



Solutions Architect Professional

AWS Essentials, Accounts, Networking

Gururaj S Bayari

IAM

- Provides authorization
- Two parts:

The policy language

– Specification: *Defining* access policies
– Enforcement: *Evaluating* policies

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["s3:Get*", "s3>List*"],  
            "Resource": "*"  
        }  
    ]  
}
```

Policy specification basics



JSON-formatted documents



Contain a statement (permissions) that specifies:

- Which actions a principal can perform
- Which resources can be accessed

```
{  
  "Statement": [{  
    "Effect": "effect",  
    "Principal": "principal",  
    "Action": "action",  
    "Resource": "arn",  
    "Condition": {  
      "condition": {  
        "key": "value" }  
    }  
  }]  
}
```

Principal
Action
Resource
Condition

You can have multiple statements and each statement is comprised of PARC

Principal: Examples

Principal
Action
Resource
Condition

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

<!-- Everyone (anonymous users) -->

```
"Principal": "AWS": "*.*"
```

<!-- Specific account or accounts -->

```
"Principal": {"AWS": "arn:aws:iam::123456789012:root" }
```

```
"Principal": {"AWS": "123456789012" }
```

<!-- Individual IAM user -->

```
"Principal": "AWS": "arn:aws:iam::123456789012:user/username"
```

<!-- Federated user (using web identity federation) -->

```
"Principal": {"Federated": "accounts.google.com"}
```

<!-- Specific role -->

```
"Principal": {"AWS": "arn:aws:iam::123456789012:role/rolename"}
```

<!-- Specific service -->

```
"Principal": {"Service": "ec2.amazonaws.com"}
```

Replace
with your
account
number

Action: Examples

Principal
Action
Resource
Condition

- Describes the type of access that should be allowed or denied
- You can find actions in the docs or use the policy editor to get a drop-down list
- Statements must include either an Action or NotAction element

```
<!-- EC2 action -->
"Action":"ec2:StartInstances"

<!-- IAM action -->
"Action":"iam:ChangePassword"

<!-- Amazon S3 action -->
"Action":"s3:GetObject"

<!-- Specify multiple values for the Action element-->
"Action":["sns:SendMessage", "sns:ReceiveMessage"]

<-- Wildcards (*) or (?) in the action name. Below covers create/delete/list/update-->
"Action":"iam:*AccessKey*"
```

Understanding NotAction

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using Action and excludes many actions
- Example: Let's say you want to allow everything but IAM APIs

```
{  
  "Version": "2012-10-17",  
  "Statement": [ {  
    "Effect": "Allow",  
    "NotAction": "iam:*",  
    "Resource": "*"  
  }]  
}
```

This is not a Deny. A user could still have a separate policy that grants IAM:*

Is there a difference?

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

If you want to prevent the user from ever being able to call IAM APIs, use an explicit Deny.

Resource: Examples

Principal
Action
Resource
Condition

- The object or objects being requested
- Statements must include either a Resource or a NotResource element

```
<-- S3 bucket -->  
"Resource": "arn:aws:s3:::my_corporate_bucket/*"
```

```
<-- All S3 buckets, except this one -->  
"NotResource": "arn:aws:s3:::security_logging_bucket/*"
```

```
<-- Amazon SQS queue-->  
"Resource": "arn:aws:sqs:us-west-2:123456789012:queue1"
```

```
<-- Multiple Amazon DynamoDB tables -->  
"Resource": ["arn:aws:dynamodb:us-west-2:123456789012:table/books_table",  
            "arn:aws:dynamodb:us-west-2:123456789012:table/magazines_table"]
```

```
<-- All EC2 instances for an account in a region-->  
"Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*"
```

Replace
with your
account
number

Condition example

Principal
Action
Resource
Condition

What if you wanted to restrict access to a time frame and IP address range?

```
“Condition” : {  
    "DateGreaterThan" : {"aws:CurrentTime" : "2017-01-01T11:00:00Z"},  
    "DateLessThan": {"aws:CurrentTime" : "2017-12-31T15:00:00Z"},  
    "IpAddress" : {"aws:SourceIp" : ["192.0.2.0/24", "203.0.113.0/24"]}  
}
```

AND

OR

- Allows a user to access a resource under the following conditions:
 - The time is after 11 a.m. on 01/01/2017 AND
 - The time is before 3 p.m. on 12/31/2017 AND
 - The request comes from an IP address in the 192.0.2.0 /24 OR 203.0.113.0 /24 range

- Predefined variables based on service request context

- **Global keys** (aws:SourceIP, aws:MultiFactorAuthPresent, etc.)
- **Principal-specific keys** (aws:username, aws:userid, aws:PrincipalType)

Policy variables

- **Provider-specific keys** (graph.facebook.com:id, www.amazon.com:user_id)
- **SAML keys** (saml:cn, saml:edupersonassurance)
- See documentation for service-specific variables

- Benefits

- Simplify policy management
- Reduce the need for hard-coded, user-specific policies

The anatomy of a policy with variables

