Part 1 of 2

AWS Identity and Access Management (IAM)

Table of Contents

1. 🔑 What is AWS IAM?
2. 🌐 Service Availability
3. 💰 Cost
4. 🤔 Why should I use IAM?
5. ⚙️ How IAM Works
6. 🔐 Ways to authenticate
7. 🖥️ AWS Console Sign-In Methods
8. 🆔 Sign-In and Account ID Access for IAM Users
9. 🔥 Root User Login
10. 👤 IAM User Login
11. 🧩 IAM Components
12. 👥 IAM Users
13. 👨‍👩‍👧‍👦 IAM User Groups
14. 🎭 IAM Roles
15. 📘 Roles terms and concepts Explained
16. 📜 Types of Policies
17. 🕴️ Identity-based Policies in AWS
18. 📦 Resource-based Policies in AWS
19. 🛑 IAM Permissions Boundaries
20. 🏢 Service Control Policies in AWS Organizations
21. 🚦 Understanding ACLs in AWS
22. ⏱️ Session Policies in AWS IAM
23. 📋 JSON Policy Document Structure
24. 🔍 Granting Least Privilege in IAM Policies
25. 🔧 Refining IAM Permissions Strategy
26. ✅ AWS Managed Policies Example
27. 🛠️ Customer Managed Policies Example
28. 📝 Inline Policies Example
29. 🔒 Security Best Practices in IAM
30. 🛡️ Root User Best Practices for AWS
31. ⚠️ Tasks Requiring AWS Root User Credentials

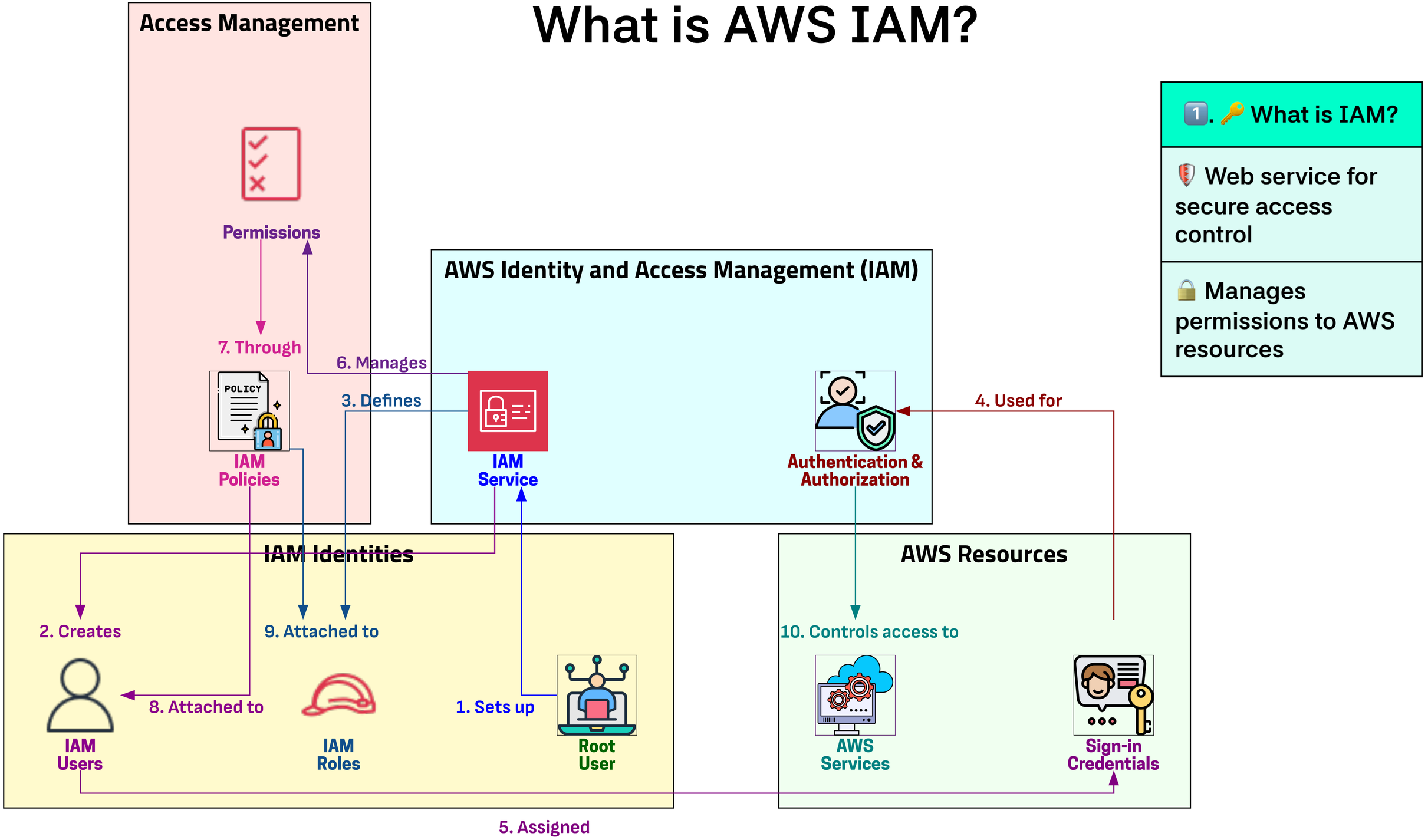
AWS Identity and Access Management

What is AWS IAM?

1️⃣. 🔑 What is IAM?

🛡️ Web service for secure access control

🔒 Manages permissions to AWS resources



What is AWS IAM?

Access Management

Permissions

7. Through

IAM Policies

6. Manages

3. Defines

AWS Identity and Access Management (IAM)

IAM Service

Authentication & Authorization

4. Used for

IAM Identities

2. Creates

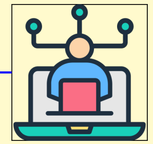
IAM Users

8. Attached to

IAM Roles

9. Attached to

1. Sets up



Root User

AWS Resources

10. Controls access to

AWS Services

Sign-in Credentials

5. Assigned

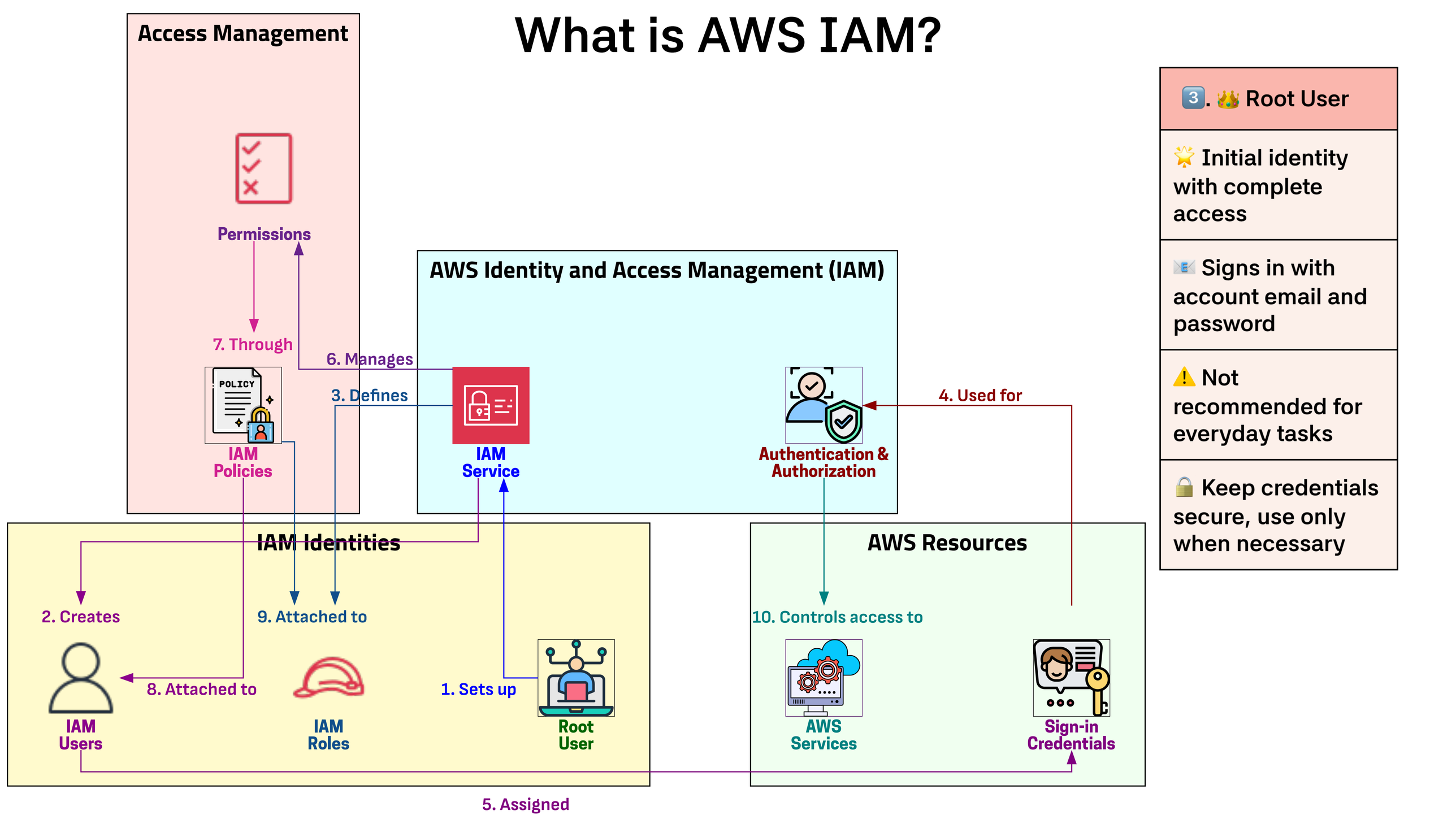
2️⃣. ✅ Core Function

🔐 Controls authentication (who can sign in)

🔑 Manages authorization (who has permissions)

⚙️ System for access management

What is AWS IAM?



3️⃣. 👑 Root User

🌟 Initial identity with complete access

📧 Signs in with account email and password

⚠️ Not recommended for everyday tasks

🔒 Keep credentials secure, use only when necessary

What is AWS IAM?

Access Management

Permissions

7. Through

IAM Policies

6. Manages

3. Defines

AWS Identity and Access Management (IAM)

IAM Service

Authentication & Authorization

4. Used for

IAM Identities

2. Creates

9. Attached to

8. Attached to

IAM Users

IAM Roles

1. Sets up

Root User

AWS Resources

10. Controls access to

AWS Services

Sign-in Credentials

5. Assigned

4️⃣. 👥 IAM Identities

👨‍💼 Create specific roles (administrators, analysts, developers)

🎯 Grant only required permissions

🔧 Principle of least privilege

What is AWS IAM?

Access Management

Permissions

7. Through

IAM Policies

6. Manages

3. Defines

AWS Identity and Access Management (IAM)

IAM Service

Authentication & Authorization

4. Used for

IAM Identities

2. Creates

9. Attached to

8. Attached to

IAM Users

IAM Roles

1. Sets up

Root User

AWS Resources

10. Controls access to

AWS Services

Sign-in Credentials

5. Assigned

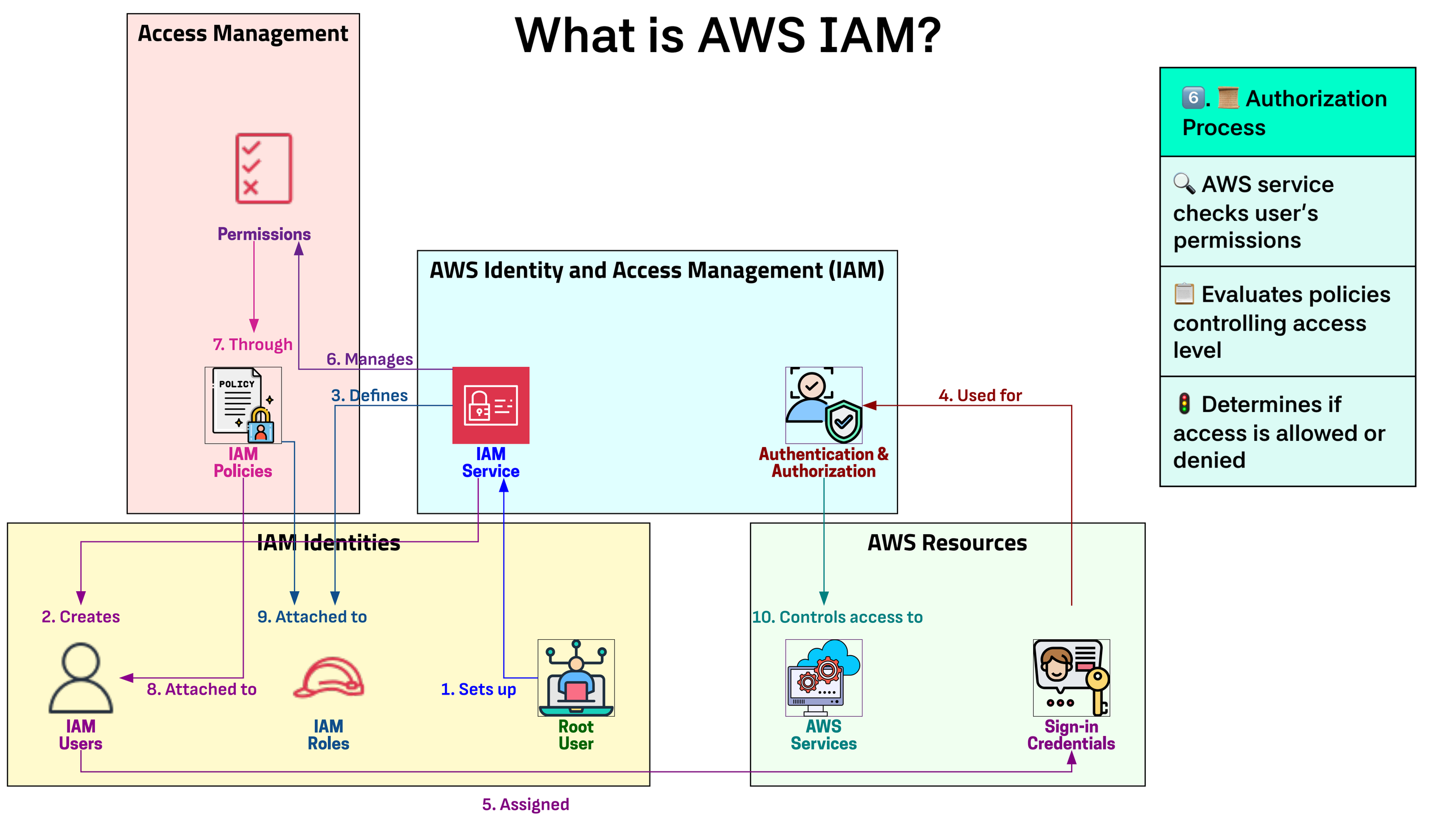
5. ID Authentication Process

User provides sign-in credentials

AWS matches credentials to trusted principal

Principals include IAM users, federated users, roles, applications

What is AWS IAM?



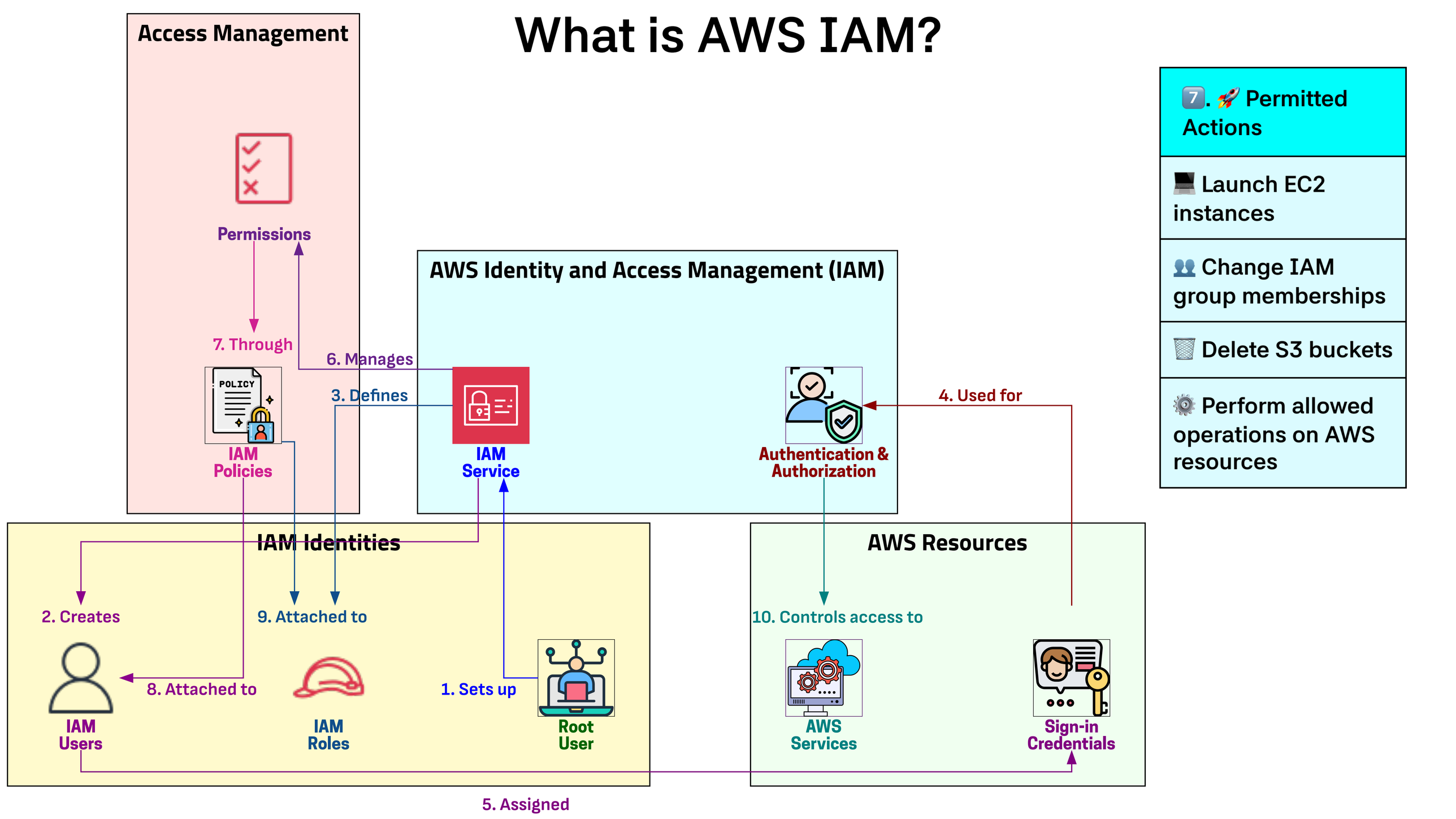
6️⃣. 📜 Authorization Process

🔍 AWS service checks user's permissions

📋 Evaluates policies controlling access level

🚦 Determines if access is allowed or denied

What is AWS IAM?



