source of truth
- Ability to enable SCIM
- Simple authorization schemes of humans into AWS
- Rules to map groups of users to AWS environments

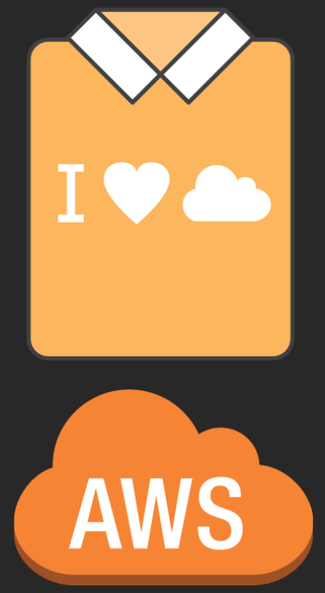
Choices for AD deployment on AWS

AWS Managed Microsoft AD

Self-managed, On-premises

Self-managed, Amazon EC2

AWS-managed, AWS Cloud



AWS services commonly used with AD

Amazon EC2

- .NET apps
- SharePoint Server
- SQL Server Always-On
- Amazon EC2 Linux

Amazon RDS and Amazon FSx

- Amazon RDS for PostgreSQL
- Amazon RDS for Oracle
- Amazon RDS for SQL Server
- Amazon FSx for Windows File Server

AWS SSO integrated applications

- AWS SSO
- AWS Management Console
- Amazon SageMaker Studio
- AWS IoT SiteWise (Preview)
- salesforce
- Office 365
- SAML

AWS applications

- Amazon WorkSpaces
- Amazon AppStream 2.0
- Amazon WorkDocs
- Amazon QuickSight
- Amazon Chime
- Amazon WorkMail
- Amazon Connect
- AWS Client VPN

Quick Demo .NET SDK – IAM Manipulation

IDE integration

AWS Toolkit for Visual Studio

AWS Toolkit for Visual Studio Code

AWS Toolkit for Rider

Programmable SDK

AWS SDK for .NET

AWS CDK for .NET

Command line tools

AWS Tools for PowerShell

'dotnet' CLI extensions

AWS CLI

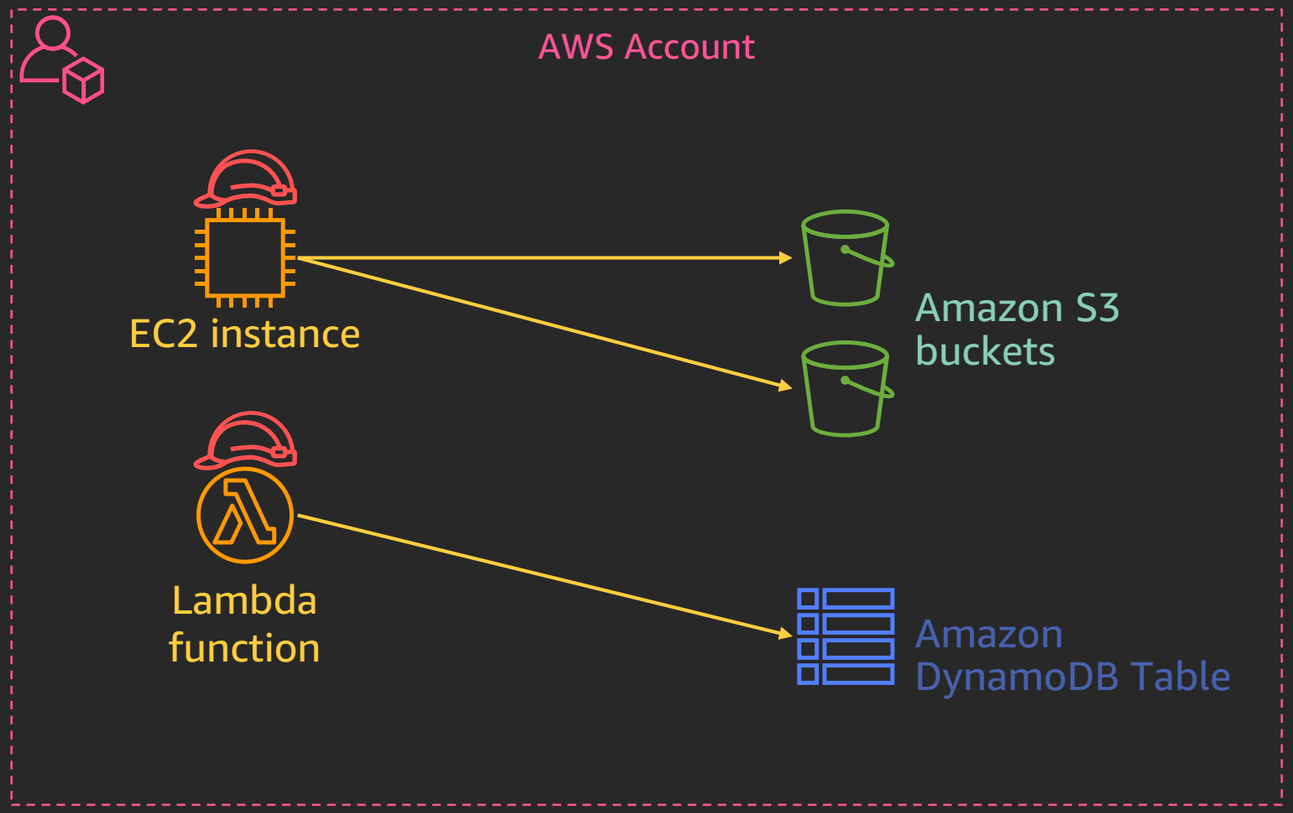
AWS SAM for Windows

CI/CD integration

AWS Tools for Azure DevOps

AWS CodePipline/ CodeBuild

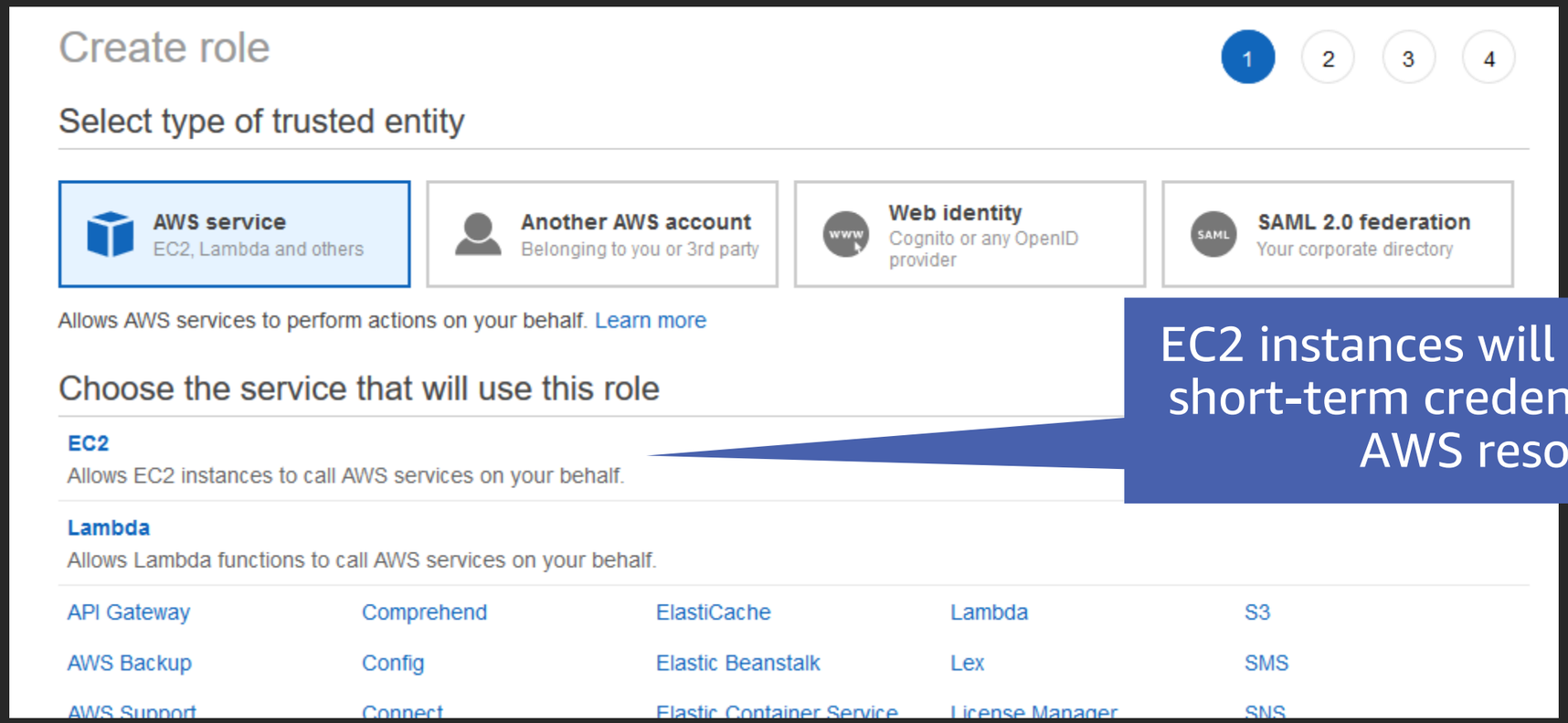
IAM roles for non-human access



Use IAM roles for access to AWS resources from:

- Your application running on an AWS compute environment, e.g., EC2 instance, Lambda function, etc.
- Permission to an AWS service to access your resources (not shown)

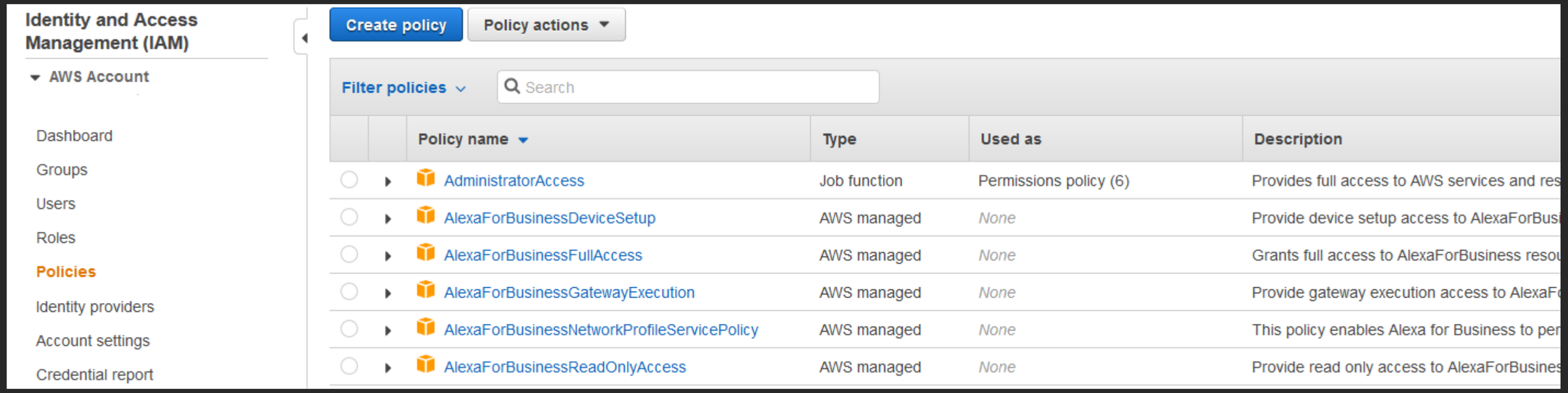
Creating IAM roles for non-human access



EC2 instances will have access to short-term credentials to access AWS resources

Assigning Permissions

Assigning AWS managed policies



. . . and many other AWS-written policies

Example: Administrator policy

Policies > AdministratorAccess

Summary

Policy ARN arn:aws:iam::aws:policy/AdministratorAccess

Description Provides full access to AWS services and resources.

Permissions | **Policy usage** | **Policy versions** | **Access Advisor**

Policy summary | { } JSON

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": "*",
7              "Resource": "*"
8          }
9      ]
10 }
```

Example: Read-only policy

Policies > ReadOnlyAccess

Summary

Policy ARN arn:aws:iam::aws:policy/ReadOnlyAccess

Description Provides read-only access to AWS services and

Permissions | **Policy usage** | **Policy versions** | **Access Advisor**

Policy summary | { } JSON

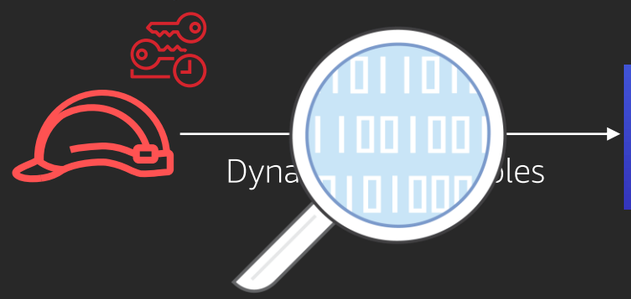
```
1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Action": [
6                 "a4b:Get*",
7                 "a4b:List*",
8                 "a4b:Describe*",
9                 "a4b:Search*",
```

. . . and many other ReadOnly APIs

Auth-Auth in AWS

How authentication works in AWS

Short-term credentials for IAM role



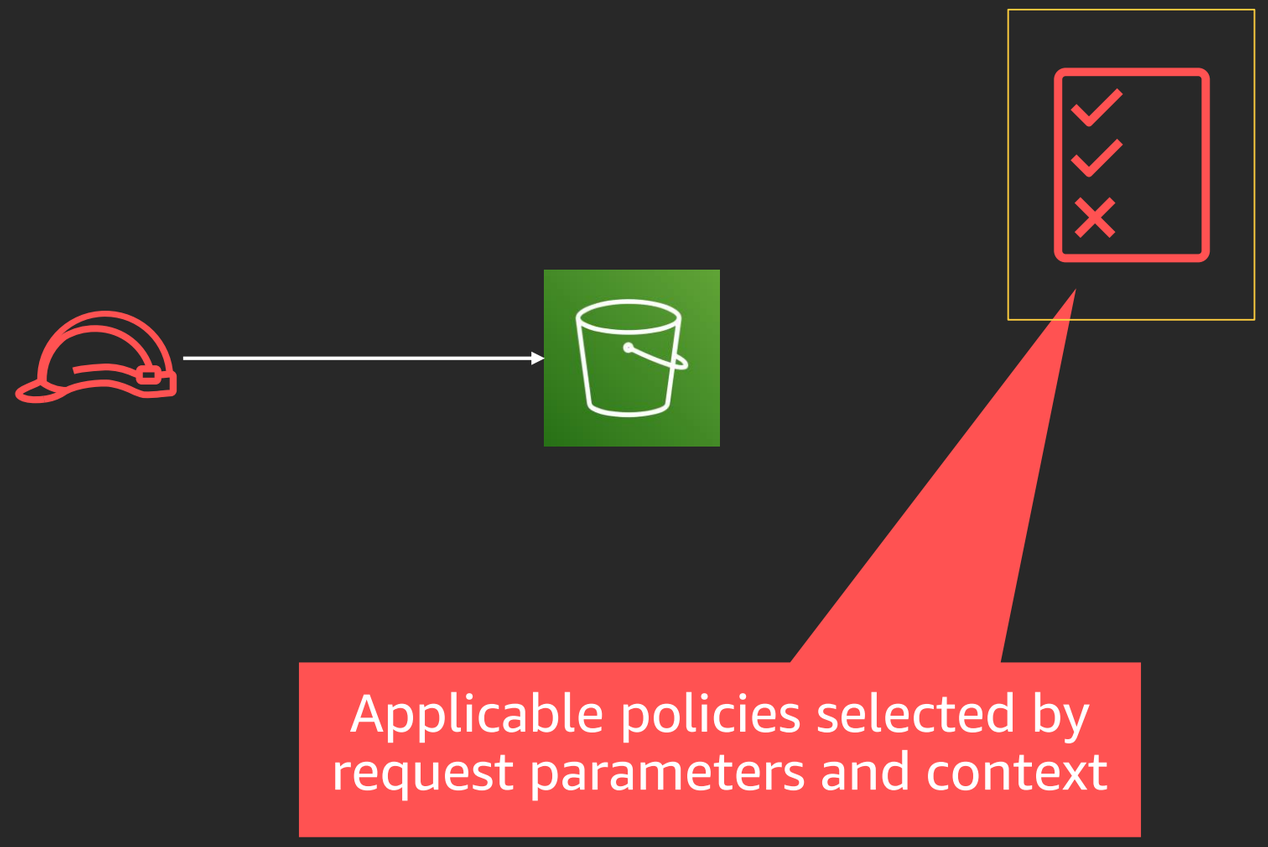
```
POST https://dynamodb.us-east-2.amazonaws.com/ HTTP/1.1
Host: dynamodb.us-east-2.amazonaws.com
X-Amz-Date: 20180918T150746Z
X-Amz-Target: DynamoDB_20120810.ListTables
X-Amz-Security-Token: FQoGZXIvYXdzEKH///////// …
Content-Type: application/x-amz-json-1.0
Authorization: AWS4-HMAC-SHA256
Credential=ASIAXXXXXXXXXXXXXXXX/20180918/us-east-
1/dynamodb/aws4_request, SignedHeaders=content-
type;host;x-amz-date;x-amz-security-token;x-amz-target,
Signature=c1b4bc2df0c47c86cbcfa54d932e8aaa455b6b7c38e65d84
0f722254add1ea9e
```

Access key ID: Identifies calling IAM principal to AWS

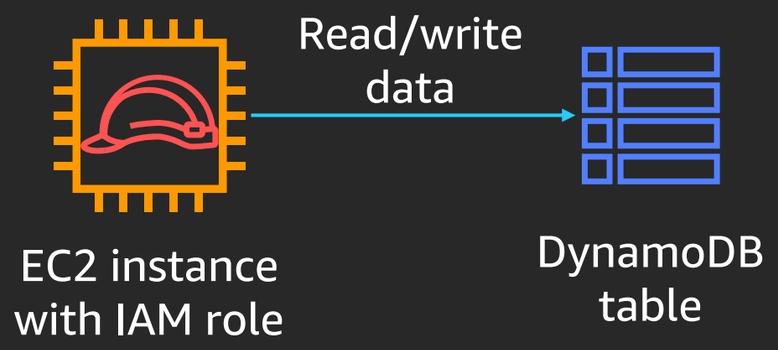
HMAC signature: Generated with secret key; validated by AWS

How authorization works in AWS

How authorization works in AWS



Example 1: Read data from DynamoDB



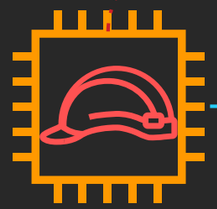
Example 1: Read data from DynamoDB

Least-privilege face of judgment

