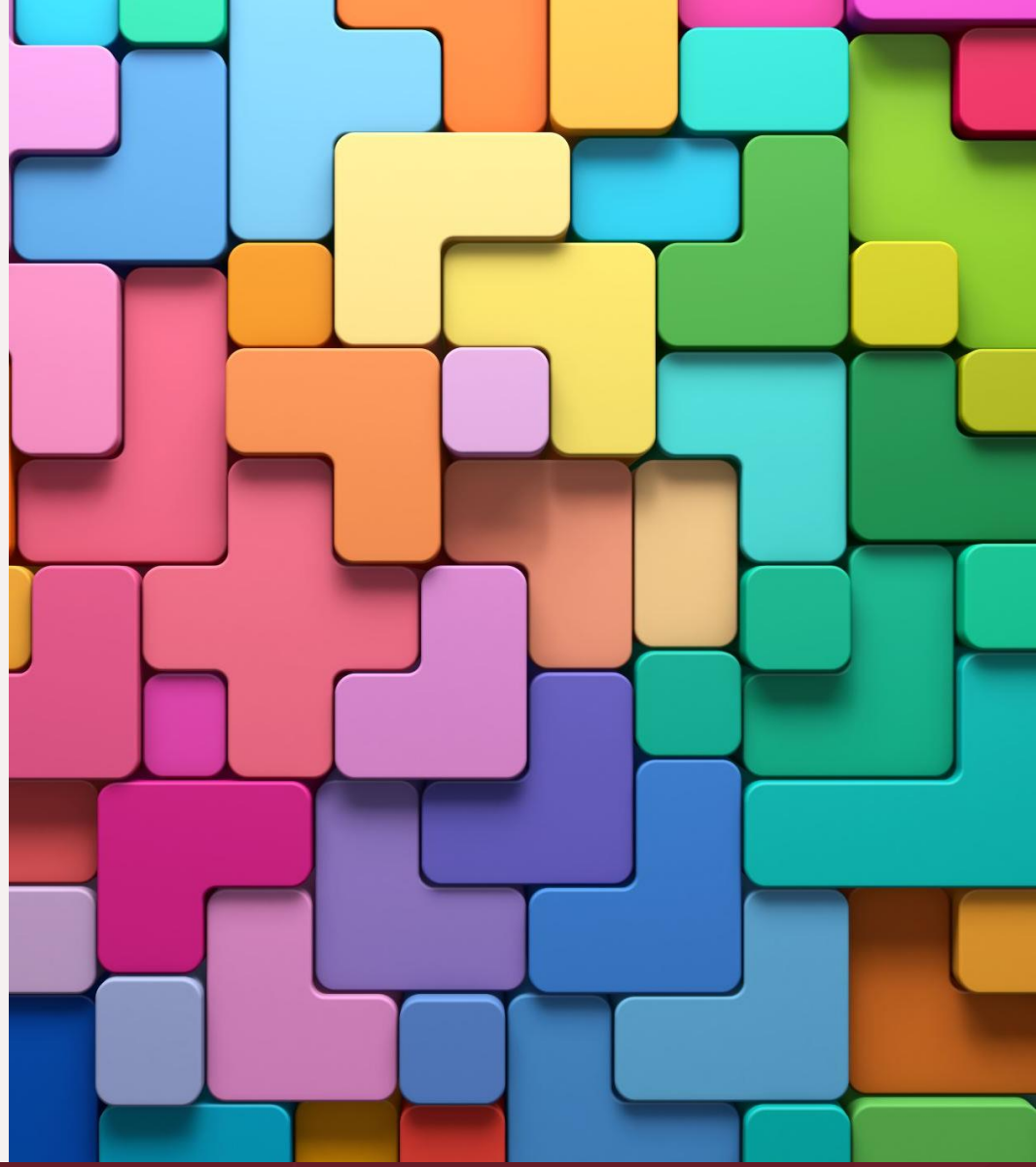
Cloud Computing Models

ECE 4150 – Spring 2024

Jan 10, 2024

Vijay Madisetti

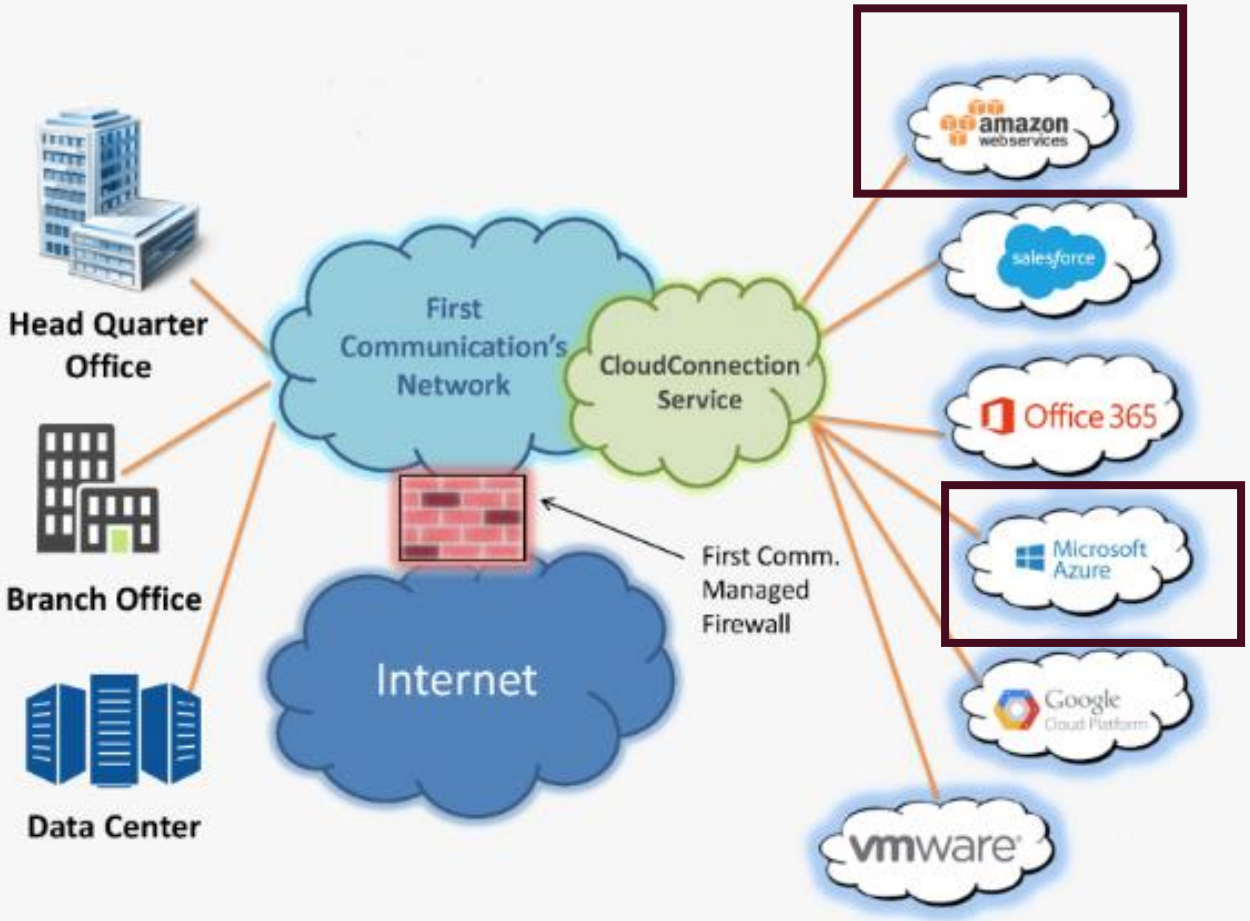
Georgia Tech



Old Way – All IT Infrastructure On Premises

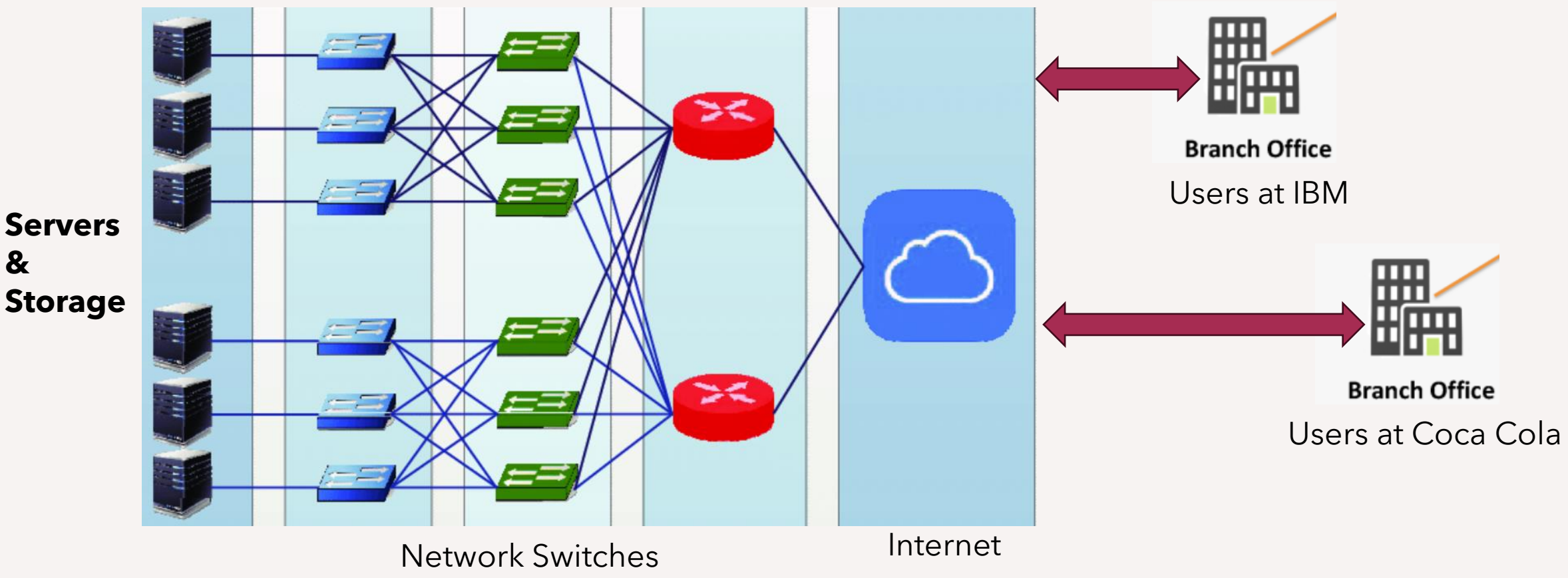
Servers,
Routers,
Switches,
Storage
(all on site or
In local Data Center)