Service Availability

1.  **Eventually consistent service**:  Replicates data across multiple servers,  Data centers around the world,  Achieves high availability

2.  **Changes take time to propagate**:  Successfully committed changes are safely stored,  Replication across IAM system requires time,  Affects user, group, role, and policy changes

Service Availability

3

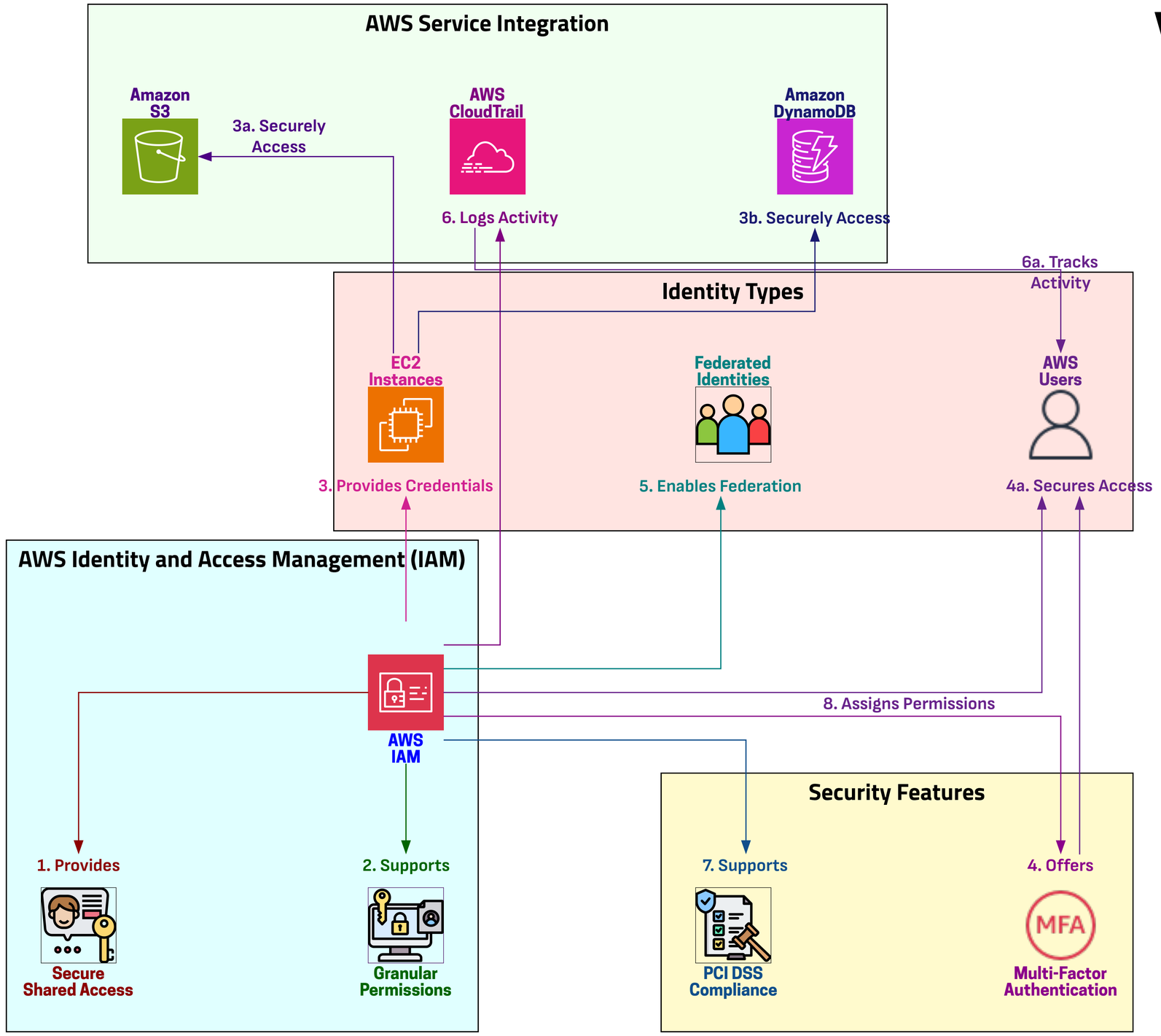
 **Best practices for developers**: ⚠ Avoid IAM changes in critical application paths,  Use separate initialization routines,  Run IAM changes less frequently,  Verify propagation before production dependencies

Cost

1. **AWS Identity and Access Management** (IAM)  , **AWS IAM Identity Center**  and **AWS Security Token Service** (AWS STS)  are **features** of your **AWS account**  offered at **no additional charge**  . You are **charged** only when you **access** other **AWS services**  using your **IAM users** or **AWS STS temporary security credentials**  .

2. **IAM Access Analyzer external access analysis**  is offered at **no additional charge**  . However, you will incur **charges** for **unused access analysis** and **customer policy checks**  .

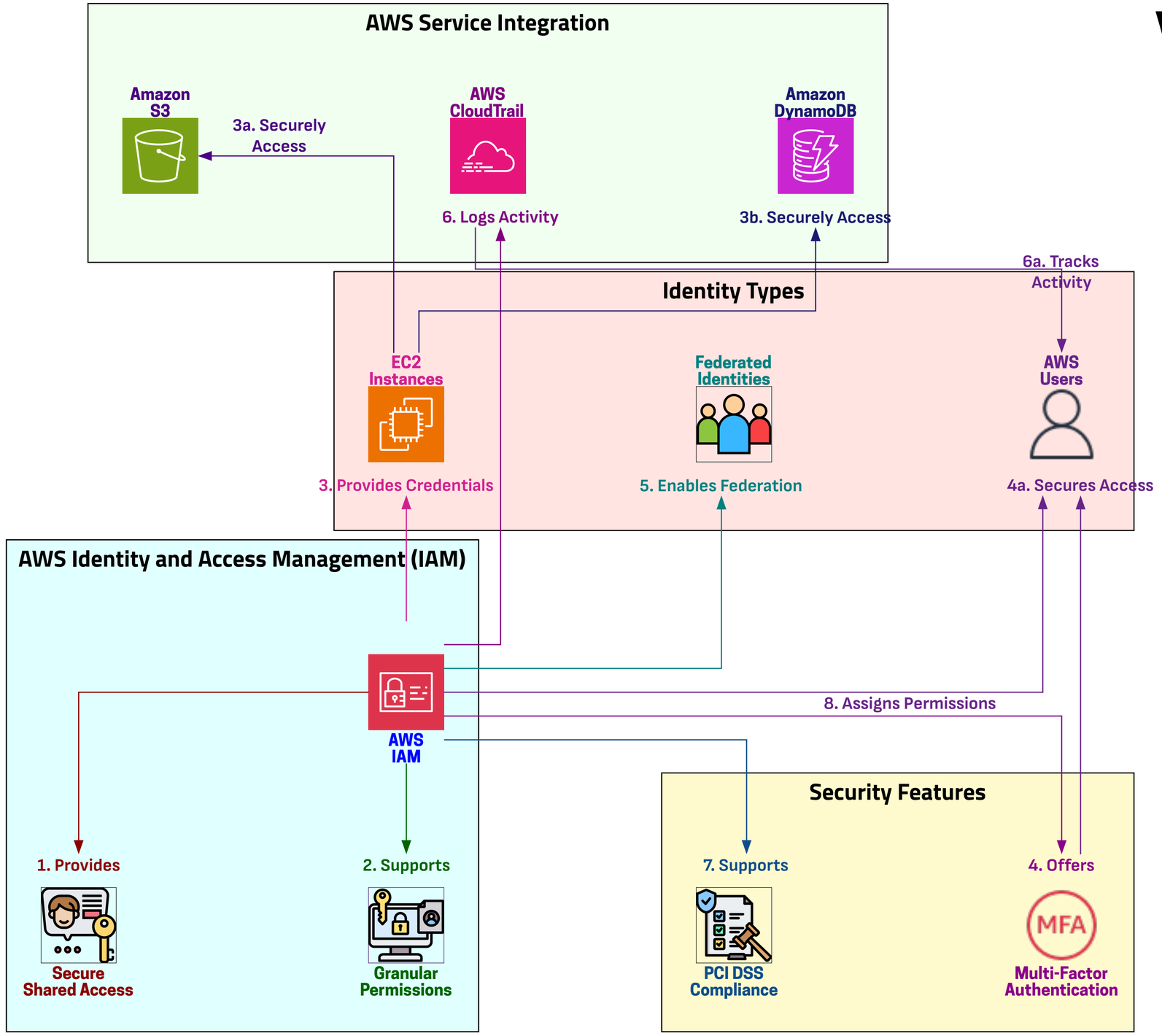
Why should I use IAM?



1

Secure shared access: No sharing primary password, No sharing access keys, Multiple users, one account

Why should I use IAM?

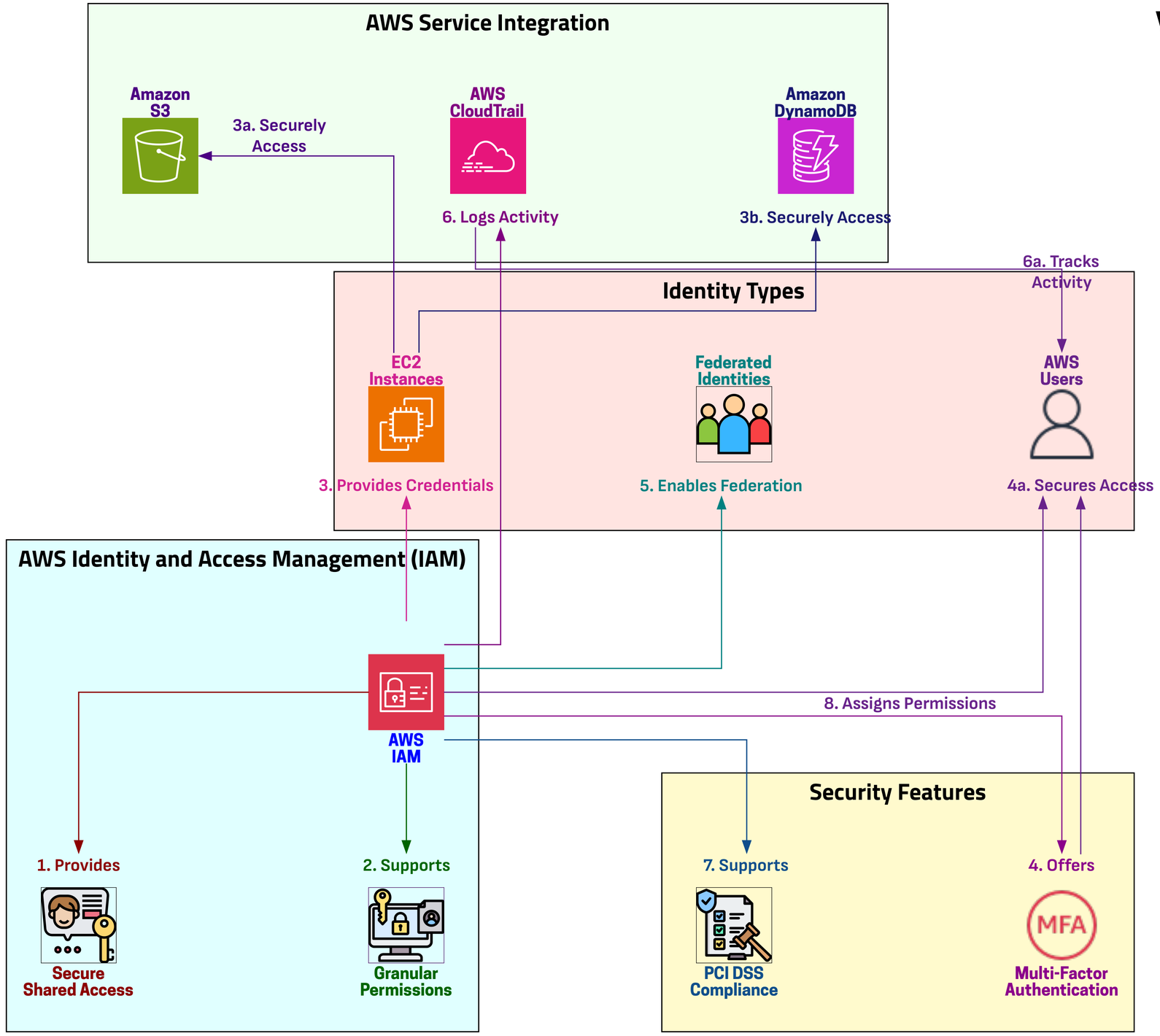


2

Granular permissions:

Different access levels, Resource-specific permissions, User-based controls

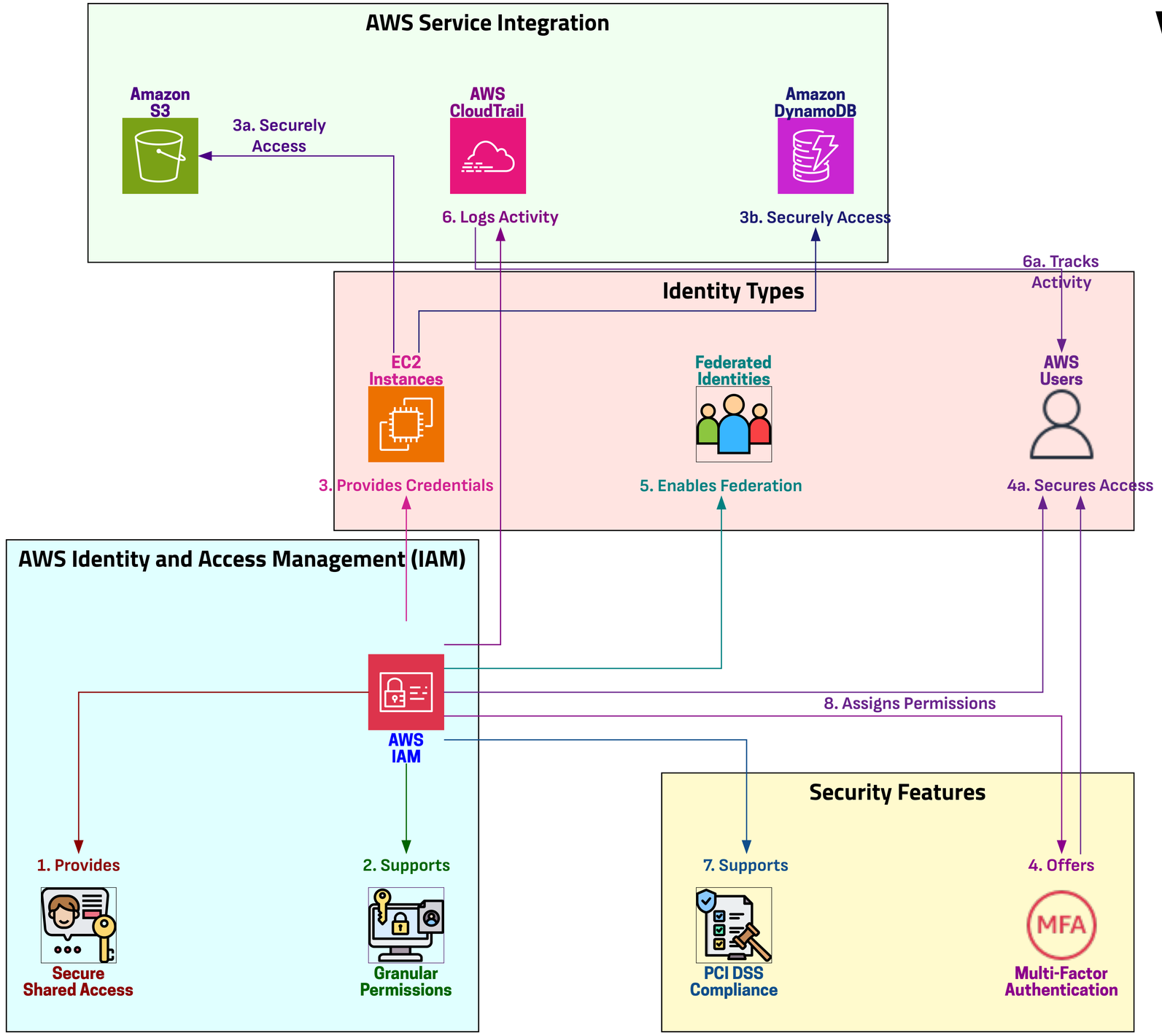
Why should I use IAM?



3

EC2 application credentials: Secure access to AWS resources, S3 bucket access, DynamoDB table access

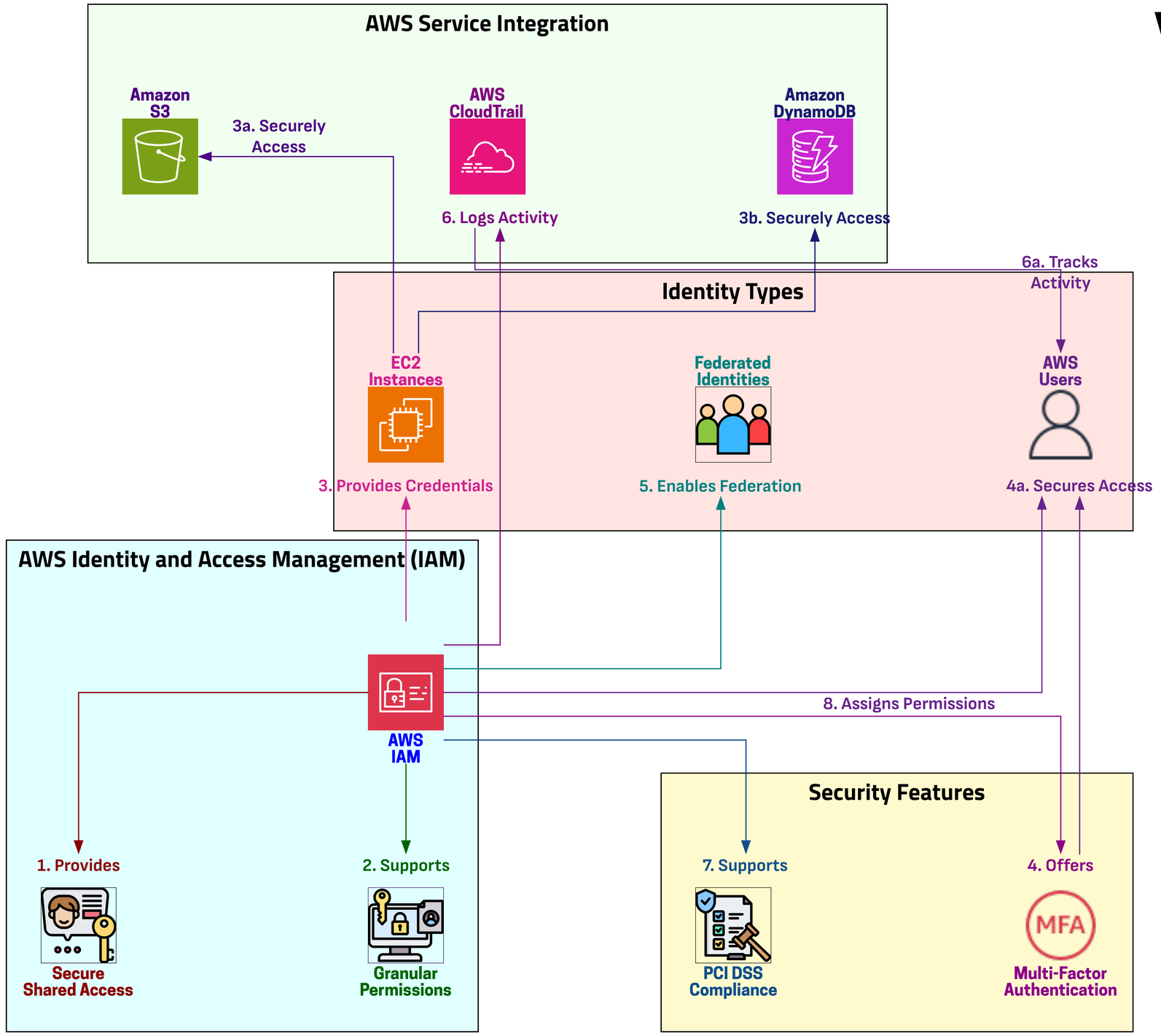
Why should I use IAM?



4

Multi-Factor Authentication: Device code verification, FIDO security key support, Extra security layer

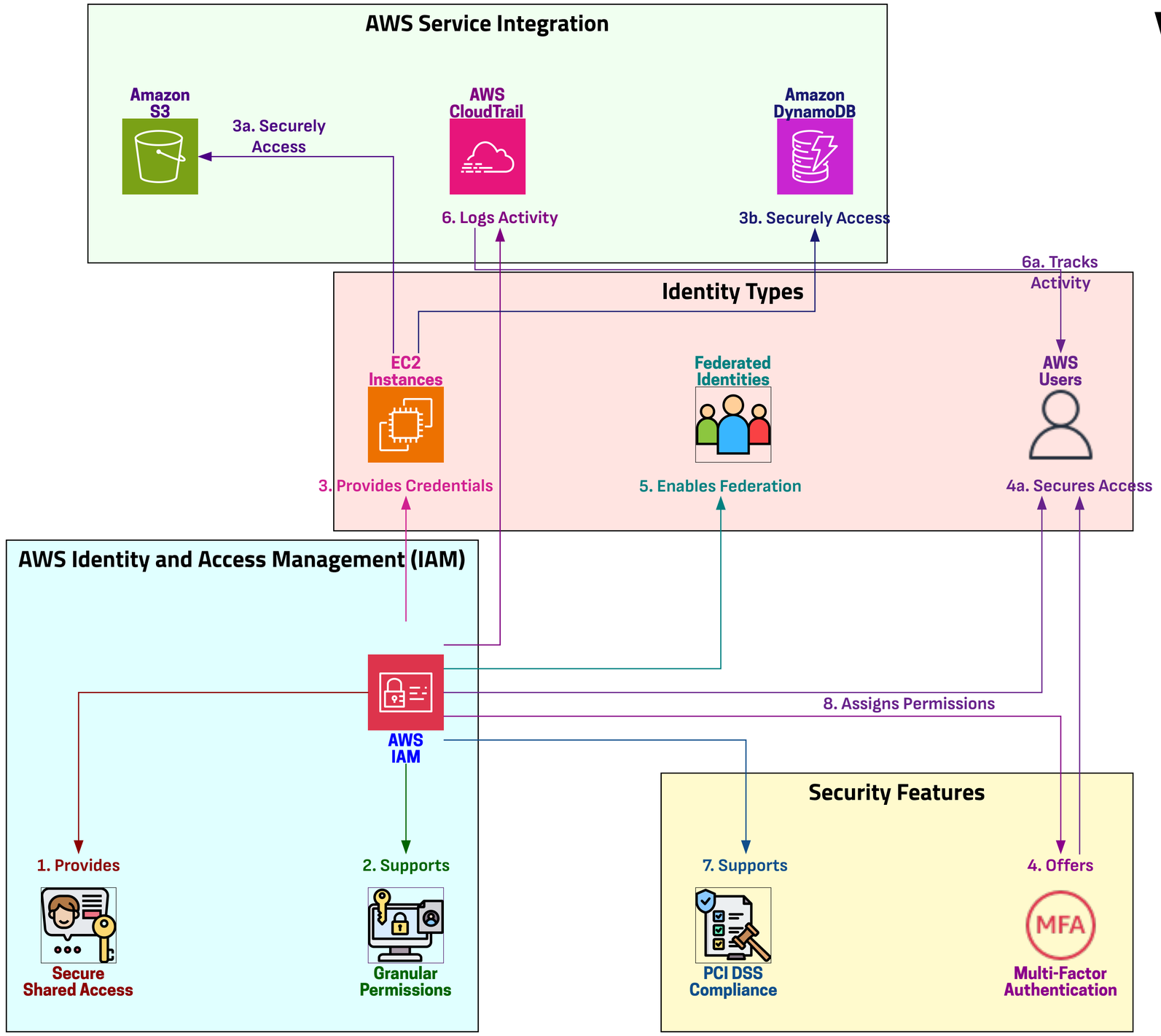
Why should I use IAM?