```
{  
  "Version": "2012-10-17", ← Version is required  
  "Statement": [  
    {"Effect": "Allow",  
     "Action": ["s3>ListBucket"],  
     "Resource": ["arn:aws:s3:::myBucket"],  
     "Condition":  
       {"StringLike":  
         {"s3:prefix": ["home/${aws:username}/*"]} ← Variable in conditions  
       }  
    },  
    {  
      "Effect": "Allow",  
      "Action": ["s3:*"],  
      "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}", ← Variable in resource ARNs  
                  "arn:aws:s3:::myBucket/home/${aws:username}/*"]  
    }  
  ]  
}
```

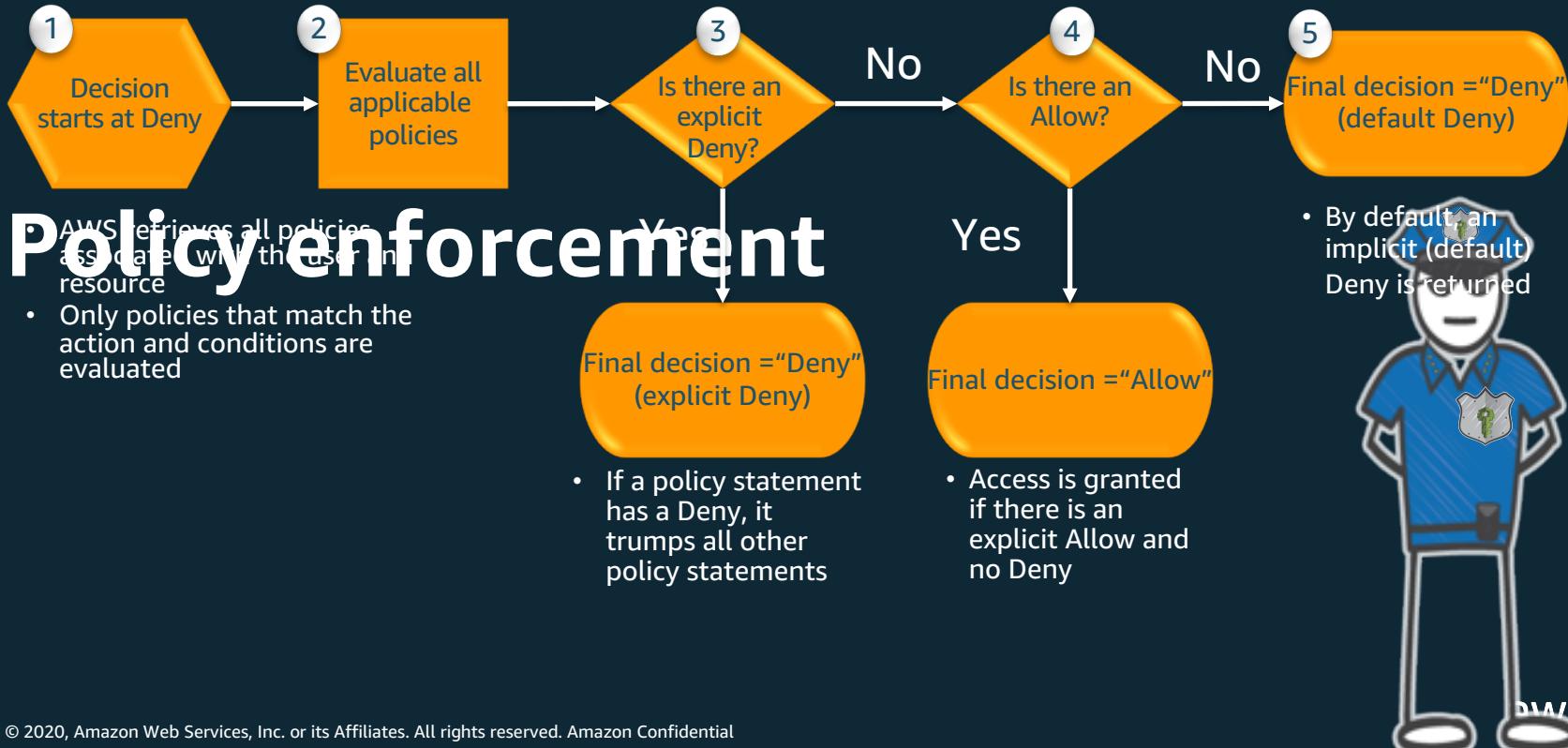
Grants a user access to a home directory in Amazon S3 that can be accessed programmatically



Policy enforcement

AWS retrieves all policies associated with the user and resource

- Only policies that match the action and conditions are evaluated



Federation

Identity federation benefits

Users



Before: Unique credentials

After: **Single sign-on (SSO)**

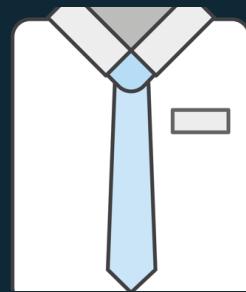
Security



Long-term access keys

Temporary security credentials

Compliance

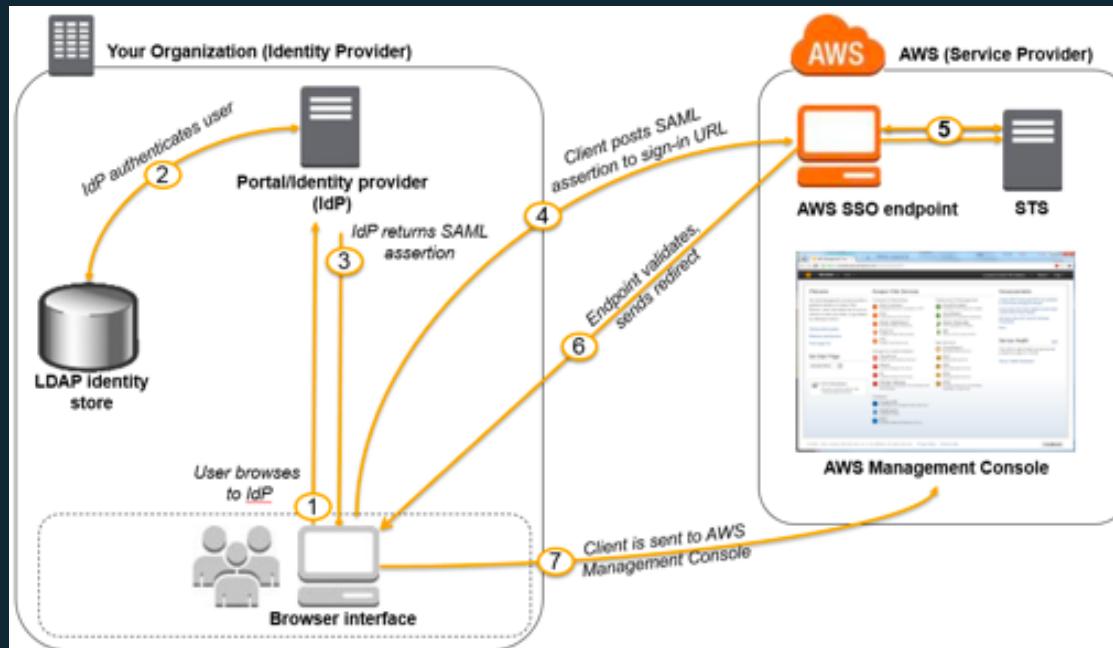


One-off

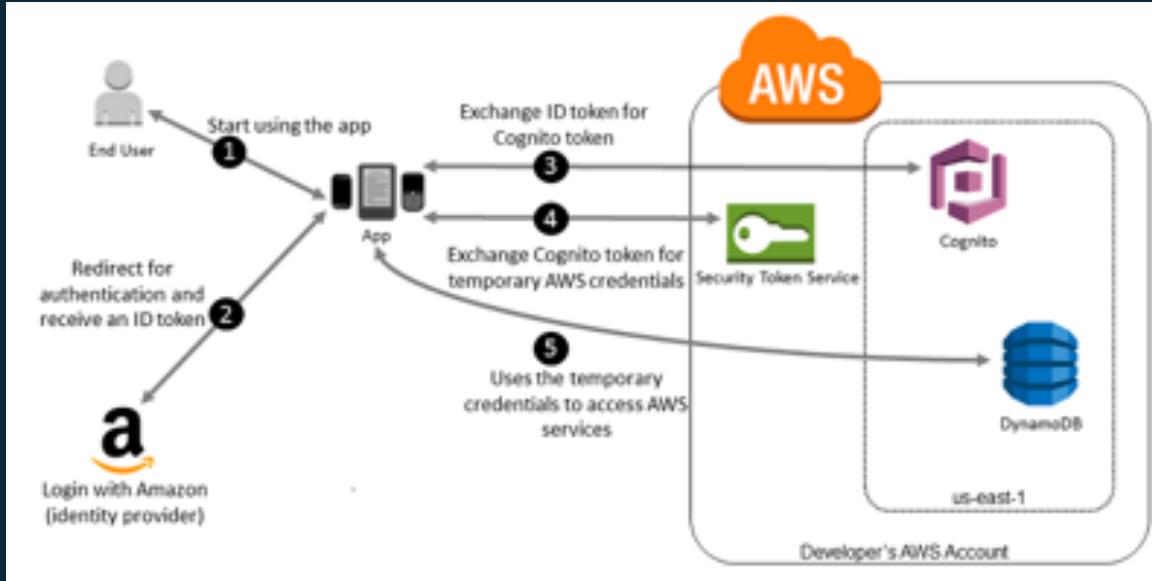
Aligned with internal controls

SAML 2.0 Federation

Example
SAML 2.0
Providers



Web Identity Federation (OpenID Connect)



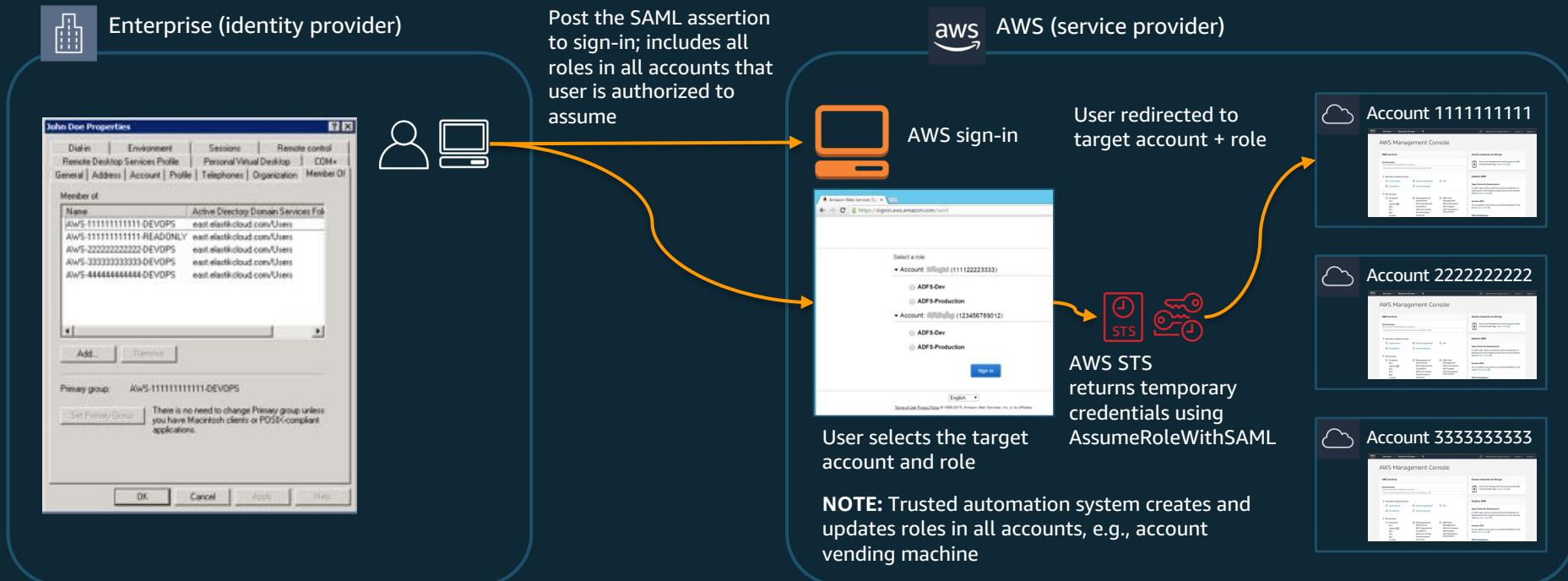
Example
OIDC
Providers



Federation patterns

Model 1: Direct federation (fine-grained)

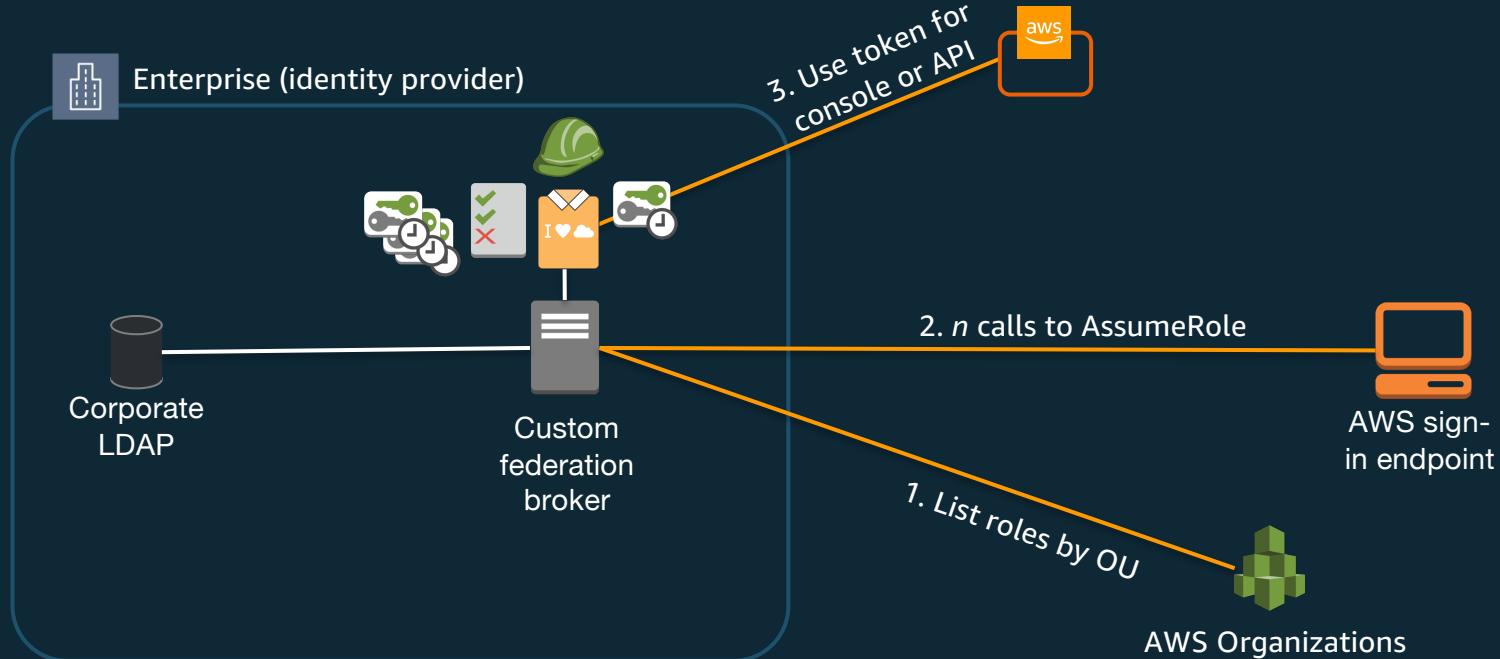
Description: One LDAP group per AWS account + role combination; naming convention for maintainable IdP rules



IdP maps group **AWS-111111111111-RoleName** to **arn:aws:iam::111111111111:role/RoleName**

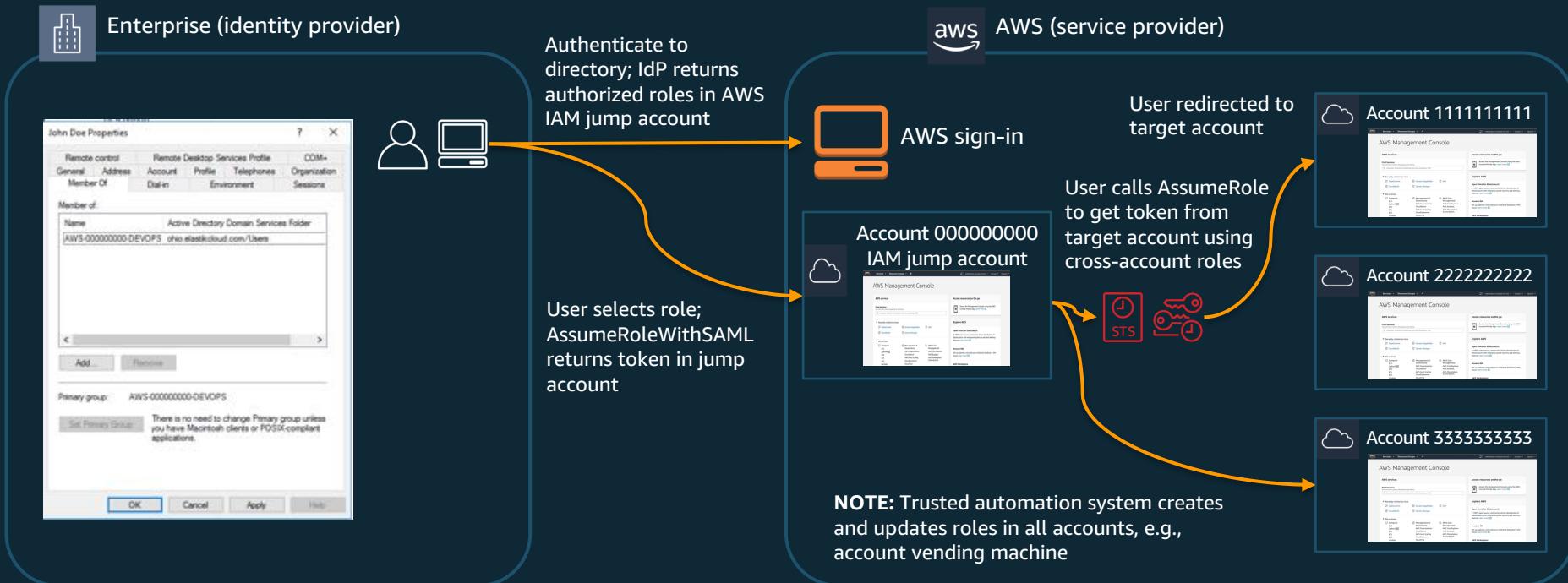
Model 2: Direct federation (coarse-grained)

Description: One LDAP group per AWS Organizations OU + role combination; custom or third-party API-aware broker



Model 3: Hub-and-spoke

Description: Federate into AWS jump account, then make AuthZ decision and assume second role in target account