New Way – Cloud (Remote) Infrastructure

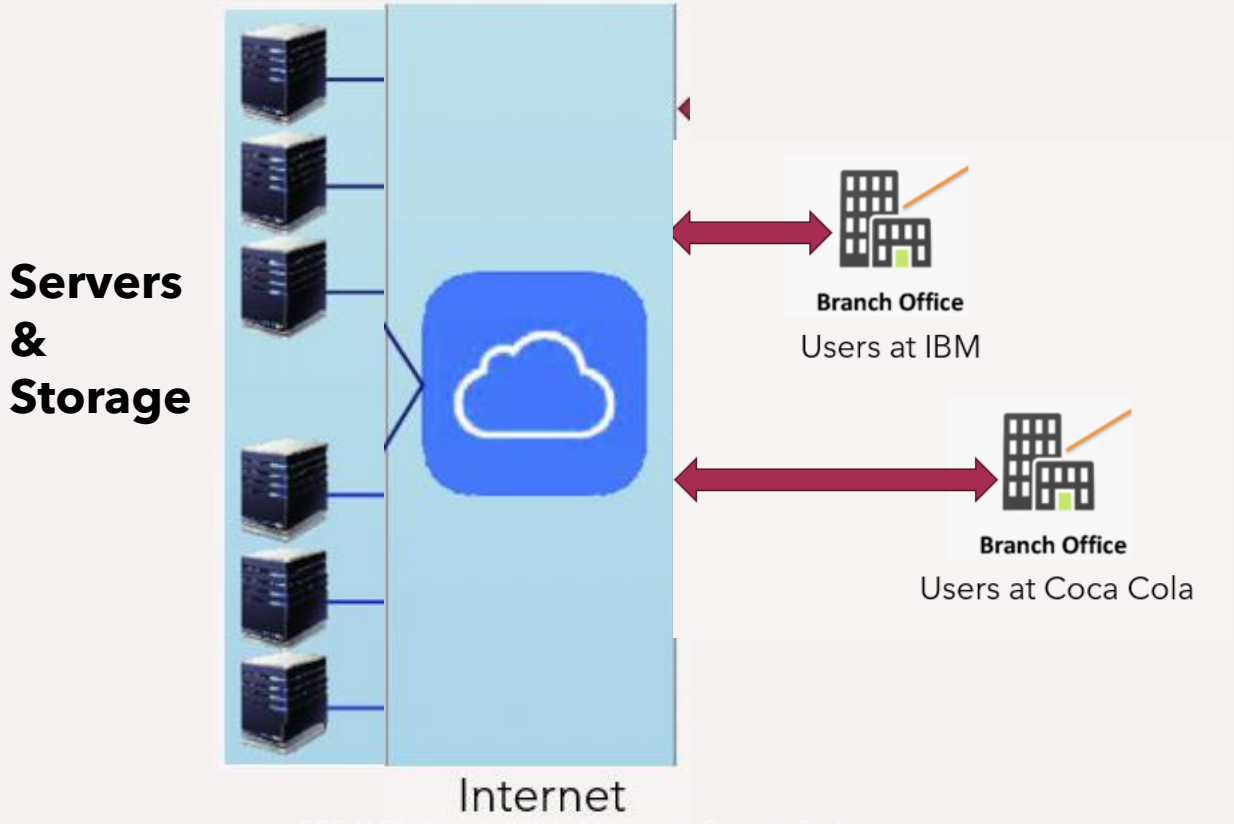


How is it done?

"Common" Computing Facilities accessed over Internet



But, users from IBM do not want to "share" their data or servers with Coca Cola

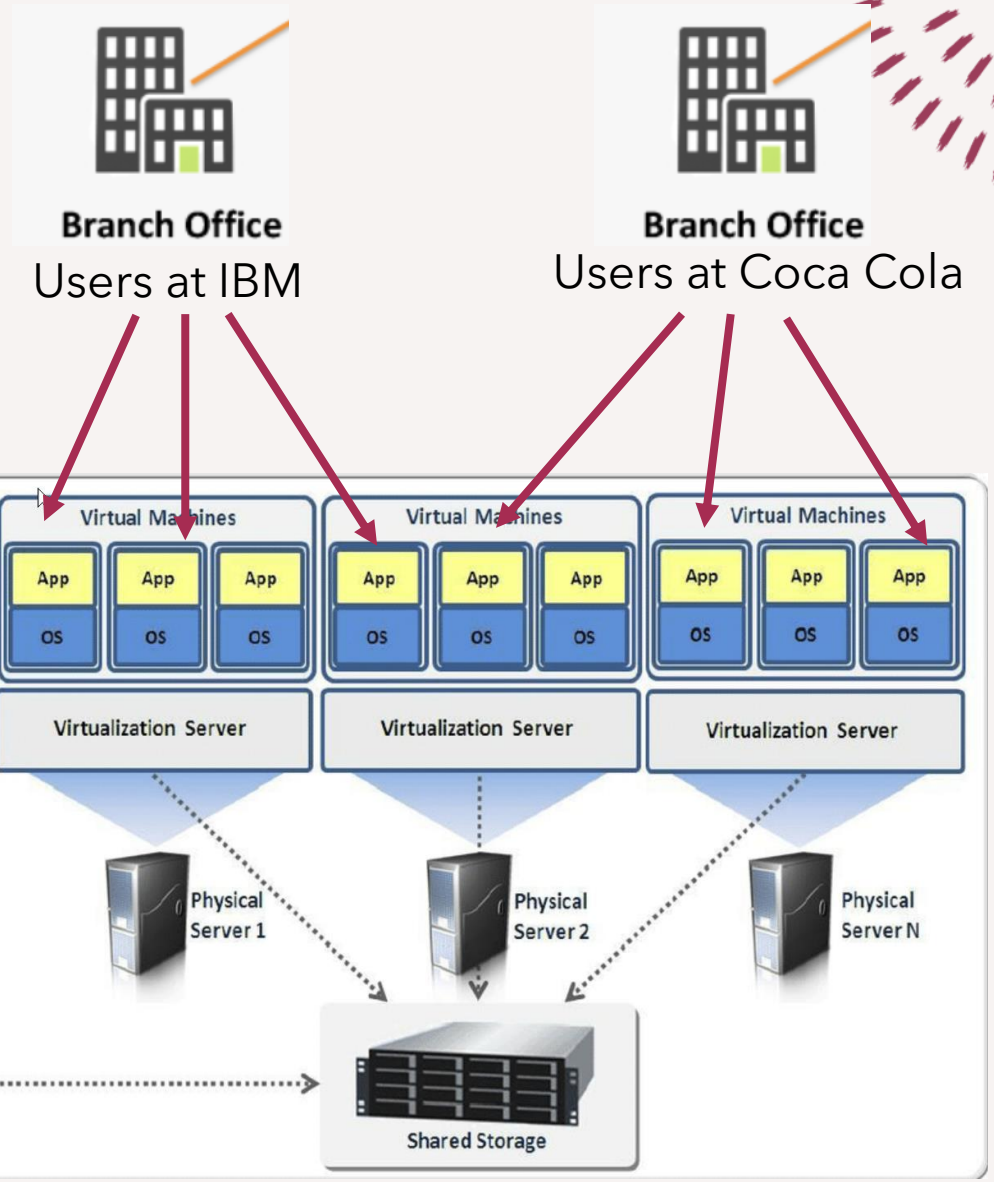


Solution - Virtualization using Virtual Machines or VMs

IBM is assigned "virtual machines" that look to it as if they are "physical servers" dedicated only to IBM users

Coca Cola is assigned "virtual machines" that look to it as if they are "physical servers" dedicated only to Coca Cola users

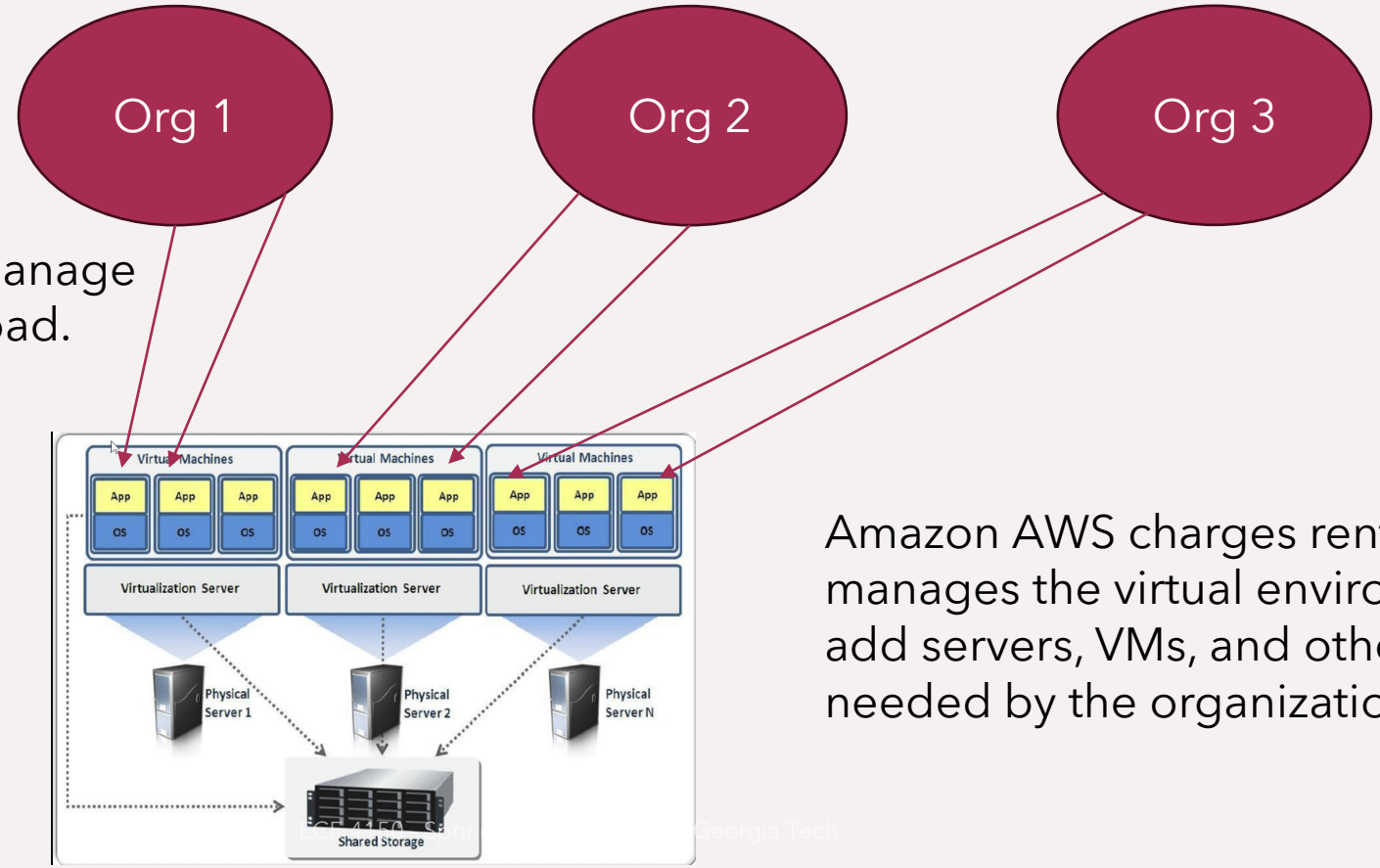
Amazon AWS



Amazon AWS makes sure that the virtual machines are managed in a suitable manner'
(IBM does not know Coca Cola exists at the Cloud)