5

Identity federation: Corporate network identities, Internet identity providers, Temporary AWS account access

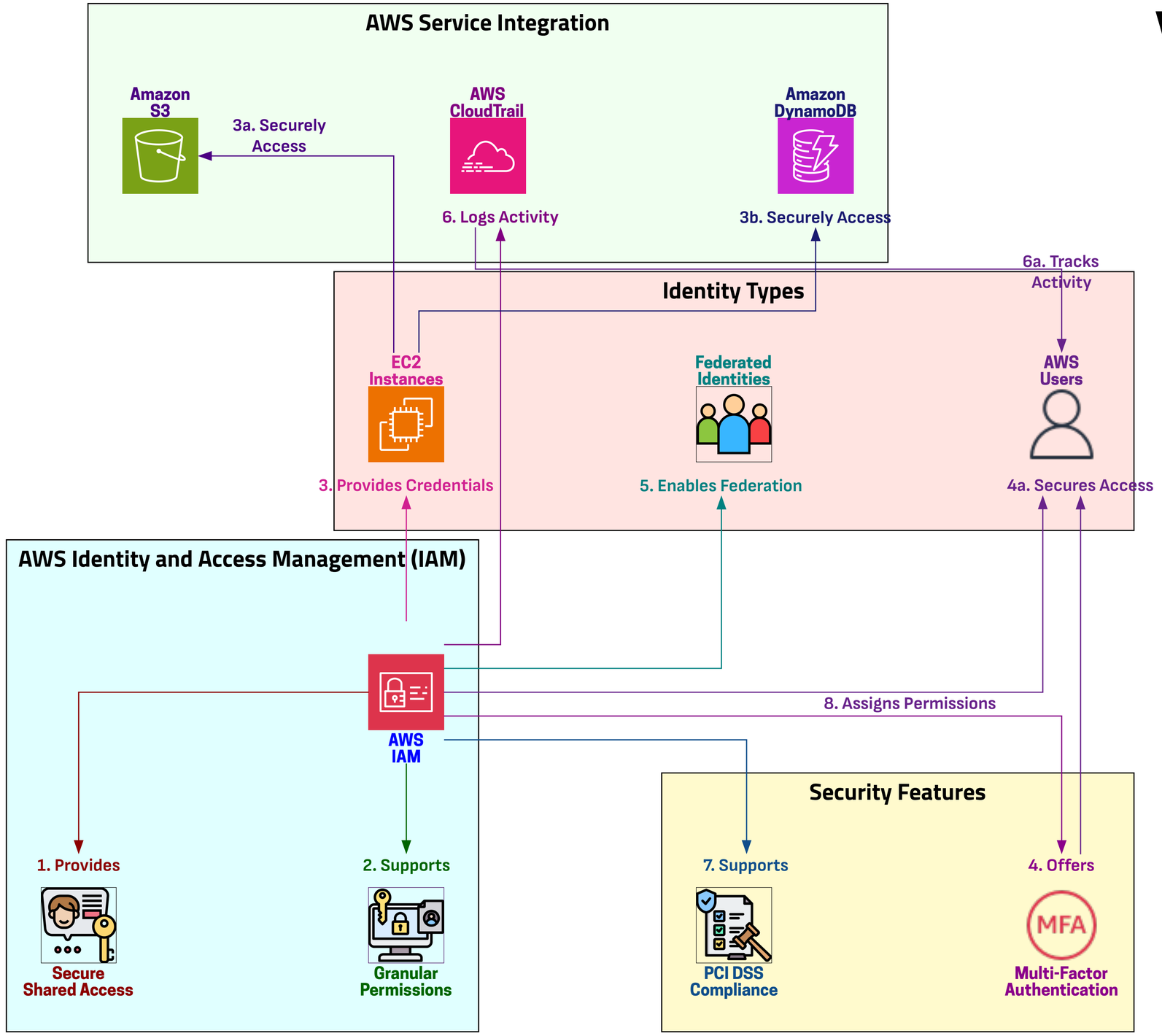
Why should I use IAM?



6

Identity assurance: AWS CloudTrail logs, Records identity-based requests, Aids in auditing

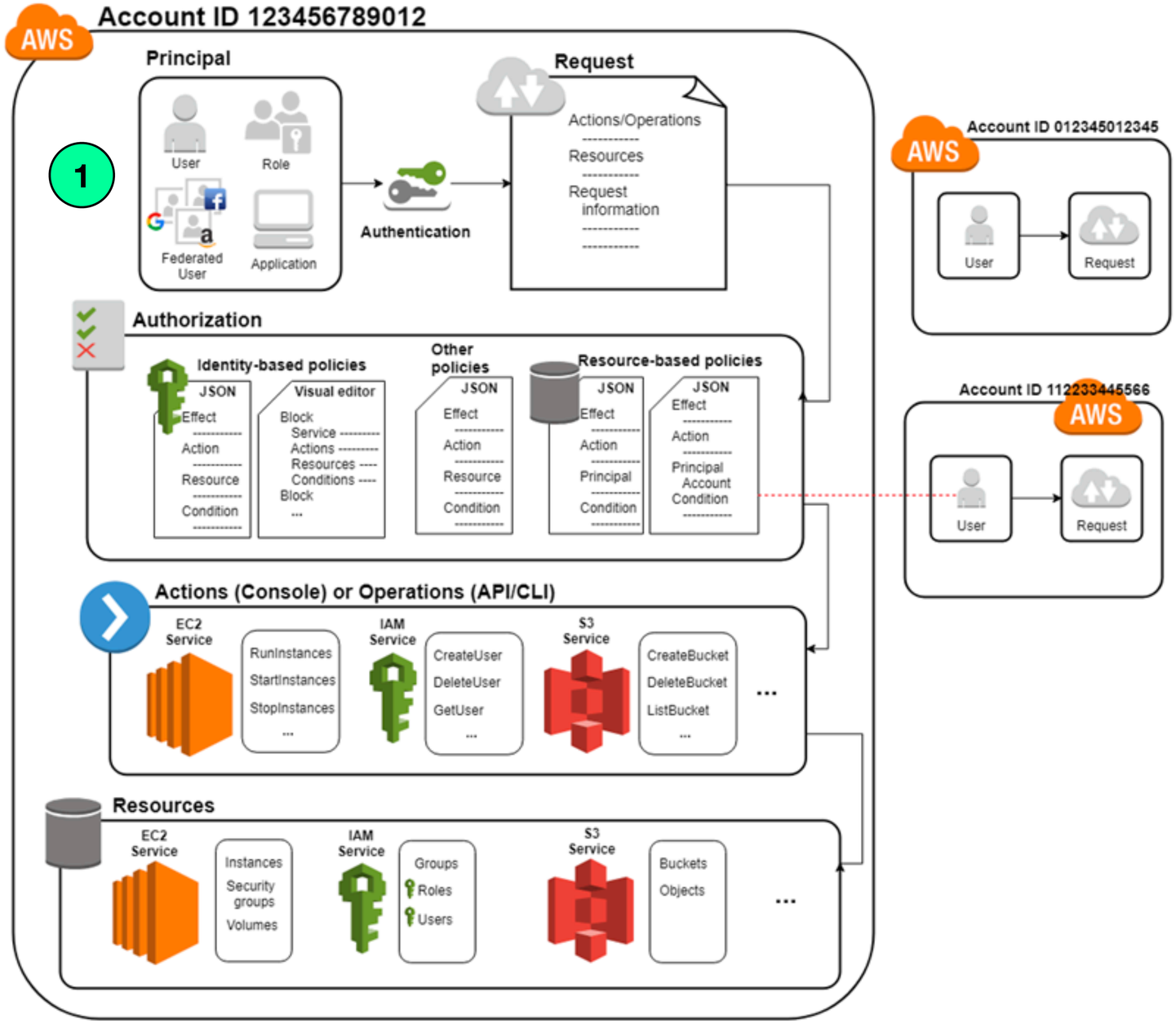
Why should I use IAM?



7

PCI DSS compliance: Secure credit card data processing, Secure data storage, Secure data transmission

How IAM Works



1

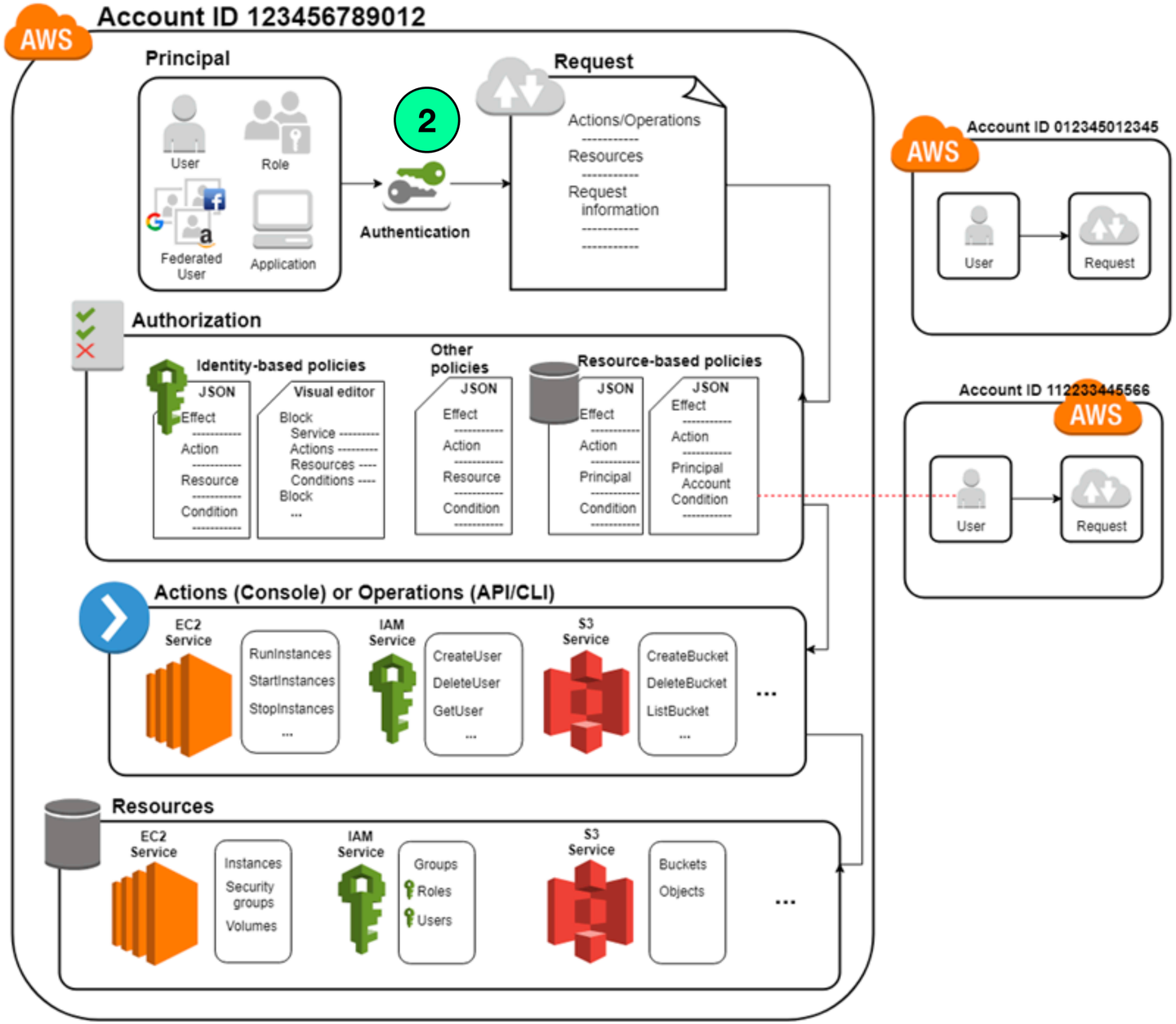
Principal:

Person or application,

Authenticated via AWS account,

Federated user identity

How IAM Works

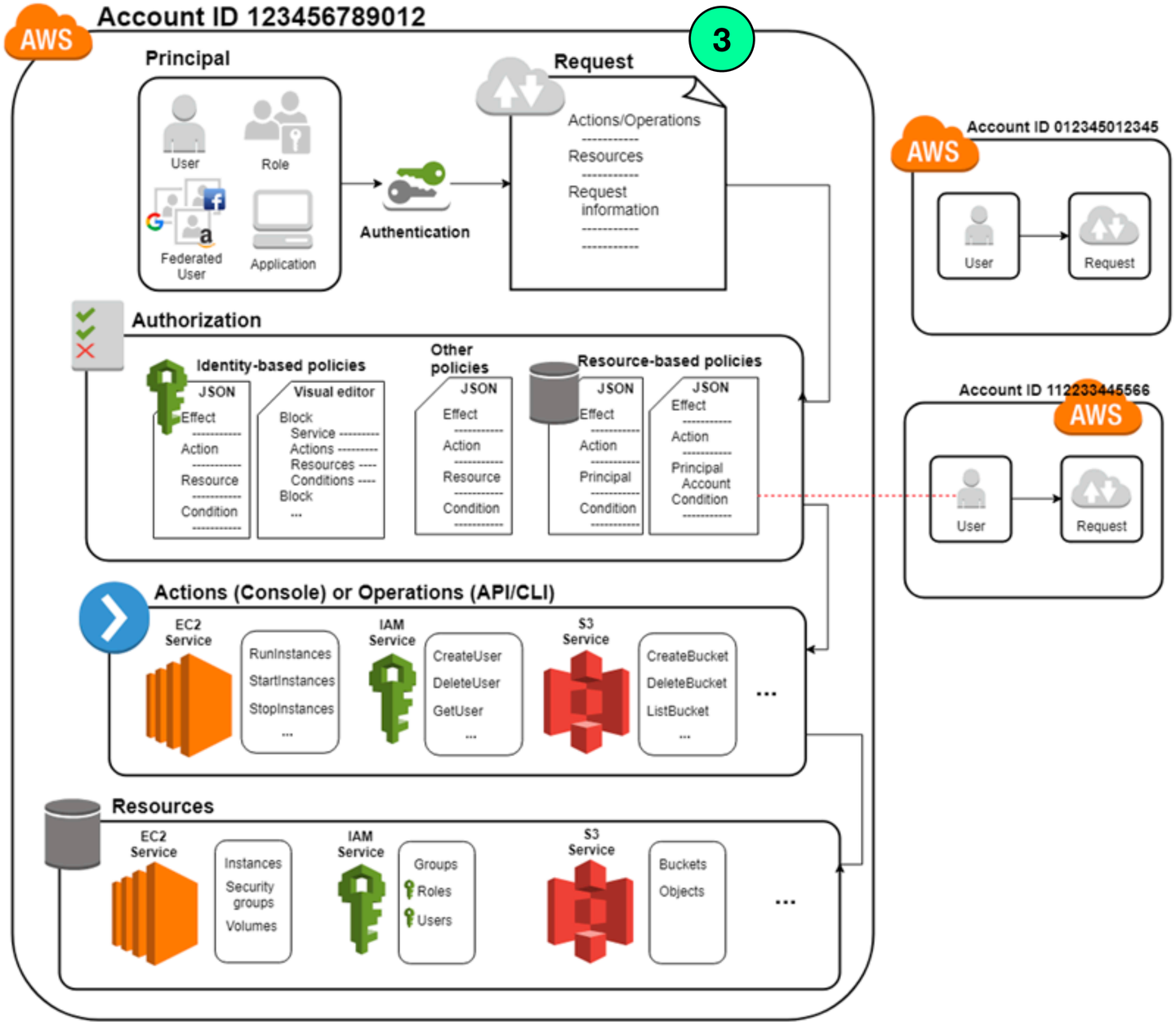


2

Authentication:

Verifies principal's identity, Uses user sign-in credentials, Or assumes a role

How IAM Works



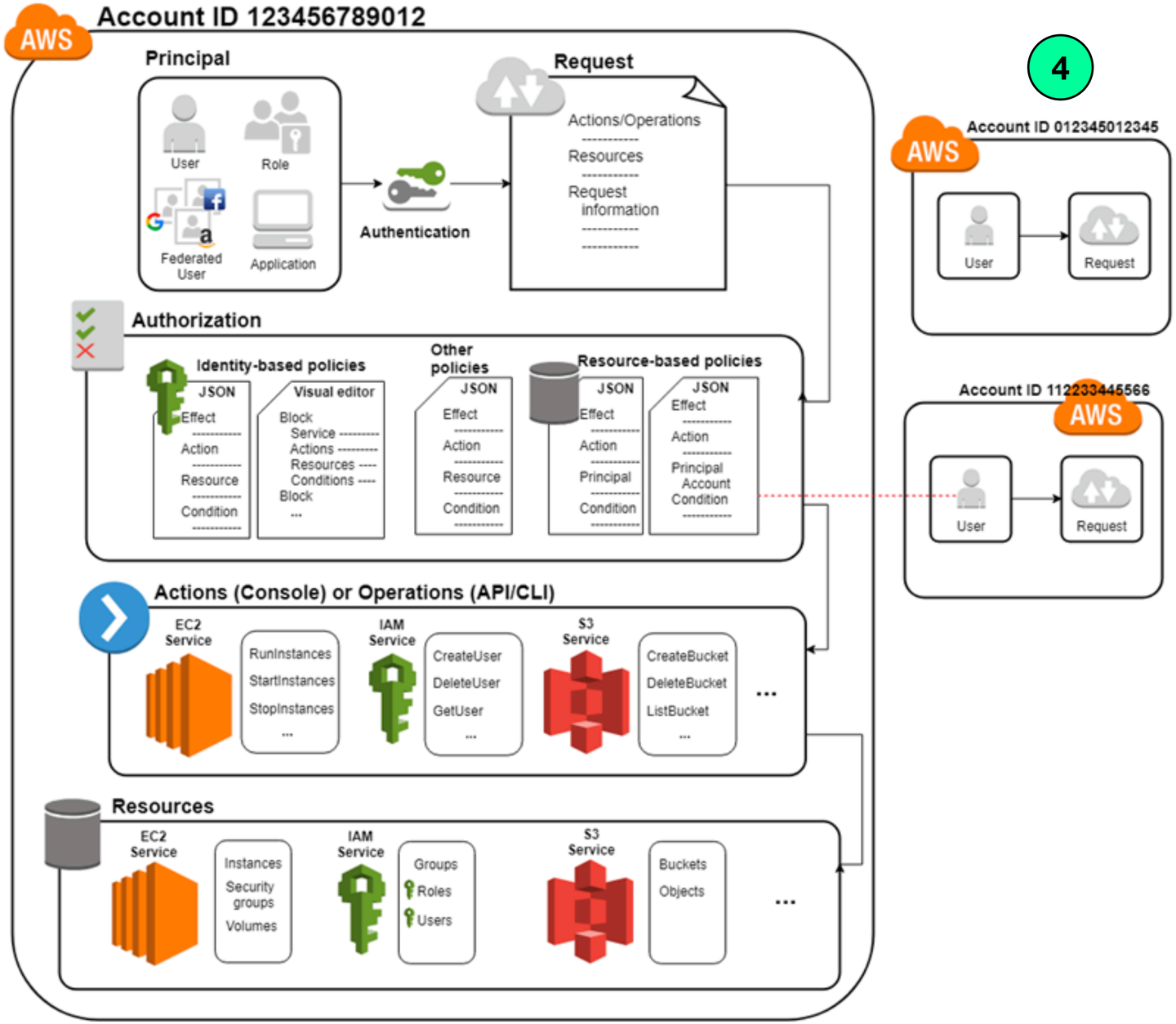
3

Request:

Actions or operations,

Resources to access, S3 bucket or EC2 instance example

How IAM Works

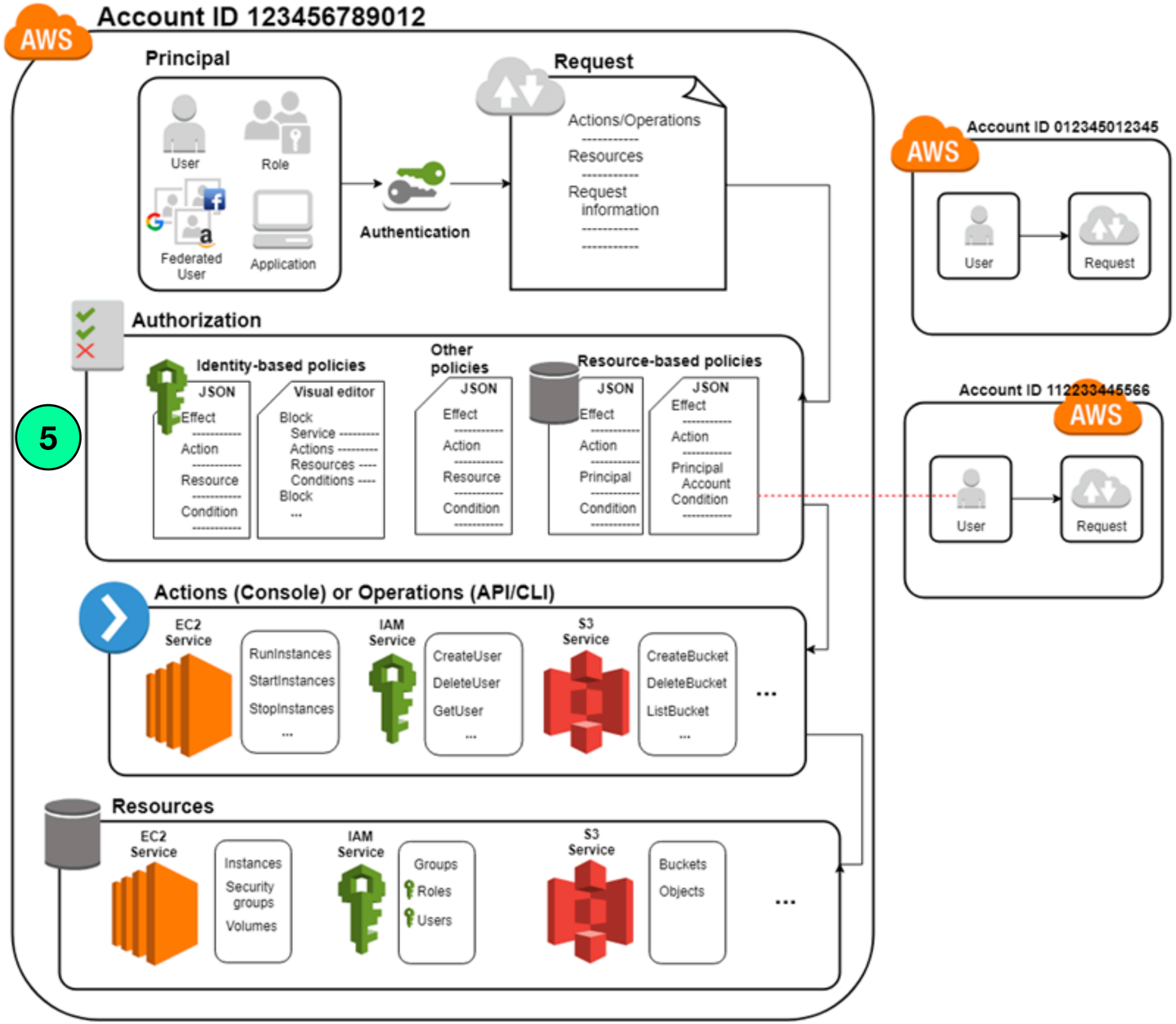


4

Account ID:

Unique AWS account identifier, Request tied to specific account, Specifies organizational context

How IAM Works



5

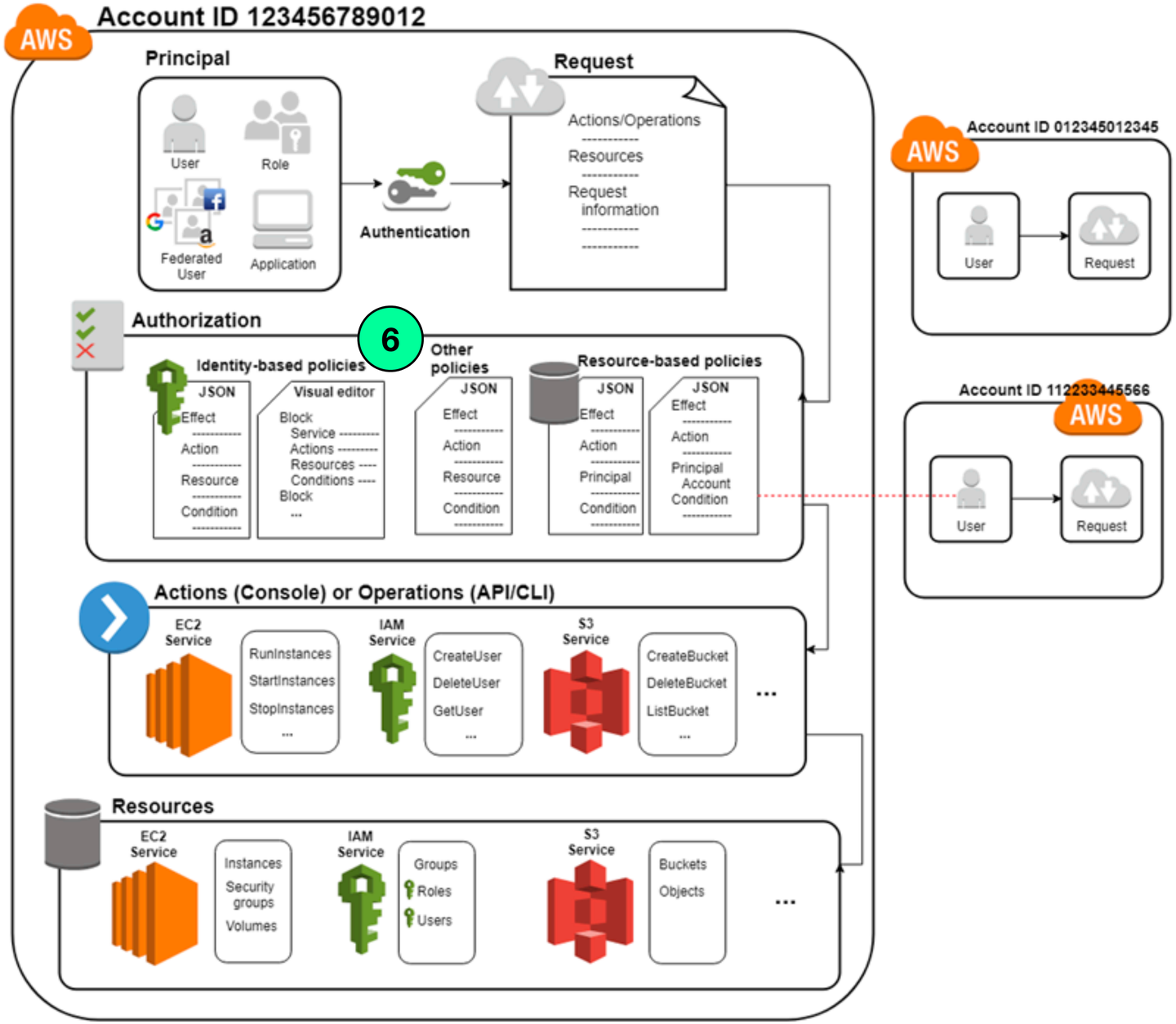
Authorization:

Determines access permissions,

Governed by policies,

Allows or denies actions

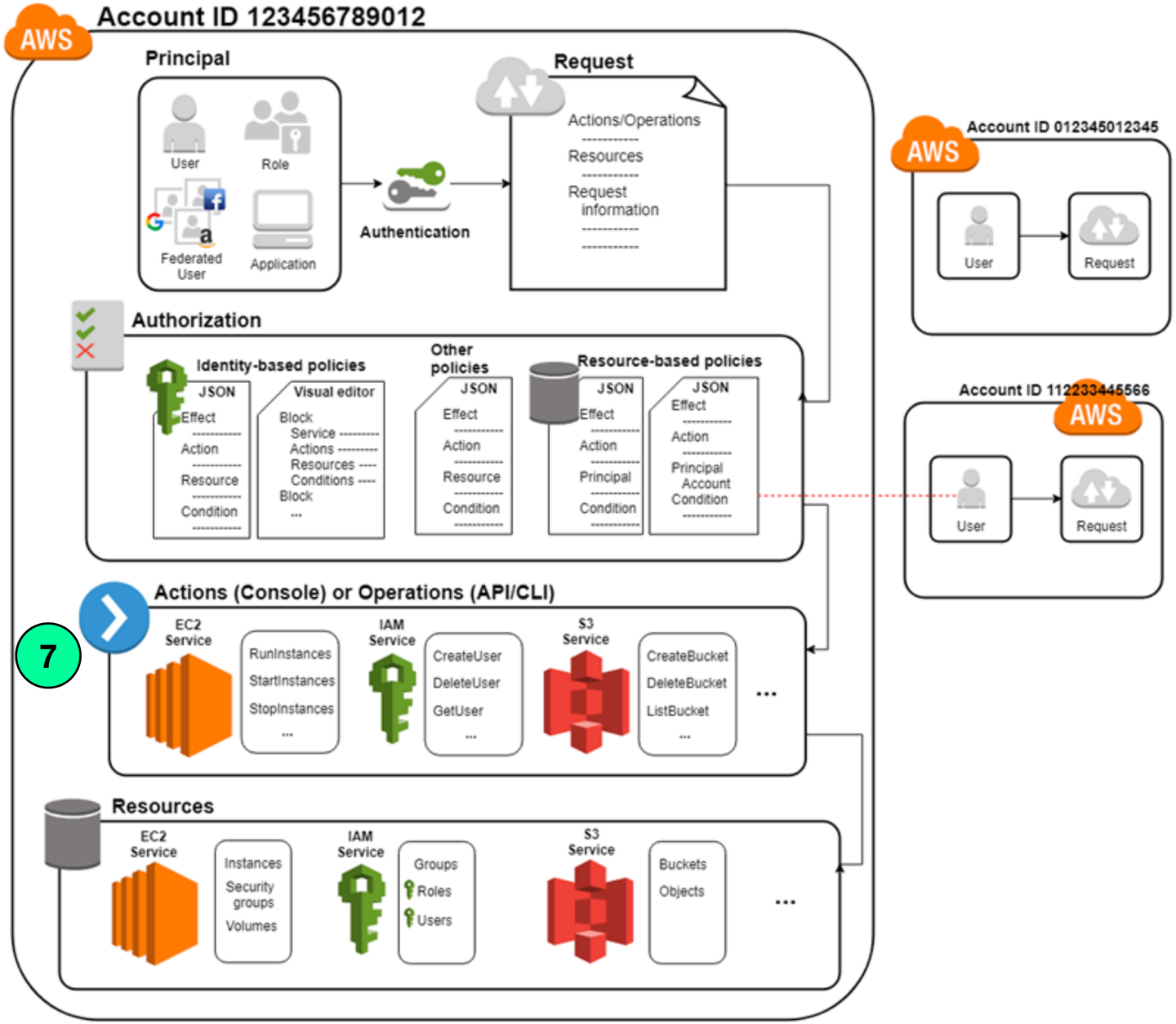
How IAM Works



6

Policy Types: Identity-based policies – attached to IAM entities, Resource-based policies – attached to AWS resources, Other policies include permission boundaries, SCPs, and session policies

How IAM Works



7

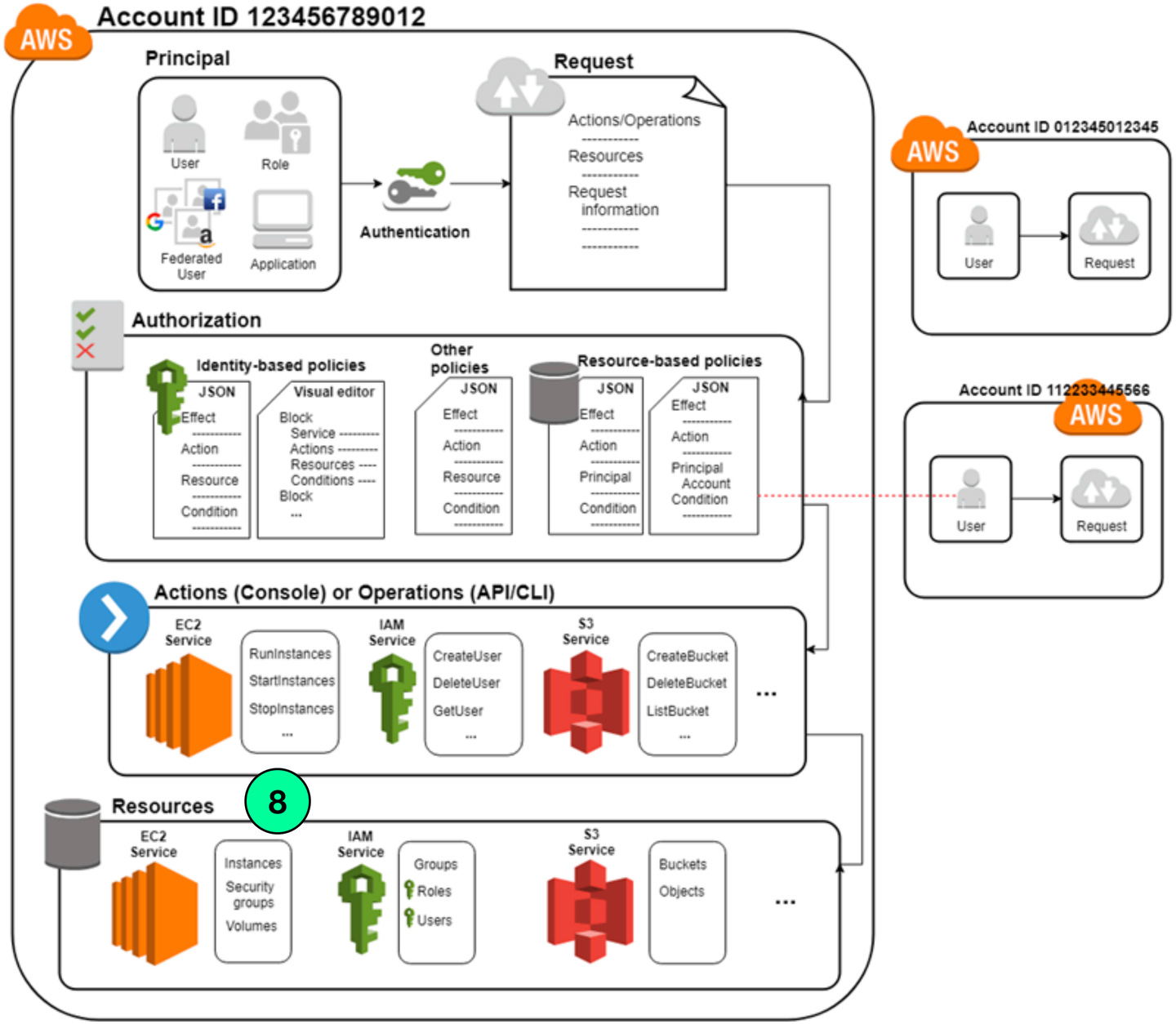
Actions/Operations:

Console actions,

API or CLI operations,

Examples: RunInstances, CreateUser, CreateBucket

How IAM Works



8

Resources: EC2 instances, Security groups, IAM entities (groups, roles, users), S3 buckets or objects

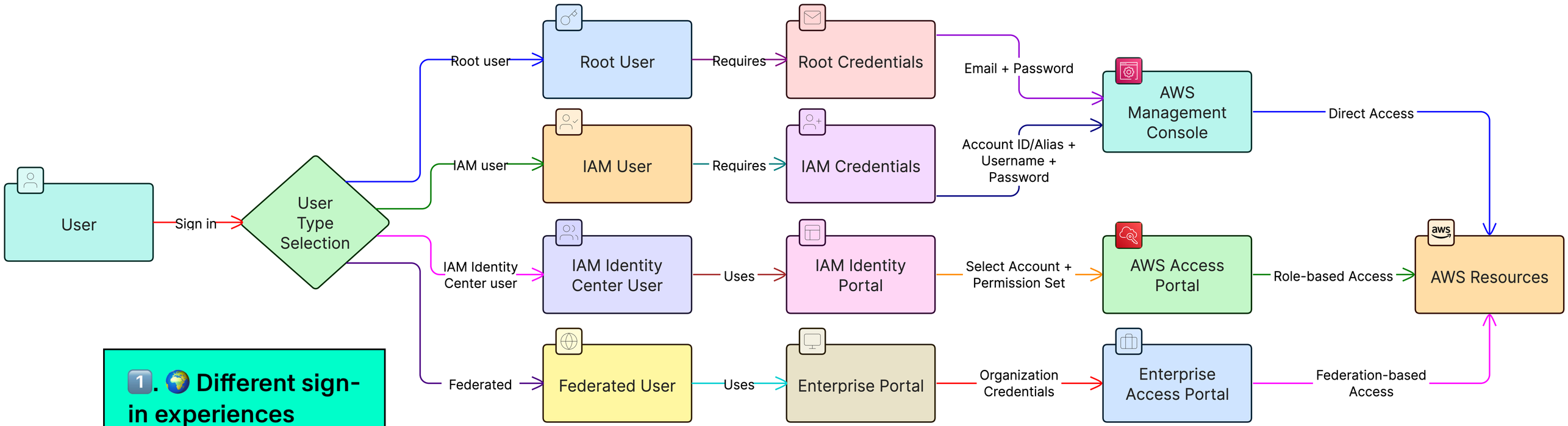
Ways to authenticate

1. **AWS Management Console**: Web-based interface, Username and password, Optional MFA

2. **AWS Command Line Interface**: Terminal-based access, Access keys, Text commands

3. **Software Development Kits**: Programmatic access, API credentials, Multiple programming languages

AWS Console Sign-In Methods



1. 🌍 Different sign-in experiences

- 👤 Various user types
- 🔒 IAM users
- 👑 Root users
- 🏢 IAM Identity Center users
- 🔄 Federated users

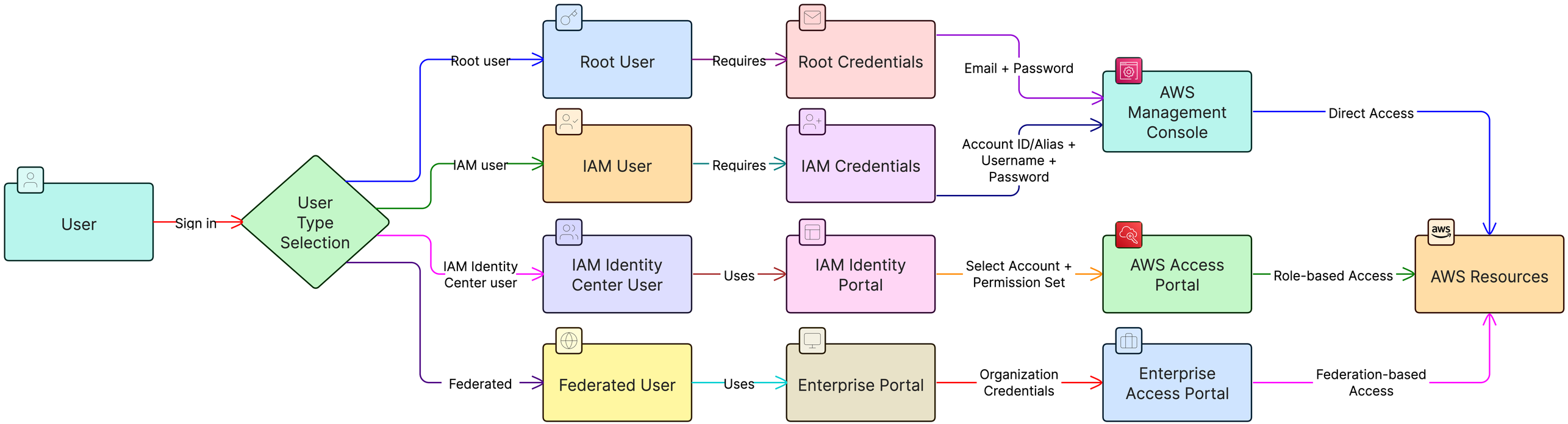
2. 🔑 Root and IAM user sign-in

- 🔗 Main AWS sign-in URL
- 🛡️ Access to account resources
- 🔒 Based on permissions