Custom logic or policy in your control plane determines which roles can be assumed



Model 4: No federation; native IAM + automation

Description: Manage entitlements with IAM users, groups, and roles; consider SSO and token vending solution

- A unique IAM user object for every authorized AWS user
 - Consider commercial SSO/password synchronization solutions
- Vault credentials for non-humans, JIT access from code
 - Use Amazon EC2 instance profile/execution roles when available (AWS Lambda)
- Password expiration and complexity policies enforced
- Users mapped to IAM groups by job function
- Groups granted minimal AWS control plane and resource access
- Token vending solution layered on for higher-privileged operations

Summary

	Direct federation (fine-grained)	Direct federation (coarse-grained)	Hub-and-spoke	Native IAM
Audit complexity	Low	Medium	High	High
Joiners, leavers, and movers complexity	Low	Low	Medium	High
Scalability	Highly scalable	Highly scalable	Hard to scale	Hard to scale
Developer impact	Low	Low	High	Medium
Impact on directory ops	High	Medium	Low	Very low
Impact on IdP ops	Low	Medium	Low	N/A
Permissions granularity	Fine	Coarse	Coarse	Fine
Custom code required	Low	High	Medium	Medium

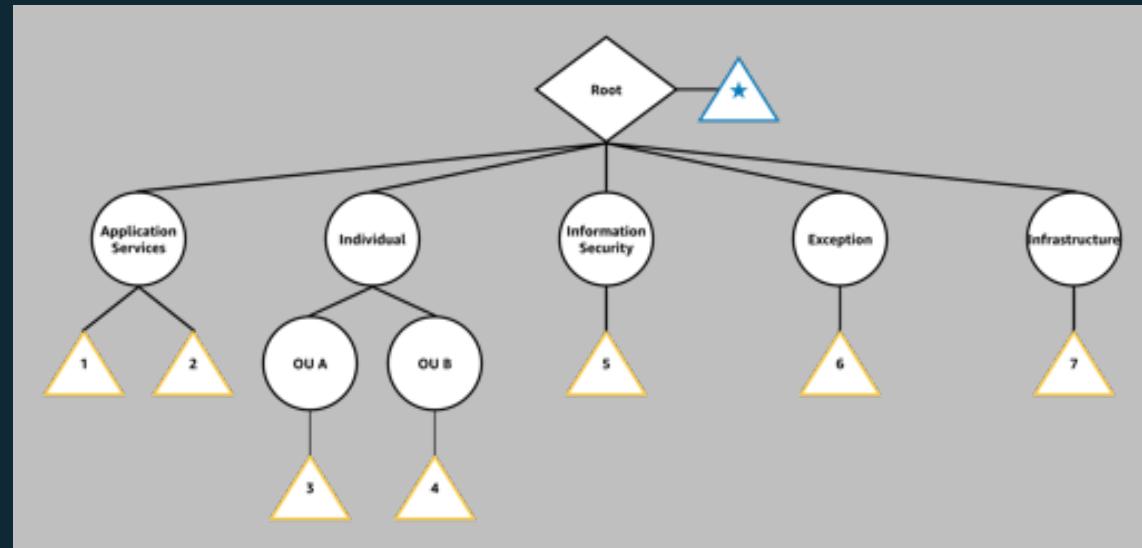
AWS Organizations

Service Control Policies (SCPs)

Define the maximum available permissions for IAM entities in an account

SCPs do not grant permission

Attach SCPs to the organization root, OUs, and individual accounts



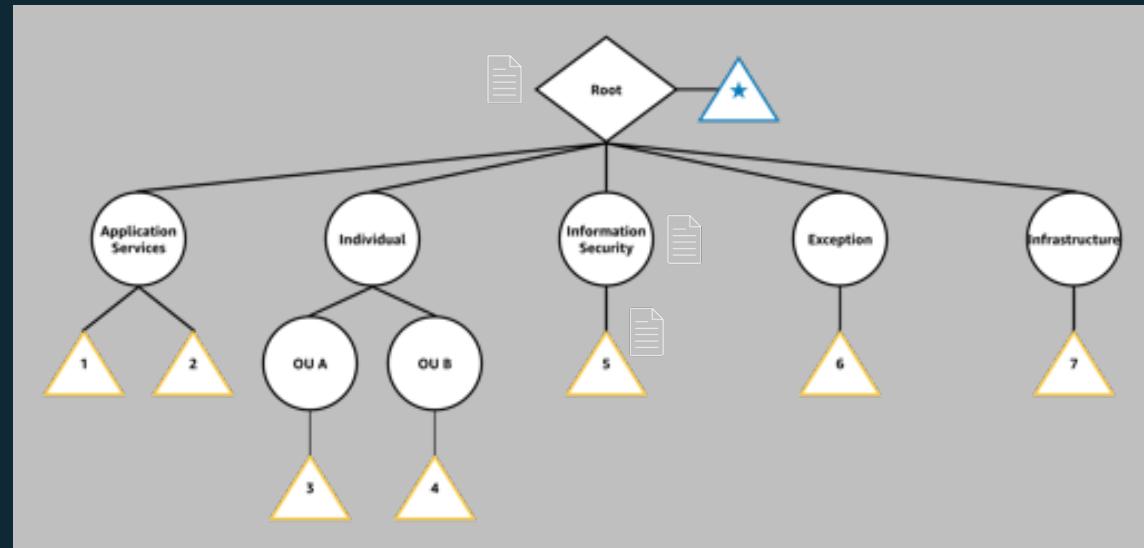
SCPs attached to the root and OUs apply to all OUs and accounts inside of them

Service Control Policies (SCPs)

Define the maximum available permissions for IAM entities in an account

SCPs do not grant permission

Attach SCPs to the organization root, OUs, and individual accounts



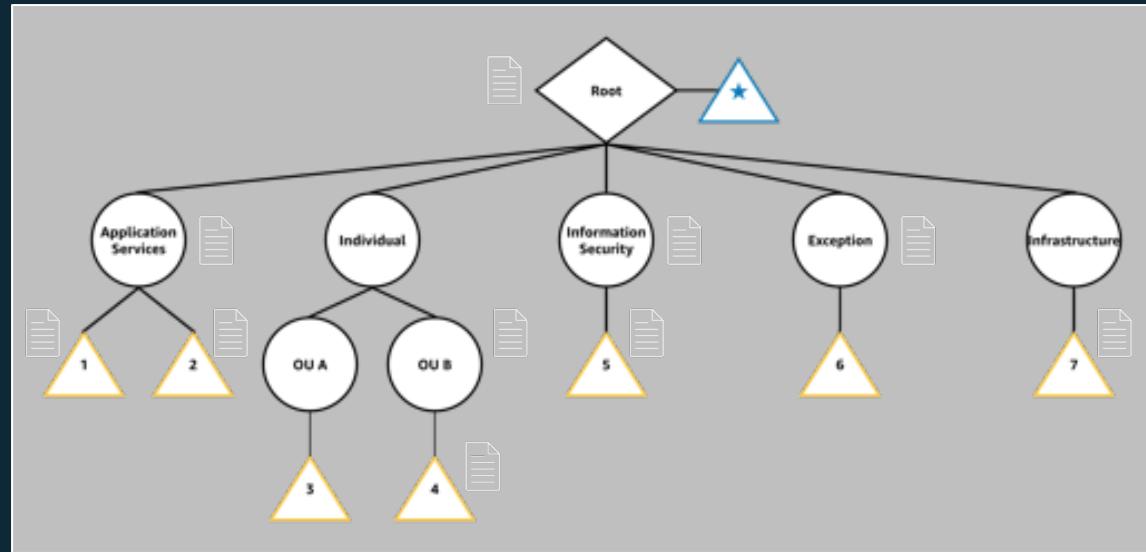
SCPs attached to the root and OUs apply to all OUs and accounts inside of them

Service Control Policies (SCPs)