```
{
  "Effect": "Allow",
  "Action": "*",
  "Resource": "*"
}
```

All actions

All resources



EC2 instance with IAM role

Read/write data

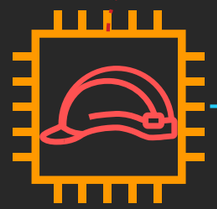
DynamoDB table

Example 1: Read data from DynamoDB

```
{
  "Effect": "Allow",
  "Action": "dynamodb:*",
  "Resource": "*"
}
```

All DynamoDB actions

All resources

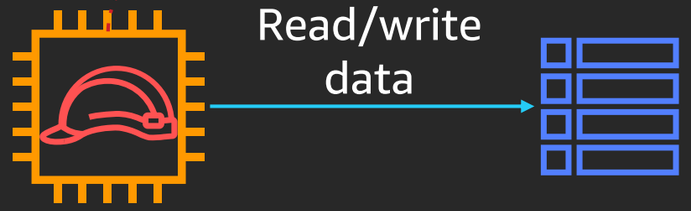


EC2 instance
with IAM role

Read/write
data

DynamoDB
table

Example 1: Read data from DynamoDB



EC2 instance with IAM role

DynamoDB table

```
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:GetItem",
    "dynamodb:PutItem"
  ],
  "Resource": "*"
}
```

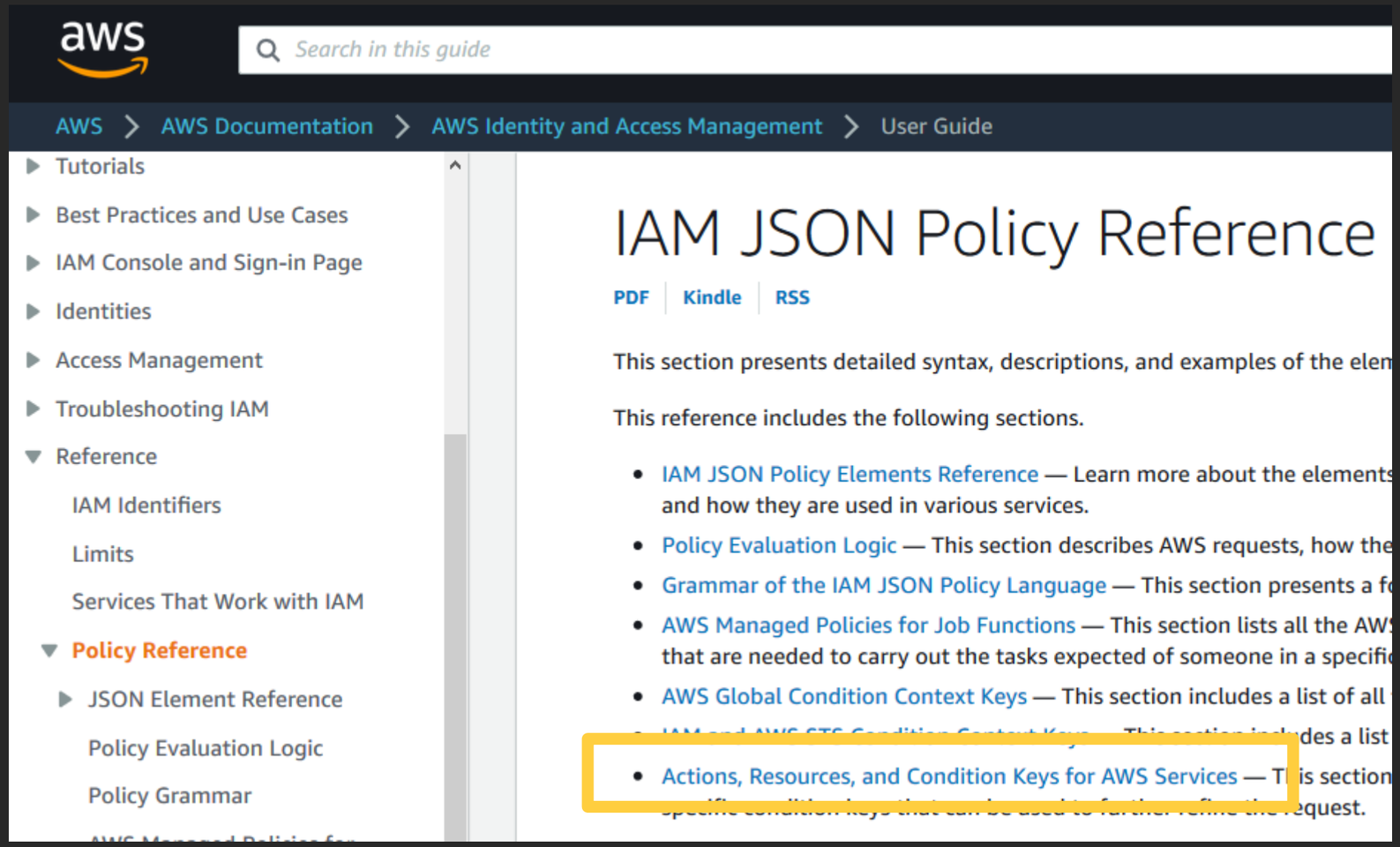
Specific DynamoDB actions for reading items from a table

All resources

Reading the IAM documentation page

Bookmark this page:

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies.html



Reading the IAM documentation page

Action

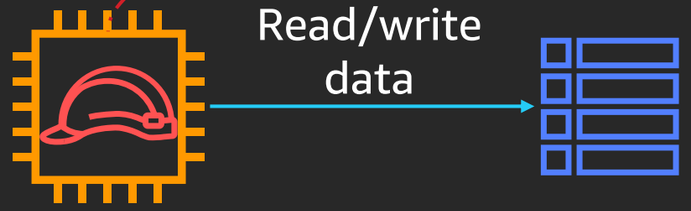
Resource

Condition

GetItem	The GetItem operation returns a set of attributes for the item with the given primary key	Read	table*		
				dynamodb:Attributes	
				dynamodb:EnclosingOperation	
				dynamodb:LeadingKeys	
				dynamodb:ReturnConsumedCapacity	
				dynamodb:Select	

Reference: https://docs.aws.amazon.com/IAM/latest/UserGuide/list_amazondynamodb.html

Example 1: Read data from DynamoDB



EC2 instance
with IAM role

DynamoDB
table