3. 📧 Root user credentials requirement

- 📮 Email address
- 🔐 Password
- 👑 Associated with root account

AWS Console Sign-In Methods



4️⃣. 👨‍💼 IAM user credentials requirement
🔢 AWS account number or alias
👤 User name
🔐 Password

5️⃣. 🏢 IAM Identity Center user sign-in
🌐 Unique AWS access portal
📋 Account/application selection
🔒 Permission set selection

6️⃣. 🌐 Federated user sign-in
🔄 Enterprise access portal
🏢 Based on organization policies
🔗 Access to AWS resources

Sign-In and Account ID Access for IAM Users

1	**Sign-in with URL: Account alias or ID**: IAM users sign in through web URL, Requires account alias or ID, Ensures direct access with secure identification
2	**Viewing Your Account ID: User type specific**: Different user types have distinct methods, View account ID within AWS console, Tailored to access level and roles

```
https://Your_Account_ID.signin.aws.amazon.com/console/
```

```
https://211125437318.signin.aws.amazon.com/console
```

```
https://Your_Account_Alias.signin.aws.amazon.com/console/
```

```
https://rehearsal-1.signin.aws.amazon.com/console
```

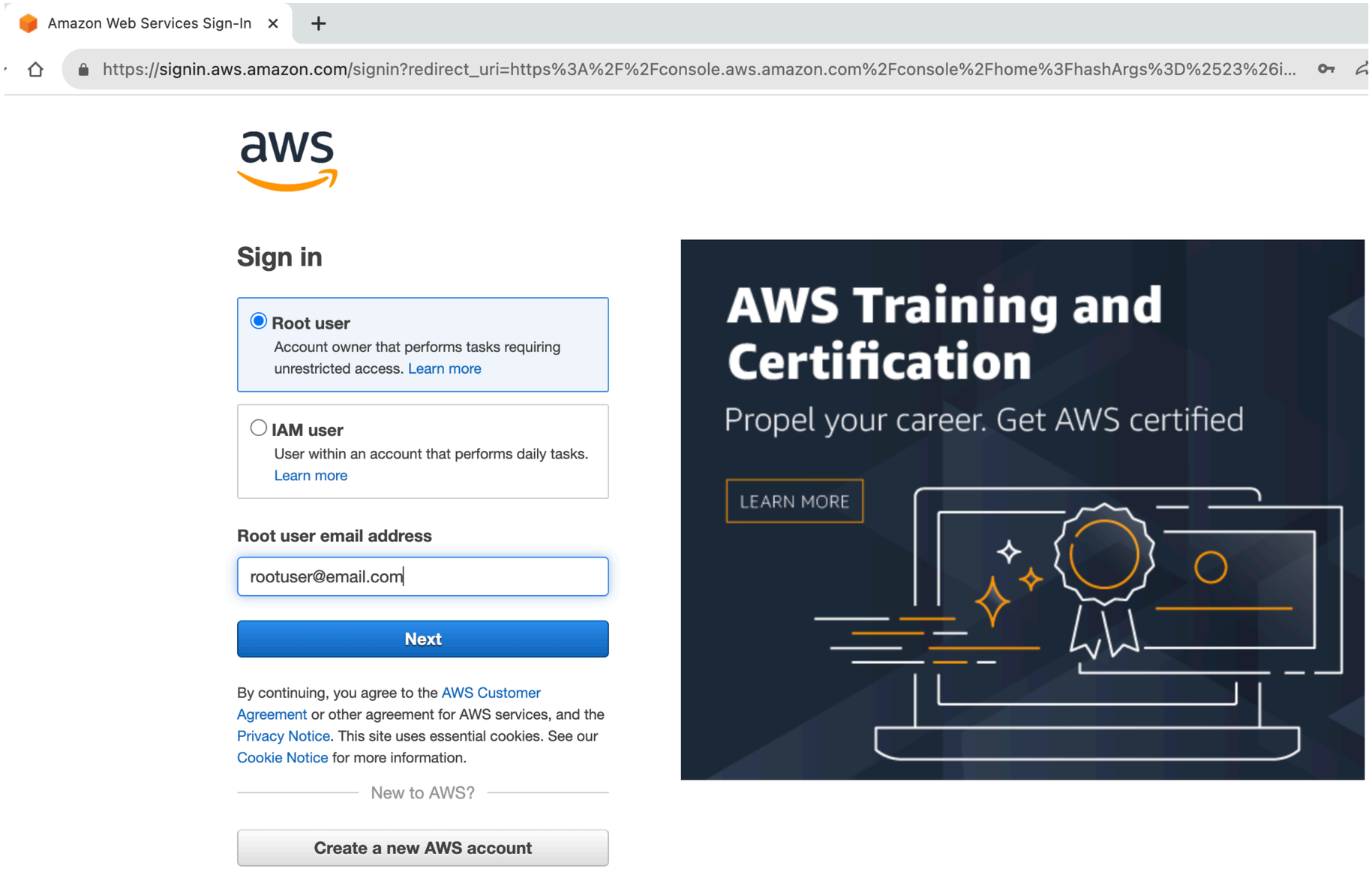
Sign-In and Account ID Access for IAM Users

3 **Root User: Security credentials navigation**: Navigate to username, Select Security credentials, Find under Account identifiers

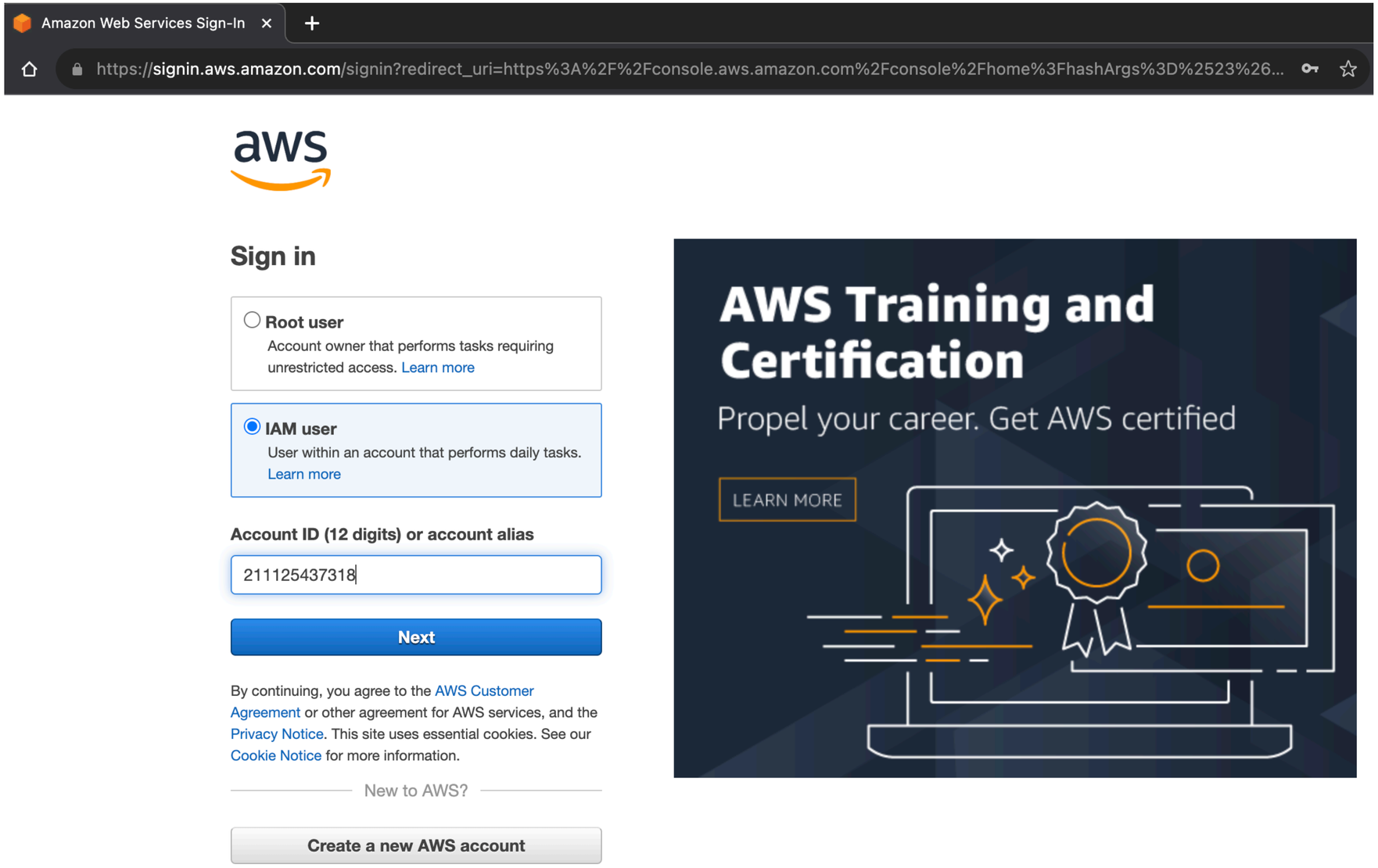
4 **IAM User: Account details access**: Similar to root users, Select Security credentials from user menu, Locate under Account details

5 **Assumed Role: Support Center method**: Choose Support option, Access Support Center, 12-digit account number displayed in navigation pane

Root User Login



IAM User Login



aws

Sign in as IAM user

Account ID (12 digits) or account alias

211125437318

IAM user name

iamuser001

Password

••••••••••

Remember this account

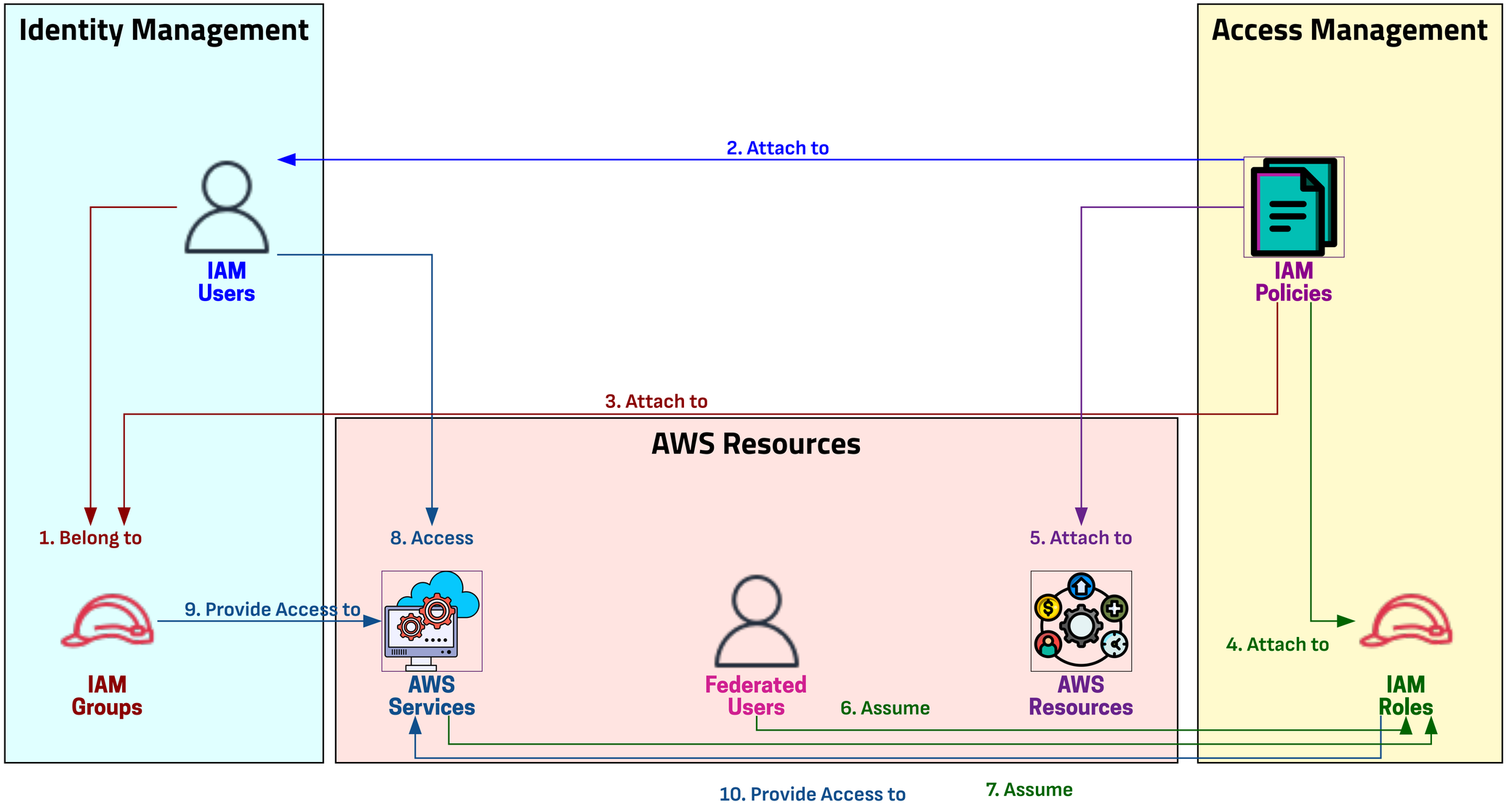
Sign in

Sign in using root user email

Forgot password?

English

IAM Components



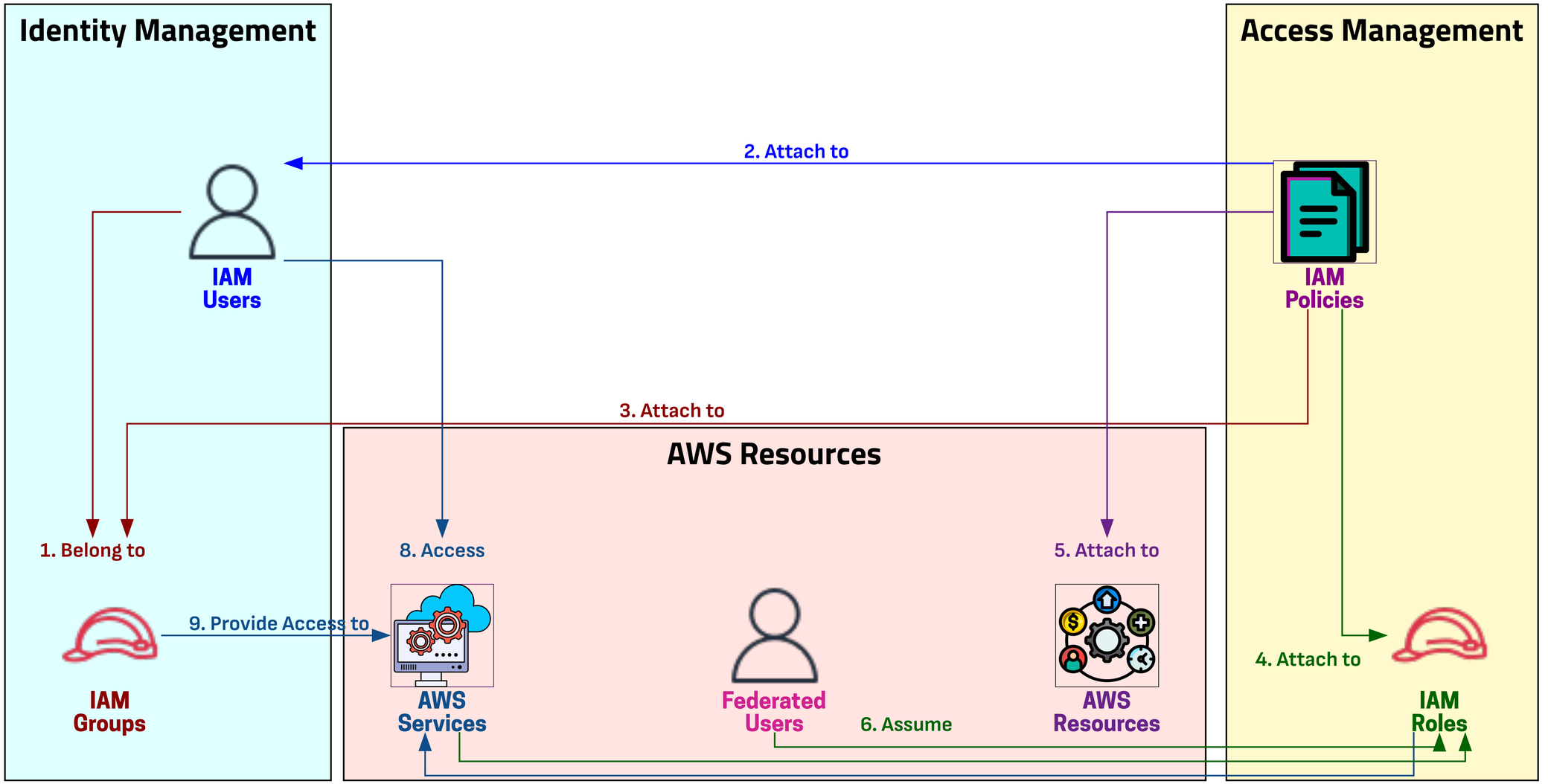
1. 👨 IAM Users

- 👤 Individual or application identities
- 🔑 Unique permissions
- 🖥️ Direct interaction with AWS services

2. 🤝 IAM Groups

- 👥 Collections of IAM users
- 📋 Single set of permissions
- 🔄 Simplify permission management
- 🛡️ Enhance organizational security

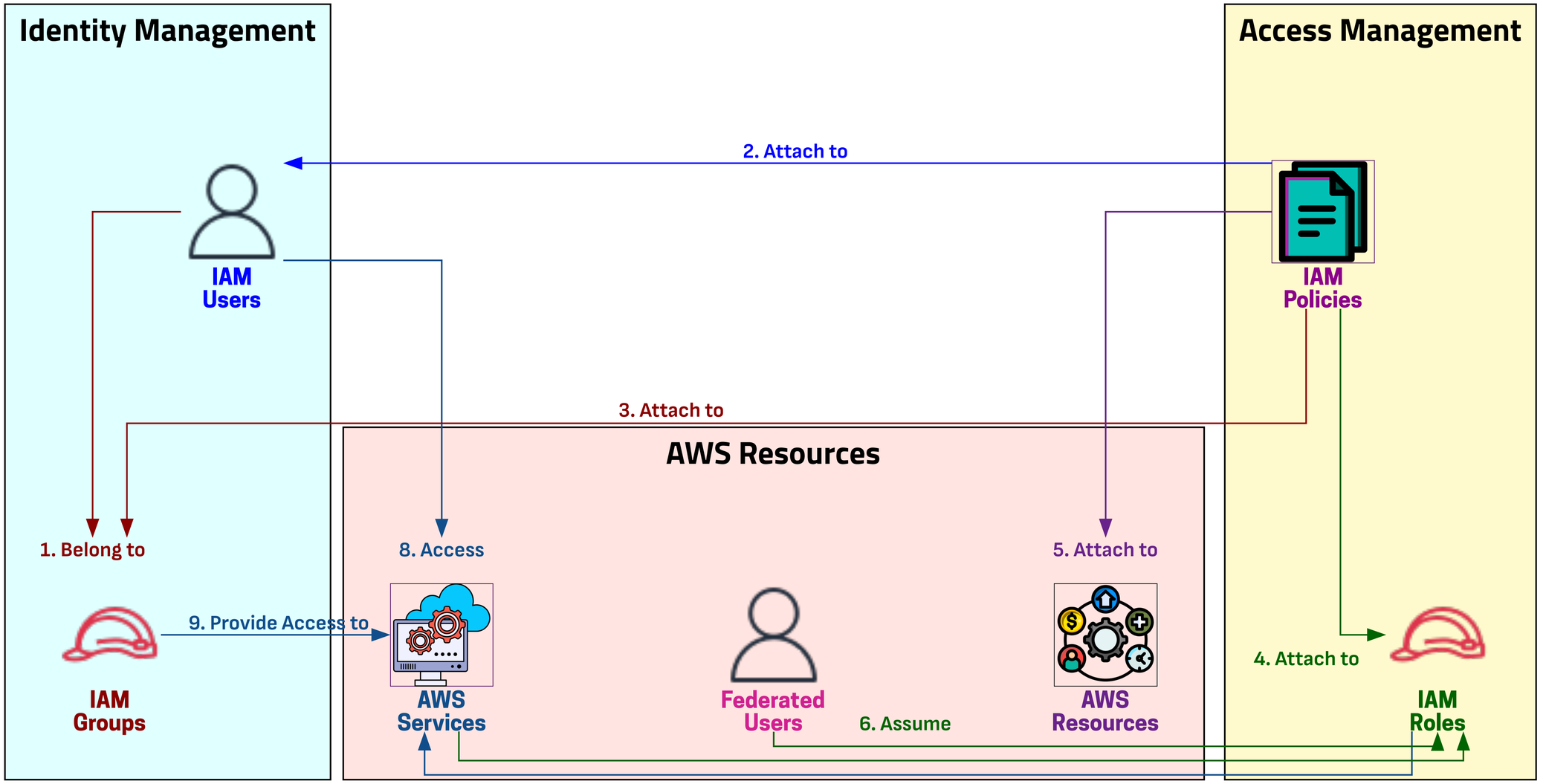
IAM Components



3. 🎭 IAM Roles

- ⏱️ Temporary security credentials
- 🔄 No permanent credentials needed
- 🌐 Assumed by trusted entities
- 🔐 Secure and flexible access