Define the maximum available permissions for IAM entities in an account

SCPs do not grant permission

Attach SCPs to the organization root, OUs, and individual accounts



SCPs attached to the root and OUs apply to all OUs and accounts inside of them

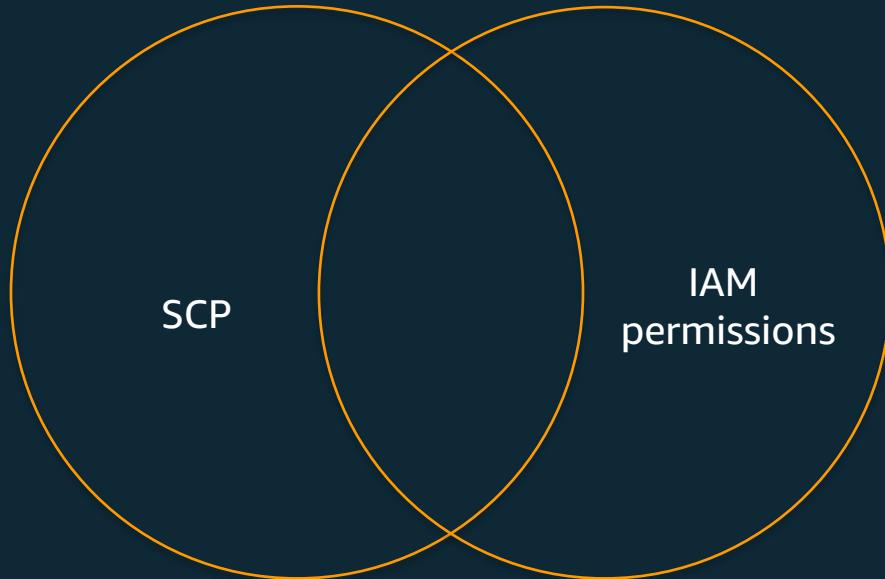
Service Control Policies (SCPs)

Define the maximum available permissions for IAM entities in an account

SCPs do not grant permission

Attach SCPs to the organization root, OUs, and individual accounts

SCPs attached to the root and OUs apply to all OUs and accounts inside of them



Allow EC2:
Allow SQS:*

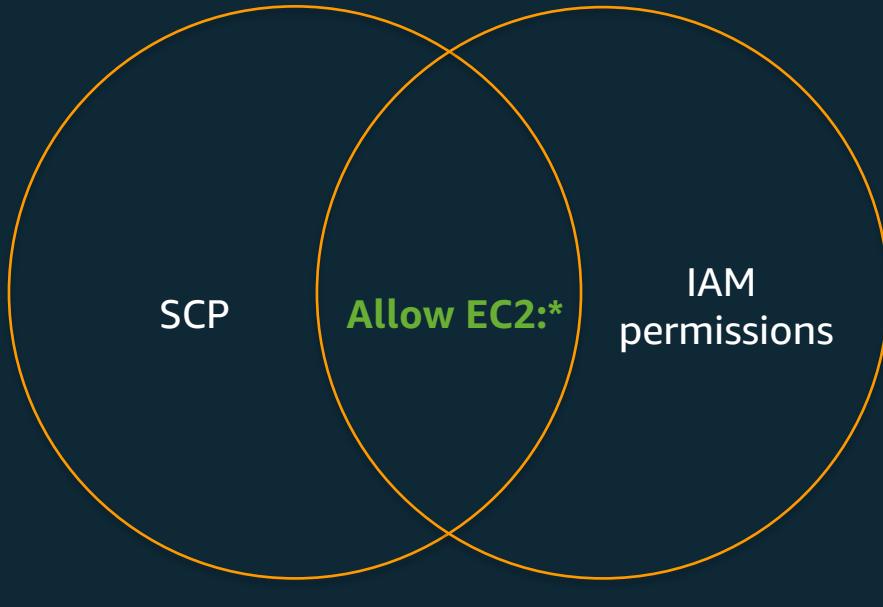
Service Control Policies (SCPs)

Define the maximum available permissions for IAM entities in an account

SCPs do not grant permission

Attach SCPs to the organization root, OUs, and individual accounts

SCPs attached to the root and OUs apply to all OUs and accounts inside of them



Allow EC2:
Allow S3:*

Allow EC2:
Allow SQS:*

Authoring SCPs

Whitelist - only allow specific actions¹

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ec2:*", "ds:*", "s3:*"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

1. Resource/conditionals supported on Deny statements only

Blacklist – block specific actions²

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": [  
        "organizations:LeaveOrganization",  
        "ec2:TerminateInstances"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

2. Requires an Allow: * policy to avoid denying all services

Authoring SCPs

Deny access to AWS based on the requested region

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyAllOutsideEU",  
      "Effect": "Deny",  
      "NotAction": [  
        "iam:*", "organizations:*", "route53:*",  
        "cloudfront.*", "support:*"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "StringNotEquals": {  
          "aws:RequestedRegion": [  
            "eu-central-1", "eu-west-1"  
          ]  
        }  
      }  
    }  
  ]  
}
```