```
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:GetItem",
    "dynamodb:PutItem"
  ],
  "Resource": [
    "arn:aws:dynamodb:us-east-2:111122223333:table/MyTable"
  ]
}
```

DynamoDB row:
Reading/writing actions

Specific resource: Table

Reading the IAM documentation page

Action | Resource | Condition

GetItem	The GetItem operation returns a set of attributes for the item with the given primary key	Read	table*	
				dynamodb:Attributes
				dynamodb:EnclosingOperation
				dynamodb:LeadingKeys
				dynamodb:ReturnConsumedCapacity
				dynamodb:Select

Reference: https://docs.aws.amazon.com/IAM/latest/UserGuide/list_amazondynamodb.html

Reading the IAM documentation page

Resources Defined by Amazon DynamoDB

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

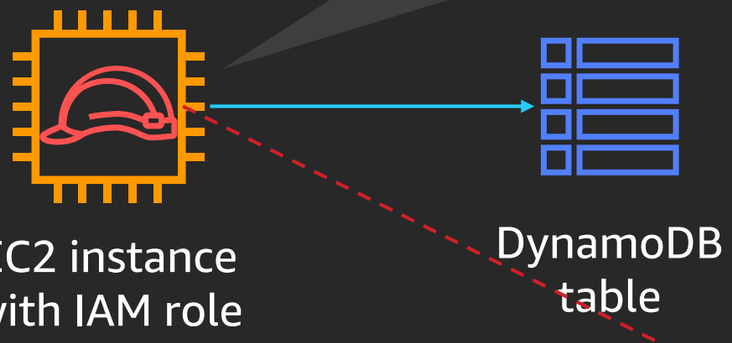
"arn:aws:dynamodb:us-east-2:111122223333:table/MyTable"

Resource Types	ARN	Condition Keys
index	dynamodb:${Region}:${Account}:table/${TableName}/index/${IndexName}	
stream	arn:${Partition}:dynamodb:${Region}:${Account}:table/${TableName}/stream/${StreamLabel}	
table	arn:${Partition}:dynamodb:${Region}:${Account}:table/${TableName}	
backup	arn:${Partition}:dynamodb:${Region}:${Account}:table/${TableName}/backup/${BackupName}	
global-table	arn:${Partition}:dynamodb::${Account}:global-table/${GlobalTableName}	

Reference: https://docs.aws.amazon.com/IAM/latest/UserGuide/list_amazondynamodb.html

How authorization works in AWS

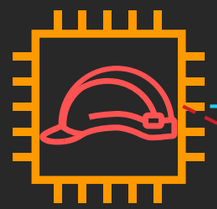
```
dynamodb.putItem({
  TableName:"MyTable",
  Item: {
    "Id": {
      S: "a1b2c3d4"
  …
});
```




```
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:GetItem",
    "dynamodb:PutItem"
  ],
  "Resource": [
    "arn:aws:dynamodb:us-east-2:111122223333:table/MyTable"
  ]
}
```

How authorization works in AWS

```
dynamodb.putItem({
  TableName:"MyTable",
  Item: {
    "Id": {
      S: "a1b2c3d4"
  …
});
```



EC2 instance
with IAM role

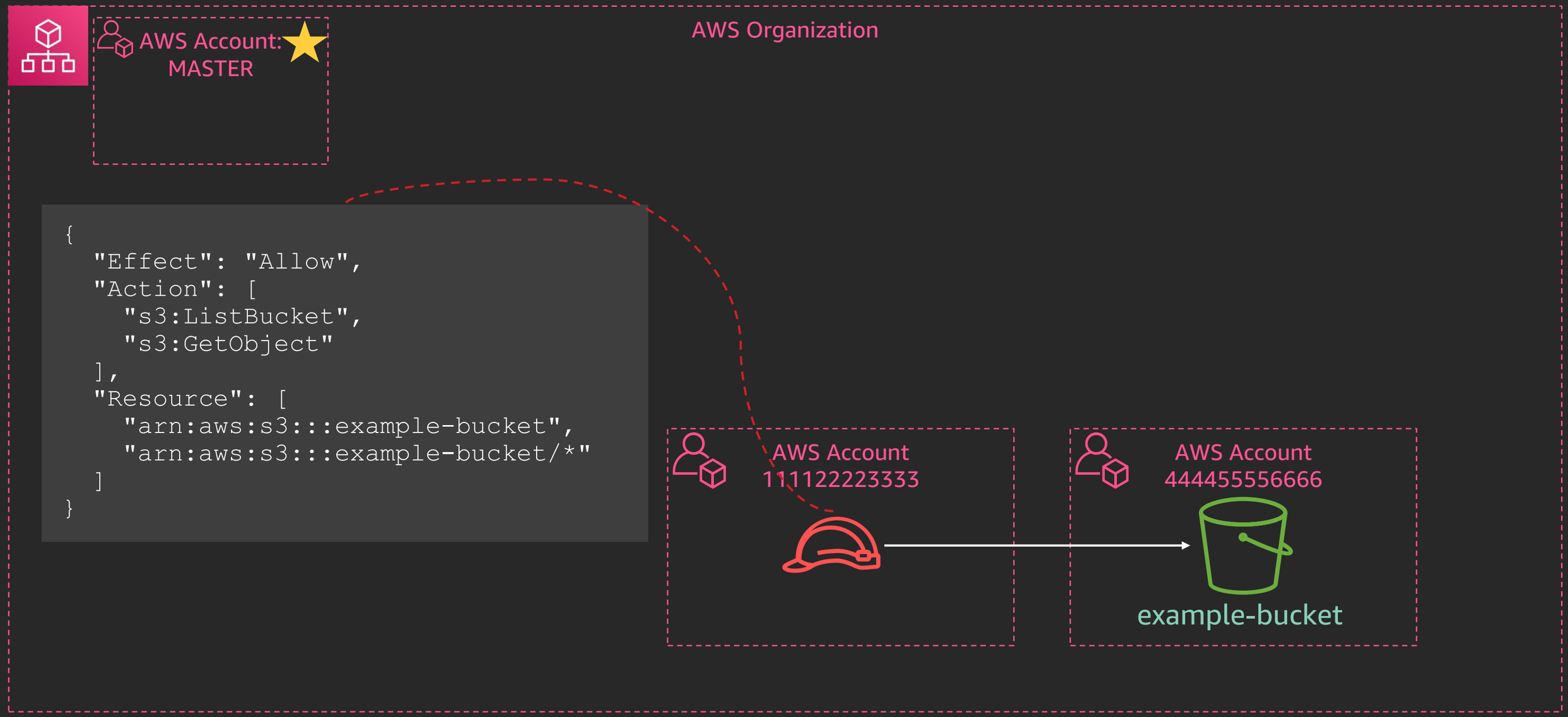
DynamoDB
table

```
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:GetItem",
    "dynamodb:PutItem"
  ],
  "Resource": [
    "arn:aws:dynamodb:us-east-2:111122223333:table/MyTable"
  ]
}
```

The “PutItem” action and the “table” resource match the Allow statement, so the request is allowed

Permission across AWS Accounts

Accessing resources in another AWS account




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

Accessing resources in another AWS account

AWS Account:

Resource-based policy for S3 bucket

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "111122223333"
  },
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/*"
  ]
}
```