IAM Components



4. 📜 IAM Policies

- 📝 Define permissions explicitly
- 🔗 Attached to identities or resources
- ✅ Detail allowed actions
- ✕ Specify denied actions
- 🔒 Central to security management

IAM Users

1. **IAM User Identity**: Created with friendly name, Unique ARN identifier, Used in policies and AWS environments

2. **IAM User Credentials**: Console passwords for AWS Management Console, Access keys for programmatic calls, SSH keys for CodeCommit, Server certificates for SSL/TLS

3. **Administering IAM Security**: Tools for managing credentials, Multi-factor authentication (MFA), Regular security audits, Removal of unused credentials

IAM Users

4 **Default Permissions**: No permissions by default, Explicit permission assignment, Principle of least privilege

5 **IAM Users and AWS Accounts**: Single AWS account association, Resource access facilitation, Activity billing without separate payment

6 **Service Accounts**: Application identities, Long-term AWS credentials, Avoid embedding access keys in code, Use AWS SDKs for secure key management

IAM User Groups

1. **Definition of IAM User Groups**: Collection of IAM users, Simplifies permission management, Collectively assigns permissions

2. **Managing Permissions Efficiently**: User groups like 'Admins', Streamlined management, Easy addition or removal of users, Based on role changes

3. **Attaching Identity-Based Policies**: Policies attached to groups, Grant shared permissions, Cannot be specified as principals, Related to permissions, not authentication

IAM User Groups

4

 **Important Characteristics**:  Can include many users,  Users can belong to multiple groups,  No nesting of groups,  No default group for all users

5

 **Limits and Restrictions**:  Limited number of groups per account,  Limited groups per user,  Requires careful planning,  Necessitates management within IAM

IAM Roles

1. **IAM Role Identity**: Designed to be assumable by anyone, No permanent credentials, No passwords or access keys

2. **Temporary Security Credentials**: Provided when assuming a role, Secure, time-limited access, Temporary AWS resource access

3. **Use Cases for IAM Roles**: Delegation within same account, Access across different accounts, Access to AWS services, External user authentication

IAM Roles

4 **Cross-Account Access**: Access resources across accounts, Users from one account access another, Enhanced collaboration, Improved resource sharing

5 **Mobile and External User Access**: Grant access to mobile apps, No embedded AWS keys needed, Corporate directory integration, Third-party identity services

Thanks for Watching

Part 2 of 2

AWS Identity and Access Management (IAM)

Roles terms and concepts Explained

1. **Service Role**  : A **service role** allows **AWS services** to **perform actions** on your **behalf**, managed within **IAM** for **creation**, **modification**, and **deletion**  .

2. **EC2 Instance Role**  : A **specific service role** for **applications** on **EC2 instances** to **perform actions** in your **account**, using **temporary security credentials**  .

3. **Service-Linked Role**  : A **role** linked to an **AWS service**, allowing it to **act on your behalf**. These roles are **visible** in your **account** but cannot be **edited** by **IAM administrators**  .

4. **Role Chaining**  : The **process** of using a **role** to **assume another role** through **AWS CLI** or **API**, allowing for **complex permissions management**  .

Roles terms and concepts Explained

5. **Delegation** : **Granting permissions** to **access** your **resources** to another **account**, involving a **trust policy** and a **permissions policy** .

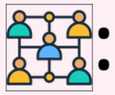
6. **Federation & Federated User** : **Establishing** a **trust relationship** with **external identity providers** for **user access** to **AWS resources** without **creating** an **IAM user** .

7. **Trust & Permissions Policy** : **Trust policy** defines **who can assume** the **role**, while **permissions policy** outlines the **role's access capabilities** .

8. **Permissions Boundary** : A **feature** to **limit** the **maximum permissions** a **role** can have, not **applicable** to **service-linked roles** .

Roles terms and concepts Explained

9. **Principal**  : An **entity** (**user**, **role**, **account**) in **AWS** that can **perform actions** and **access resources**, **defined** in **policies**  .

10. **Role for Cross-Account Access**  : A **role** that enables **access** to **resources** across different **AWS accounts**, facilitating **secure collaboration**  .

Types of Policies

1. **Identity-based Policies**: Attach to IAM identities, Users, groups, or roles, Grant permissions directly

2. **Resource-based Policies**: Attach to AWS resources, Example: S3 buckets, Grant to any principal, Allow cross-account access

3. **Permissions Boundaries**: Define maximum permissions, Act as a limit, Don't grant permissions directly

Types of Policies

4 **Organizations SCPs**: Control maximum permissions, For AWS Organization members, Limit account entities

5 **ACLs**: Grant cross-account access, Without using JSON, Not for same-account permissions

6 **Session Policies**: Limit permissions for sessions, For AWS CLI or API, Restrict assumed role, Restrict federated user

Identity-based Policies in AWS

1️⃣. 📝 JSON permissions policy documents

📋 Specify what actions identities can execute

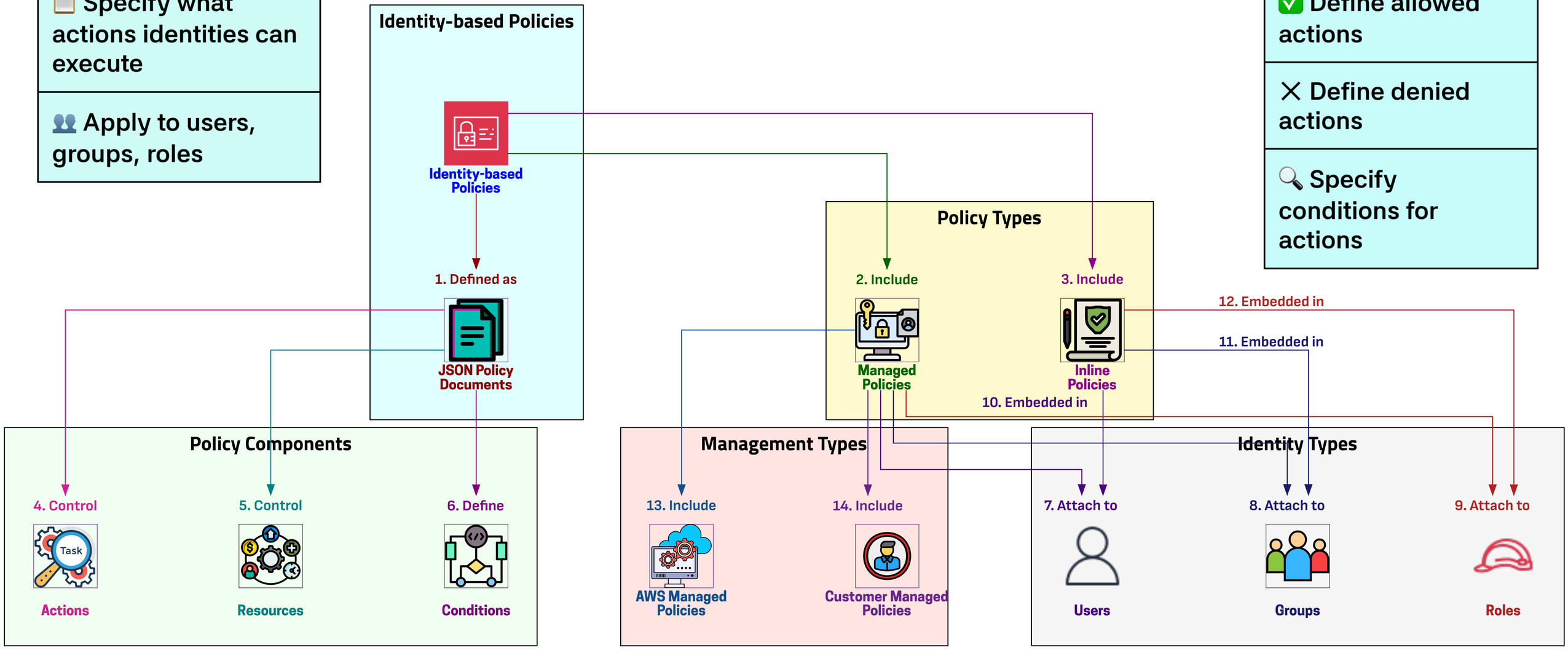
👥 Apply to users, groups, roles

2️⃣. 🎮 Control actions, resources, conditions

✅ Define allowed actions

✕ Define denied actions

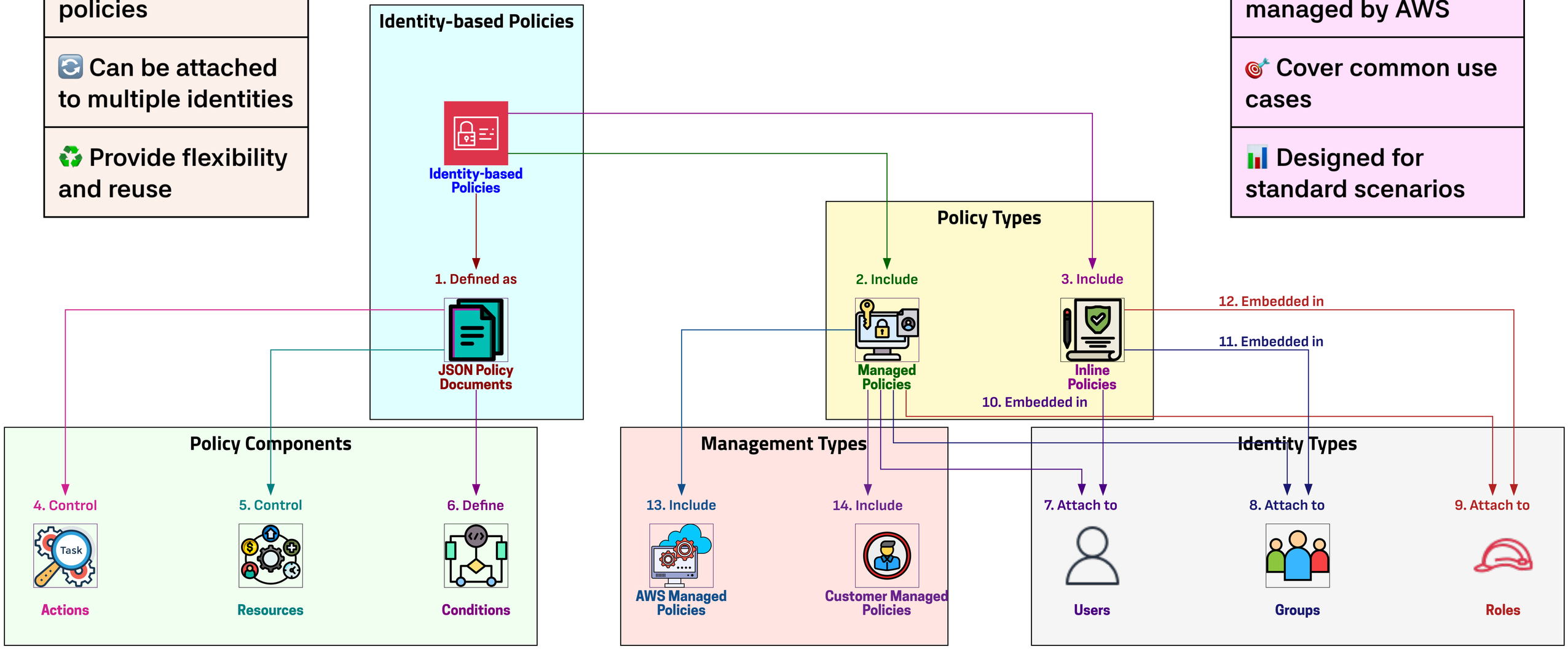
🔍 Specify conditions for actions



Identity-based Policies in AWS

3️⃣. 📊 Managed policies
🧩 Standalone policies
🔄 Can be attached to multiple identities
♻️ Provide flexibility and reuse

4️⃣. ☁️ AWS managed policies
🛠️ Created and managed by AWS
🎯 Cover common use cases
📊 Designed for standard scenarios



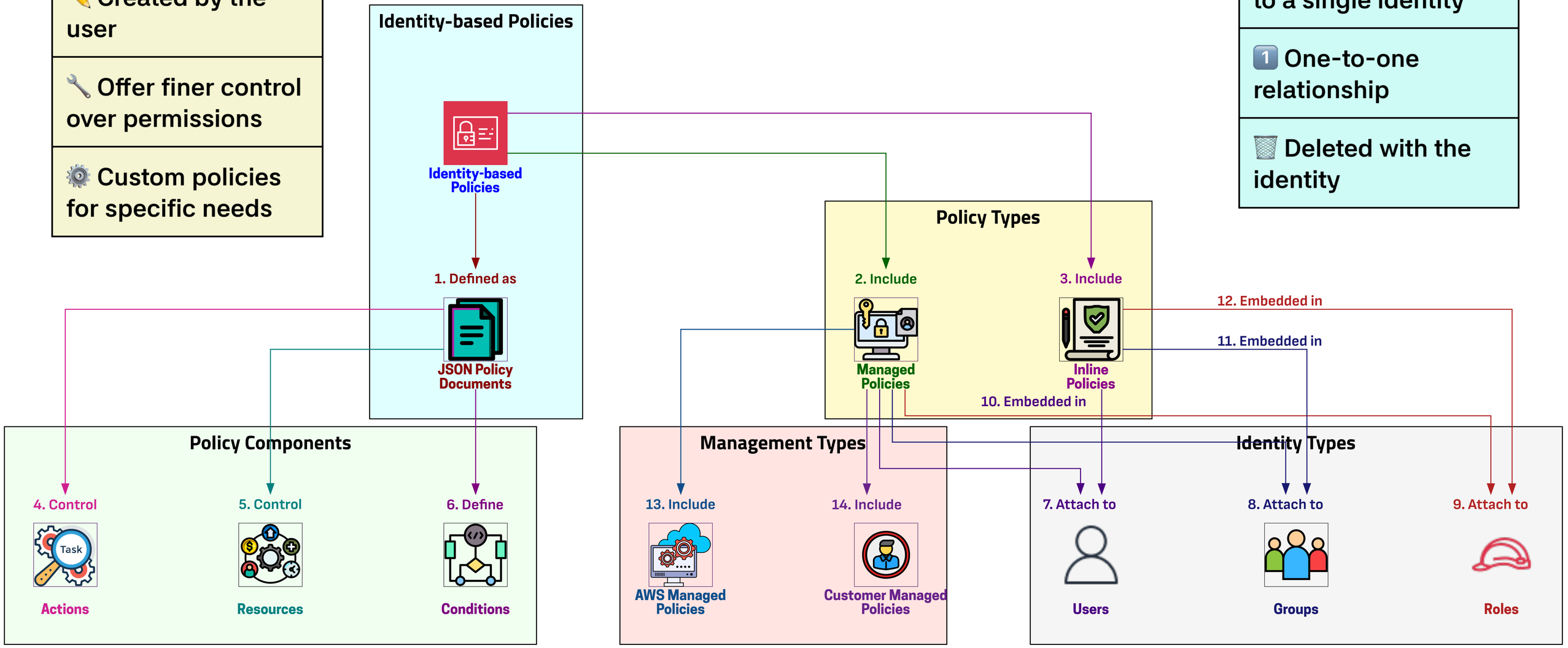
Identity-based Policies in AWS

5️⃣. 👤 Customer managed policies

- ✏️ Created by the user
- 🔧 Offer finer control over permissions
- ⚙️ Custom policies for specific needs

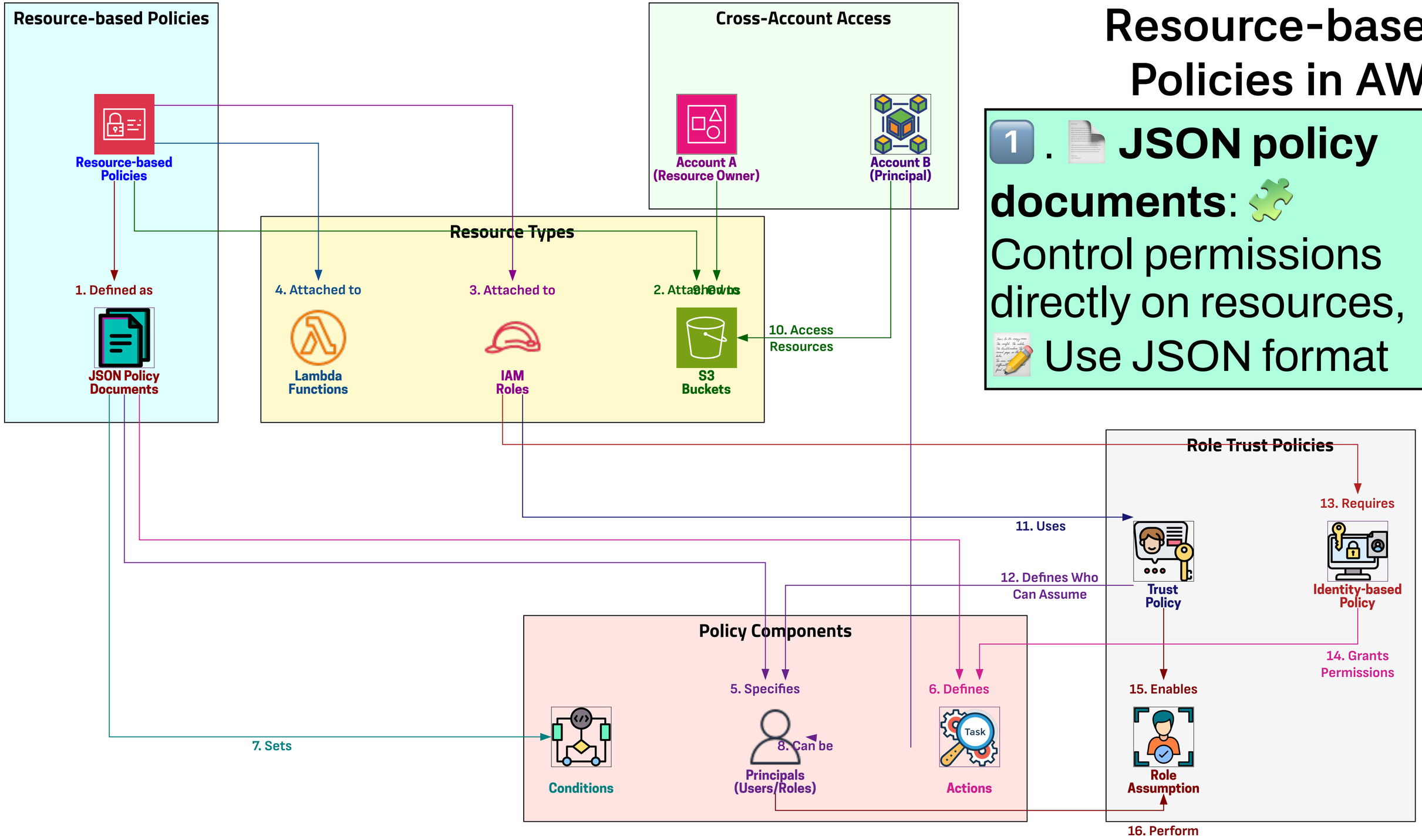
6️⃣. 📌 Inline policies

- 🔗 Directly attached to a single identity
- 1️⃣ One-to-one relationship
- 🗑️ Deleted with the identity



