Organizing Your AWS Environment Using Multiple Accounts

AWS Whitepaper

Organizing Your AWS Environment Using Multiple Accounts: AWS Whitepaper

Table of Contents

Organizing Your AWS Environment Using Multiple Accounts

Publication date: **July 19, 2021** (*Document history* (p. 86))

Abstract

Businesses that are starting to adopt AWS, expanding their footprint on AWS, or planning to enhance an established AWS environment, can benefit from considering the latest guidance for organizing their AWS environments.

Using multiple AWS accounts to help isolate and manage your business applications and data can help you optimize across most of the AWS Well-Architected Framework pillars including operational excellence, security, reliability, and cost optimization. This paper provides best practices for organizing your overall AWS environment. The extent to which you use these best practices depends on your stage of the cloud adoption journey and your specific business needs.

Introduction

Businesses that are starting to adopt AWS, expanding their footprint on AWS, or planning to enhance an established AWS environment, can benefit from considering the latest guidance for organizing their AWS environments.

As AWS customers look to establish their AWS Cloud presence, they look for a way to build out their cloud foundations to enable them to deploy production workloads. Many customers want the ability to build and move fast while staying secure.

Customers might have multiple teams with different security and compliance controls that need to be isolated from one other. Some might have different business processes entirely or be part of different business lines that need clarity of costs incurred.

Customers need explicit security boundaries, a mechanism to have direct control and visibility of their limits and any throttling, and a complete billing separation to directly map costs to underlying projects. The isolation designed into an AWS account can help you meet these needs.

Using multiple AWS accounts to help isolate and manage your business applications and data can help you optimize across most of the AWS Well-Architected Framework pillars including operational excellence, security, reliability, and cost optimization.

Topics

AWS accounts

Your cloud resources and data are contained in an AWS account. An account acts as an identity and access management isolation boundary. When you need to share resources and data between two accounts, you must explicitly allow this access.

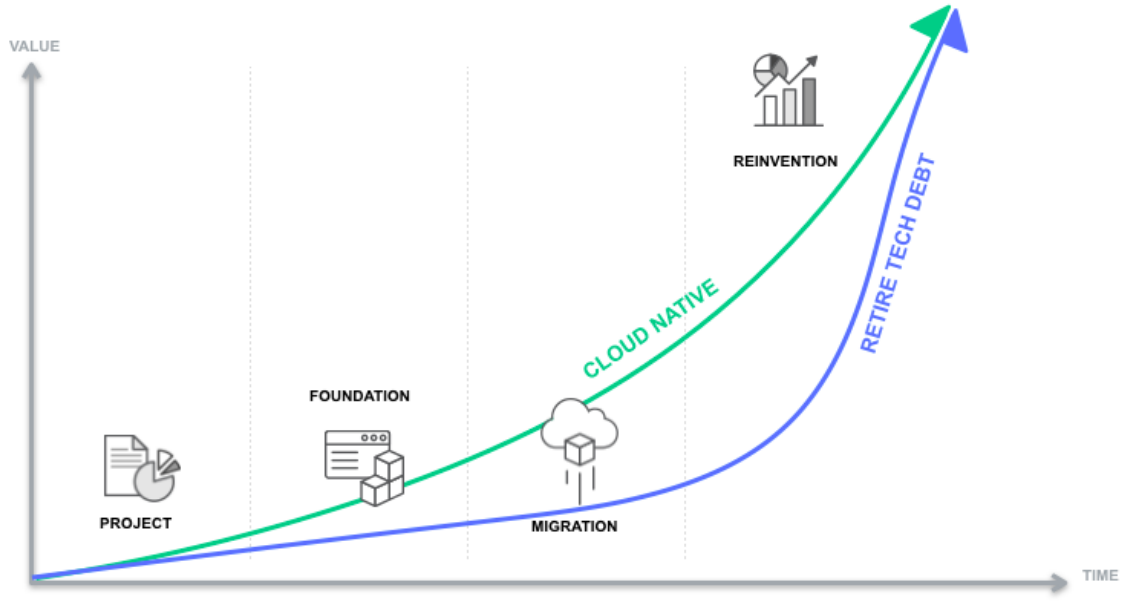
By default, no access is allowed between accounts. For example, if you designate different accounts to contain your production and non-production resources and data, by default, no access is allowed between those environments.

