PATTERNS, CHALLENGES AND BEST PRACTICES FOR AWS ACCOUNTS IN YOUR CLOUD STRATEGY

AN OVERVIEW OF THE ACCOUNT MANAGEMENT APPROACH
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Introduction</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>The AWS User Account</td>
<td>3</td>
</tr>
<tr>
<td>AWS Accounts as Boundaries</td>
<td>4</td>
</tr>
<tr>
<td>Security &amp; Access Control</td>
<td>4</td>
</tr>
<tr>
<td>IAM</td>
<td>4</td>
</tr>
<tr>
<td>Security Groups</td>
<td>4</td>
</tr>
<tr>
<td>Communication</td>
<td>4</td>
</tr>
<tr>
<td>AWS Service Limits</td>
<td>5</td>
</tr>
<tr>
<td>Cost &amp; Billing</td>
<td>5</td>
</tr>
<tr>
<td>Resource Sharing</td>
<td>6</td>
</tr>
<tr>
<td>Account Linking and Consolidated Billing</td>
<td>6</td>
</tr>
</tbody>
</table>

## Best Practice Account Usage Patterns

| Single Account Model (Fully Centralized) | 7 |
| Autonomous Model                       | 7 |

## Identity and Access Management in multiple accounts

| Discreet IAM configurations per account | 10 |
| Cross-Account IAM Roles                | 10 |
| Identity Providers and Federation      | 11 |

## Simplifying your AWS Account strategy

| AWS IAM Configuration and Management   | 11 |
| Centralized Billing                   | 12 |

## Summary

| 12 |
INTRODUCTION

Given the flexibility of AWS, there are multiple possible methods for how organizations manage user accounts. Each method has implications, both good and bad, that should be considered prior to granting user accounts at scale. Security and access control, cost and billing, resource sharing, and many other considerations can make a particular user management method either a great or poor fit for your organization. In this paper, James Cowe, Director of Solution Architecture for Fanatical Support for AWS, will discuss methods of user account management, and the potential benefits and detriments for each approach.

THE AWS USER ACCOUNT

Let’s begin by defining the purpose and structure of the AWS account itself. The AWS account is a formally defined entity that describes the relationship between AWS and the customer. An AWS account has complete, unrestricted access to any resources created within the account by means of the root credentials (which are created during account setup under the owners email/login). This level of access is in contrast to “users,” which are created within an account using AWS Identity and Access Management (IAM). Users can have their access permissions controlled at a granular level.

Two unique and unchangeable identifiers are assigned to each account during creation:

AWS Account ID: A unique 12-digit number is used as the reference point for each AWS account as well as in the construction of Amazon Resource Names (ARN’s). The account ID is used to distinguish between resources, such as IAM users or SQS queues, in different AWS accounts.

Canonical User ID: This ID is a long string primarily used to enable cross-account access on S3 buckets. It uses another AWS account’s canonical user ID when writing an S3 bucket policy.

More details on both can be found here: http://docs.aws.amazon.com/general/latest/gr/acct-identifiers.html.

AWS ACCOUNTS AS BOUNDARIES

Now we’ve defined what an AWS account is, let’s now look into the properties or behaviors associated with an AWS account. These behaviors or properties are generally used to form a boundary between the applications or services running in the AWS account. They can generally be classified into four discreet types:

SECURITY & ACCESS CONTROL

IAM
AWS Identity and Access Management (IAM) exists on an account-by-account basis, and controls both the access and permissions within your AWS account. It can be used to create users, roles and permission sets that control any and all access to resources within your account. By default, when you create an IAM configuration within your account, it prevents any
accidental access being granted to users outside of your AWS account. Although there are specific ways you can enable cross-account access, each requires configuration work on both the source and target AWS accounts. In this way, your AWS account can be considered a natural security boundary from an IAM perspective.

Many organizations consider an AWS account to be a natural organizational divider. A typical usage pattern for organizations with multiple divisions would be to issue an AWS account to each separate division or business unit. That way, each has full control of their AWS resources but is blocked from impacting another part of the organization by this boundary.

**Security Groups**

EC2 Security Groups are used to control ingress and egress from EC2 instances, and as such are the typical method for securing access to individual resources within EC2. Security Groups allow you to control access by either instance ID, CIDR range or by referencing existing security groups. While it is possible to reference any CIDR range whether it exists within an AWS account or not, referencing Instance IDs and other Security Groups works only within your AWS account. This helps avoid accidental configurations that would grant access across accounts or between separate workloads. It also allows for segregation of responsibilities/access across different parts of an organization or team.