Infrastructure as a Service (IaaS)

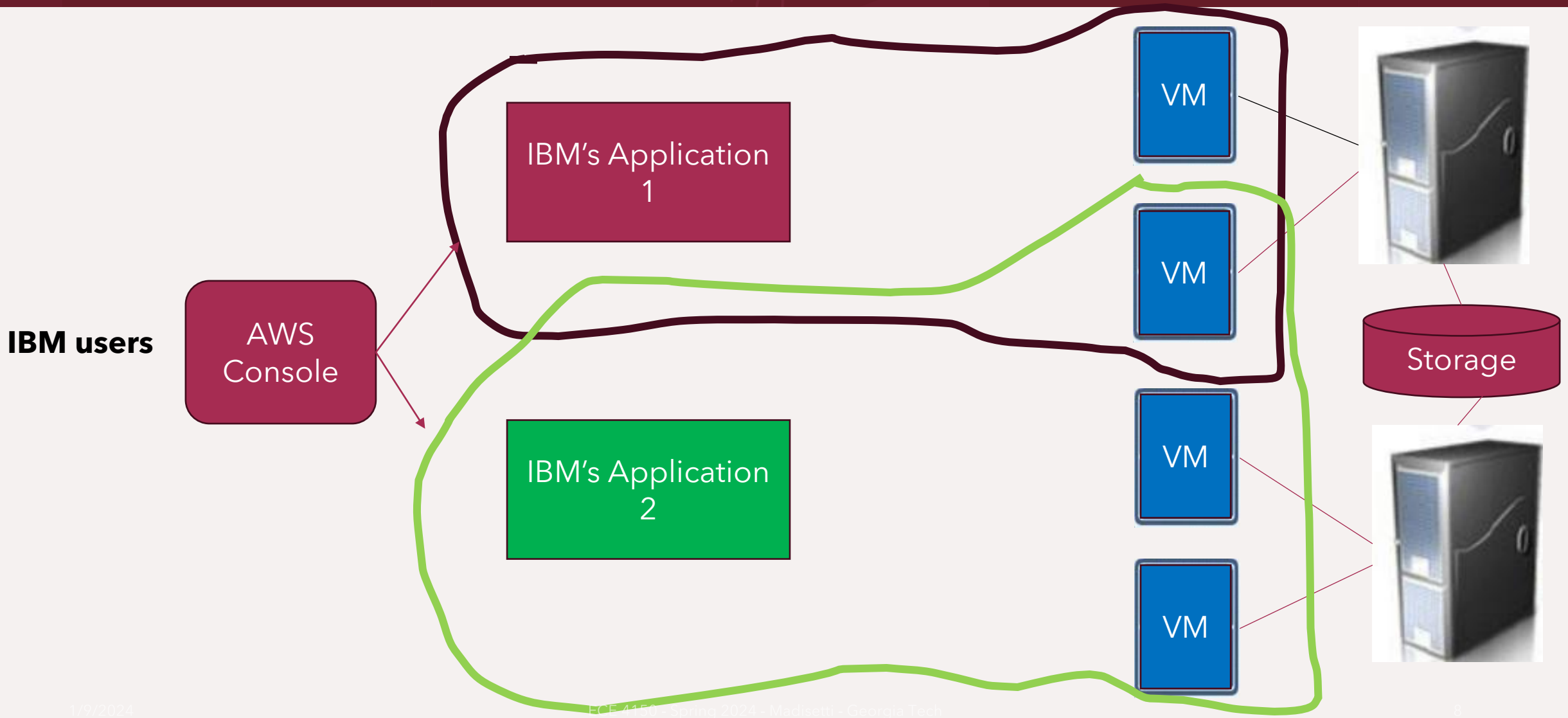
Organizations rent Virtual Machines (VMs) and build their virtual data center, write their own applications, manage their resources, and balance load.



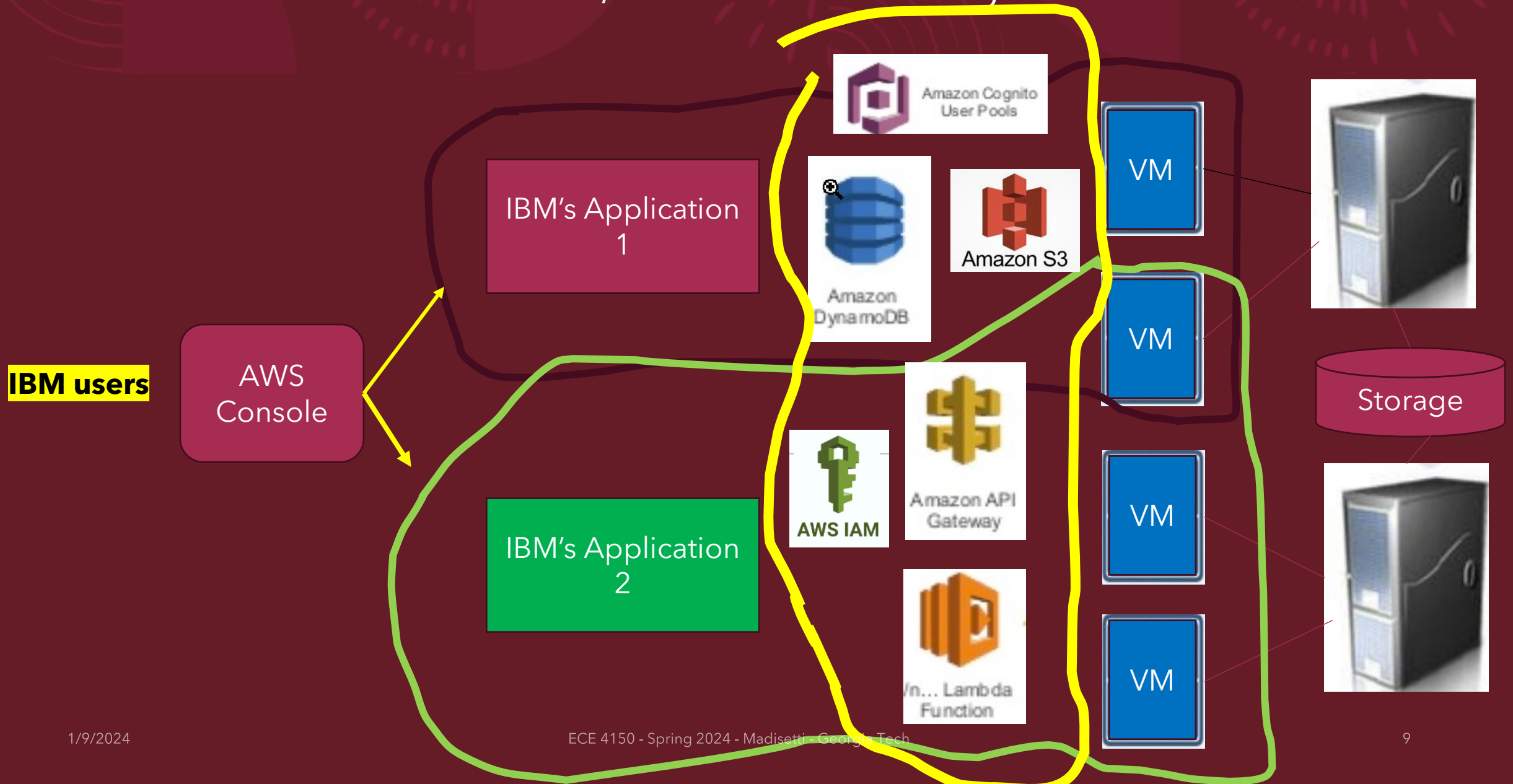
Amazon AWS

Amazon AWS charges rent for IaaS and manages the virtual environment (add servers, VMs, and other storage as needed by the organizations)