The number of accounts that best meets your needs can range from a few, to hundreds, or even thousands. Management of many accounts requires use of automation to help minimize your operational complexity and ensure efficient alignment with your security, governance, and operational requirements. AWS does not charge per account. Rather, you incur charges based on resources used, regardless of account quantity.

Stages of adoption

Through experience working with thousands of customers, AWS has outlined a common set of stages of cloud adoption. These best practices are designed to help you meet your needs throughout your cloud adoption journey. You can start out with a small AWS environment and progressively grow and evolve it as you gain experience and expand your adoption.

When your organization is new to AWS, you might start by creating one or more personal or team-managed accounts for initial experimentation. This work is usually done informally and before more concerted efforts are made to evaluate the value of AWS. In this experimental and often informal stage, there's usually little investment made in organizing and rationalizing the number and purpose of accounts.



Stages of cloud adoption

In the **project** stage of adoption, you begin to formally plan for your first few production deployments on AWS. It's common to establish an initial cloud foundation that meets your security, governance, and operational requirements.

A workload identifies a set of components that together deliver business value. A workload is usually the level of detail that business and technology leaders communicate about. Some examples of workloads are:

- Marketing websites
- Ecommerce websites
- Mobile app backends
- Analytic platforms

Workloads vary in levels of architectural complexity, from static websites to complex microservices each with potentially different requirements on cost or billing identification.

Rather than using a single account, we recommend that you use at least several accounts to separate your workloads. This approach is designed to make it easier for you to meet your requirements, even in the early project stage of adoption. Based on the success of those first few workloads, you'll likely want to gain further business benefits by expanding your adoption of AWS. This motivation often leads to the **foundation** stage of adoption. In this stage, you invest in evolving your cloud foundational capabilities before greatly expanding adoption.

Evolving your foundation in AWS often includes formalizing and expanding the structure of your accounts to meet the needs of onboarding more teams and workloads. At this stage, it's important for you to design and prepare to manage your AWS environment so that it can scale to meet your needs

without the need for a corresponding linear increase in headcount. As you plan for and perform large-scale migrations and deployment of net new cloud native workloads, you can continue to adjust and enhance your approach to using multiple accounts.

See AWS Cloud Transformation and Maturity Model for more information about the stages of cloud adoption.

Best practices

The best practices described in this paper are designed to help you more easily achieve your security, governance, and operational requirements through multiple accounts. The best practices were assembled based on the experiences of thousands of customers who have progressed through their cloud adoption journeys.

The best practices can help you quickly establish the initial right-sized scope of your AWS environment. The best practices can also help you adjust and expand your AWS environment as you gain experience both with the AWS services and how you work with the AWS Cloud.

Although your needs will likely be similar to the needs of other organizations, each organization also has some unique requirements. Accordingly, these best practices are intended to offer guidance rather than a one-size-fits-all solution to organizing your AWS environment. As a result, the design of your AWS environment may differ from the examples provided in this document. However, we believe that these best practices will help you make informed decisions as you design your environment.

Relation to AWS Well-Architected

AWS Well-Architected helps cloud architects build secure, high-performing, resilient, and efficient infrastructure for their applications and workloads. Based on five pillars—operational excellence, security, reliability, performance efficiency, and cost optimization—AWS Well-Architected provides a consistent approach for customers and partners to evaluate architectures, and implement designs that can scale over time.

The best practices for organizing your AWS environment addressed in this guide augment and support the best practices represented in the Well-Architected pillars:

See Appendix A: Relation to AWS Well-Architected (p. 62) for a detailed list of areas of the Well-Architected framework that are related to how you organize your AWS environment.

Intended audience

These best practices are oriented toward cloud architects and technical leads who are responsible for the overall security and architecture of your AWS environment. Whether you are new to AWS or you have already been using AWS for years, your team will benefit from reviewing these best practices and comparing them to your requirements and current AWS environment.

These best practices are intended to apply to organizations largely independent of their industry, size, expected scale of adopting AWS, and workload portfolio. Depending on your needs, not all of the best practices may apply to your situation.

If you're just starting to experiment and starting to learn about AWS via a single AWS account, you do not need to consider these best practices until you begin planning for your first few production workloads.