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

AWS Account
111122223333

AWS Account
444455556666

example-bucket

Accessing resources in another AWS account

AWS Account:
MASTER

Allow the other account

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "111122223333"
  },
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/*"
  ]
}
```

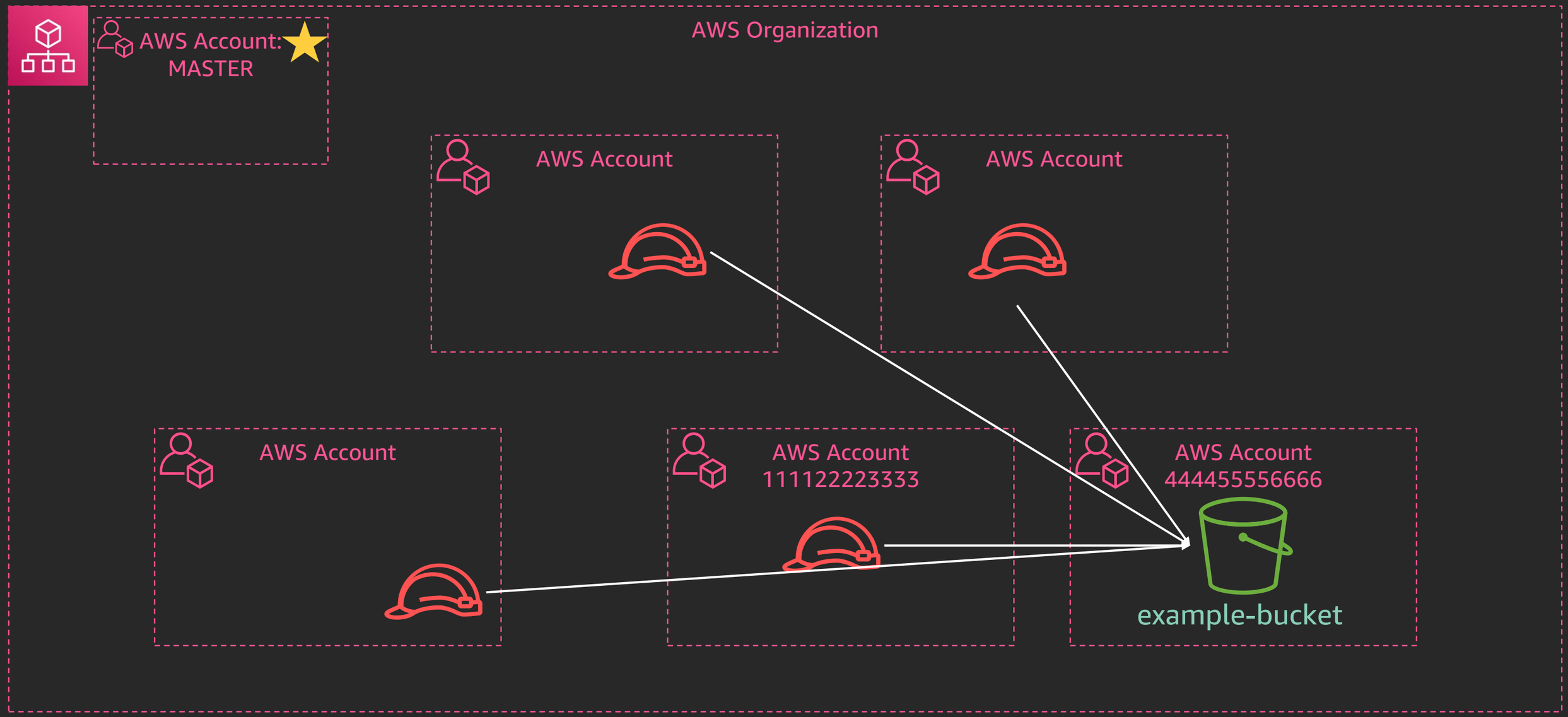
```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/*"
  ]
}
```

AWS Account
111122223333

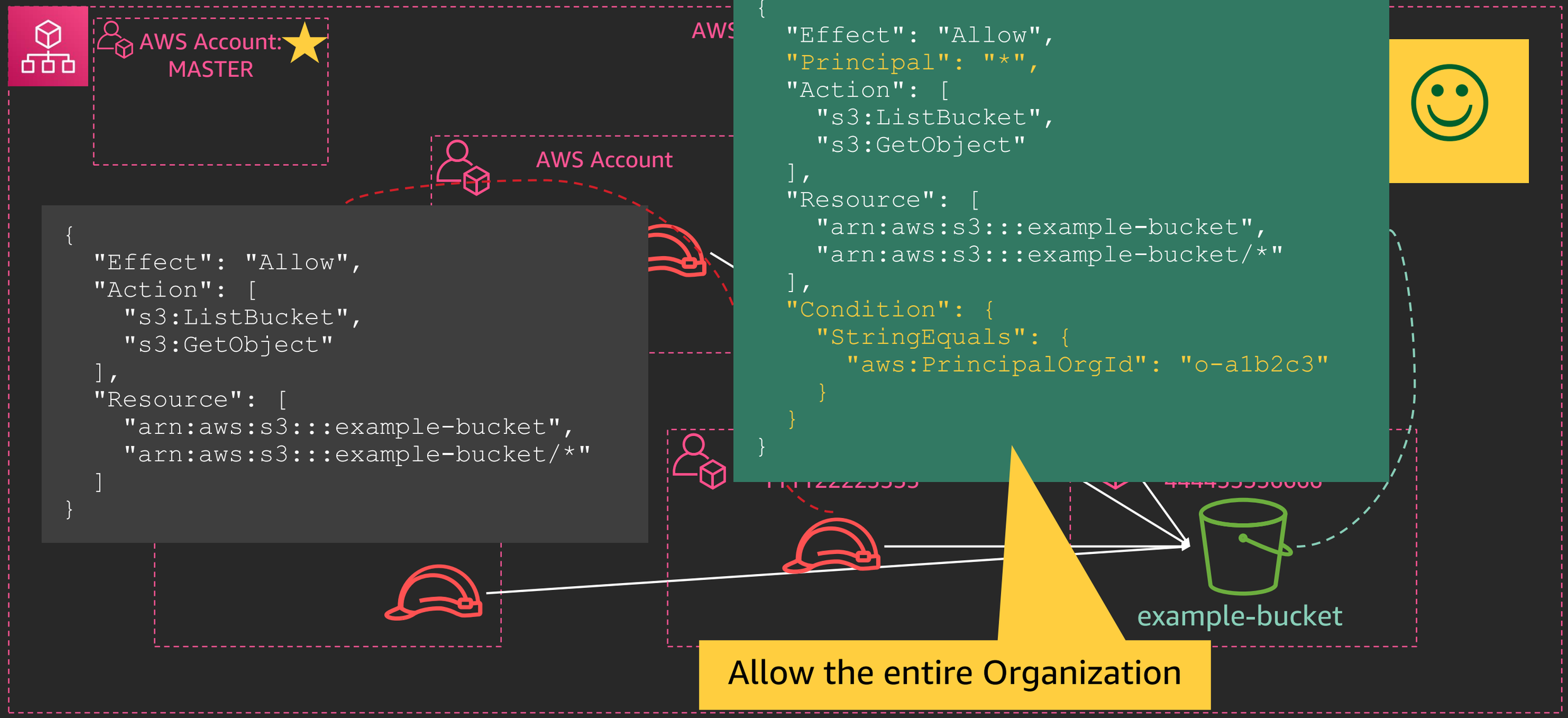
AWS Account
444455556666

example-bucket

Accessing resources in another AWS account



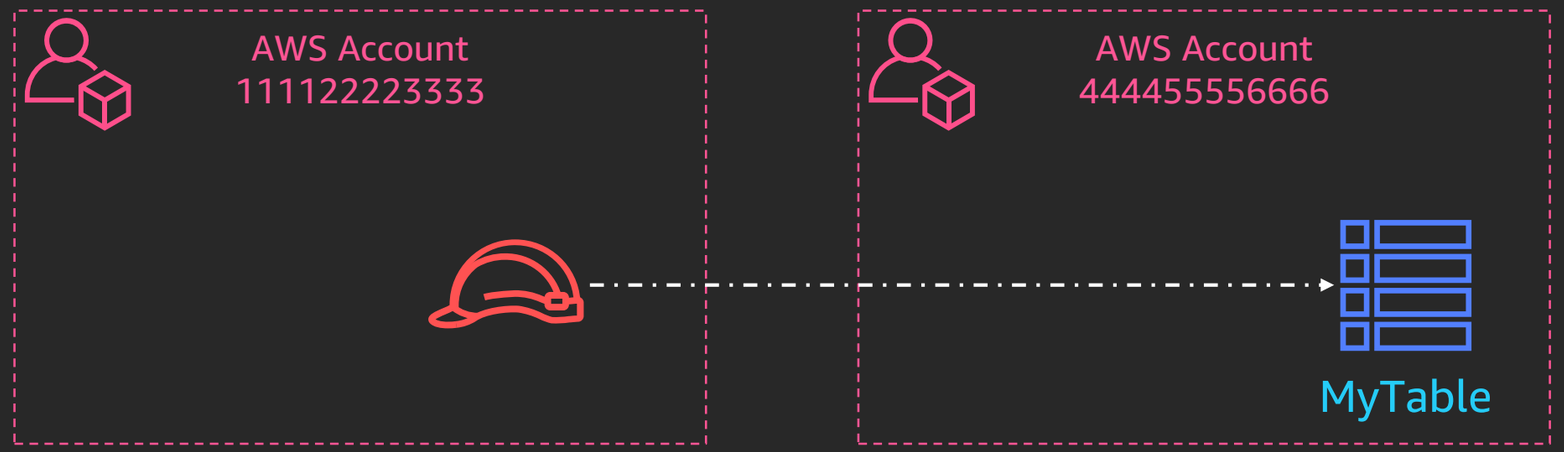
Accessing resources in another AWS account




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

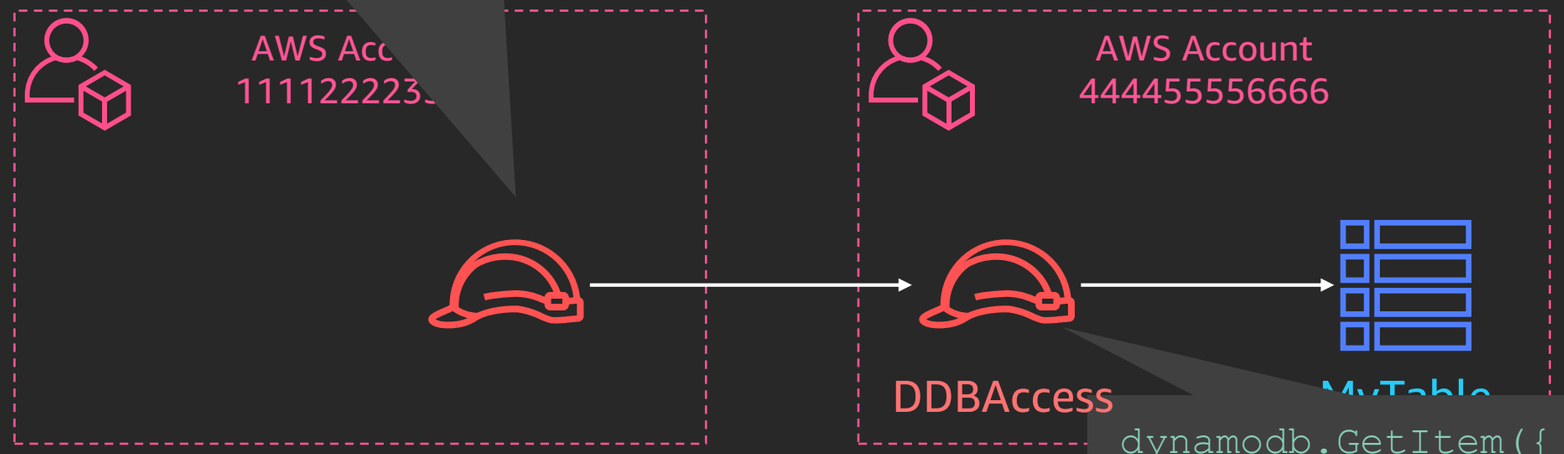
```
{
  "Effect": "Allow",
  "Principal": "*",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:PrincipalOrgId": "o-a1b2c3"
    }
  }
}
```

Accessing resources in another AWS account



Accessing resources in another AWS account: Using IAM roles across accounts