**COMMUNICATION**

AWS Virtual Private Clouds (VPCs) are logically isolated portions or allocations of network within the AWS EC2 environment, allowing for the creation of discreet network constructs for your workload or organization. By utilizing VPCs, users gain complete control over all aspects of the network architecture (such as subnets, routing tables, internet connectivity, etc.), and act as another security boundary within your AWS account.

As with all AWS products or services, access to configuration of the VPC is controlled by IAM and by default it is impossible for instances in different VPCs to communicate privately with each other (although it should be noted that public IPs will be accessible between VPCs unless explicitly blocked). VPCs support a feature known as VPC Peering, which is a logical method of connecting and non-transitively routing traffic between the private IP space of two separate VPC. This can allow private traffic to traverse between subnets (public or private) within two or more VPCs.

To configure a VPC Peering connection, the requestor VPC must request a peering connection with the target or peer VPC and the administrator on the target or peer VPC must then accept the request. Once a peering connection has been established, routing tables can be modified to allow the flow of traffic between the connected VPCs. Without specific IAM configuration in place, it is likely that the person initiating the VPC Peering request would have permissions to accept the request from the target VPC, thus allowing inter-VPC access.

VPC Peering supports cross-account peering, however by their very nature the administrative control between different AWS accounts is typically segregated. This IAM boundary again acts as a limit to the kind of change and communication flow that can be established between your workloads or business units.
AWS SERVICE LIMITS

There are a number of AWS Service limits that exist by default. These limits are typically region specific, and control the amount of different resource types that can be created in your AWS account.

Examples of These Limits Include:

- Number of VPCs per region
- Number of EC2 Auto Scaling Groups per region
- Number of EC2 On-Demand instances running per region

Full details of AWS Service Limits can be found here: http://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html.

While most of these limits are “soft” and can be increased or decreased by request, such service limit changes are not instantaneous. When requesting a service limit change, it will most likely take a minimum of 24 hours or more for a limit change to take effect — it depends on the particular AWS Service and size of the change.

Using separate AWS accounts provides multiple sets of these AWS Service limits. If you are running multiple business units that manage and govern their own infrastructure, it is possible to put a logical boundary in place using multiple accounts to prevent service limits being encountered due to behavior of another business unit within the AWS account.

COST & BILLING

The AWS account acts as a billing entity, making it a boundary for the allocation and payment of costs associated with the resources within an account. Each AWS account can have its own method for payment, produce detailed billing statements, and may be owned by separate individuals, groups or governance entities within an organization. Tagging makes it possible to identify individual resources within an AWS account, allowing for detailed cost allocation models. Tags can be associated with the billed resources by using a detailed billing report. However, some resources within AWS cannot be tagged and therefore will not show up in a detailed billing file. If you are sharing an AWS account for multiple entities — each of which require costs to be apportioned — then the simplest way to achieve this is via separate AWS accounts. Resources that cannot be tagged include:

- Ingress/Egress bandwidth from EC2
- Ingress/Egress bandwidth from S3 (although you can use bucket naming strategies to get some intelligence)
- VPC Endpoints
- VPC Flowlogs
- Elastic IPs

Although an exhaustive list of taggable and untaggable resources is published, more information can be found here: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html.

Reserved Instances (RIs) serve a major function in cost optimizing your AWS infrastructure spend. RIs allow a customer to access significant discounts over on-demand EC2 instances by committing to an instance type, region/availability zone, and period of utilization up front. When purchasing an RI it is constrained to an instance type, a region/availability zone, and an AWS account. While it is possible to request changes to the instance type or region/availability zone, you cannot request that an RI be moved between
accounts. It is possible to sell an unwanted or unused RI on the Reserved Instance Marketplace. However, this is not the same as “returning” or moving an unwanted RI between accounts.

**RESOURCE SHARING**

While we have discussed boundaries inherent in AWS accounts that prevent cross-account sharing or interaction, there are some key AWS resources that have been designed to be shared across accounts. Features such as VPC Peering have been designed to explicitly allow cross-account interaction, with appropriate security controls in place.

Some AWS resources that are sharable between accounts allow for the transfer of useful data between accounts. The key mechanisms are:

- EC2 Amazon Machine Images (AMIs)
- AWS EBS volume snapshots
- Amazon RDS backup/snapshots
- Amazon S3 Buckets

These provide mechanisms to share data from key services and effectively provide the ability to migrate workload data between AWS accounts. Without the ability to share these resources between accounts, any data migration would need to be undertaken manually or using third-party tooling.