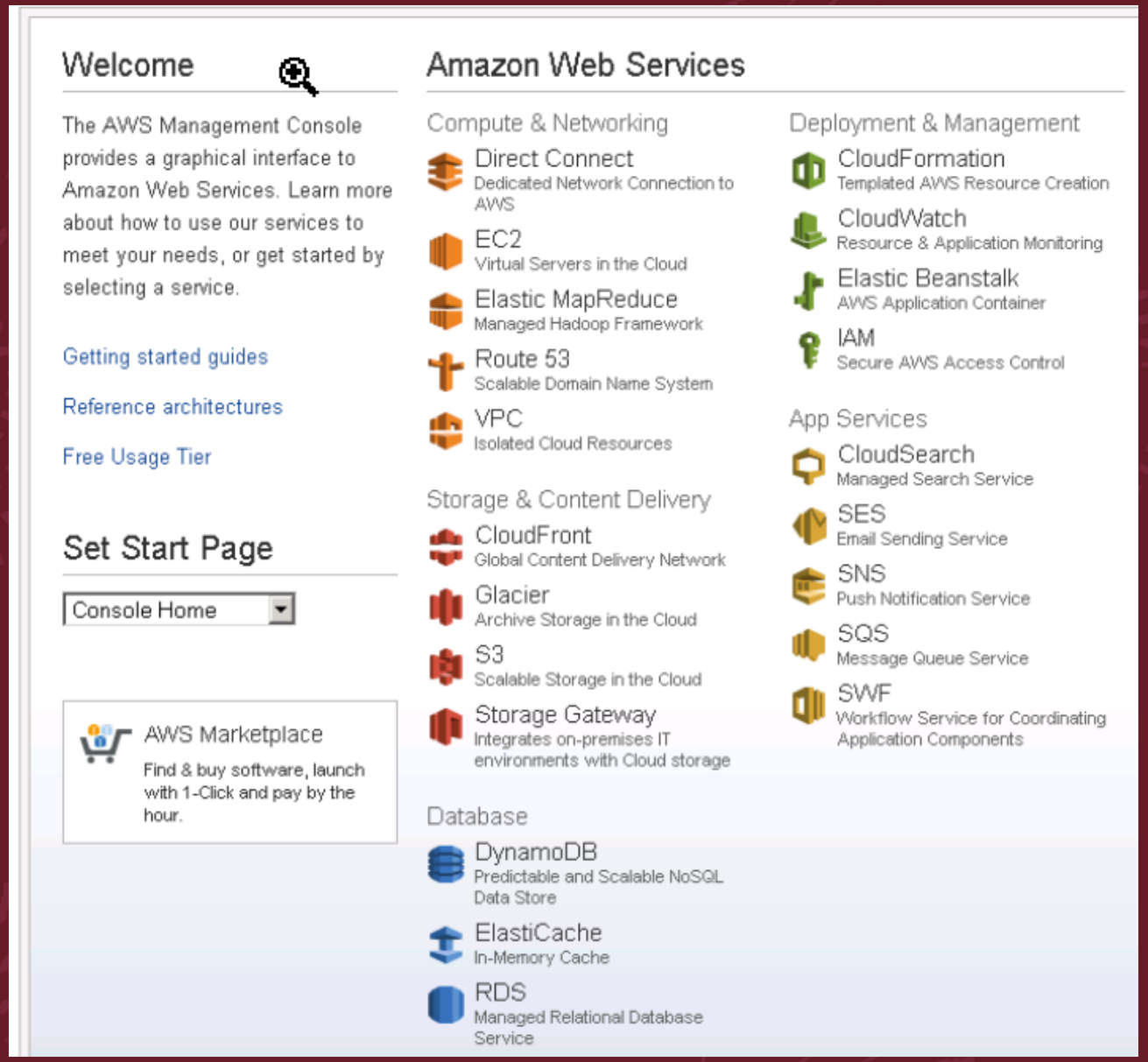
How IBM and Amazon AWS Work Together



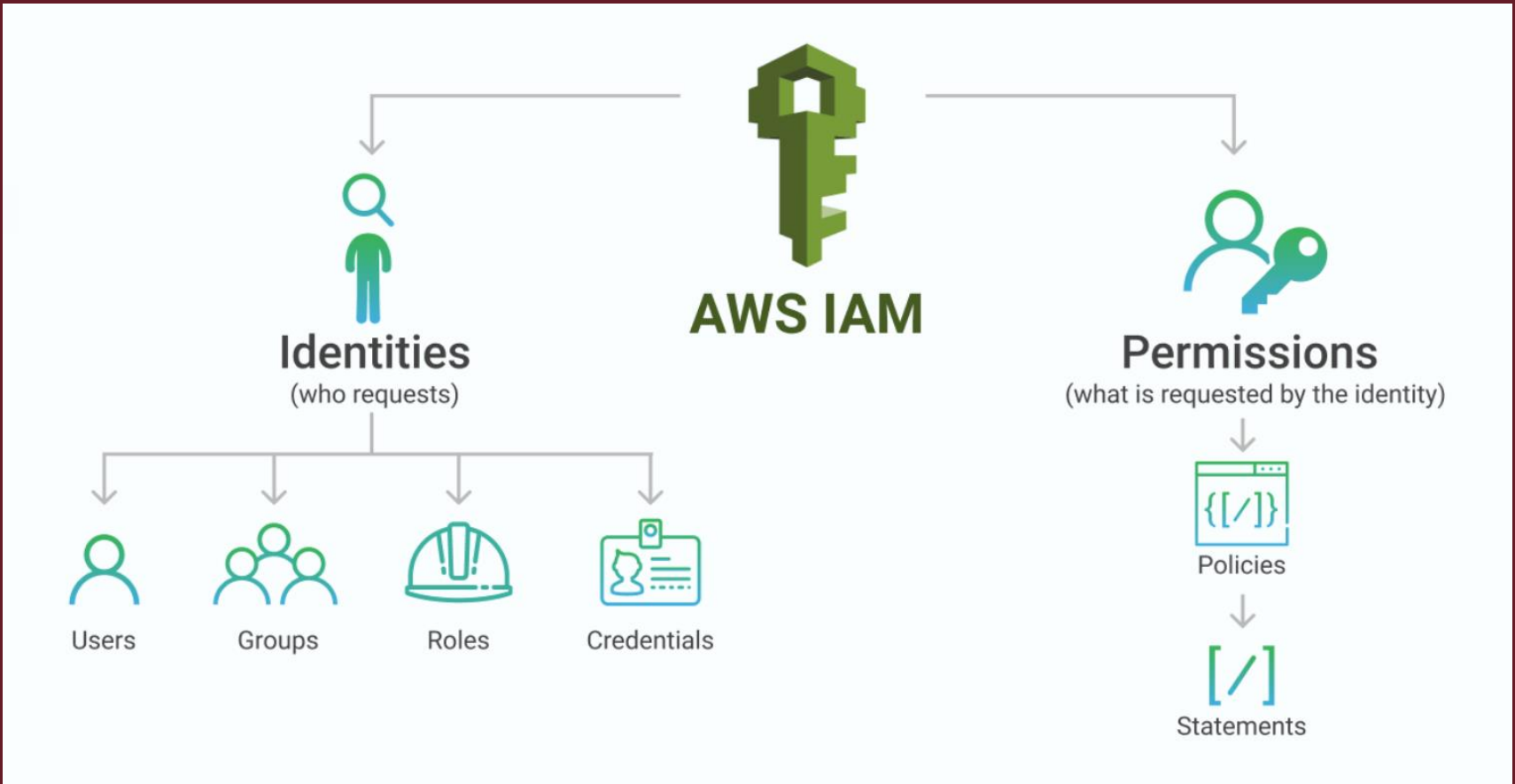
Amazon AWS offers more "services" to make IaaS VM's Useable – Benefit: AWS easier to use, and makes money for AWS

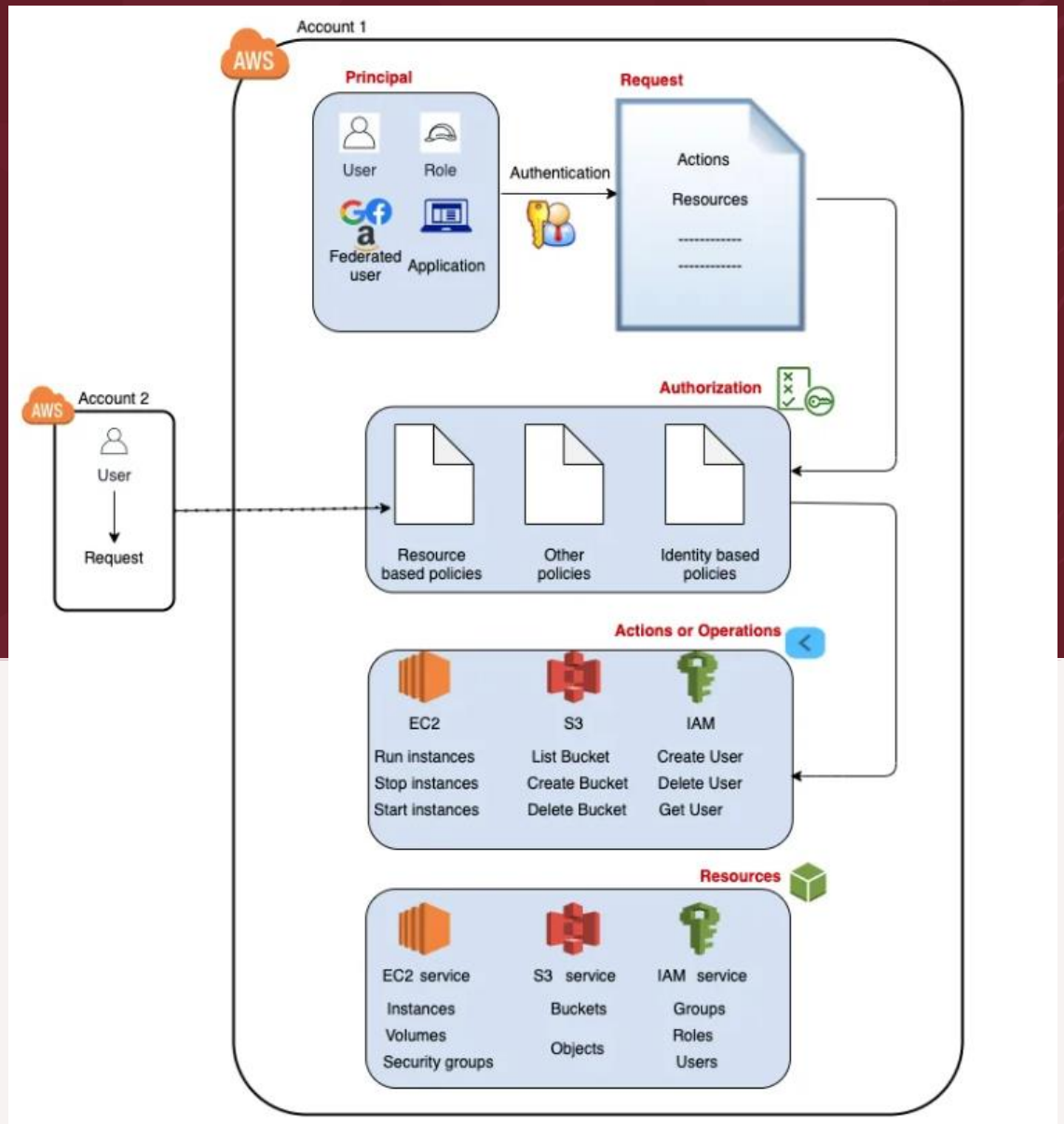


Indeed, AWS offers a whole lot of “services” and “resources” for “users”



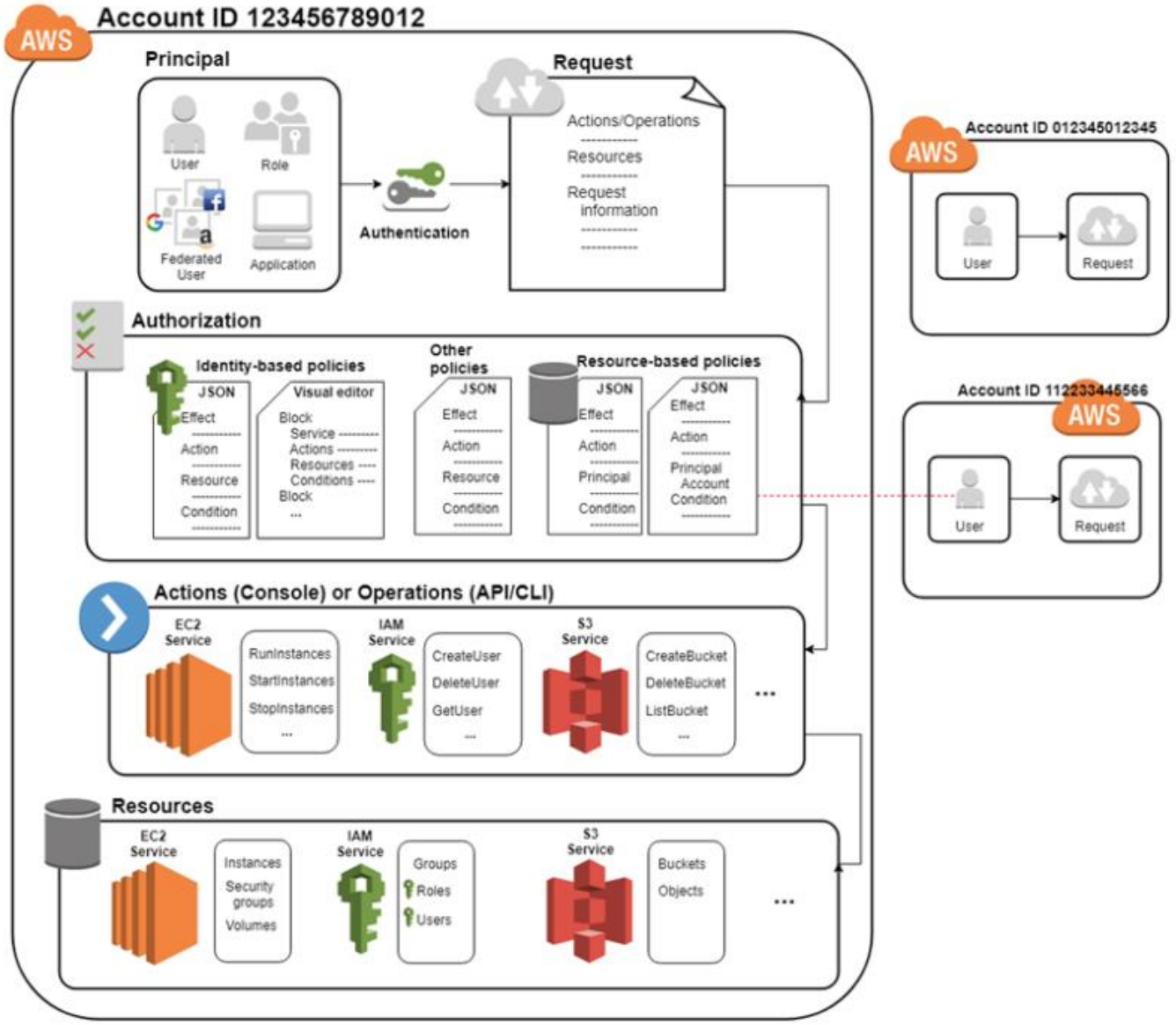
How do the “cloud” permissions work to access these services and resources?