Benefits of using multiple AWS accounts

As you adopt AWS, we recommend that you determine how your business, governance, security, and operational requirements can be met in AWS. Use of multiple AWS accounts plays an important role in how you meet those requirements.

The use of multiple accounts enables you to realize the benefits in the following sections.

Topics

Group workloads based on business purpose and ownership

You can group workloads with a common business purpose in distinct accounts. As a result, you can align the ownership and decision making with those accounts and avoid dependencies and conflicts with how workloads in other accounts are secured and managed.

Different business units or product teams may have different processes. Depending on your overall business model, you might choose to isolate distinct business units or subsidiaries in different accounts. Isolation of business units can help them operate with greater decentralized control, but still provides the ability for you to provide overarching guardrails. This approach might also ease divestment of those units over time.

Guardrails are governance rules for security, operations, and compliance that you can define and apply either across your AWS environment or to specific groups of accounts. Guardrails help protect your users from making choices that aren't aligned with your overall requirements.

If you acquire a business that is already operating in AWS, you can move the associated accounts intact into your existing organization. This movement of accounts can be an initial step toward integrating acquired services into your standard account structure.

Apply distinct security controls by environment

Workloads often have distinct security profiles that require separate control policies and mechanisms to support them. For example, it's common to apply different security and operational policies for the non-

production and production environments of a given workload. By using separate accounts for the non-production and production environments, by default, the resources and data that make up a workload environment are separated from other environments and workloads.

Constrain access to sensitive data

When you limit sensitive data stores to an account that is built to manage it, you can more easily constrain the number of people and processes that can access and manage the data store. This approach simplifies the process of achieving least privilege access. Limiting access at the coarse-grained level of an account helps contain exposure to highly sensitive data.

For example, designating a set of accounts to house publicly accessible Amazon S3 buckets would enable you to implement policies for all your other accounts to expressly forbid making S3 buckets publicly available.

Promote innovation and agility

At AWS, we refer to your technologists as builders in that they are all responsible for building value using AWS products and services. Your builders likely represent diverse roles such as application developers, data engineers, data scientists, data analysts, security engineers, and infrastructure engineers.

In the early stages of a workload's lifecycle, you can help promote innovation by providing your builders with separate accounts in support of experimentation, development, and early testing. These environments often provide greater freedom than more tightly controlled production-like test and production environments by enabling broader access to AWS services while using guardrails to help prohibit access to and use of sensitive and internal data.

- **Sandbox accounts** are typically disconnected from your enterprise services and do not provide access to your internal data, but offer the greatest freedom for experimentation.
- **Development accounts** typically provide limited access to your enterprise services and development data, but can more readily support day-to-day experimentation with your enterprise approved AWS services, formal development, and early testing work.

In both cases, we recommend security guardrails and cost budgets so that you limit risks and proactively manage costs.

In support of later stages of the workload lifecycle, you can use distinct test and production accounts for workloads or groups of related workloads. Having an environment for each set of workloads can enable owning teams to move faster by reducing dependencies on other teams and workloads and minimizing the impact of changes.

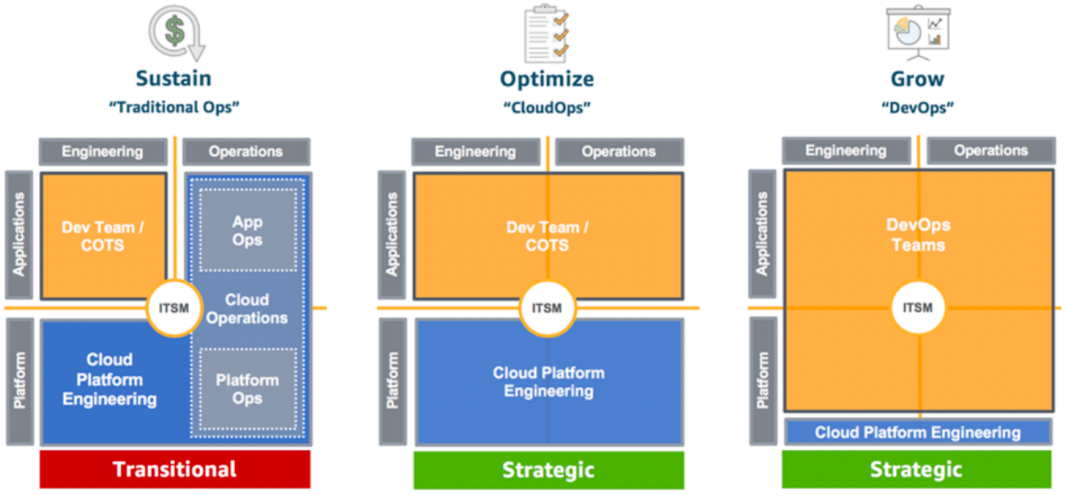
Limit scope of impact from adverse events