Authoring SCPs

Deny access to AWS based on the requested region

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DenyAllOutsideEU",  
            "Effect": "Deny",  
            "NotAction": [  
                "iam:*", "organizations:*", "route53:*,  
                "cloudfront.*", "support:*"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringNotEquals": {  
                    "aws:RequestedRegion": [  
                        "eu-central-1", "eu-west-1"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

Authoring SCPs

Deny access to AWS based on the requested region

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyAllOutsideEU",  
      "Effect": "Deny",  
      "NotAction": [  
        "iam:*", "organizations:*", "route53:*",  
        "cloudfront:*", "support:*"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "StringNotEquals": {  
          "aws:RequestedRegion": [  
            "eu-central-1", "eu-west-1"  
          ]  
        }  
      }  
    }  
  ]  
}
```

Prevent IAM principals from making changes to a common administrative IAM role

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyAccessToASpecificRole",  
      "Effect": "Deny",  
      "Action": [  
        "iam:AttachRolePolicy", "iam:DeleteRole",  
        "iam:DeleteRolePermissionsBoundary",  
        "iam:DeleteRolePolicy", "iam:DetachRolePolicy",  
        "iam:PutRolePermissionsBoundary",  
        "iam:PutRolePolicy", "iam:UpdateRole",  
        "iam:UpdateAssumeRolePolicy",  
        "iam:UpdateRoleDescription"  
      ],  
      "Resource": [  
        "arn:aws:iam::*:role/AWS-ADMINISTRATOR-ROLE"  
      ]  
    }  
  ]  
}
```

Authoring SCPs

Deny access to AWS based on the requested region

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyAllOutsideEU",  
      "Effect": "Deny",  
      "NotAction": [  
        "iam:*", "organizations:*", "route53:*",  
        "cloudfront:*", "support:*"  
      ],  
      "Resource": "*",  
      "Condition": {  
        "StringNotEquals": {  
          "aws:RequestedRegion": [  
            "eu-central-1", "eu-west-1"  
          ]  
        }  
      }  
    }  
  ]  
}
```

Prevent IAM principals from making changes to a common administrative IAM role

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "DenyAccessToASpecificRole",  
      "Effect": "Deny",  
      "Action": [  
        "iam:AttachRolePolicy", "iam:DeleteRole",  
        "iam:DeleteRolePermissionsBoundary",  
        "iam:DeleteRolePolicy", "iam:DetachRolePolicy",  
        "iam:PutRolePermissionsBoundary",  
        "iam:PutRolePolicy", "iam:UpdateRole",  
        "iam:UpdateAssumeRolePolicy",  
        "iam:UpdateRoleDescription"  
      ],  
      "Resource": [  
        "arn:aws:iam::*:role/AWS-ADMINISTRATOR-ROLE"  
      ]  
    }  
  ]  
}
```

Best Practice #1

Use Federated Access from Outside of AWS

Use the same identity management system you use for everything else in your organization.



<https://aws.amazon.com/iam/details/manage-federation/>

Best Practice #2

Use IAM roles not accounts

IAM access keys are just waiting to be leaked

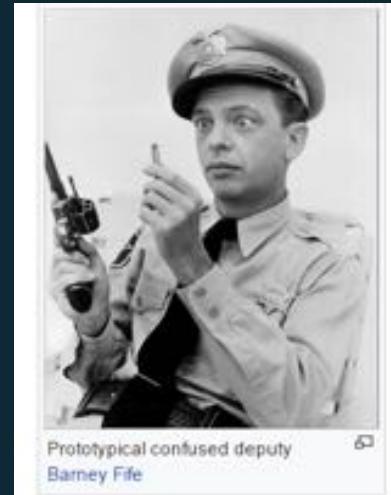
Examples:

Cross-account roles, instance roles, etc....

*When using IAM cross-account roles
don't forget about the confused
deputy problem:*

https://en.wikipedia.org/wiki/Confused_deputy_problem

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential



Best Practice #3

Rotate all IAM access and secret access keys on a regular basis

Set a policy for how often

Policy can vary based on situation

Wait to see how hard it is to actually accomplish

<https://aws.amazon.com/blogs/security/how-to-rotate-access-keys-for-iam-users/>



Best Practice #4

Don't leak your AWS access keys

Pushing Access Keys to GitHub is a common accident

What to Do If You Inadvertently Expose an AWS Access Key

<https://aws.amazon.com/blogs/security/what-to-do-if-you-inadvertently-expose-an-aws-access-key/>

<https://wptavern.com/ryan-hellyers-aws-nightmare-leaked-access-keys-result-in-a-6000-bill-overnight>

Ryan Hellyer's AWS Nightmare:
Leaked Access Keys Result in a \$6,000 Bill Overnight

• Sarah Gooding • September 26, 2014 • 26

WordPress developer Ryan Hellyer had always wanted to open source his website. As a strong supporter of open source software and an avid plugin developer, he enjoys sharing his code and learning from others. This desire

Best Practice #5

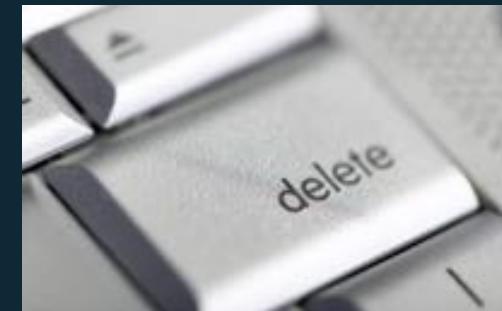
Delete Stale or Unused Access Keys

If an IAM access key is not being used, delete it

More complicated deleting root access keys

Create a policy around aging of stale or unused
90 day? 365 day?

Look at AWS method ***generate-credential-report***



Best Practice #6

Set IAM Password Policies

Account by Account setting

Nine Password Policy Options

- Minimum password length
- Require at least one uppercase letter
- Require at least one lowercase letter
- Require at least one number
- Require at least one nonalphanumeric character
- Allow users to change their own password
- Enable password expiration
- Prevent password reuse
- Password expiration requires administrator reset

http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_account-policy.html

Best Practice #7

*Do not grant permissions to IAM users,
grant to IAM groups and assign users to groups*

Managing user permissions is complex

Control permissions on groups

Assign users to groups



Best Practice #8

Find unused IAM accounts

Attack surface that doesn't need to be there



User left organization? Is a default password still on the account?

Set your organizational policy on how long is unused or stale

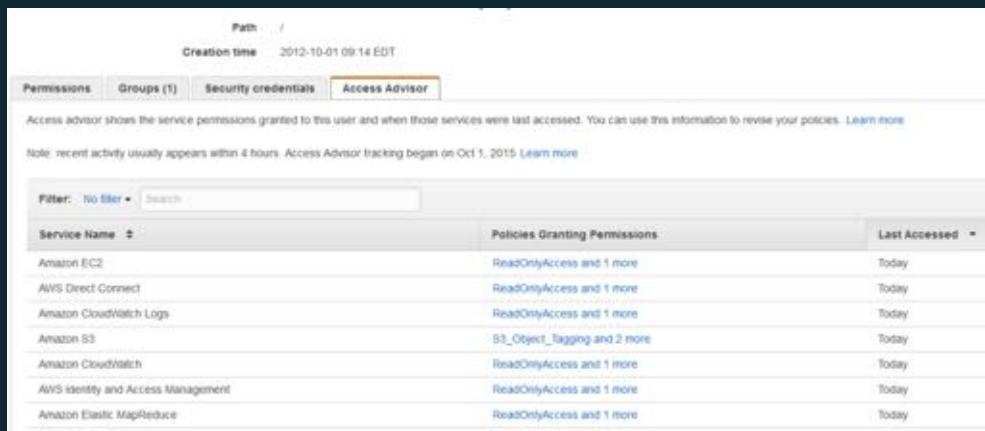
Best Practice #9

Find accounts with permissions they don't actually need

Principle of least privileges

Use IAM Access Advisor tab

Revoke what you don't need



The screenshot shows the AWS IAM Access Advisor interface. At the top, there are tabs for 'Permissions', 'Groups (1)', 'Security credentials', and 'Access Advisor', with 'Access Advisor' being the active tab. Below the tabs, there is a note about recent activity and a note about Access Advisor tracking. A filter bar allows for searching by service name. The main table lists various AWS services along with the policies granting them permissions and the last time they were accessed.