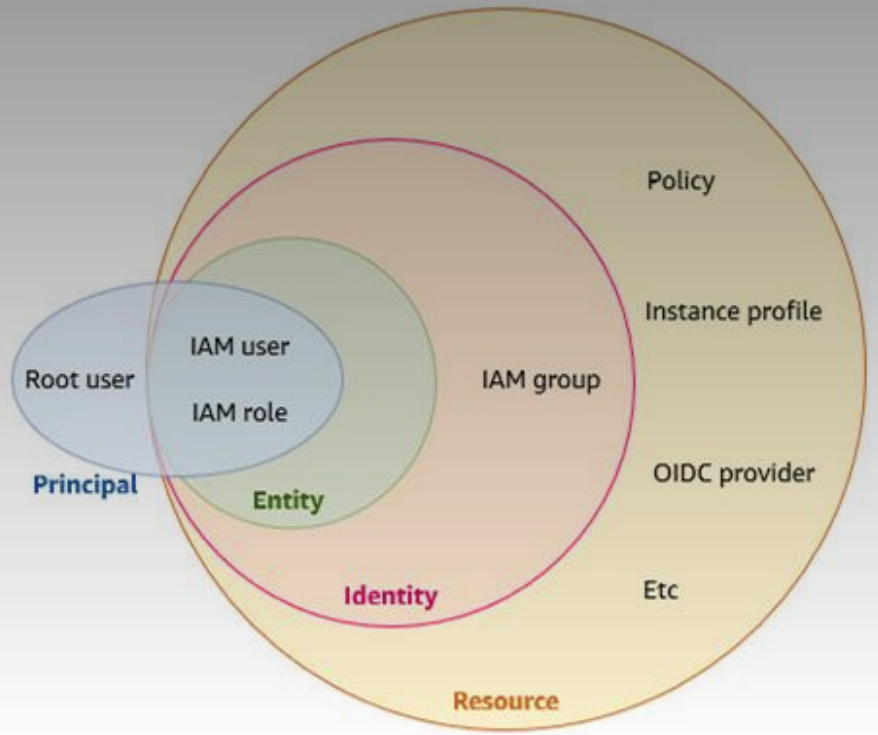


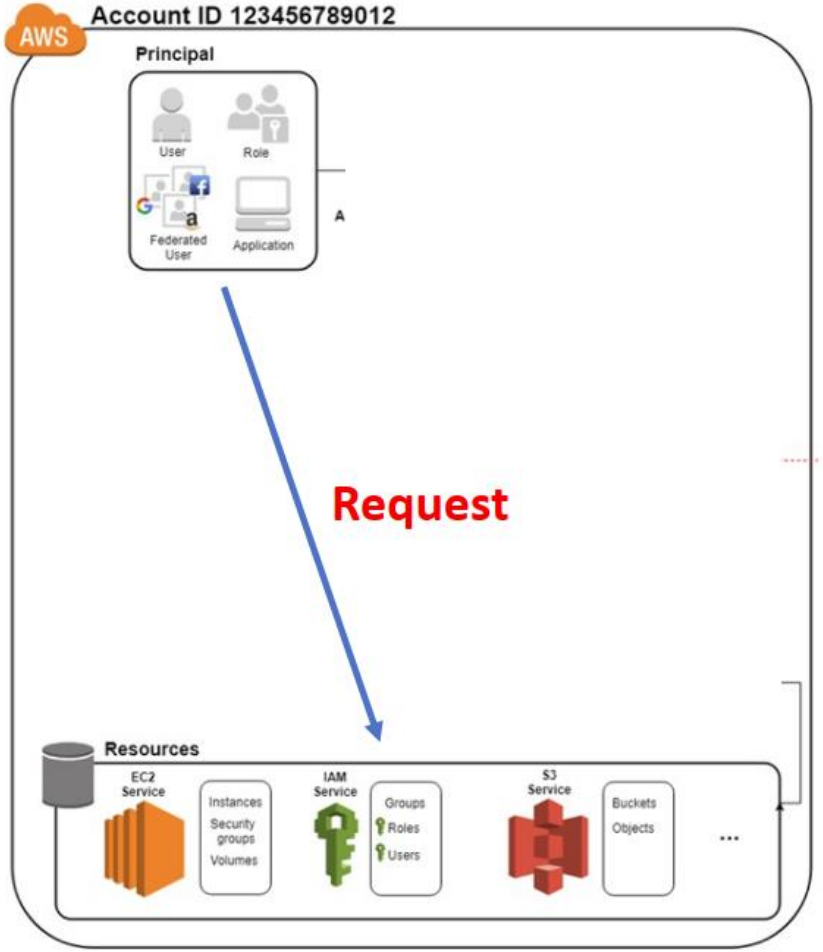
How does IAM work?

IAM Operation



IAM





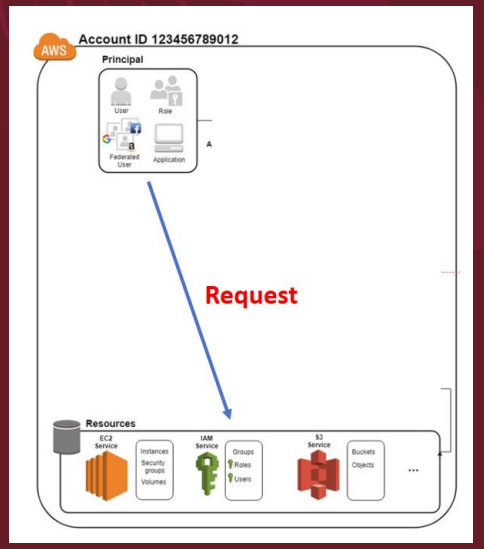
Principal

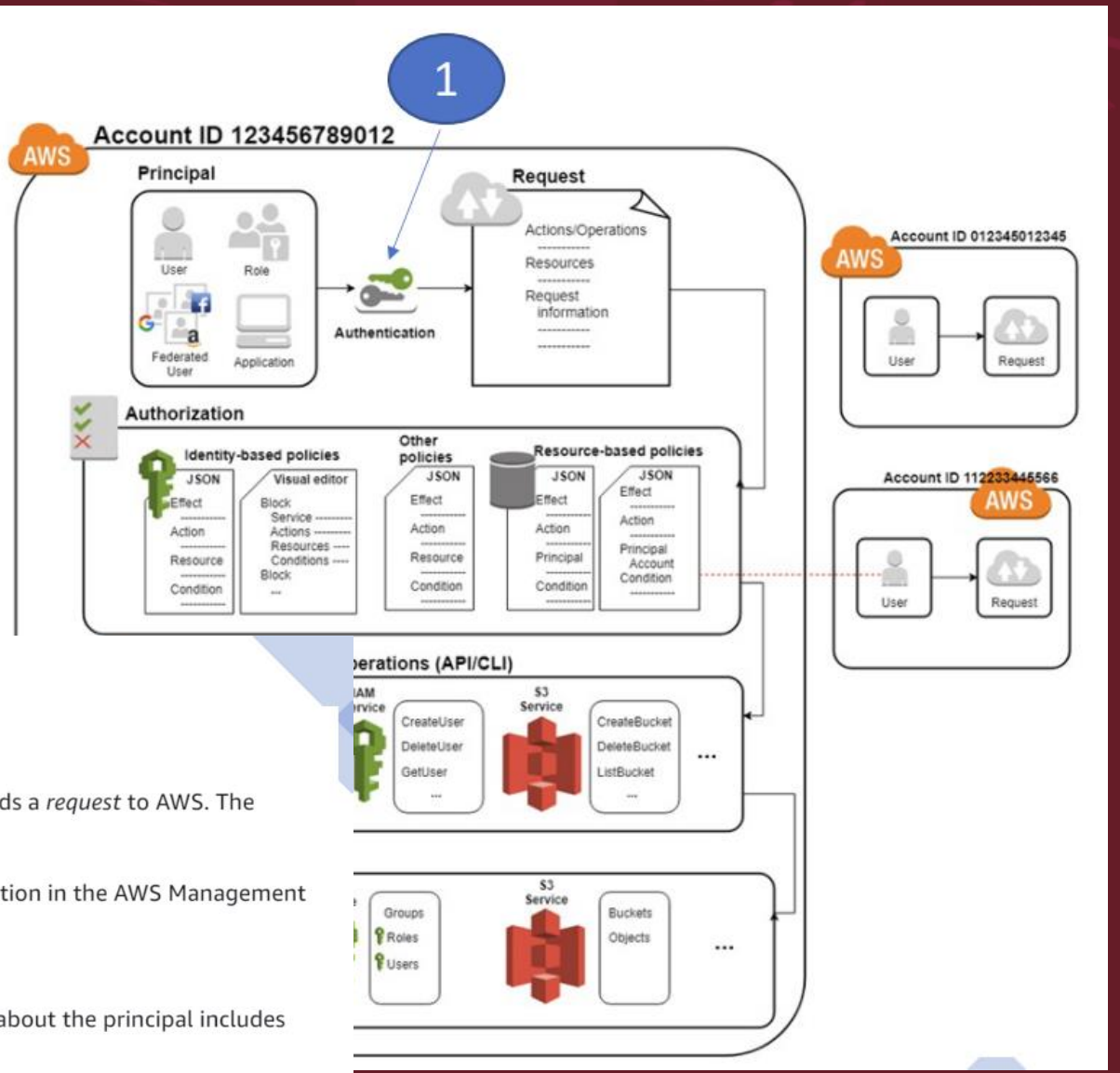
A *principal* is a person or application that can make a request for an action or operation on an AWS resource. The principal is authenticate as the AWS account root user or an IAM entity to make requests to AWS. As a best practice, do not use your root user credentials for your daily work. Instead, create IAM entities (users and roles). You can also support federated users or programmatic access to allow an application to access your AWS account.