An AWS account provides security, access, and billing boundaries for your AWS resources that can help you achieve resource independence and isolation. By design, all resources provisioned within an account are logically isolated from resources provisioned in other accounts, even within your own AWS environment.

This isolation boundary provides you with a way to limit the risks of an application-related issue, misconfiguration, or malicious actions. If an issue occurs within one account, impacts to workloads contained in other accounts can be either reduced or eliminated.

Support multiple IT operating models

Organizations often have multiple IT operating models or ways in which they divide responsibilities among parts of the organization to deliver their application workloads and platform capabilities. The following figure shows three example operating models.



Example operating models

In the *Traditional Ops* model, teams who own custom and commercial off-the-shelf (COTS) applications are responsible for engineering their applications, but not for their production operations. A cloud platform engineering team is responsible for engineering the underlying platform capabilities. A separate cloud operations team is responsible for the operations of both applications and platform.

In the *CloudOps model*, application teams are also responsible for production operations of their applications. In this model, a common cloud platform engineering team is responsible for both engineering and operations of the underlying platform capabilities.

In the *DevOps model*, the application teams take on the additional responsibilities of engineering and operating platform capabilities that are specific to their applications. A cloud platform engineering team is responsible for engineering and operations of shared platform capabilities that are used by multiple applications.

As a practice, IT Service Management (ITSM) is a common element across all of the models. Your overall goals and requirements of ITSM might not change across these models, but the responsible individuals and solutions for meeting those goals and requirements may vary depending on the model.

Given the implications of centralized operations versus more distributed operational responsibilities, you will likely benefit from establishing separate groups of accounts in support of different operating models. Use of separate accounts enables you to apply distinct governance and operational controls that are appropriate for each of your operating models.

To learn more about operating models and their implications on your cloud adoption, see the AWS Well-Architected Operational Excellence Pillar Operating Model.

Manage costs

An account is the default means by which AWS costs are allocated. Because of this fact, using different accounts for different business units and groups of workloads can help you more easily report, control, forecast, and budget your cloud expenditures.

In addition to cost reporting at the account level, AWS has built-in support to consolidate and report costs across your entire set of accounts. When you require fine-grained cost allocation, you can apply cost allocation tags to individual resources in each of your accounts.

For more information about cost optimization, see the AWS Well-Architected Cost Optimization Pillar's Expenditure and Usage Awareness best practices.

Distribute AWS Service Quotas and API request rate limits

AWS Service Quotas, also known as limits, are the maximum number of service resources or operations that apply to an account. For example, the number of Amazon Simple Storage Service (Amazon S3) buckets that you can create for each account.

You can use Service Quotas to help protect you from unexpected excessive provisioning of AWS resources and malicious actions that could dramatically impact your AWS costs.

AWS services can also throttle or limit the rate of requests made to their API operations.

Because Service Quotas and request rate limits are allocated for each account, use of separate accounts for workloads can help distribute the potential impact of the quotas and limits.

To learn more about managing service quotas, see AWS Well-Architected Reliability Pillar Manage Service Quotas and Constraints.

Core concepts

This section covers the following core concepts:

Topics

AWS Organizations

The AWS Organizations service helps you centrally govern your environment as you grow and scale your workloads on AWS. Whether you are a growing startup or a large enterprise, Organizations helps you to centrally provision accounts and resources; secure and audit their environment for compliance; share resources; control access to accounts, regions, and services; as well as optimize costs and simplify billing. Additionally, Organizations supports aggregation of health events, consolidated data on use of access permissions, and centralized management of backups and tagging for multi-account environments.