**ACCOUNT LINKING AND CONSOLIDATED BILLING**

AWS provides a method of “linking” multiple AWS accounts in a parent-child model known as Linked Accounts. This enables a customer to consolidate billing from multiple child AWS accounts into a single master AWS account. Linked Accounts have the advantage of giving the user a unified view of resource charges across a group of AWS accounts in a single billing report, as well as individual reports per child account. This allows more visibility into an organization’s spend across AWS, while still leveraging the AWS account as a boundary for the reasons described above.

Our recommended best practice when setting up a linked account model is to create an empty master AWS account used only for consolidated billing. Creation of AWS resources within the master account adds unnecessary complexity and attack surface to an account holding payment and consolidated billing data/configuration.

Consolidated billing also proves beneficial when purchasing RIs for cost optimization purposes. RIs purchased at the master account level can be associated at instance start-up time across any of the child accounts linked to the master account.
BEST PRACTICE ACCOUNT USAGE PATTERNS

SINGLE ACCOUNT MODEL (FULLY CENTRALIZED)

In this pattern, a single AWS account is created and used to house all AWS resources used by an organization. This model is often preferred by smaller organizations who plan to have very few or simplistic workloads inside their AWS account, or an organization with a centrally managed business and IT function. This model means the customer will receive a single billing report against a single payment method. They do not have to manage the complexity of multiple accounts for communications or access management.

Benefits:
- Low complexity
- Resource sharing is retained within an account
- No cross-account configuration required
- Single billing report and payment method
- Simple centralized governance across AWS resources

Drawbacks:
- Account limits will need to be continuously increased as account usage grows
- A configuration or resource change issue in a development environment will impact any other workloads (including production)
- Potential large impact for erroneous changes to items such as IAM policies or Security Group changes
- Granularity of cost allocation relies on tagging alone, meaning untaggable resources such as bandwidth cannot be correctly attributed
- No method of isolating access control outside of IAM policies (i.e., only users with genuine production access requirements have an IAM account)
- No segregation between production and non-production workloads

AUTONOMOUS MODEL

The autonomous pattern uses multiple unlinked AWS accounts to provide the benefits of the single account model, while allowing autonomous business or IT functions (geographic, departmental, or project/workload based) to operate independently. This model allows for organizations that have a decentralized model – each department or division can govern themselves, manage and pay for their usage of AWS resources, and isolate their environments from other parts of the organization.
Benefits:

- Low complexity for each account
- Resource sharing is retained within an account
- No cross-account configuration required
- Separate billing report and payment method for each AWS account/business unit
- Simple decentralized governance across AWS resources to match the organizations needs
- Lower potential impact for erroneous changes to items such as IAM policies or Security Group changes minimized to a single organizational unit

Drawbacks:

- Allocating a single account to a business unit or function reduces the potential impact of erroneous changes outside the business unit or function, but still faces all other drawbacks of a single account model.
- Complexity introduced in creating multiple AWS accounts within a single organization but having no centralized governance
- Duplication of effort across an organization to manage billing and payment

SINGLE MASTER HIERARCHICAL MODEL

A single master account, in a traditional one-to-many linked account model, allows an organization to centralize governance and control of their AWS spend while still retaining some autonomy for their business units. This model is typically used for larger organizations that have many workloads across functional groups/projects and centralized finance. This model can also utilize separate subordinate AWS accounts to isolate application environments (production vs. non-production) within an organization.

Benefits of this model are:

- Centralized governance model
- A single payment mechanism, with a single unified billing report across all accounts
- The ability to attribute costs, whether taggable or not, to an individual business unit or environment based on AWS account boundaries
- Minimized potential impact for erroneous changes to items such as IAM policies or Security Group changes (limited to individual accounts)
- The ability to delegate IT or administrative control to appropriate business units, project teams, or functions
- RI purchases can be made at the master account level, allowing them to be leveraged across any of the child accounts

Drawbacks of this model are:

- Management of IAM users, roles and policies becomes complex across multiple AWS accounts — each account requires duplicate configuration
- Setup of each new subordinate AWS account requires manual linking to the master AWS account
- Consolidated billing with many child accounts can create large detailed billing files, making it progressively more difficult to analyze the data effectively
MULTI-MASTER HIERARCHICAL MODEL

A very large or enterprise organization may have many governing bodies or large business units, requiring independence and isolation from each other for any number of reasons, including political, legislative, or financial separation. In order to support such a requirement, a multi-master account structure is usually most useful. Multi-master account structures involve the creation of multiple master payer accounts, beneath which subordinate child accounts are created. This achieves the benefits of a single master model but allows allocation of costs to multiple payers/payments on a master-by-master account basis. While you will naturally lose some of the benefits of cost analysis and RI purchase optimization, it provides the ability to maintain separate governance bodies for each hierarchy of accounts.