Resources

After AWS approves the operations in your request, they can be performed on the related resources within your account. A resource is an object that exists within a service. Examples include an Amazon EC2 instance, an IAM user, and an Amazon S3 bucket. The service defines a set of actions that can be performed on each resource. If you create a request to perform an unrelated action on a resource, that request is denied. For example, if you request to delete an IAM role but provide an IAM group resource, the request fails. To see AWS service tables that identify which resources are affected by an action, see Actions, Resources, and Condition Keys for AWS Services.

IAM





1

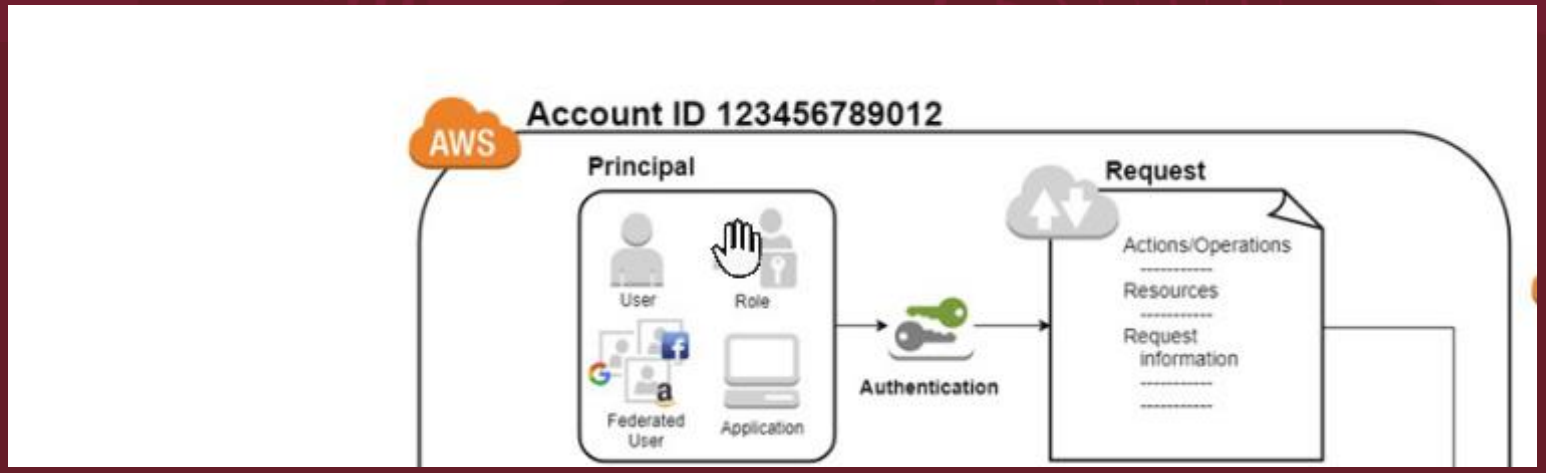
Request

When a principal tries to use the AWS Management Console, the AWS API, or the AWS CLI, that principal sends a *request* to AWS. The request includes the following information:

- **Actions or operations** – The actions or operations that the principal wants to perform. This can be an action in the AWS Management Console, or an operation in the AWS CLI or AWS API.
- **Resources** – The AWS resource object upon which the actions or operations are performed.
- **Principal** – The person or application that used an entity (user or role) to send the request. Information about the principal includes the policies that are associated with the entity that the principal used to sign in.
- **Environment data** – Information about the IP address, user agent, SSL enabled status, or the time of day.
- **Resource data** – Data related to the resource that is being requested. This can include information such as a DynamoDB table name or a tag on an Amazon EC2 instance.

AWS gathers the request information into a *request context*, which is used to evaluate and authorize the request.

IAM



Authentication

A principal must be authenticated (signed in to AWS) using their credentials to send a request to AWS. Some services, such as Amazon S3 and AWS STS, allow a few requests from anonymous users. However, they are the exception to the rule.

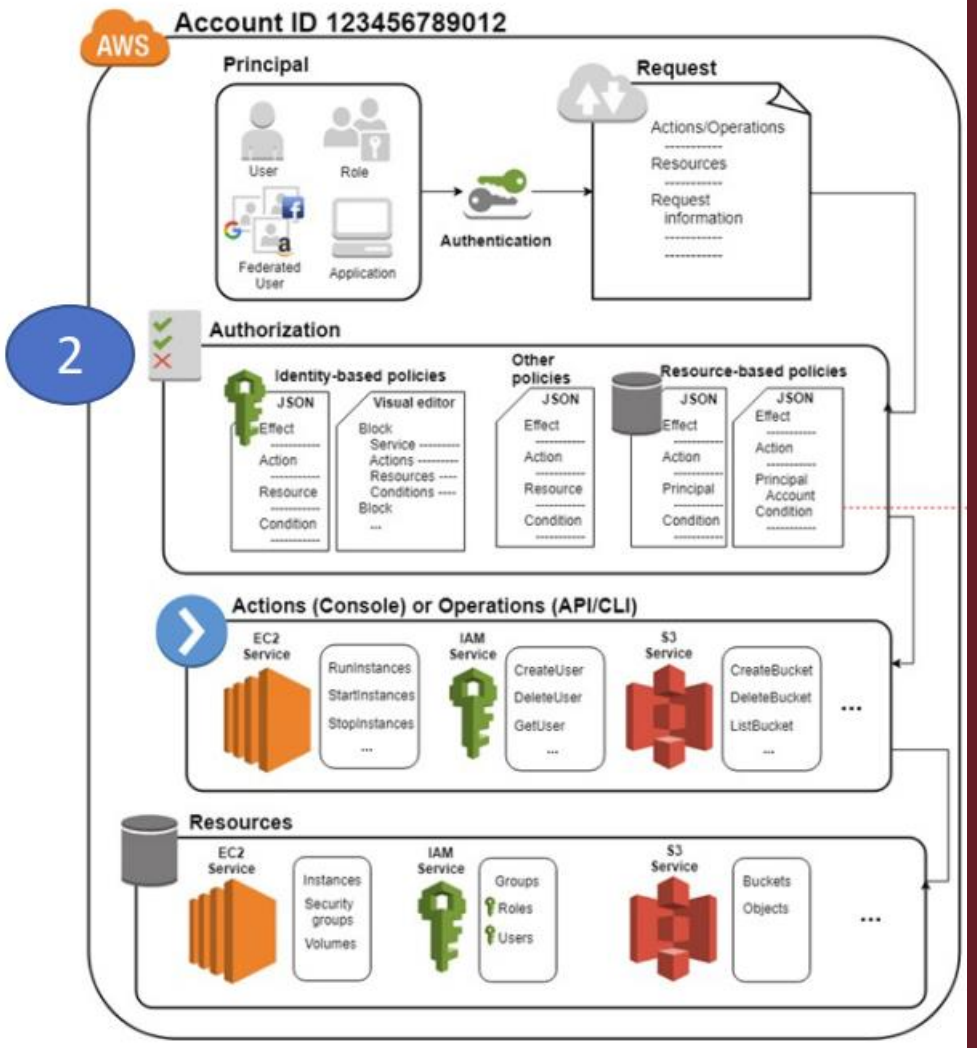
To authenticate from the console as a root user, you must sign in with your email address and password. As an IAM user, provide your account ID or alias, and then your user name and password. To authenticate from the API or AWS CLI, you must provide your access key and secret key. You might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more about the IAM entities that AWS can authenticate, see IAM users and IAM roles.

Authorization

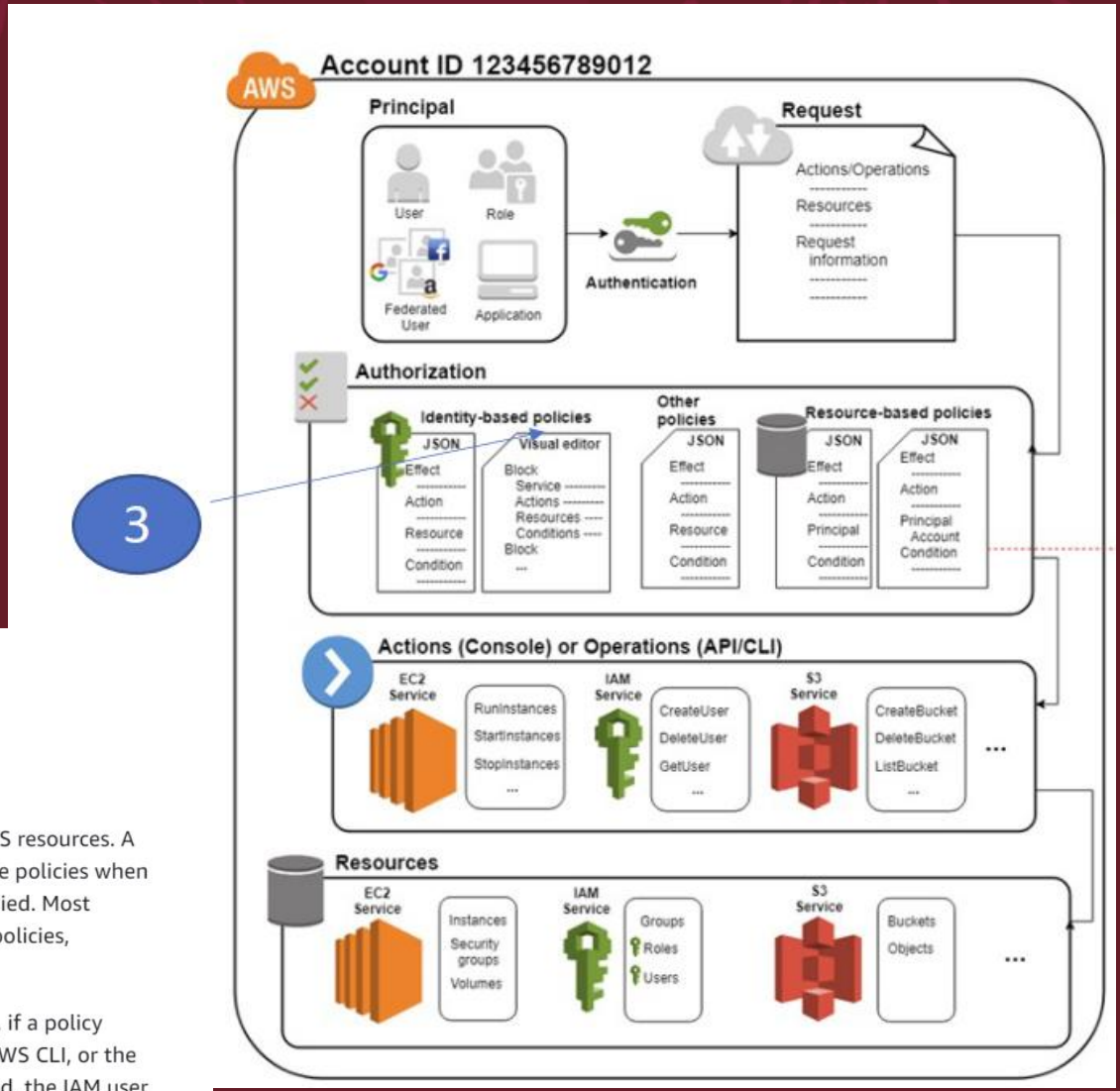
You must also be authorized (allowed) to complete your request. During authorization, AWS uses values from the request context to check for policies that apply to the request. It then uses the policies to determine whether to allow or deny the request. Most policies are stored in AWS as JSON documents and specify the permissions for principal entities. There are several types of policies that can affect whether a request is authorized. To provide your users with permissions to access the AWS resources in their own account, you need only identity-based policies. Resource-based policies are popular for granting cross-account access. The other policy types are advanced features and should be used carefully.