Service Name	Policies Granting Permissions	Last Accessed
Amazon EC2	ReadOnlyAccess and 1 more	Today
AWS Direct Connect	ReadOnlyAccess and 1 more	Today
Amazon CloudWatch Logs	ReadOnlyAccess and 1 more	Today
Amazon S3	S3_Object_Tagging and 2 more	Today
Amazon CloudWatch	ReadOnlyAccess and 1 more	Today
AWS Identity and Access Management	ReadOnlyAccess and 1 more	Today
Amazon Elastic MapReduce	ReadOnlyAccess and 1 more	Today

Best Practice #10

Monitor All IAM Activity

AWS CloudTrail records each time the AWS API is called

- Currently supports most AWS services
- You have to enable it in every account/region

Conveniently everything in AWS goes through the API

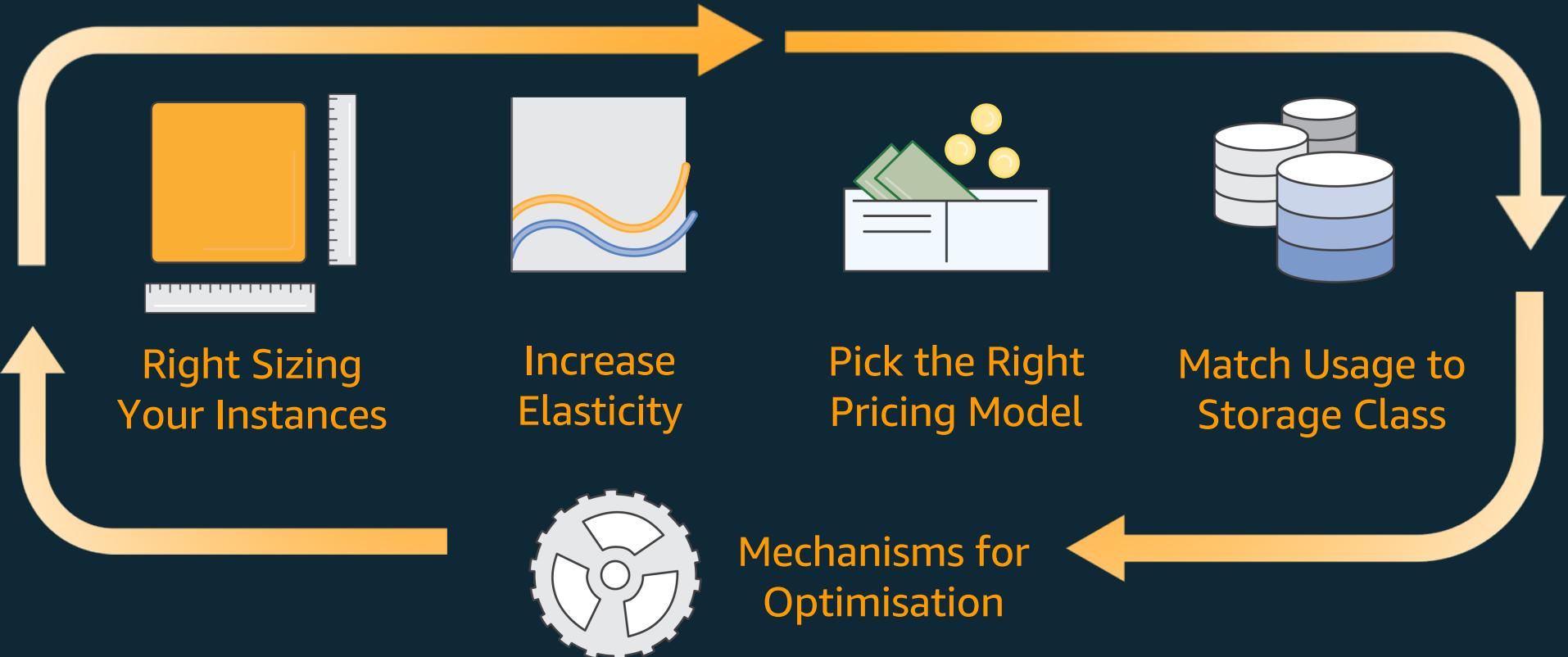
- Even actions in the AWS Management Console go through the API

CloudTrail writes files into an Amazon S3 bucket

- Near real-time (every five minutes)
- Files are in JSON format

Cost Optimization

The Five Pillars of Cost Optimization





Pillar 1: Right Sizing

**Cheapest Instance / Best
Performance
Monitor Usage**

Right Sizing example

m4.4xlarge
\$0.80 per hr

1. Migrate/provision &
Run



m4.large
\$0.10 per hr

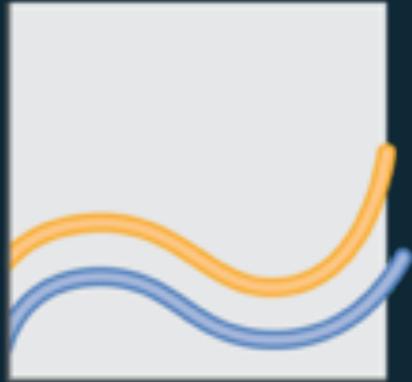
3. Right Size



4. Review Performance

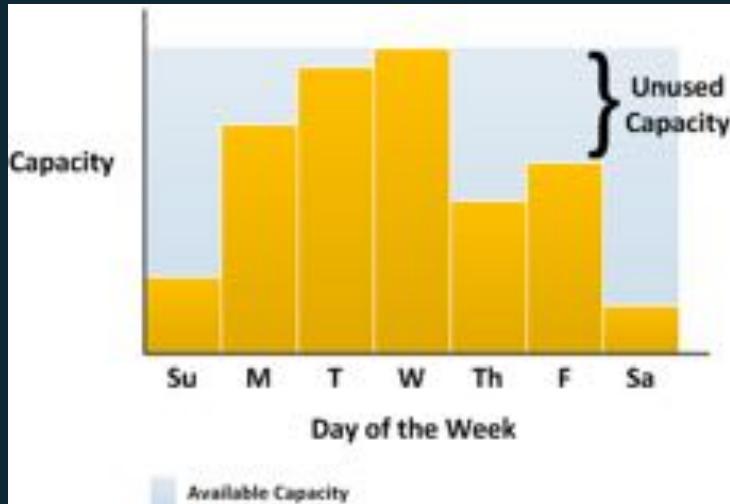
87%
Saving

5. Save!

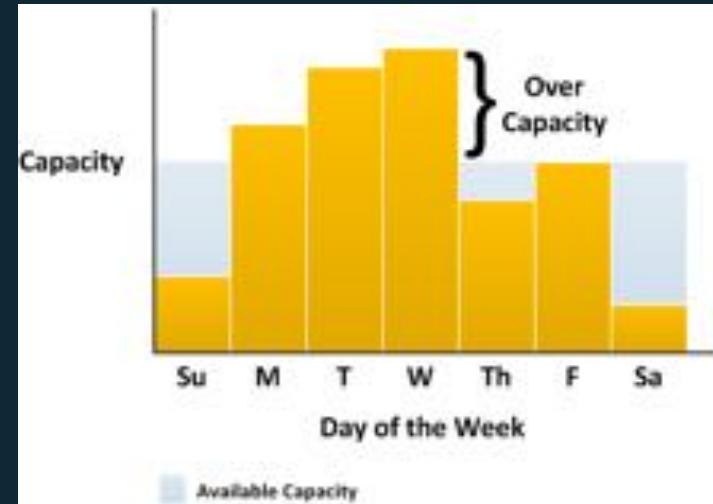


Pillar 2: Increase Elasticity Autoscaling

Traditional approaches to capacity management



Build to peak load

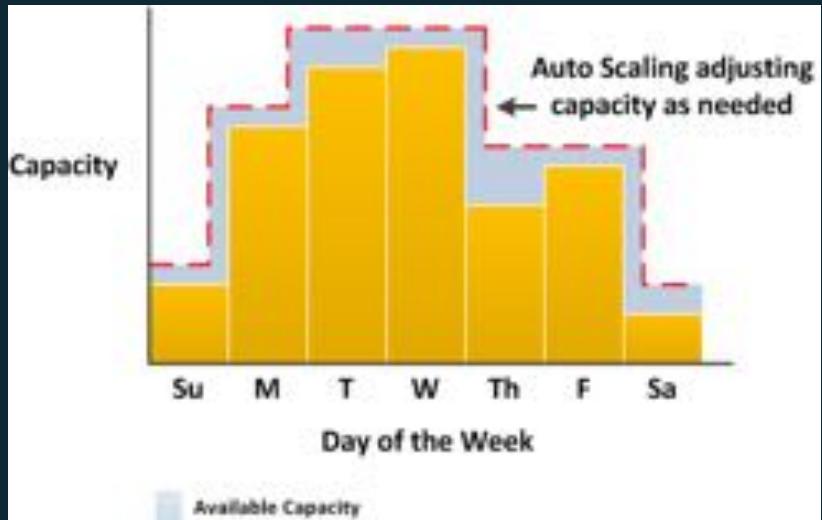


Build to average load

Lower over-provisioning via elasticity

Auto Scaling allows you to:

- React dynamically to changes in load
- Schedule regular workloads
- Optimise your instance usage
- Reduce over-provisioning
- Complimentary service!





Pillar 3: Leveraging the Right Pricing Model

Reserved Instances
EC2 Spot
On-Demand

What are Reserved Instances (RIs)?