```
sts.AssumeRole({
  RoleArn: 'arn:aws:iam::444455556666:role/DDBAccess',
  …
});
```




```
dynamodb.GetItem({
  TableName: 'MyTable',
  …
});
```

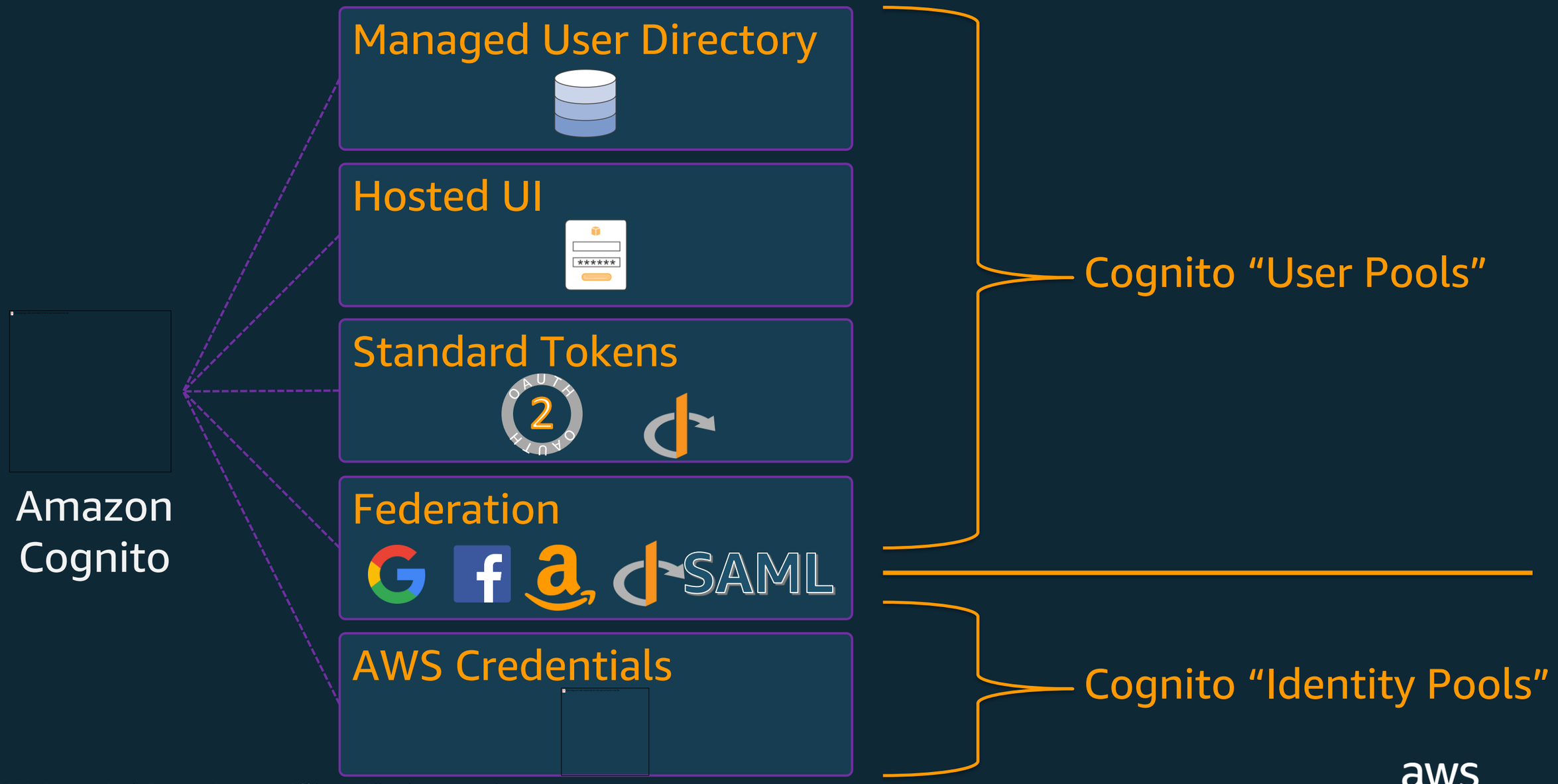
Recommendations for cross-account access

Keep it simple:

- Use resource-based policies when available
- Unless you have a specific reason to do otherwise:
  - Trust the entire other account, or
  - Trust the AWS Organization
- Use IAM roles if resource-based policies are not available
  - Follow the above rules for their trust policies (i.e., resource-based policies for IAM roles)

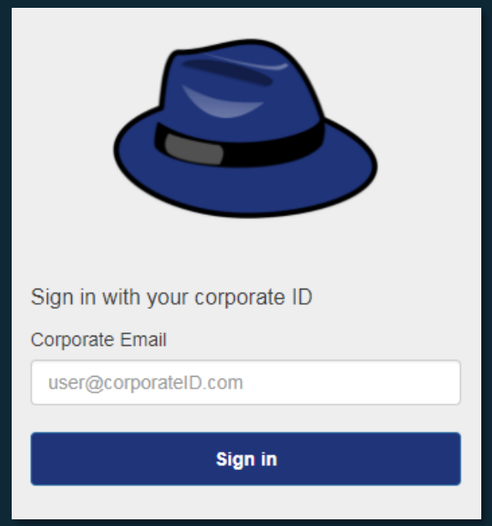
End Client Authentication….

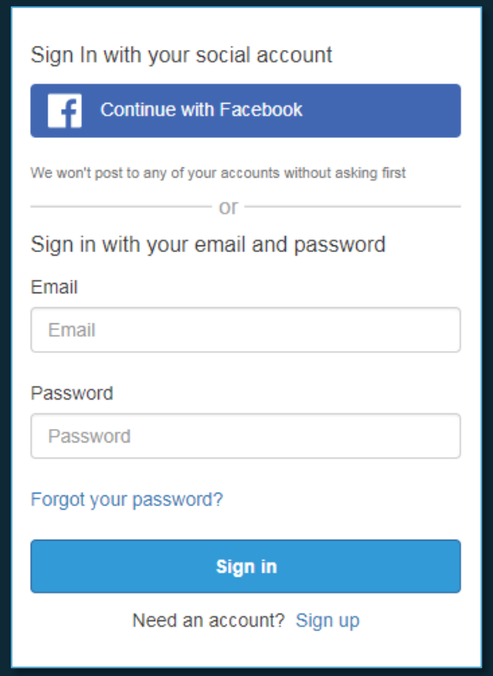
Amazon Cognito Overview

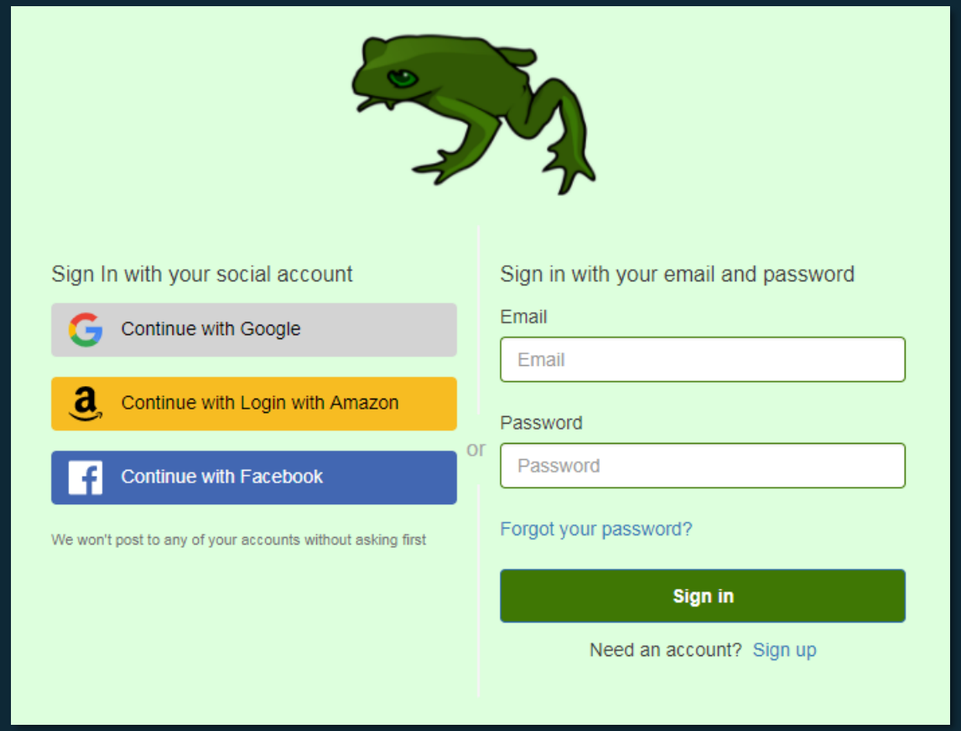


Hosted user interface

- Facilitates user flows (sign up/in, forgot password, etc)
- Customizable logo, style and branding
- Use your own domain







Cognito: Flexible and Fully Managed Application Identity

Amazon Cognito

Flexible and Scalable API & SDK Support

- AWS SDKs
- Node JS
- Angular
- React
- iOS
- Android
- Vue
- Ionic

Built-In UI for Applications

- iOS
- Android
- Web
- SPA

Extensible AuthN & AuthZ

- AWS Lambda
- Amazon ALB
- Amazon API Gateway
- AWS AppSync

Secure & Available

- Adaptive Auth
- Compromised Password DB
- MFA
- 99.9% SLA

Out of the box support for Open Standards

- SAML
- OAuth2
- OIDC

Out of the box support for Social Federation

- Google
- Facebook
- Amazon