Benefits:
- Centralized governance model for each business unit or division, while maintaining desired separation
- Separate payment mechanisms available for each master account, dependent on the business unit or divisions requirements
- Minimized potential impact for erroneous changes to items such as IAM policies or Security Group changes (limited to individual accounts)
- The ability to delegate IT or administrative control to appropriate project teams or functions within each business unit
- RI purchases can be made at each master account level, allowing RI optimization within each business unit and avoiding limitation of centralized governance

Drawbacks:
- Management of IAM users, roles and policies becomes exponentially more complex across multiple AWS master and sub-accounts
- Consolidated billing across multiple master AWS accounts with many child accounts creates multiple large detailed billing files, increasing the complexity of cost analysis even further than a single master account model
- Multi-master pattern will naturally lose some of the cost efficiencies by isolating RI purchases to each master account
- No way to nest multiple-master accounts into a n+ hierarchical structure, which means any centralized accounting or governance function requires manual effort or automated tooling

The use of a single account for all your AWS resources does reduce complexity, but at the cost of increased risk as you deploy more of your business applications into the account. For simple or singular use-cases of AWS, a single account model makes sense. But if you plan to grow your AWS footprint it can quickly become complex to manage.

Ultimately, there is no single “correct” account pattern to use for all scenarios — each must be evaluated based on the use case of the organization. However, certain aspects of the linked account models are beneficial in the majority of scenarios. Leveraging the natural resource and security boundaries of an AWS account make it best practice to isolate environments (prod vs. dev), workloads and even business units with a multi-account strategy. Whether you choose to have a single or multi-master structure will be decided by the governance policy within your organization, but the efficiencies and behaviors of linked accounts will outweigh the investment in setting up such a structure.
When choosing any account structure, it is important to plan ahead for growth, network communications (e.g., ensuring that accounts do not use overlapping CIDR ranges in the event you need to peer VPCs together in the future), and IAM configuration. Identity and access management planning is one of the more critical steps in planning a multi-account AWS account structure.

IDENTITY AND ACCESS MANAGEMENT IN MULTIPLE ACCOUNTS

AWS IAM is the web service that provides the ability to securely control access to AWS Resources at a fine-grained level. IAM controls the “who” of access controls with authentication for users, and the “how” with authorization for access permissions to AWS resources within your AWS accounts.


When operating multiple AWS accounts, there are several methods of implementing these controls across your accounts:

DISCREET IAM CONFIGURATIONS PER ACCOUNT

IAM operates by default in the isolation of a single AWS account, providing authentication and authorization within the boundaries of the AWS account where it is configured. This method is relatively straightforward to configure, and it limits the scope of resources under control to those resources directly addressable within the AWS account.

The creation of IAM policies, roles and users can give fine-grained control over all AWS resources. However, this pattern for access control does not scale well beyond two or three AWS accounts. There is significant overhead in the configuration and ongoing management of discreet IAM policies across multiple AWS accounts, exposing a customer to ongoing risk of configuration drift and exploitable misconfigurations. At the simplest level, the management of users in the event of staffing changes within your organization is multiplied by the number of AWS accounts in use.

IDENTITY PROVIDERS AND FEDERATION

An AWS Identity Provider (IdP) leverages either SAML 2.0-based federation or Web Identity Federation to allow the use of existing user identity stores based outside of AWS IAM. This effectively transitions the overhead of managing IAM users away from an individual AWS account (or even cross-account) basis, and leverages an existing single source of truth in a corporate identity store such as Microsoft Active Directory (AD).

While there is some upfront configuration work to be done in building the trust between an IdP and the directory source, it can greatly simplify the ongoing management of user management inside AWS accounts. It is important to note that IdP federation does not remove the complexity of managing and assigning IAM roles and policies across multiple AWS, and the addition of any more AWS accounts require the same configuration be duplicated at creation.

More detailed information can be found here: http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers.html.

CROSS-ACCOUNT IAM ROLES

An IAM role is effectively a collection of permissions or rights that grant access to AWS resources and define the actions that can be taken with those resources. Permissions or rights are granted to a role rather than an individual IAM user or group.

When configuring a cross-account IAM role, two IAM policies are assigned to it: a permissions policy and a trust policy. The permissions policy will define the AWS resources and their applicable accessible actions. The trust policy defines the entity that can assume the IAM role and take actions with its permissions. Both the trusting and trusted entities can be in a single AWS account, two AWS accounts under a single organization’s control, or two AWS accounts controlled by separate organizations. In this manner, it is possible to grant access to a third-party entity; Rackspace itself uses a delegated IAM cross-account role to deliver Fanatical Support® for AWS services to customers.