RI coupon



- Discount for commitment
- Expires after 1 hour
- Doesn't map directly to an Instance

Up to 75%
Savings

EC2 Spot provides **incredible savings** for the right workloads

**Up to
90%
Savings**

Spot can be used for

- Stateless
- Fault-tolerant
- Big data
- Containers
- CI/CD
- Web servers
- High Performance Compute
- Dev/Test
- See <https://amzn.to/2wKNIT1>

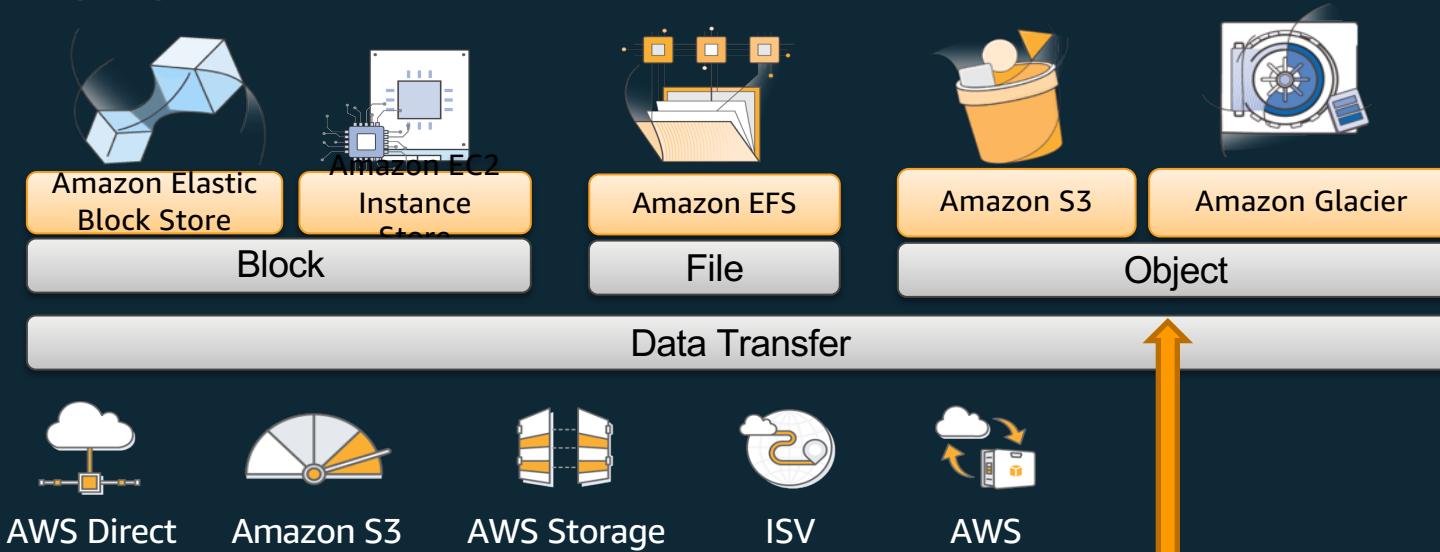


Pillar 4: Leveraging the Right Storage Class

**Elastic Block Storage
Simple Storage Service**

Amazon Storage Classes/Platforms

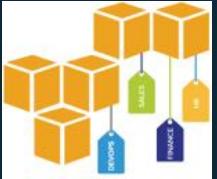
- Server/Instance Storage
 - Local Filesystems
 - Networked Volumes (EBS)
- Distributed Flesystem
- Object Storage
 - Backup Target – Snapshots,...





Pillar 5: Mechanisms for Cost Optimization

Summary of Mechanisms for Optimisation



Define, agree, and enforce
cost allocation tagging



Define metrics, set
targets, and review



Enable teams to
architect for cost



Assign optimization
responsibility

Networking

Basics quiz

What is the minimum and maximum size for a VPC CIDR?

A. /24 -- /8

B. /32 -- /16

C. /28 -- /16

D. /28 -- /8

Basics quiz

What is the minimum and maximum size for a VPC CIDR?

A. /24 -- /8

B. /32 -- /16

C. /28 -- /16

D. /28 -- /8

Basics quiz

V
n
C

/16 = 65,536 addresses
/17 = 32,728
/18 = 16,384
/19 = 8,192
/20 = 4,096
/21 = 2,048
/22 = 1,024
/23 = 512
/24 = 256
/25 = 128
/26 = 64
/27 = 32
/28 = 16

5 reserved IPs per range
+0 = Network
+1 = Gateway
+2 = DNS
+3 = Reserved
N-1 = Broadcast

Basics quiz

How many additional CIDR ranges can be assigned to a VPC?

A. 1

B. 2

C. 3

D. 4

E. 5

Basics quiz

How many additional CIDR ranges can be assigned to a VPC?

A. 1

B. 2

C. 3

D. 4

E. 5

Basics quiz

You may have 1 primary CIDR block and up to 4 additional CIDR blocks assigned to a VPC

You should avoid overlapping IP addresses



VPC Firewall

Security Group

- Stateful
- Only specify allow rules
- Default = deny all
- If traffic is allowed out, the return traffic is automatically allowed back in
- Can reference other SGs

NACL

- Stateless
- Can allow or deny
- Default = allow all
- Have to worry about ephemeral ports
- Rules processed in order

VPC Connectivity Options

- IGW Make a subnet public by routing to IGW
- NAT GW Enable private instances to connect to the internet
- Egress-only IGW Like NAT GW for IPv6
- VPG Site-to-Site VPN
- Peering Connect 2 VPCs to each other
- ClassicLink Enable EC2-Classic instances to connect to VPC
- VPC Endpoint Private connectivity between VPC and AWS Service
- Outposts LGW Route to network where Outposts resides
- Middleware Intercept traffic with an appliance (ie. Filtering, DLP)
- TGW Let's talk about it...

AWS Transit Gateway

Easily interconnect thousands of VPCs and on-premise networks



AWS Transit Gateway: Key features



- Centralized routing policies across VPCs and on-prem
- Scales to support thousands of VPCs across multi-accounts
- Increase connectivity throughput with multi-vpn connections
- Flexible segmentation and routing rules
- Horizontally scalable
- Simplified management

AWS Transit Gateway: Benefits

	Simplified Networking	<ul style="list-style-type: none">• Centrally interconnect multiple VPCs across accounts• One central connection point for VPN and Direct Connect• Reduce or eliminate need for peer to peer networking• Increase VPN throughput via ECMP routing (50 Gbps+)
	Global Connectivity	<ul style="list-style-type: none">• Peer AWS Transit Gateway across regions• Leverage the AWS Global Network for low latency cross-region connectivity• Regional construct reduces blast radius
	Easy manageability	<ul style="list-style-type: none">• Reduces time to configure on premise connectivity to AWS• Easily monitor and manage from a central point• Integrated with CloudWatch and VPC Flow Logs• Leverage existing VPC security groups and network access control lists

Use Case – Interconnecting Geographically Dispersed On-Premise and VPC resources



- Customer with multiple VPCs
- Build applications that span a large number of VPCs
- Share network services (DNS, Active Directory, FW, IDS)
- Reduce management overhead

Use Case – Edge Consolidation



- Share a common VPN or Direct Connect Gateway (DXGW) across all VPCs
- Reduce time to connect on premise resources to multiple VPCs
- No additional customer network changes required when adding a VPC to AWS Transit Gateway

Use Case – Digital security and threat intelligence



- Shared VPC hosts security tools
- Firewall as a service
- Web application Firewall (WAF), Data Loss Prevention (DLP), Intrusion Detection / Protection (IDS/IPS),
- Scales out over native AWS Service

How Does It Work

AWS Transit Gateway Reference Architecture

Administrative accounts (logging, AWS Organizations, billing, landing zone)