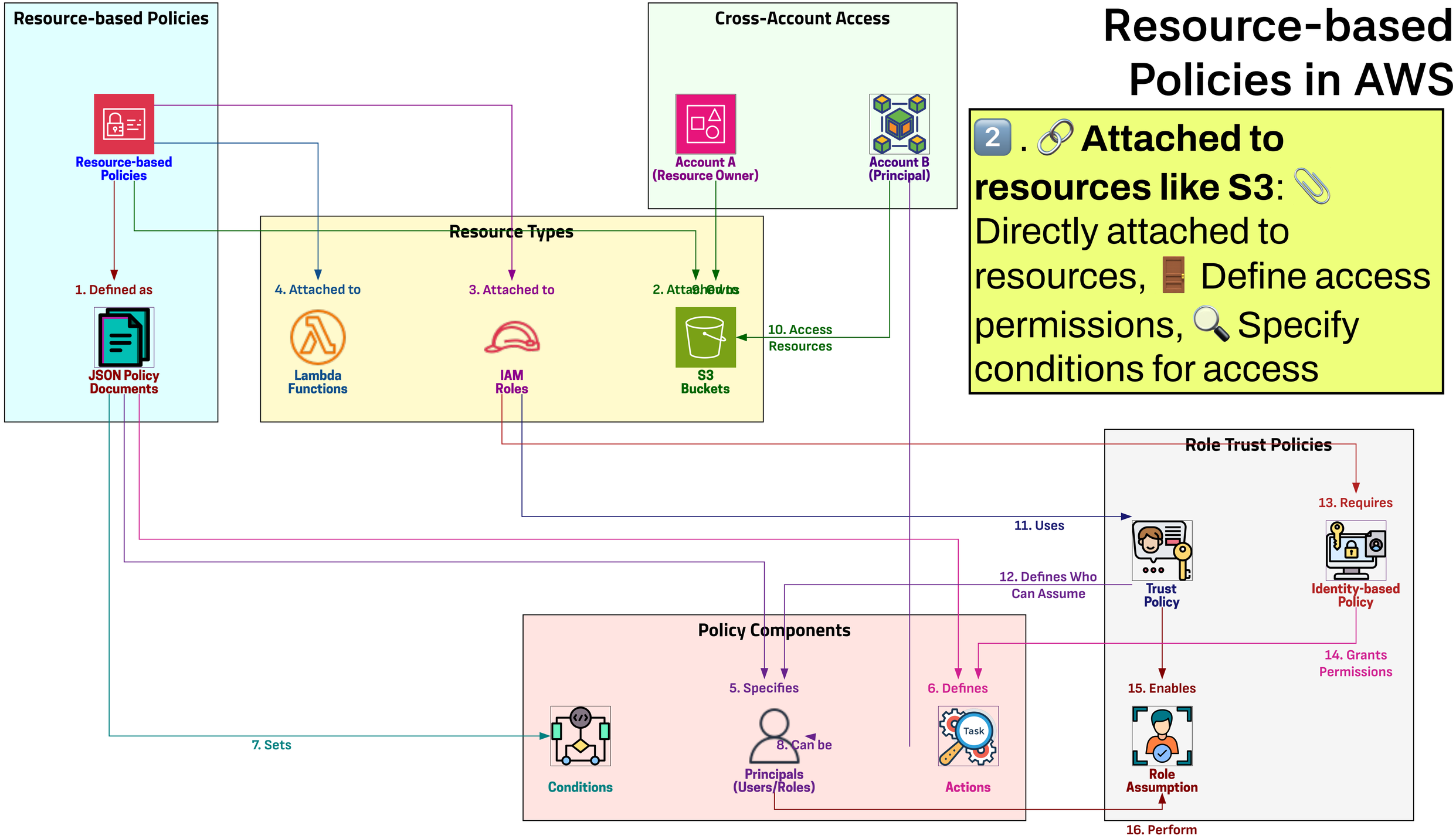
Resource-based Policies in AWS

1. JSON policy documents: Control permissions directly on resources, Use JSON format



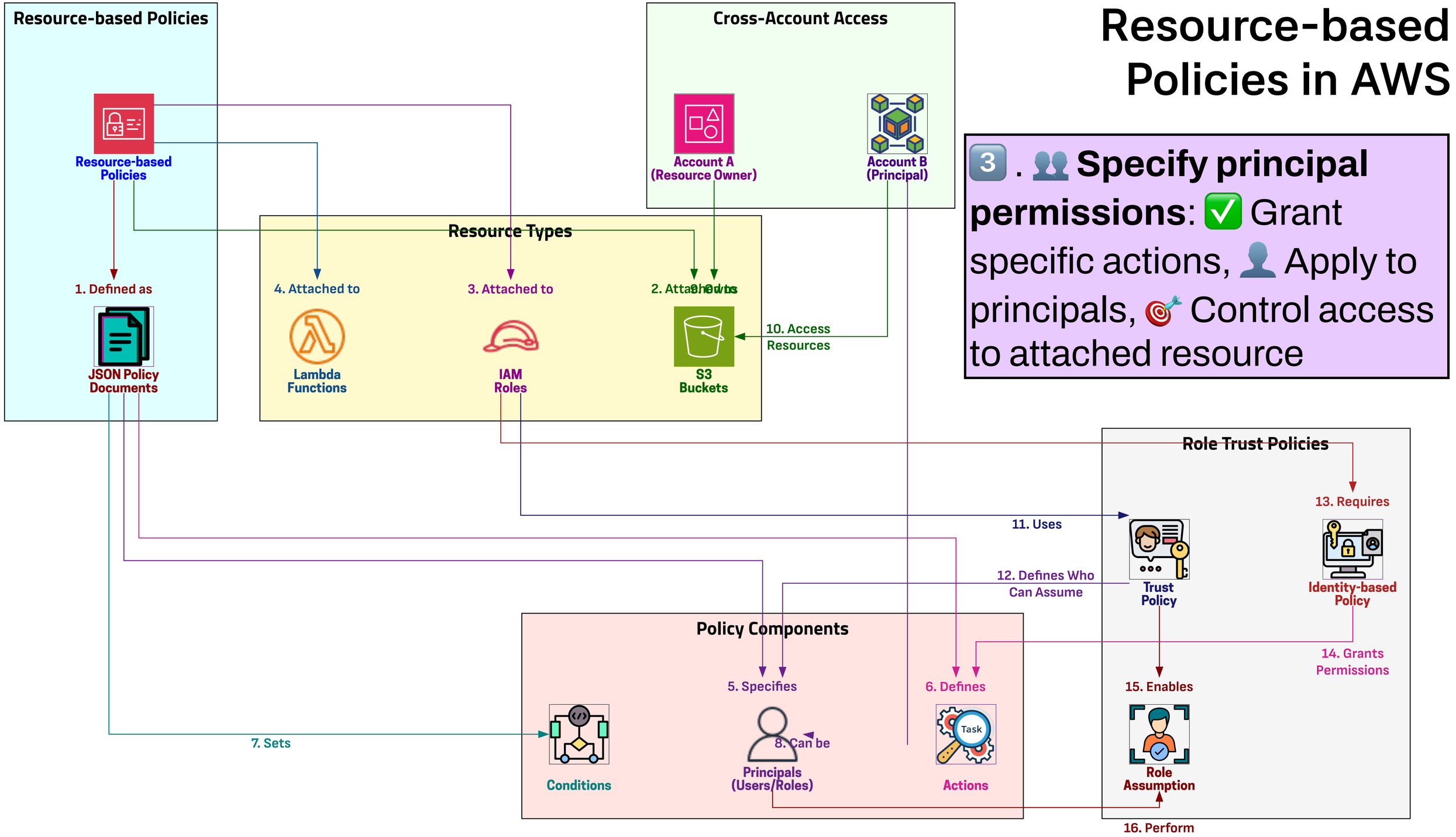
Resource-based Policies in AWS

2️⃣. 🔗 **Attached to resources like S3**: 📎 Directly attached to resources, 🚪 Define access permissions, 🔍 Specify conditions for access



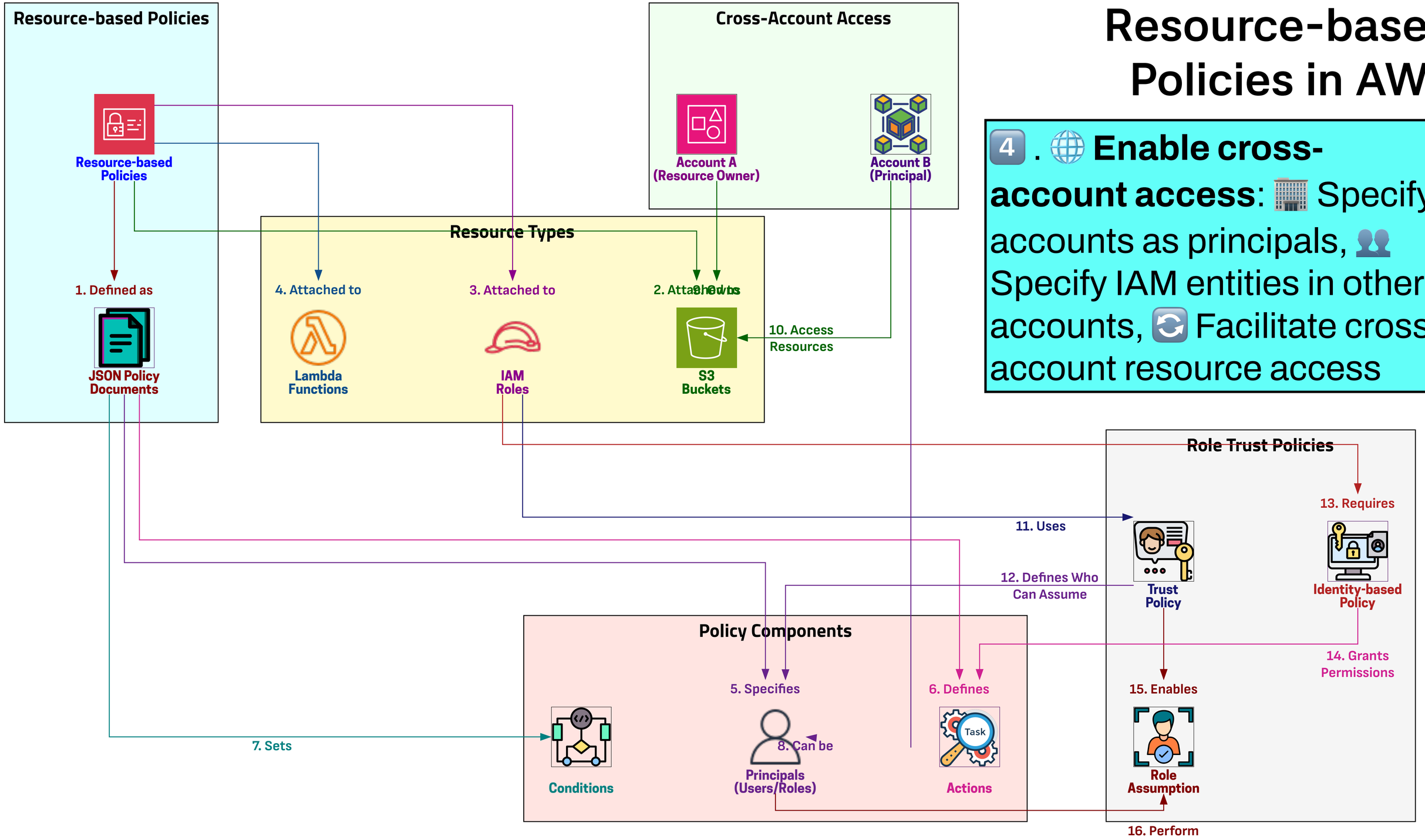
Resource-based Policies in AWS

3️⃣ . 👥 **Specify principal permissions**: ✅ Grant specific actions, 👤 Apply to principals, 🎯 Control access to attached resource



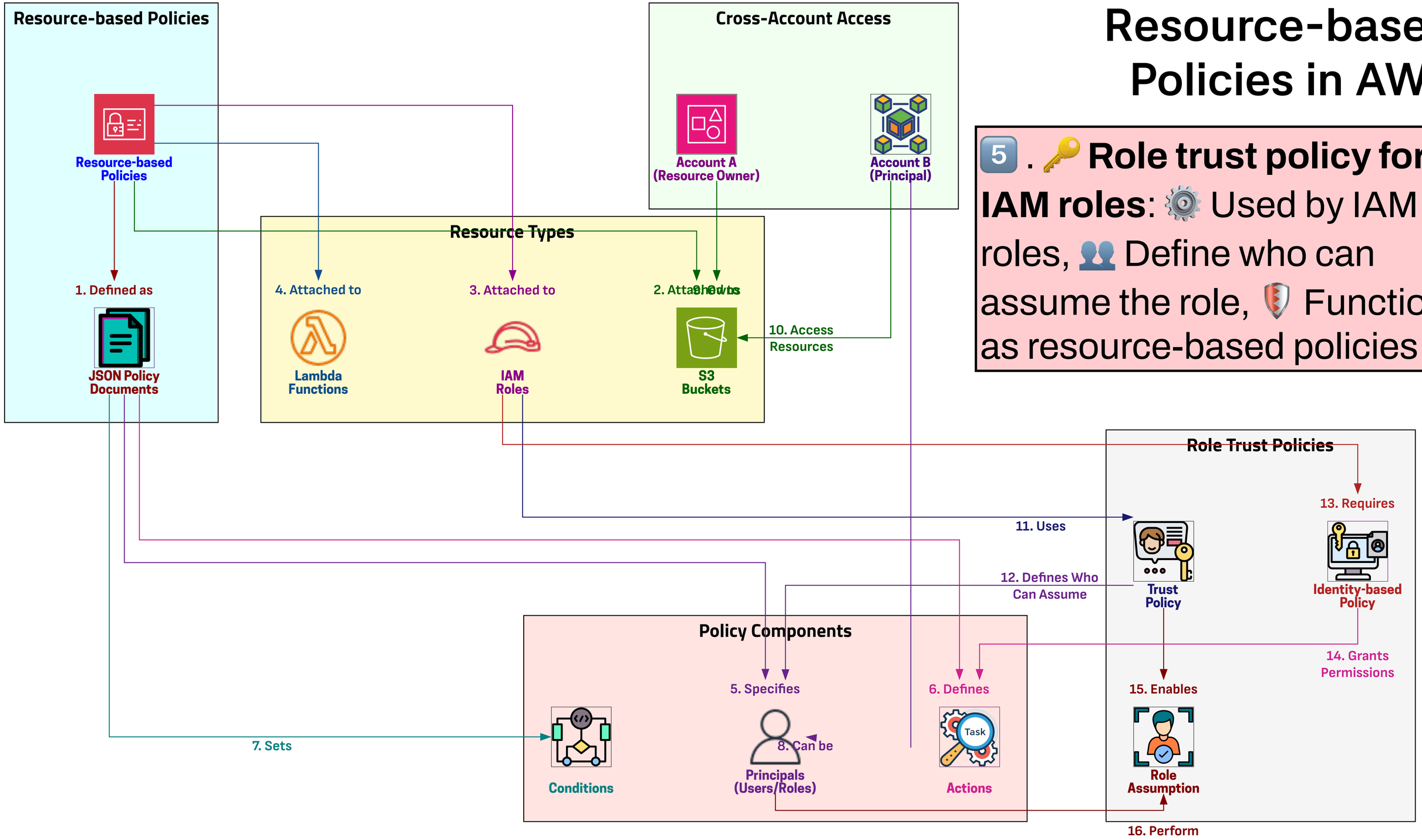
Resource-based Policies in AWS

4 . 🌐 **Enable cross-account access**: 🏢 Specify accounts as principals, 👥 Specify IAM entities in other accounts, 🔄 Facilitate cross-account resource access



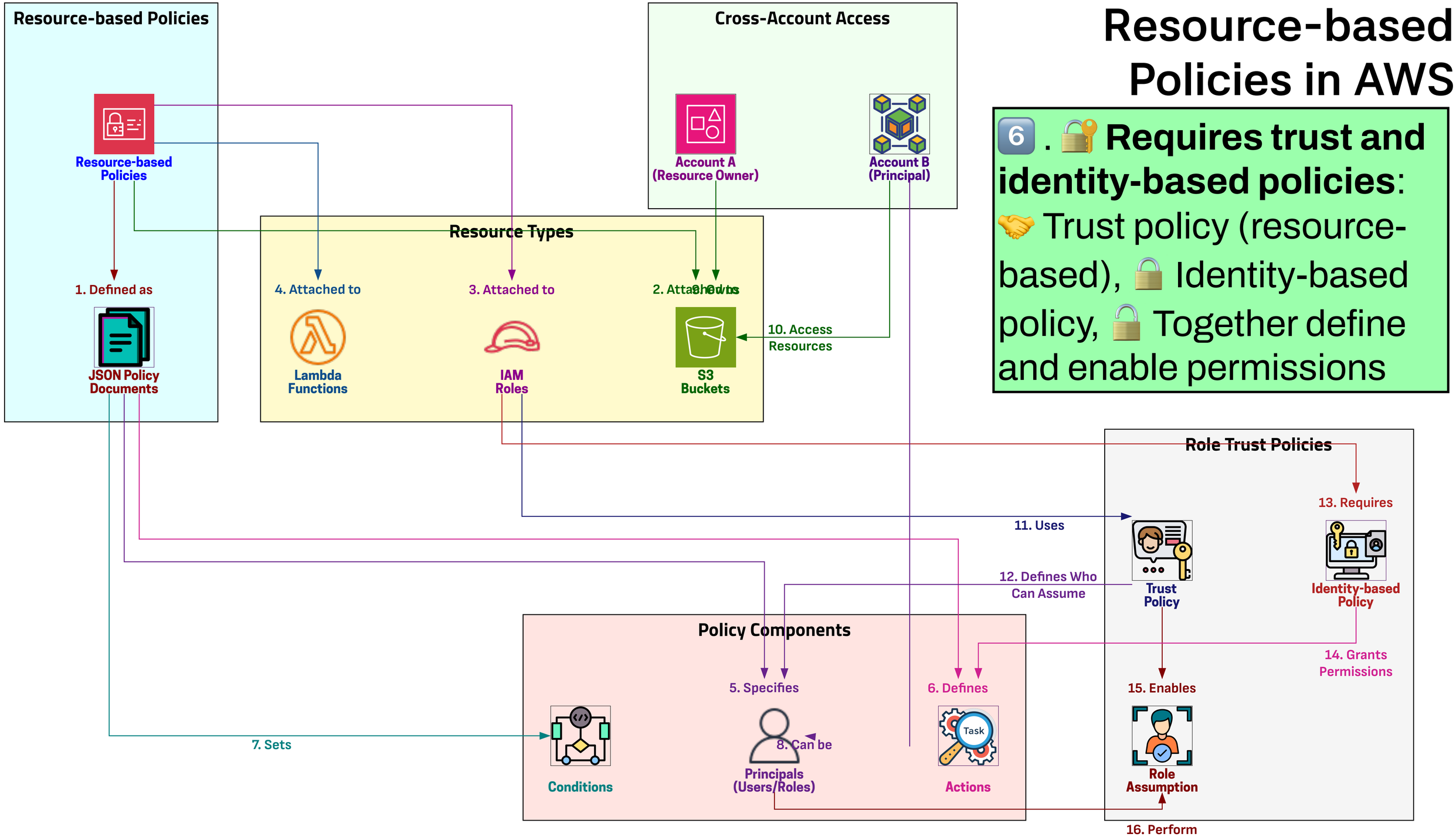
Resource-based Policies in AWS

5️⃣ . 🔑 **Role trust policy for IAM roles**: ⚙️ Used by IAM roles, 👥 Define who can assume the role, 🛡️ Function as resource-based policies



Resource-based Policies in AWS

6 . 🔐 **Requires trust and identity-based policies**: 🤝 Trust policy (resource-based), 🔒 Identity-based policy, 🔒 Together define and enable permissions



IAM Permissions Boundaries

1. **Advanced IAM feature**: Defines upper limits of permissions, Applied to IAM entities, Controls what policies can grant

2. **Sets maximum permissions for IAM entity**: Creates permission ceiling, Cannot be exceeded by identity policies, Enforces governance guardrails

3. **Requires both identity-based policies and permissions boundaries agreement**: Actions must be allowed by both, Intersection of permissions, Dual authorization model

IAM Permissions Boundaries

4

Not limited by resource-based policies: Resource policies operate separately, Entity as principal not affected, No boundary restrictions

5

Explicit deny overrides allow: Deny takes precedence, Applies to all policy types, Cannot be bypassed by boundaries

Service Control Policies in AWS Organizations

1. **AWS Organizations overview**: Grouping of AWS accounts, Central management, Business accounts administration

2. **SCPs application**: Requires all features enabled, Can be applied to any accounts, Organization-wide implementation

3. **Defines maximum permissions**: JSON policy format, Sets permission ceiling, Organization or OU level

Service Control Policies in AWS Organizations

4

 **Limits permissions for member accounts**:  Restricts all account entities,  Includes root user,  Governance control mechanism

5

Explicit deny overrides allow:  Deny takes precedence,  Cannot be bypassed, 

Standard AWS policy evaluation

Understanding ACLs in AWS

1. **ACLs defined**: Access control lists, Policies controlling resource access, For principals from another account

2. **Account-specific restrictions**: Not for same-account principals, Only for cross-account access, Inter-account permission management

3. **Non-JSON policy format**: Different structure than other policies, Alternative format specification, Unique configuration syntax

4. **Supported AWS services**: Amazon S3 buckets, AWS WAF web ACLs, Amazon VPC network ACLs

Session Policies in AWS IAM

1. 🛠️ Advanced policy mechanism

- 🔒 Define permissions for temporary sessions
- 👤 For roles or federated users
- ⏳ Time-limited access controls

2. 💻 Programmatically created sessions

- 🔄 Generated using API calls
- 🧩 Dynamic permission management
- 🔌 Runtime access configuration

3. 🔀 Intersection of permissions

- 🧱 Combined permission evaluation
- 🛡️ Identity-based policies as foundation
- ➕ Session policies as additional layer

4. ❌ Explicit deny prioritization

- 🚫 Deny overrides permissions
- ⚖️ Security-first evaluation model
- 🔑 Access control hierarchy

5. 📲 API operations for session creation

- 🔐 AssumeRole
- 📝 AssumeRoleWithSAML
- 🌐 AssumeRoleWithWebIdentity

6. 📄 Support for JSON inline and managed policies

- 🔤 Single inline JSON document
- 🔢 Up to 10 managed session policies
- 📋 Flexible configuration options

JSON Policy Document Structure

1. **Version**  : **Specify** the **version** of the **policy language** that you want to **use**. We recommend that you use the **latest 2012-10-17 version**. For more information, see **IAM JSON policy elements**: Version  .