This section includes best practices for organizing your AWS accounts, including grouping your accounts into organizational units (OUs) so that you can more effectively secure and manage your overall AWS environment.

What is an organization?

An *organization* is an entity that you create to consolidate a collection of accounts so that you can administer them as a single unit. Within each organization, you can organize the accounts in a hierarchical, tree-like structure with a root at the top and organizational units (OUs) nested under the root. Each account can be placed directly in the root, or placed in one of the OUs in the hierarchy.

Each organization consists of:

- A management account
- Zero or more member accounts
- Zero or more organizational units (OUs)
- Zero or more policies

Organizations management account

The organization management account is the account that creates the organization. Management of the organization's resources including OUs and policies occurs within the organization's management account.

Creation of member accounts and associating them with OUs is also managed from within the management account. Access to the management account does not automatically result in permissions to access each member account of the organization. Cross-account AWS Identity and Access Management (IAM) roles must be configured to allow such access.

Organizations member accounts

AWS Organizations member accounts belong to the organization and reside in the overall organization's structure. All billing for member accounts is consolidated to the management account of the organization.

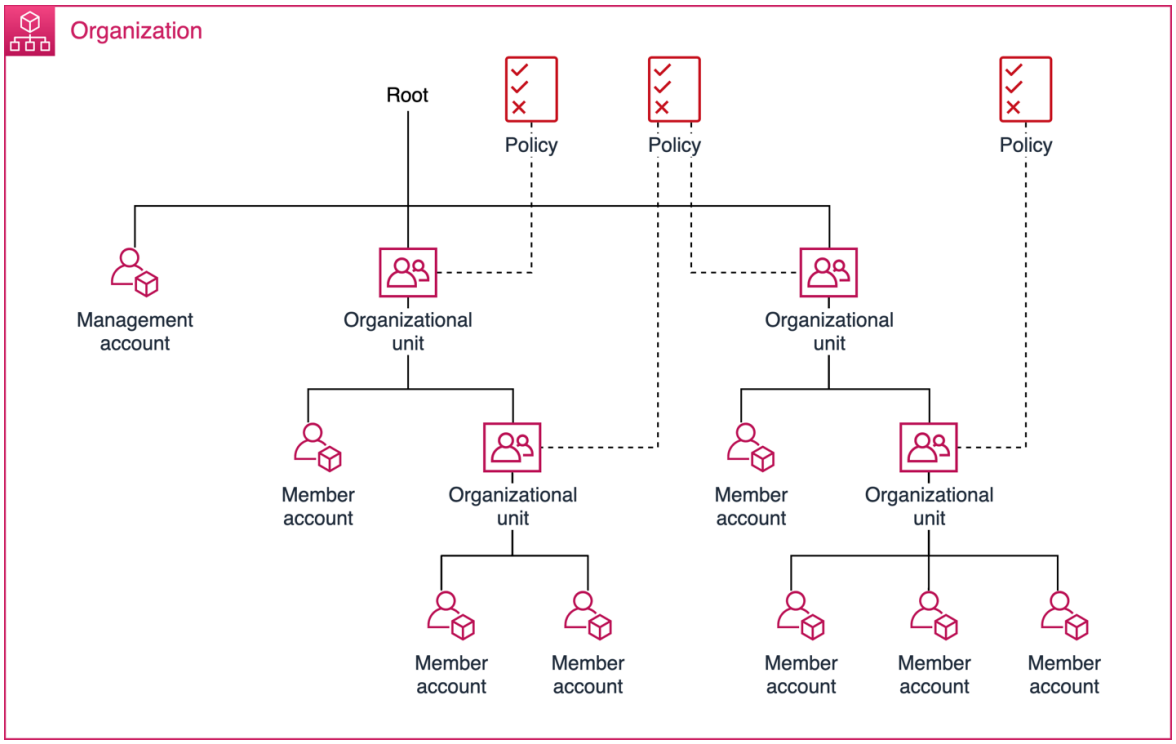
Most of your workloads will reside in member accounts, except for some centrally managed processes that must reside in either the management account or in accounts assigned as designated administrators for specific AWS services.