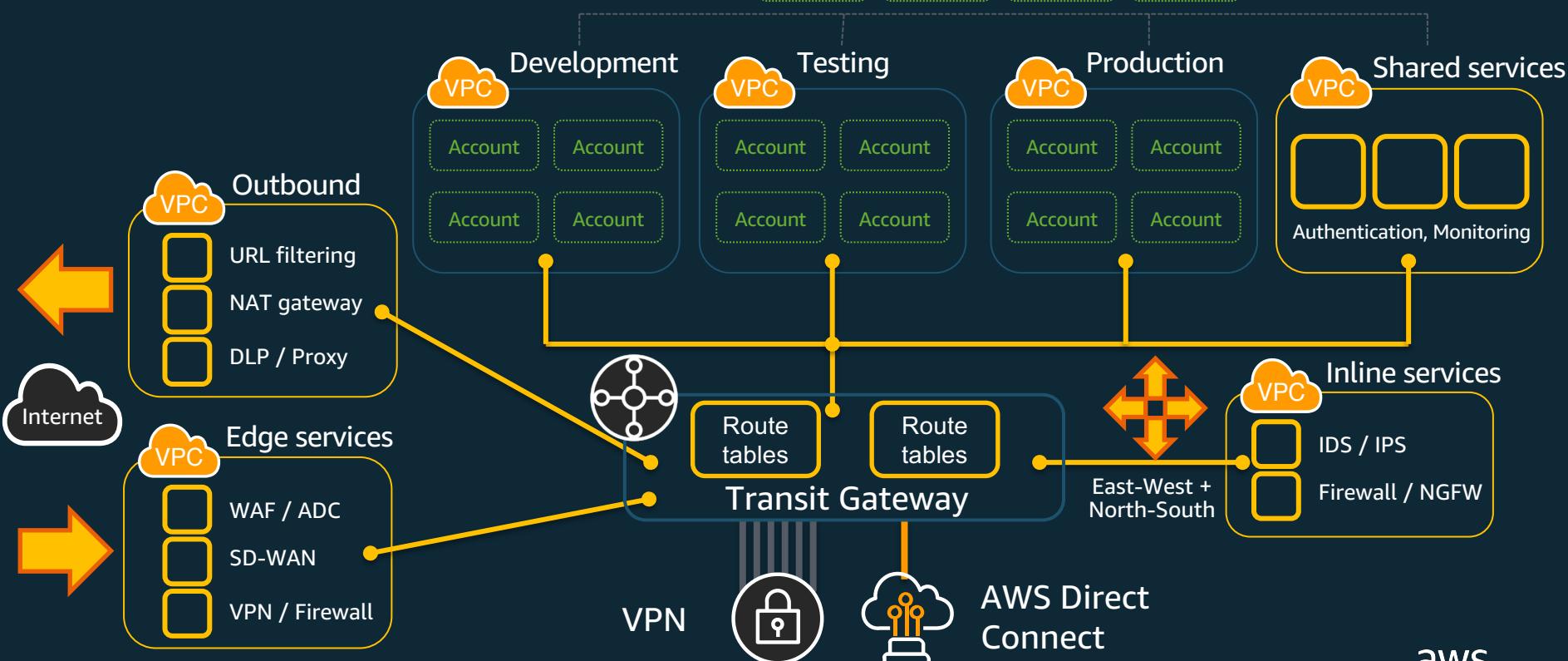
Account

Account

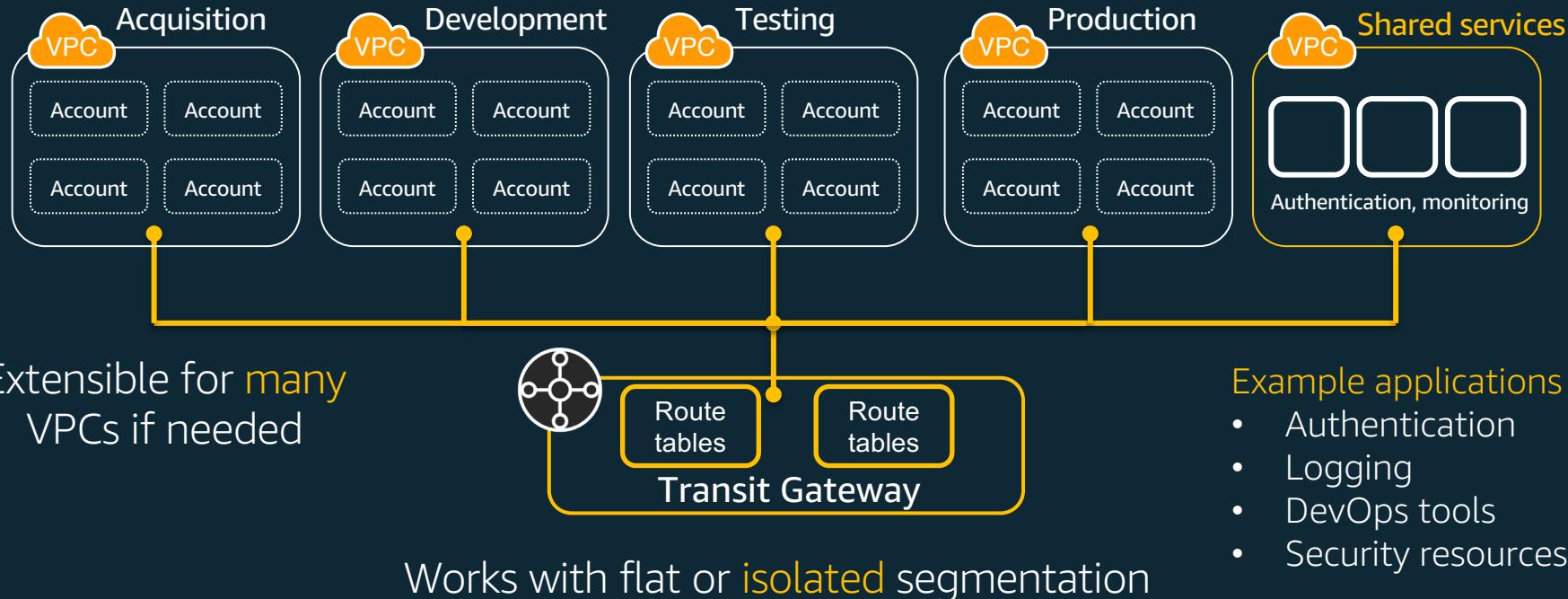
Account

Account

IAM, Cross-account roles

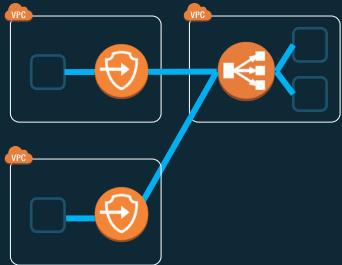


Shared Services with AWS Transit Gateway



Using AWS Transit Gateway and PrivateLink

AWS PrivateLink



- One-to-many connectivity
- Highly scalable
- Supports overlapping CIDRs
- Uses Elastic Load Balancing
- Load balancing and hourly endpoint costs

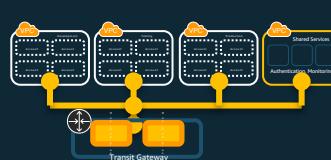
Scope: Application shared services

Trust model: No mutual trust

Dependencies: Load balancing and application architecture

Scale: Thousands of spoke VPCs

AWS Transit Gateway



- Many-to-Many or one-to-many with route tables
- Highly scalable
- Hourly per AZ endpoint costs

Scope: Network shared services to many VPCs

Trust model: Per VPC trust, centralized control

Dependencies: Centralized control of the Transit Gateway

Scale: Thousands of spoke VPCs

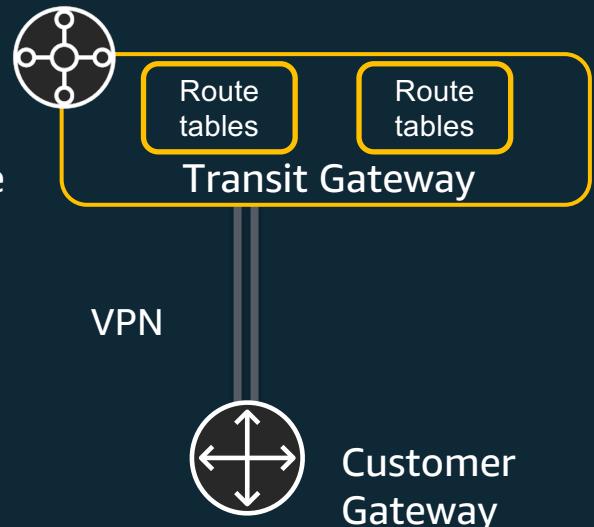
VPN With Transit Gateway

Consolidate VPN at the Transit Gateway (TGW)

- VPN acts similar to the Virtual Private Gateway (VGW)
 - Bandwidth, configuration, APIs, cost, and experience
 - VPN is attached to a TGW instead of a VGW
 - Same 1.25 gbps bandwidth per tunnel applies

Encryption to the edge of many VPCs

- Traffic is encrypted until it's inside the VPC
- Does not natively encrypt traffic between VPCs
 - Inter-region VPC peering does



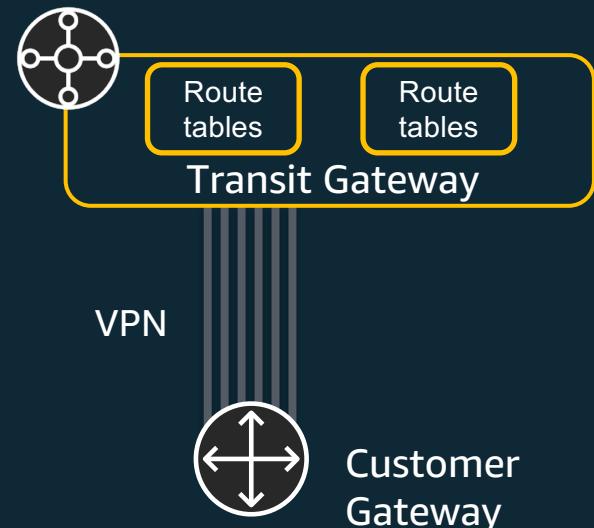
VPN with Transit Gateway: Add more bandwidth

Support for spreading traffic across many tunnels

- Equal Cost Multi-Path (ECMP) support with BGP multi-path
- Tested up to 50 Gbps of traffic
- Split traffic into smaller flows, multi-part uploads, etc.

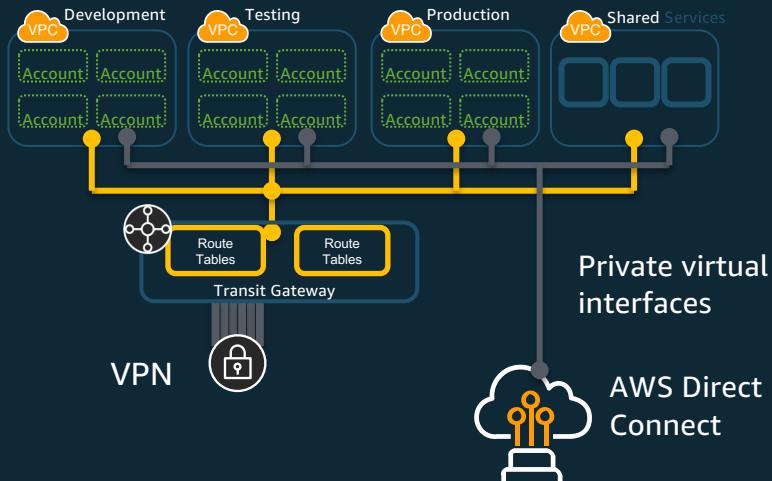
Check your on-premises configuration

- Multi-path BGP
- ECMP support, amount of equal paths, reverse-path forwarding/spoofing checks
- Only supported with BGP, not static routing



AWS Direct Connect and Transit Gateway

Use Direct Connect in parallel



Use VPN over a Direct Connect public virtual interface (VIF)