AWS checks each policy that applies to the context of your request. If a single permissions policy includes a denied action, AWS denies the entire request and stops evaluating. This is called an *explicit deny*. Because requests are *denied by default*, AWS authorizes your request only if every part of your request is allowed by the applicable permissions policies. The evaluation logic for a request within a single account follows these general rules:

- By default, all requests are denied. (In general, requests made using the AWS account root user credentials for resources in the account are always allowed.)
- An explicit allow in any permissions policy (identity-based or resource-based) overrides this default.
- The existence of an Organizations SCP, IAM permissions boundary, or a session policy overrides the allow. If one or more of these policy types exists, they must all allow the request. Otherwise, it is implicitly denied.
- An explicit deny in any policy overrides any allows.



IAM



Policies and permissions in IAM

PDF | Kindle | RSS

You manage access in AWS by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an IAM principal (user or role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. AWS supports six types of policies: identity-based policies, resource-based policies, permissions boundaries, Organizations SCPs, ACLs, and session policies.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, if a policy allows the GetUser action, then a user with that policy can get user information from the AWS Management Console, the AWS CLI, or the AWS API. When you create an IAM user, you can choose to allow console or programmatic access. If console access is allowed, the IAM user can sign in to the console using a user name and password. Or if programmatic access is allowed, the user can use access keys to work with the CLI or API.

Policies and permissions in IAM

PDF | Kindle | RSS

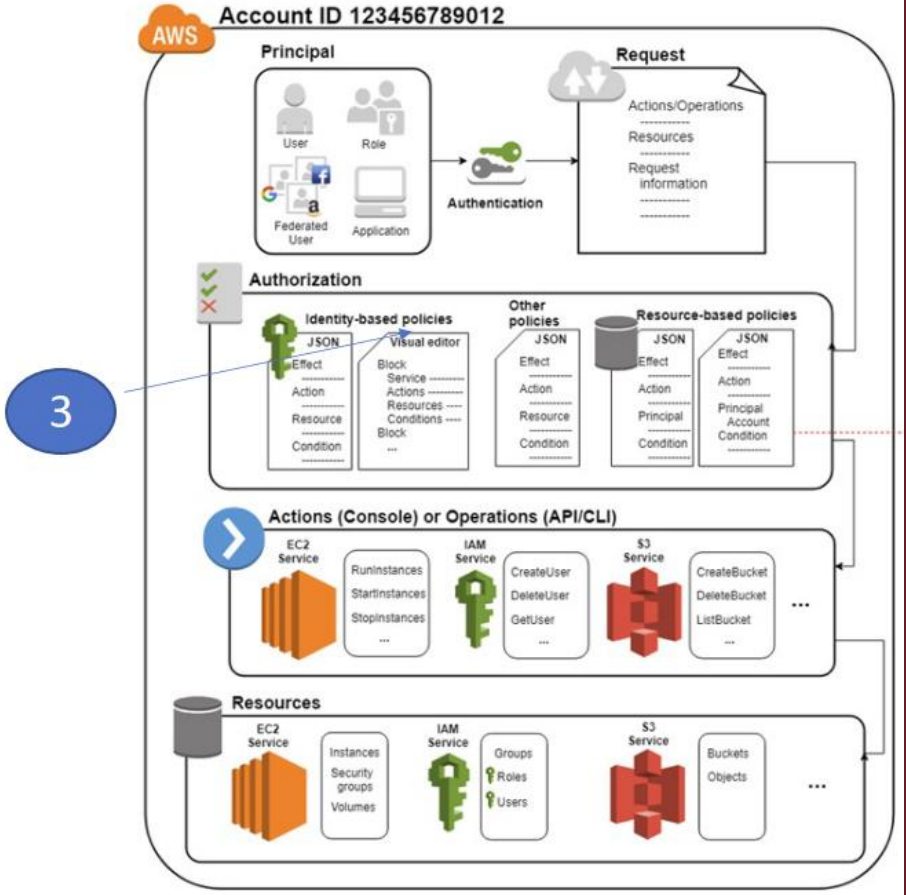
You manage access in AWS by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an IAM principal (user or role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. AWS supports six types of policies: identity-based policies, resource-based policies, permissions boundaries, Organizations SCPs, ACLs, and session policies.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, if a policy allows the GetUser action, then a user with that policy can get user information from the AWS Management Console, the AWS CLI, or the AWS API. When you create an IAM user, you can choose to allow console or programmatic access. If console access is allowed, the IAM user can sign in to the console using a user name and password. Or if programmatic access is allowed, the user can use access keys to work with the CLI or API.

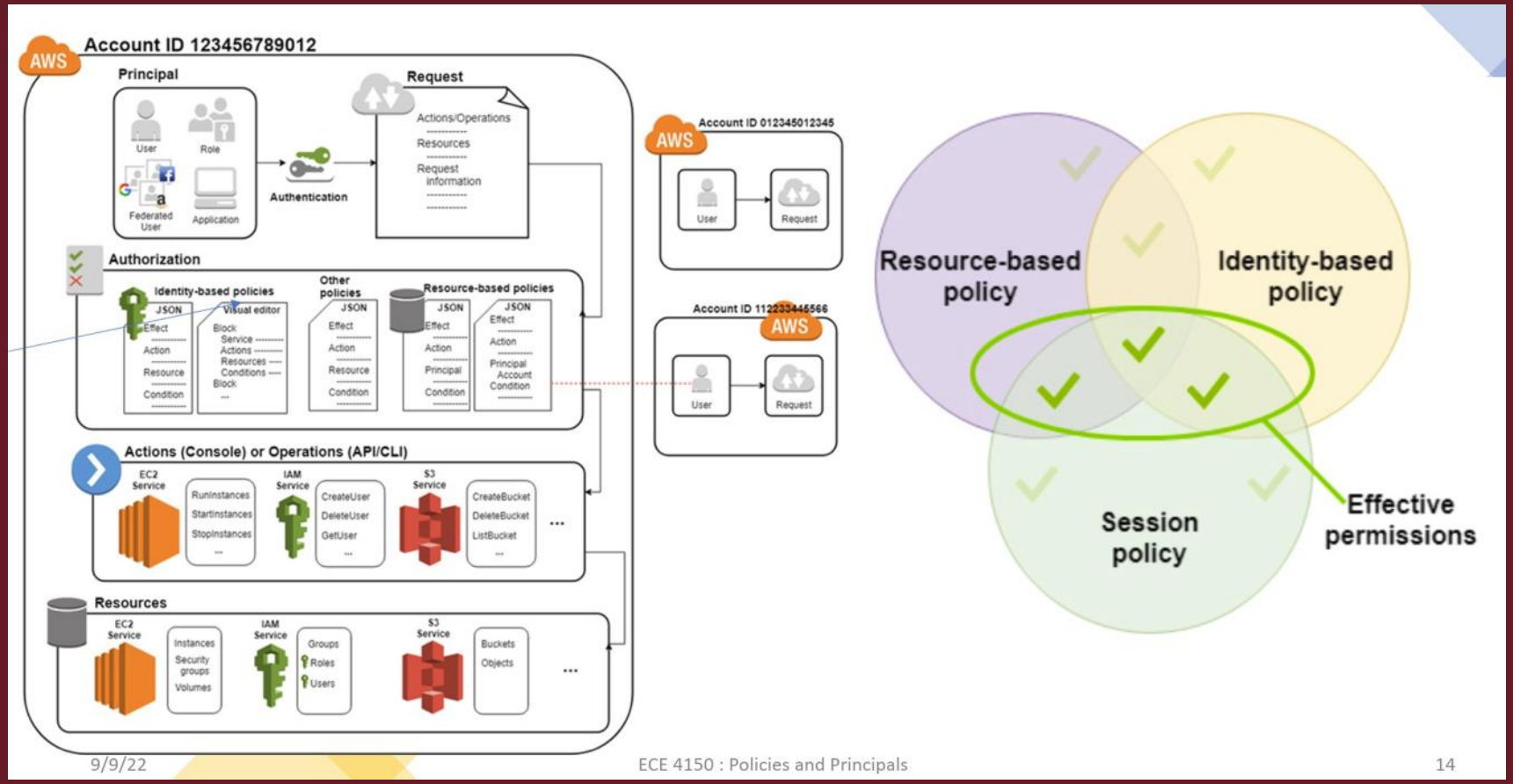
Policy types

The following policy types, listed in order of frequency, are available for use in AWS. For more details, see the sections below for each policy type.

- **Identity-based policies** – Attach managed and inline policies to IAM identities (users, groups to which users belong, or roles). Identity-based policies grant permissions to an identity.
- **Resource-based policies** – Attach inline policies to resources. The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies. Resource-based policies grant permissions to the principal that is specified in the policy. Principals can be in the same account as the resource or in other accounts.
- **Permissions boundaries** – Use a managed policy as the permissions boundary for an IAM entity (user or role). That policy defines the maximum permissions that the identity-based policies can grant to an entity, but does not grant permissions. Permissions boundaries do not define the maximum permissions that a resource-based policy can grant to an entity.
- **Organizations SCPs** – Use an AWS Organizations service control policy (SCP) to define the maximum permissions for account members of an organization or organizational unit (OU). SCPs limit permissions that identity-based policies or resource-based policies grant to entities (users or roles) within the account, but do not grant permissions.
- **Access control lists (ACLs)** – Use ACLs to control which principals in other accounts can access the resource to which the ACL is attached. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document structure. ACLs are cross-account permissions policies that grant permissions to the specified principal. ACLs cannot grant permissions to entities within the same account.
- **Session policies** – Pass advanced session policies when you use the AWS CLI or AWS API to assume a role or a federated user. Session policies limit the permissions that the role or user's identity-based policies grant to the session. Session policies limit permissions for a created session, but do not grant permissions. For more information, see Session Policies.