Organizational units

An organizational unit (OU) provides a means to group accounts within a root. An OU can also contain other OUs. When you attach a policy to one of the nodes in the hierarchy, it flows down and affects all the branches (OUs) and leaves (accounts) beneath it. An OU can have exactly one parent, and each account can be a member of exactly one OU.

OUs are not meant to mirror your own organization's reporting structure. Instead, OUs are intended to group accounts that have common overarching security policies and operational needs. The primary question to ask yourself is: How likely will the group need a set of similar policies?

The following diagram shows a basic organization that consists of seven accounts that are organized into four OUs under the root. The organization also has a few policies that are applied to OUs.



Example of a basic organization

Benefits of using OUs

The following benefits of using OUs helped shape the Recommended OUs (p. 17) and Patterns for organizing your AWS accounts (p. 45).

Group similar accounts based on function

When you have multiple accounts that perform either similar or related functions, you can benefit from grouping these accounts into distinct top-level OUs. Prudent use of top-level OUs can help your teams better understand the overall structure of your AWS accounts.

For example, these best practices recommend a set of top-level OUs (p. 17) to help you organize different sets of related accounts. At a minimum, the top-level OUs are used to distinguish between overall functions of accounts.

Apply common policies

OUs provide a way for you to organize your accounts so that it's easier to apply common overarching policies to accounts that have similar needs. Policies in AWS Organizations enable you to apply additional types of management to the accounts in your organization.

By attaching policies to OUs rather than to individual accounts, you can simplify management of policies across groups of similar accounts. As the number of accounts in your environment grows, simplifying policy management by attaching policies to OUs becomes more important.

AWS Organizations supports use of authorization and management policies. For a complete list of policy types, see Managing AWS Organizations policies.

Authorization policies

AWS Organizations service control policies (SCPs) are a type of organization policy that you can use to manage permissions in your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization.

SCPs are a means of implementing guardrails in your AWS organization. Your use of SCPs can help ensure that your accounts stay within your access control guidelines. For example, you can use SCPs to constrain the set of AWS services and actions allowed on resources.

Although you can apply SCPs to the root of your organization, you typically associate SCPs with underlying OUs. For example, based on the nature of the workloads deployed in accounts within an OU, you may choose to restrict the set of AWS services and AWS Regions that are allowed to be used by accounts in the OU.

You only apply an SCP at the root when you have an overarching security policy that applies across your entire organization and set of OUs. For example, you might apply an SCP at the root of the organization to deny AWS Organizations member accounts from attempting to leave the organization of their own accord.

Management policies

You can also apply tag policies to your parts of your organization to help you monitor and ensure compliance with your cloud resource tagging standards.

You can use Artificial Intelligence (AI) services opt-out policies, which enable you to control data collection for AWS AI services for all of your organization's accounts.

Backup policies help you centrally manage and apply backup plans to the AWS resources across your organization's accounts.

Share common resources

OUs provide a means for you to organize your accounts so that it's easier for you to share centrally managed resources across similar accounts.

AWS services have been introducing support for sharing their resources through AWS Resource Access Manager (AWS RAM) and AWS Organizations. For example, with AWS RAM, you can use OUs as the basis for sharing centrally managed network resources such as Amazon Virtual Private Cloud (Amazon VPC) subnets.

Provision and manage common resources

Sometimes you need to deploy common, centrally managed resource configurations to groups of related accounts. In cases where resource sharing doesn't apply, you can use a variety of AWS services and third-party tools that work with OUs to automatically roll out and update your own custom resources.

For example, you can use OUs as a basis for targeting automation to deploy and update your own sets of IAM roles and customer managed IAM policies that help establish common baseline and/or workload-specific security controls to groups of related accounts.

Multiple organizations

Day-to-day work is often performed within a single organization. However, there are some scenarios in which using more than one organization helps you to meet your requirements. Multiple organizations can help you with the following tasks:

Test changes to your overall AWS environment

You may need to develop code that interacts with APIs and other mechanisms fundamental to the management of your organization. In these cases, to determine whether your changes break something without having to make the changes in your production organization, we recommend that you test your changes in an organization different from the one running your production workloads.

For example, you may need to either modify the automation that creates new accounts to change the configuration baseline of accounts it creates or change the configuration of a workflow management system you're using to modify SCPs.