Managed user directory

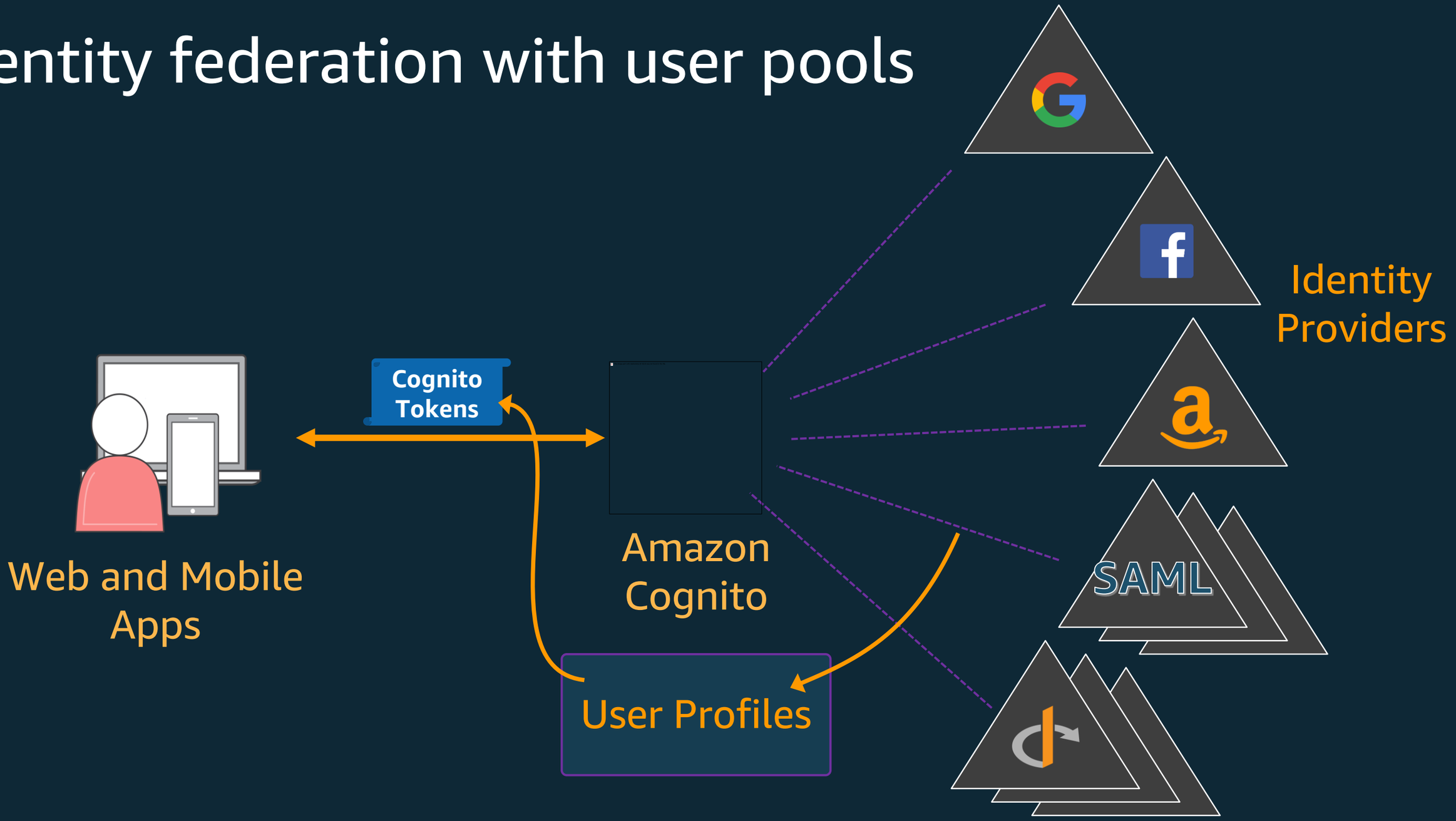
- Serverless directory
  - Nothing to manage
  - API driven
  - Multi-AZ redundancy
- User & group storage
  - Profile information (name, email, etc)
  - Credential & device information (SRP verifier, MFA, etc)
  - Extensible with custom attributes

Support for OAuth 2.0 in Cognito User Pools

- OAuth 2.0 flows:
  - Authorization code
  - Implicit
  - Client credentials
  - Resource owner password credentials

- Custom scopes defined for resource servers

Identity federation with user pools

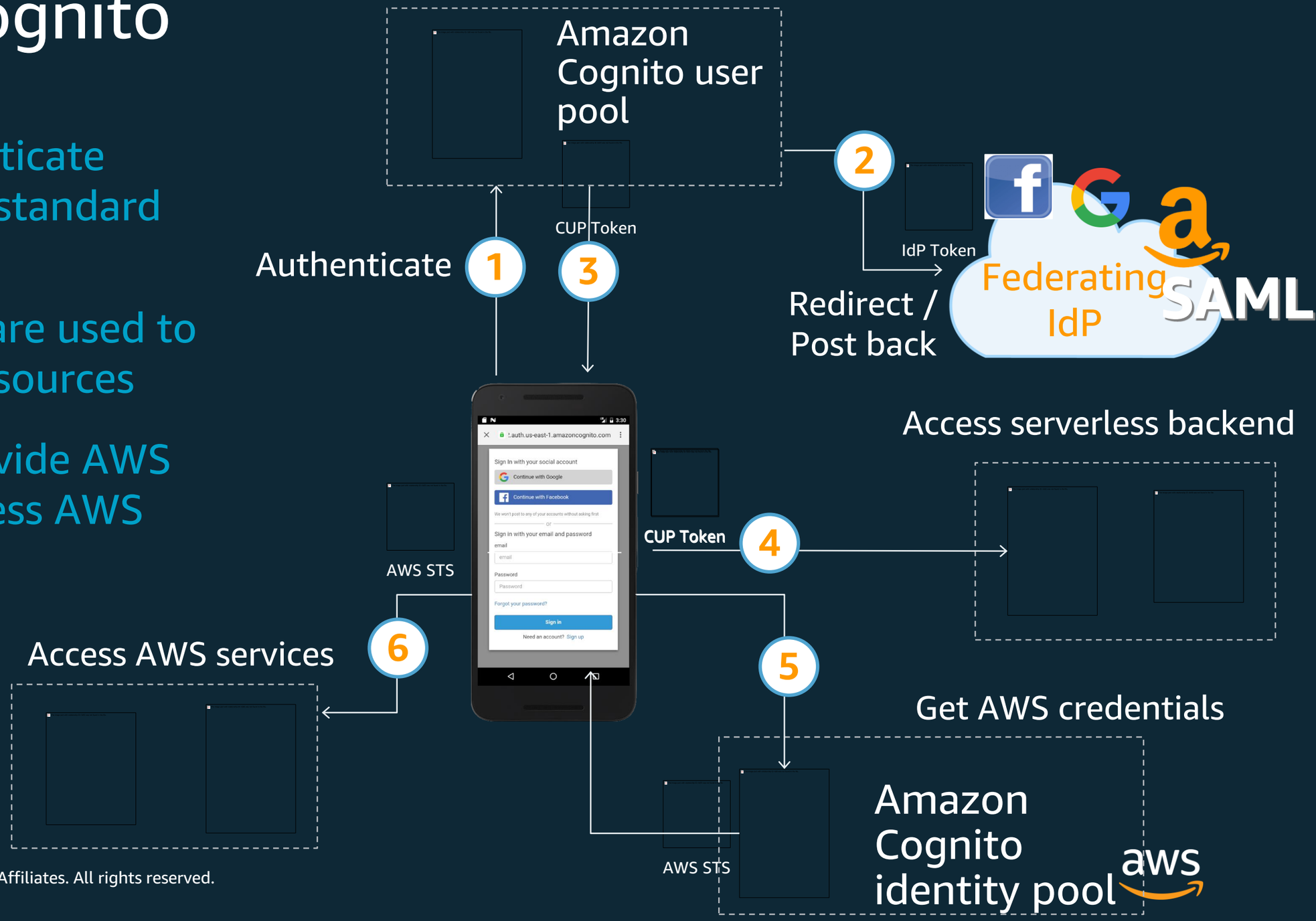


Amazon Cognito

User pools authenticate users and returns standard tokens

User pool tokens are used to access backend resources

Identity pools provide AWS credentials to access AWS services



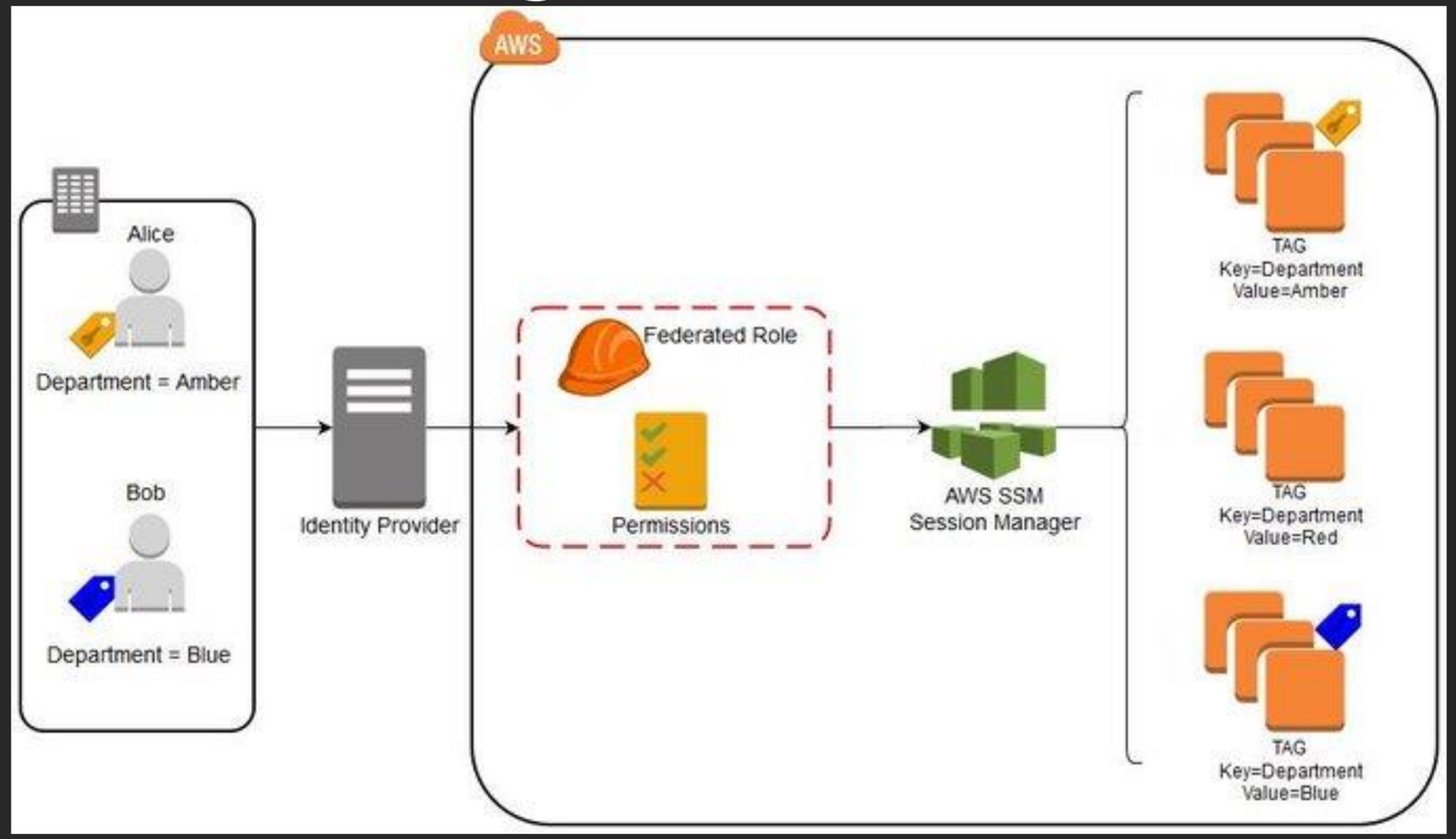
Fine grained permissioning

AWS lake formation helps you set up a secure data lake in days. A data lake is a centralized, curated, and secured repository that stores all your data, both in its original form and prepared for analysis.

- You can use Lake Formation to centrally define security, governance, and auditing policies in one place, versus doing these tasks per service

- Eliminates the need to manually configure them across security services like AWS Identity and Access Management and AWS Key Management Service, storage services like S3, and analytics and machine learning services like Redshift, Athena, and (in beta) EMR for Apache Spark. This reduces the effort in configuring policies across services and provides consistent enforcement and compliance.