SIMPLIFYING YOUR AWS ACCOUNT STRATEGY

Even when deploying into a single AWS account pattern, some elements of managing your AWS account remain operationally challenging. Setting up and managing IAM users, groups and roles is time consuming and can be fraught with opportunity for misconfiguration and unexpected behaviors. It is possible to reduce some of this overhead by leveraging cross-account IAM roles or IdP federation, but configuring these methods bring their own complexities.
Fanatical Support for AWS provides both human experts to assist in the design, deployment and management of your AWS resources as well as the operational tooling and automation to simplify operation of your AWS infrastructure. Built with best cloud practices, Rackspace has developed tooling to automate the setup and management of multiple AWS accounts under a single cloud account, removing the need for configuration and lessening the burden of ongoing management of multiple AWS accounts.

**AWS IAM Configuration and Management**

Our tooling provides cross-account IAM role access by leveraging users created in the Fanatical Support for AWS control panel for authorization and multiple account identity. As a Fanatical Support for AWS customer, you can create as many child AWS accounts as needed under a single cloud account. This cross-account capability is automatically configured according to best practices from both Rackspace and AWS. The creation (and management) of users is controlled from a single interface with pre-created IAM roles for common access requirements (or custom IAM roles can be created in the AWS accounts by working with our certified AWS architects) available for association with a user into each AWS account. Cross-account identity management is now automated and simplified, allowing a Fanatical Support for AWS customer to leverage the multi-account model without the administrative overhead of IAM configuration.

**Centralized Billing**

One of the benefits of linked accounts is consolidated billing — by default an AWS account created with Fanatical Support for AWS is linked to the Rackspace master account. It is impossible to nest AWS accounts more than one deep, however, consolidated billing is available for each customer via Rackspace consolidated billing. Each AWS account created under your Rackspace account has its monthly AWS resource usage rolled up into a single invoice coming from Rackspace, giving customers the benefit of a single bill aggregating all AWS account usage. No initial configuration for this is required from the user. Detailed billing files are also available, and have pre-configured AWS tagging pass-through with tags designed to allow charge-back models and AWS resource governance.

**Summary**

Whichever usage patterns you choose for your AWS environment, there are benefits and challenges to adopting each model. Some of these operational challenges can be addressed with careful planning or additional tooling and automation. Rackspace has invested in building the tooling and automation to enable you to use the AWS account strategy that best fits your organization's needs. By leveraging Fanatical Support for AWS, you can simplify the provisioning of your AWS accounts. In addition, our AWS certified Solutions Architects will assist in building and managing the most efficient and cost optimal AWS infrastructure for your organization.
ABOUT RACKSPACE

Rackspace (NYSE: RAX), the #1 managed cloud company, helps businesses tap the power of cloud computing without the challenge and expense of managing complex IT infrastructure and application platforms on their own. Rackspace engineers deliver specialized expertise on top of leading technologies developed by OpenStack®, Microsoft®, VMware® and others, through a results-obsessed service known as Fanatical Support®.

Learn more at www.rackspace.com or call us at 1-800-961-2888.

© 2016 Rackspace US, Inc.

This whitepaper is provided “AS IS” and is a general introduction to the service described. You should not rely solely on this whitepaper to decide whether to purchase the service. Features, benefits and/or pricing presented depend on system configuration and are subject to change without notice. Rackspace disclaims any representation, express or implied warranties, including any implied warranty of merchantability, fitness for a particular purpose, and non-infringement, or other legal commitment regarding its services except for those expressly stated in a Rackspace services agreement. This document is a general guide and is not legal advice, or an instruction manual. Your implementation of the measures described may not result in your compliance with law or other standards.

This document may include examples of solutions that include non-Rackspace products or services. Except as expressly stated in its services agreements, Rackspace does not support, and disclaims all legal responsibility for, third party products and services. Unless otherwise agreed in a Rackspace service agreement, you must work directly with third parties to obtain their products and services and related support under separate legal terms between you and the third party.

Rackspace®, Fanatical Support® and other Rackspace marks are service marks or registered services of Rackspace US, Inc. and are registered in the United States and other countries. Other Rackspace or third party trademarks, service marks, images, products and brands remain the sole property of their respective holders and do not imply endorsement or sponsorship.

Amazon Web Services sold by Rackspace, an authorized reseller of Amazon Web Services. Amazon Web Services, the Amazon Web Services logo, AWS, and the Powered by Amazon Web Services logo are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.