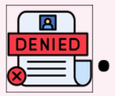
2. **Statement**  : **Use** this **main policy element** as a **container** for the following **elements**. You can include **more than one statement** in a **policy**  .

3. **Sid** (Optional)  : **Include** an **optional statement ID** to **differentiate** between your **statements**  .

4. **Effect**  : **Use Allow** or **Deny** to indicate whether the **policy allows** or **denies access**  .

JSON Policy Document Structure

5. **Principal**  (Required in only some circumstances): If you **create** a **resource-based policy**, you must **indicate** the **account**, **user**, **role**, or **federated user** to which you would like to **allow** or **deny access**. If you are **creating** an **IAM permissions policy** to **attach** to a **user** or **role**, you cannot **include** this **element**. The **principal** is implied as that **user** or **role** .

6. **Action**  : **Include** a **list of actions** that the **policy allows** or **denies** 

.

JSON Policy Document Structure

7. **Resource**  (Required in only some circumstances): If you **create** an **IAM permissions policy**, you must **specify** a **list of resources** to which the **actions** apply. If you **create** a **resource-based policy**, this **element** is optional. If you do not **include** this **element**, then the **resource** to which the **action** applies is the **resource** to which the **policy** is attached ☑ .

8. **Condition** 

(Optional): **Specify** the **circumstances** under which the **policy grants permission**  .

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FirstStatement",
      "Effect": "Allow",
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
    },
    {
      "Sid": "SecondStatement",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "ThirdStatement",
      "Effect": "Allow",
      "Action": [
        "s3:List*",
        "s3:Get*"
      ],
      "Resource": [
        "arn:aws:s3:::confidential-data",
        "arn:aws:s3:::confidential-data/*"
      ],
      "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
    }
  ]
}
```

```
{
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::example_bucket"
   }
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "1",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::account-id:root"]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }]
}
```

Granting Least Privilege in IAM Policies

1. **Principle of least privilege**: Grant only necessary permissions, Enhances security
2. **Specific task permissions**: Analyze user and role tasks, Tailor policies to specific needs
3. **Start with minimum permissions**: Begin with least access, Add permissions incrementally, Avoid excessive access rights
4. **Adjust permissions as needed**: Update policies dynamically, Adapt to evolving responsibilities
5. **Alternatives: AWS managed policies**: Use managed policies for initial setup, Wildcard permissions provide broad access, Refine access over time

Refining IAM Permissions Strategy

1. **Understand access level groupings**  : **Utilize access level groupings** to **comprehend** the **extent of access** a **policy grants**, categorized under **List**, **Read**, **Write**, **Permissions management**, or **Tagging**  .

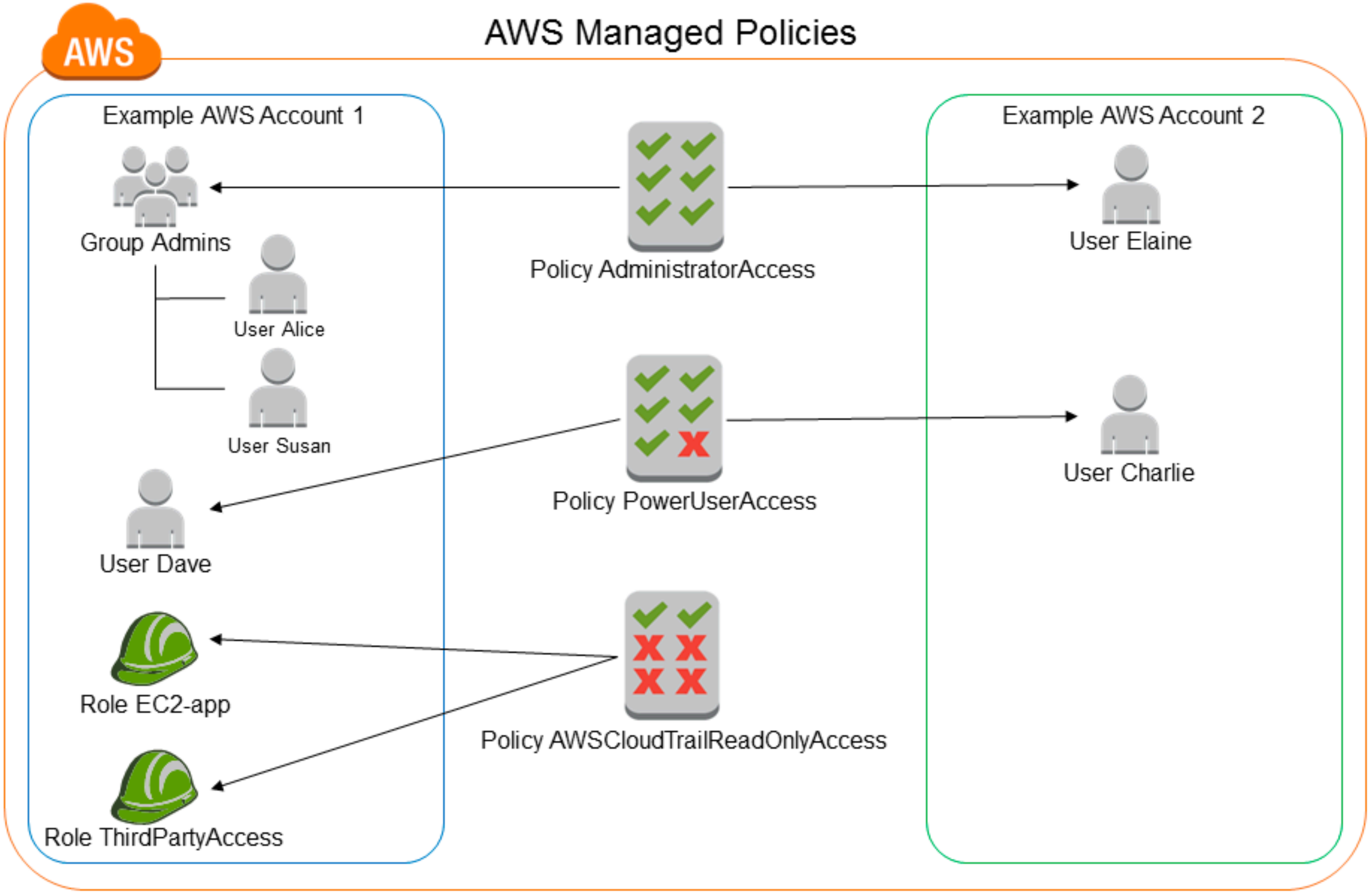
2. **Validate your policies**  : **Employ IAM Access Analyzer** for **policy validation** to **ensure policies** are **secure** and not **overly permissive**, utilizing over **100 policy checks**  .

3. **Generate a policy based on access activity**  : **Leverage IAM Access Analyzer** to **create policies** based on **actual access activity**, allowing for more **precise permission grants** based on **CloudTrail logs**  .

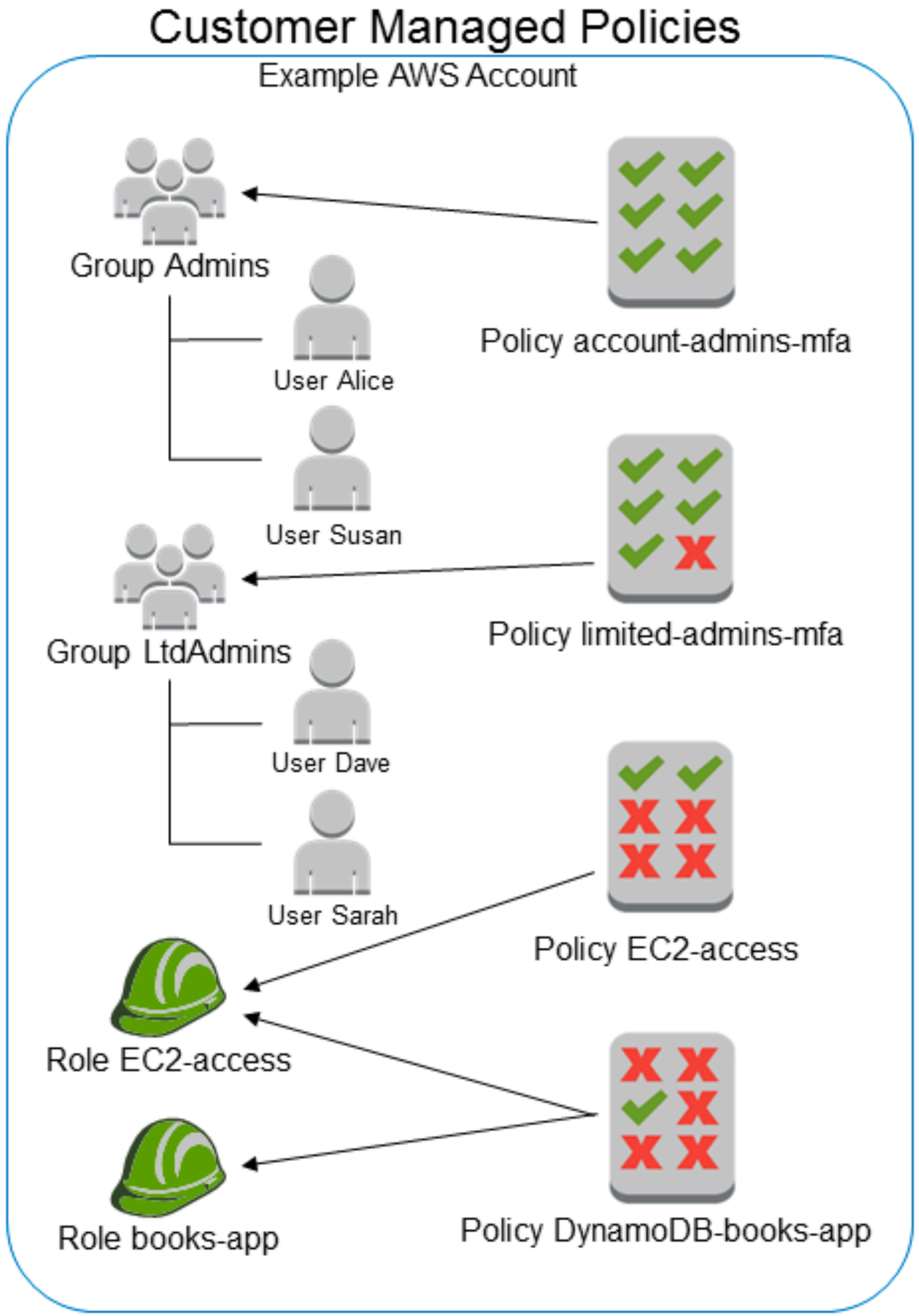
Refining IAM Permissions Strategy

4. **Use last accessed information**  : **Analyze** the **Access Advisor tab** or **AWS CLI/API** for **last accessed data** to **identify** and **remove unnecessary permissions**, further **aligning** with the **principle of least privilege**  .

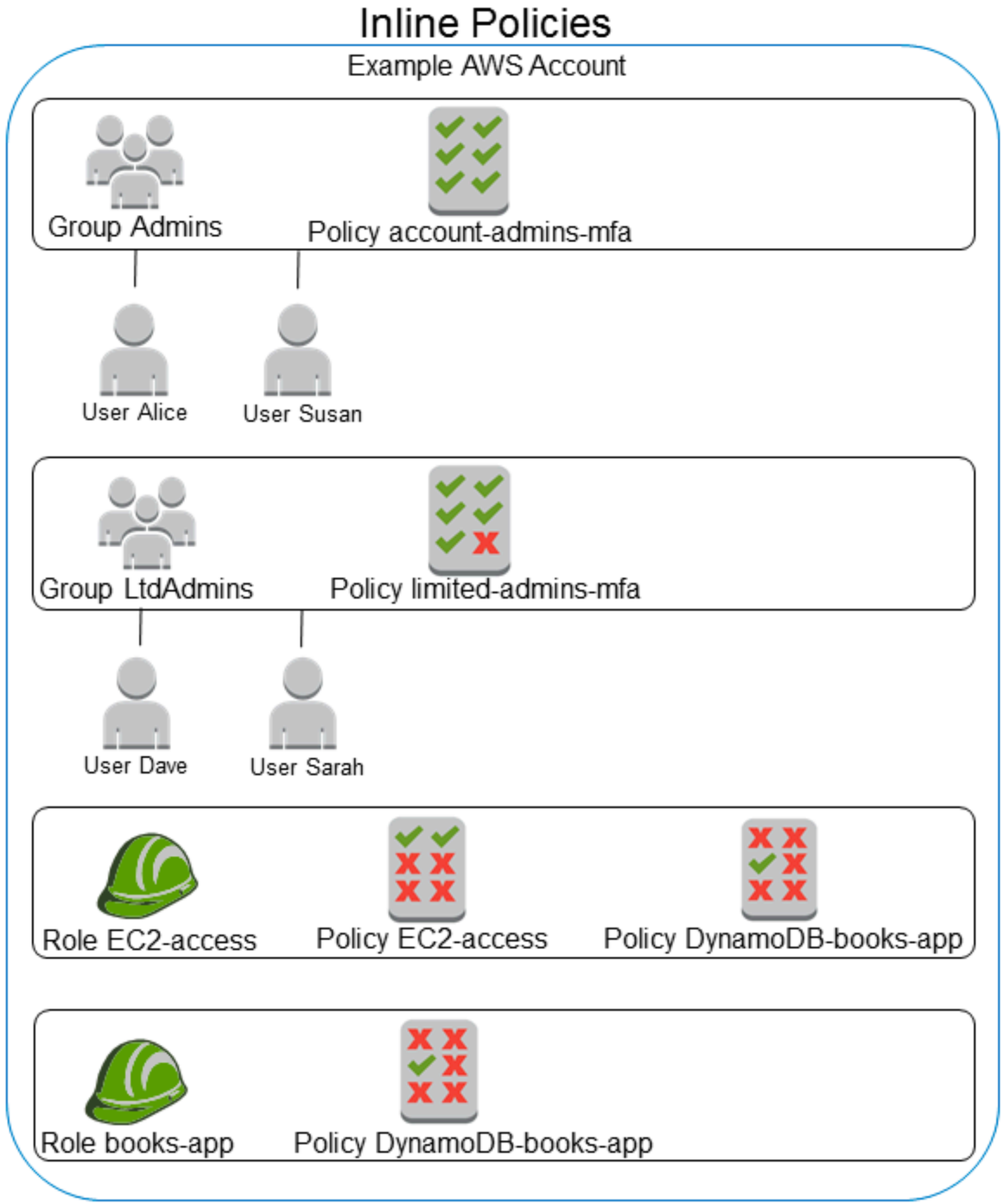
5. **Review account events in AWS CloudTrail**  : **Inspect CloudTrail Event history** for **detailed actions** and **resources accessed** by **IAM entities** to **fine-tune permissions** accordingly  .


AWS
AWS Managed Policies
Example AWS Account 1
Example AWS Account 2
Group Admins
User Alice
User Susan
User Dave
Role EC2-app
Role ThirdPartyAccess
Policy AdministratorAccess
Policy PowerUserAccess
Policy AWSCloudTrailReadOnlyAccess
User Elaine
User Charlie

Customer Managed Policies Example



Inline Policies Example



Security Best Practices in IAM

1. **Federation with identity providers**: Human users access with temporary credentials, Use identity federation, Enhances security

2. **Use of temporary credentials**: Workloads use temporary credentials, Utilize IAM roles for access, Bolsters security posture

3. **Require MFA**: Multi-factor authentication, Adds additional security layer

4. **Update access keys**: Regularly rotate credentials, Especially for long-term credentials, Mitigates security risks

Security Best Practices in IAM

5 **Protect root user credentials**: Safeguard against unauthorized access, Critical for account security

6 **Apply least-privilege permissions**: Provide only necessary access rights, Match permissions to function needs

7 **Start with AWS managed policies**: Begin with pre-defined policies, Progress toward custom policies, Adhere to least-privilege principles

8 **Use IAM Access Analyzer**: Create policies based on activity, Ensure minimal privilege, Optimize access rights

Security Best Practices in IAM

9 **Review and remove unused assets**: Regularly audit resources, Eliminate unused components, Minimize security vulnerabilities

10 **Use conditions in policies**: Control access tightly, Create context-aware permissions

11 **Verify public and cross-account access**: Check resource accessibility, Ensure appropriate controls, Use IAM Access Analyzer

12 **Validate IAM policies**: Confirm policies are secure, Ensure operational effectiveness, Use IAM Access Analyzer

Security Best Practices in IAM

13

 **Establish permissions guardrails**:  Consistent controls across accounts,  Maintain security posture,  Apply to multiple accounts

14

 **Use permissions boundaries**:  Delegate permission management,  Control maximum permissions,  Effective within accounts

Root User Best Practices for AWS

1️⃣. 🛡️ Secure root user credentials
🔒 Prevent unauthorized use
🔐 Safeguard credentials

2️⃣. 🔐 Use a strong password
🔢 Complex password
🛡️ Enhanced protection

3️⃣. 📱 Secure sign-in with MFA
🔒 Extra security layer
🔐 Multi-factor authentication

4️⃣. ⛔ Avoid root user access keys
📉 Reduce risk
🚫 Do not create access keys

5️⃣. 👥 Multi-person approval for sign-in
🛡️ Enhanced security
✅ Multiple approvals required

6️⃣. 📧 Use group email for credentials
👥 Shared responsibility
📩 Group email management

7️⃣. 🔒 Restrict access to recovery mechanisms
🛡️ Tighten security
🔐 Limit recovery options access

8️⃣. 🏢 Secure Organizations account credentials
🔐 Safeguard Organization root user
🛡️ Protect management account

9️⃣. 👁️ Monitor access and usage
📊 Track root user activity
🔍 Identify unauthorized access

Tasks Requiring AWS Root User Credentials

1️⃣. ⚙️ Change account settings
📝 Alter account name, email
🔐 Change password, access keys
👑 Root-only permissions

2️⃣. 🔄 Restore IAM user permissions
🛠️ Fix IAM admin access loss
🔓 Regain administrative control

3️⃣. 💰 Activate IAM access to Billing
📊 Monitor AWS expenses
📈 Track usage metrics
💳 Manage cost controls

4️⃣. 📄 View certain tax invoices
🧾 Access specific tax documents
🇪🇺 Particularly for VAT in Europe

5️⃣. 🛑 Close AWS account
❌ Terminate account completely
🔒 Ensure security during closure
💾 Protect data integrity

6️⃣. 🏪 Register as Reserved Instance seller
📝 Sign up for RI Marketplace
💹 Manage AWS resource selling

Tasks Requiring AWS Root User Credentials

7️⃣. 🔐 Configure S3 bucket for MFA
📱 Require multi-factor authentication
🛡️ Enhance bucket security
🔑 Protect critical actions

8️⃣. 📝 Edit/Delete SQS resource policy
🔄 Modify restrictive policies
🗑️ Remove policies denying all principals
🛡️ Maintain security safeguards

9️⃣. 📝 Edit/Delete S3 bucket policy
🔄 Change policies that deny all access
🔓 Override restrictive permissions

🔟. 🏛️ Sign up for AWS GovCloud
🇺🇸 US government cloud services
🔐 Exclusive root user enrollment
👥 For government and affiliates

1️⃣1️⃣. 🔑 Request GovCloud root user access keys
🎫 Contact AWS Support
🔐 Secure access management

1️⃣2️⃣. 🔄 Recover AWS KMS key
🔑 Fix unmanageable encryption keys
🛡️ Critical security recovery
🆘 Last resort intervention

Thanks for Watching