- E.g. https://aws.amazon.com/blogs/big-data/enable-fine-grained-permissions-for-amazon-quicksight-authors-in-aws-lake-formation/

ABAC (Attribute Based Access Control) – Leverage session tags



https://aws.amazon.com/blogs/mt/configure-session-manager-access-for-federated-users-using-saml-session-tags/

Largest ecosystem of security partners and solutions

Network and infrastructure security

- ALERT LOGIC Security. Compliance. Cloud.
- APPGATE
- ARMOR
- Barracuda
- Check Point SOFTWARE TECHNOLOGIES LTD.
- CISCO
- ExtraHop
- FORTINET
- Guardicore
- paloalto NETWORKS
- PROTECTWISE
- SKinfosec co.,ltd.
- SOPHOS
- zscaler

Host and endpoint security

- CROWDSTRIKE
- Symantec
- TREND MICRO

Application security

- Barracuda
- Checkmarx
- f5

Identity and access control

- Auth0
- CLOUDKNOX
- okta
- onelogin
- Ping Identity
- SAVIYNT

Vulnerability and configuration analysis

- bridgecrew
- ExtraHop
- Qualys
- RAPID7
- tenable
- threat stack

Data protection and encryption

- CYBERARK
- DataSunrise Data & Database Security
- gemalto security to be free
- HashiCorp
- PRIVITAR
- THALES

Logging, monitoring, SIEM, threat detection, and analytics

- ALIEN VAULT
- LACEWORK
- McAfee Together is power.
- SECURONIX Security Analytics. Delivered.
- splunk>
- sumo logic

aws

AWS IDENTITY

.NET on AWS

Thank You!