POLICIES


Account ID 123456789012
Principal
User
Role
Federated User
Application
Authentication
Request
Actions/Operations
Resources
Request information
Authorization
Identity-based policies
Other policies
Resource-based policies
JSON
Visual editor
Effect
Action
Resource
Condition
Block
Service
Actions
Resources
Conditions
Principal
Account
Actions (Console) or Operations (API/CLI)
EC2 Service
RunInstances
StartInstances
StopInstances
IAM Service
CreateUser
DeleteUser
GetUser
S3 Service
CreateBucket
DeleteBucket
ListBucket
Resources
Instances
Security groups
Volumes
Groups
Roles
Users
Buckets
Objects
Account ID 012345012345
Account ID 112233445566
Resource-based policy
Identity-based policy
Session policy
Effective permissions
9/9/22
ECE 4150 : Policies and Principals
14

4

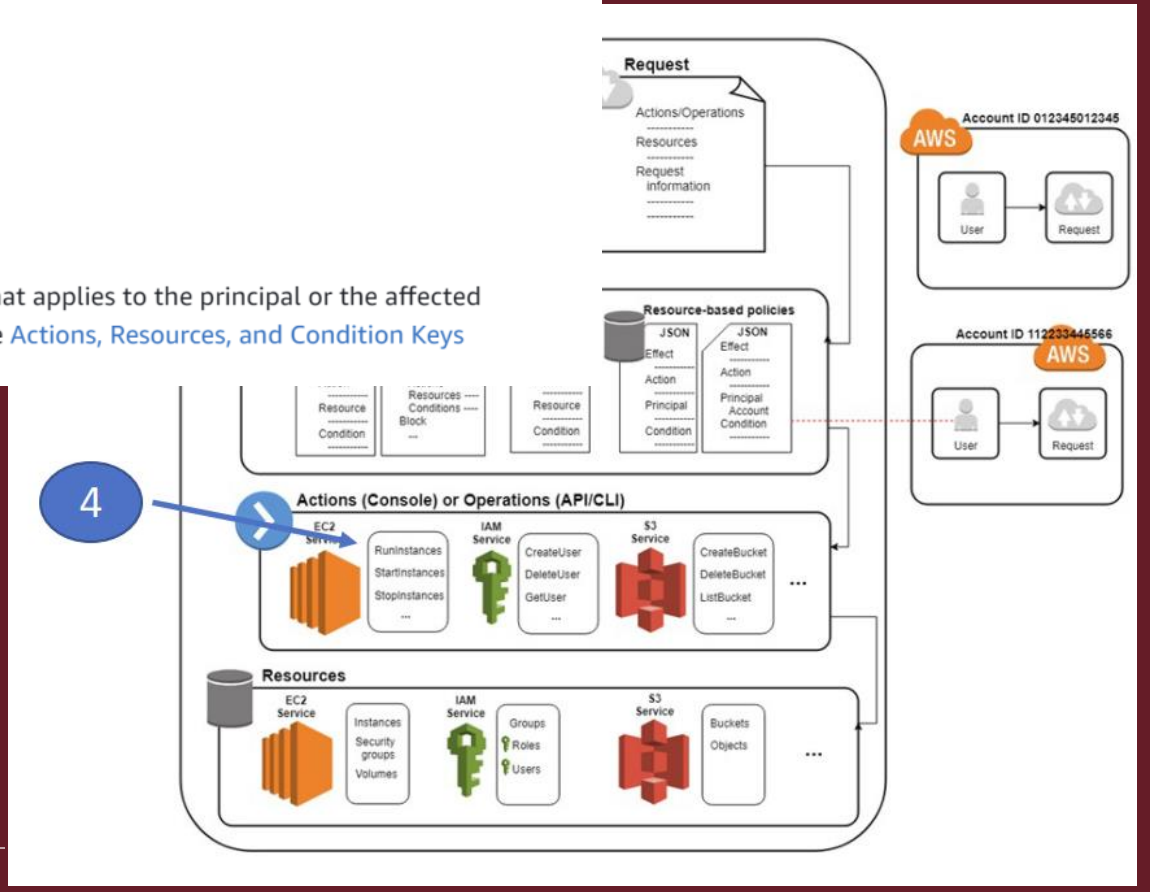
Actions or operations

After your request has been authenticated and authorized, AWS approves the actions or operations in your request. Operations are defined by a service, and include things that you can do to a resource, such as viewing, creating, editing, and deleting that resource. For example, IAM supports approximately 40 actions for a user resource, including the following actions:

- `CreateUser`
- `DeleteUser`
- `GetUser`
- `UpdateUser`

To allow a principal to perform an operation, you must include the necessary actions in a policy that applies to the principal or the affected resource. To see a list of actions, resource types, and condition keys supported by each service, see Actions, Resources, and Condition Keys for AWS Services.

ACTIONS



ROLES

IAM roles

PDF | Kindle | RSS

An IAM *role* is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but not want to embed AWS keys within the app (where they can be difficult to rotate and where users can potentially extract them). Sometimes you want to give AWS access to users who already have identities defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.

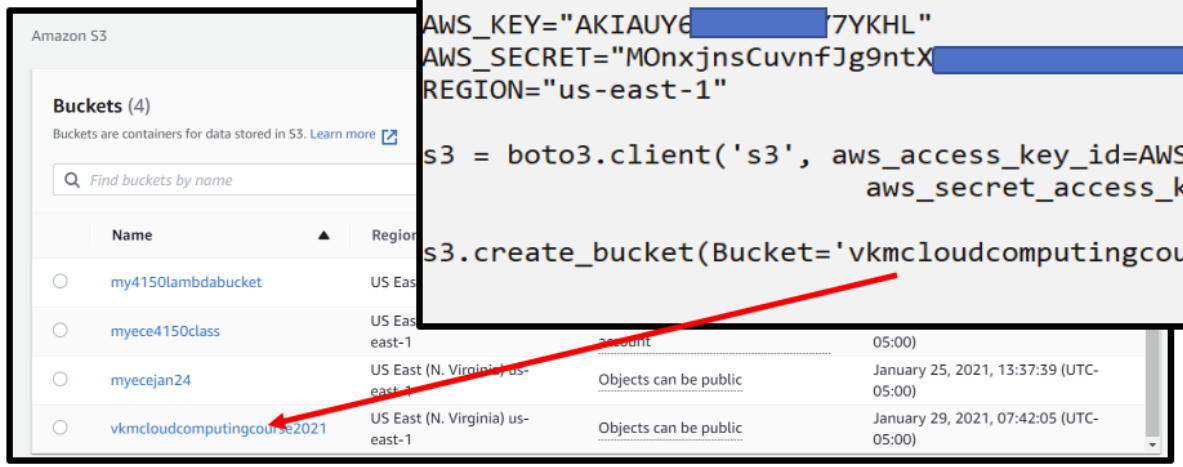
For these scenarios, you can delegate access to AWS resources using an *IAM role*. This section introduces roles and the different ways you can use them, when and how to choose among approaches, and how to create, manage, switch to (or assume), and delete roles.

PROGRAM IS PRINCIPAL

Using Programmatic Access API to create S3 bucket objects using Authenticated Identity Access Keys

Using Boto3 and Python to Create and Managed AWS Services

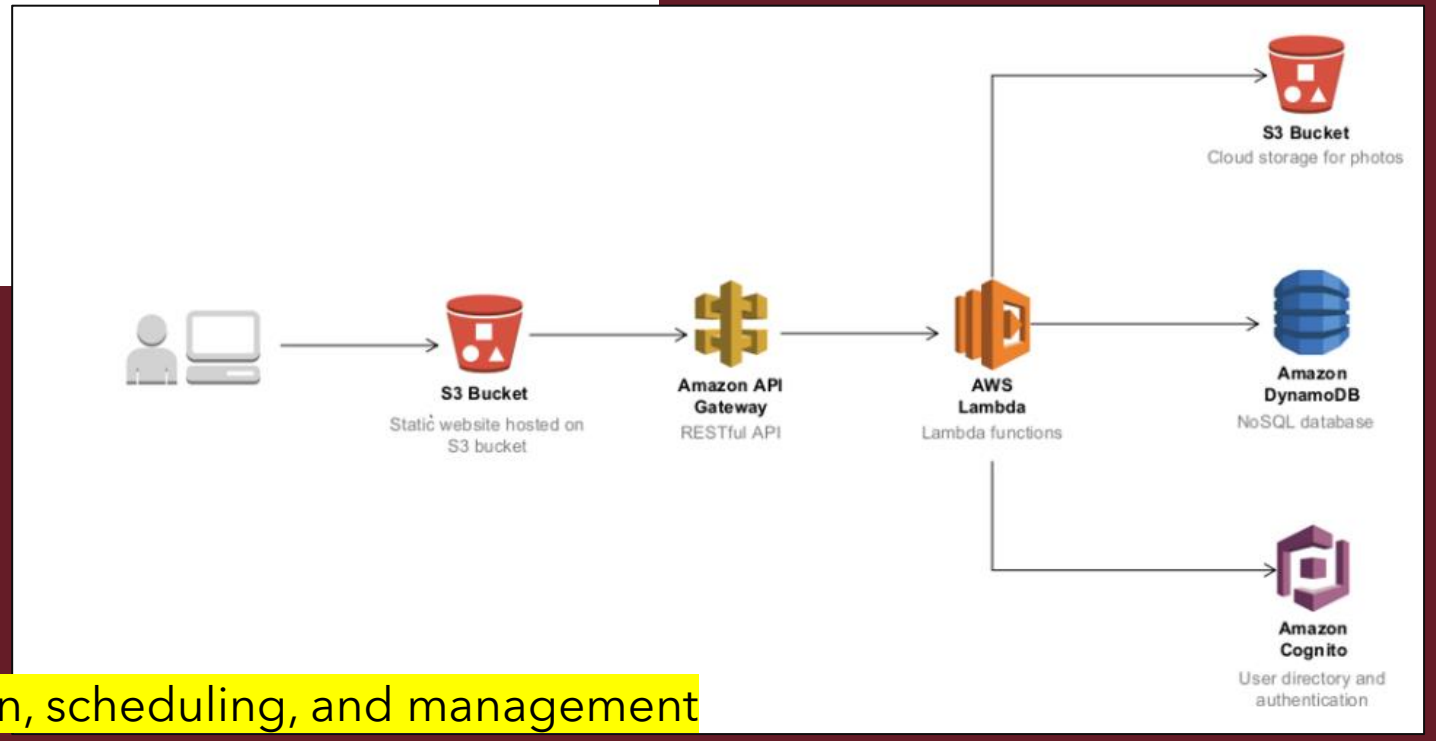
```
import boto3
import json

AWS_KEY="AKIAUY6[redacted]7YKHL"
AWS_SECRET="MOnxjnsCuvnfJg9ntX[redacted]/kVG"
REGION="us-east-1"

s3 = boto3.client('s3', aws_access_key_id=AWS_KEY,
                        aws_secret_access_key=AWS_SECRET)

s3.create_bucket(Bucket='vkmcloudcomputingcourse2021')
```



How AWS IAM and Services are used in creating a secure photo gallery application

The application uses the following:

1. A web interface implemented in HTML & JS, which can be served through S3 static website hosting
2. AWS API Gateway endpoints
3. Lambda functions
4. Cognito for user pool management
5. DynamoDB for storing records of photos
6. S3 for storing photos



AWS manages virtual machines and allocation, scheduling, and management
If you choose their "Lambda Service"

Summary

- You now know about the IaaS cloud computing model
- You learnt about the services that AWS offers you in addition to providing your virtual machines
- You learnt how IAM provides secure access to you shared resources in the cloud without allowing other organizations or users to access your applications and data
- You will learn how to use this knowledge in Lab 1