In these circumstances, we recommend that you establish an additional organization for testing that resembles your more formally managed production organization. You would perform testing of changes to how you manage your organization in your test organization before promoting those changes to be applied to your production organization.

Support acquisitions and divestments

You may acquire an entity that has already established an organization. If you decide to merge the acquired entity's AWS environment with your AWS environment, you can move member accounts from

the acquired organization to your mainstream organization. In this case, you can later decommission the acquired entity's organization.

If you plan to potentially divest a portion of your portfolio, you can manage the workloads and supporting AWS accounts for that portion of your portfolio in a separate organization to support simpler divesture and isolated billing.

Support large AWS environments

If you need more accounts than the maximum number supported by an organization, we recommend that you divide your accounts between multiple organizations. For more details about the maximum number of accounts supported in an organization, see Quotas for AWS Organizations.

Align with your billing requirements

An organization gathers billing information from all member accounts into a single AWS bill. If you have use cases where different sets of accounts require distinct bills and/or payments, then multiple organizations might be required.

Design principles for organizing your AWS accounts

The following design principles helped develop the best practices described in this paper. You can also use these principles to help guide your initial account design and evolve it over time.

These design principles complement the Benefits of using multiple accounts (p. 5) and Benefits of using OUs (p. 10).

Topics

Organize based on security and operational needs

We recommend that you organize accounts using OUs based on function (p. 11), compliance requirements, or a common set of controls rather than mirroring your organization's reporting structure.

Apply security guardrails to OUs rather than accounts

Where feasible, we recommend that you apply security guardrails, for example SCPs, to OUs rather than accounts so that you can more efficiently manage the distribution of guardrails across accounts that have the same or similar requirements.

For more information about managing security guardrails, see Permissions Management in the AWS Well-Architected Security Pillar.

Avoid deep OU hierarchies

Overly complicated structures can be difficult to understand and maintain. Although AWS Organizations supports a depth of five levels of OUs, the recommended structure strives to use OUs only when there is sufficient benefit.

When you consider the addition of new OU levels, you should review the Benefits of using OUs (p. 10) and these principles to decide whether the additional complexity adds sufficient value.

Start small and expand as needed

We recommend that you review the example Patterns for organizing your AWS accounts (p. 45), start with a subset of the Recommended OUs (p. 17), and expand the structure of your AWS accounts when your needs call for the creation of new OUs.

You shouldn't need to invest a lot of time at the beginning of your adoption journey designing what you expect your AWS account structure will look like in several years.

We provide examples of successive degrees of building out an AWS account structure in Patterns for organizing your AWS accounts (p. 45).

Avoid deploying workloads to the organization's management account

Since privileged operations can be performed within an organization's management account and SCPs do not apply to the management account, we recommend that you limit access to an organization's management account. You should also limit the cloud resources and data contained in the management account to only those that must be managed in the management account.

Separate production from non-production workloads

We recommend that you separate production workloads from non-production workloads. For overall recommendations on designing this separation, see Organizing workload-oriented OUs (p. 37).

Assign a single or small set of related workloads to each production account

In support of your production workloads, we recommend that you either assign a single workload to each production account or assign a small set of closely related workloads to each production account.

Consider separating workloads that have different owners into their own production accounts to simplify access management, streamline change approval processes, and limit the scope of impact for misconfigurations.

Use federated access to help simplify managing human access to accounts

We recommend that you use AWS identity federation capabilities via either AWS Single Sign-on (AWS SSO) or IAM integration with a third-party identity provider. These capabilities enable you to use a common identity provider and your existing processes for controlling human user access to your AWS accounts.

By using federated access and a common identity provider, you avoid the need to manage individual IAM users in each account. Instead, your human users can use their existing credentials to access authorized accounts. You also gain the benefit of keeping personally identifiable information (PII) out of IAM.

With federated access, your human users use temporary credentials instead of long term access keys for programmatic access to their AWS environments.

Use of federated access avoids the creation and management of IAM users in your AWS accounts for humans. Instead, use of IAM users can be limited to those exceptional cases such as third-party applications that do not support the use of IAM roles.

For more information about managing identities, see Identity Management in the AWS Well-Architected Security Pillar and Identity federation in AWS.

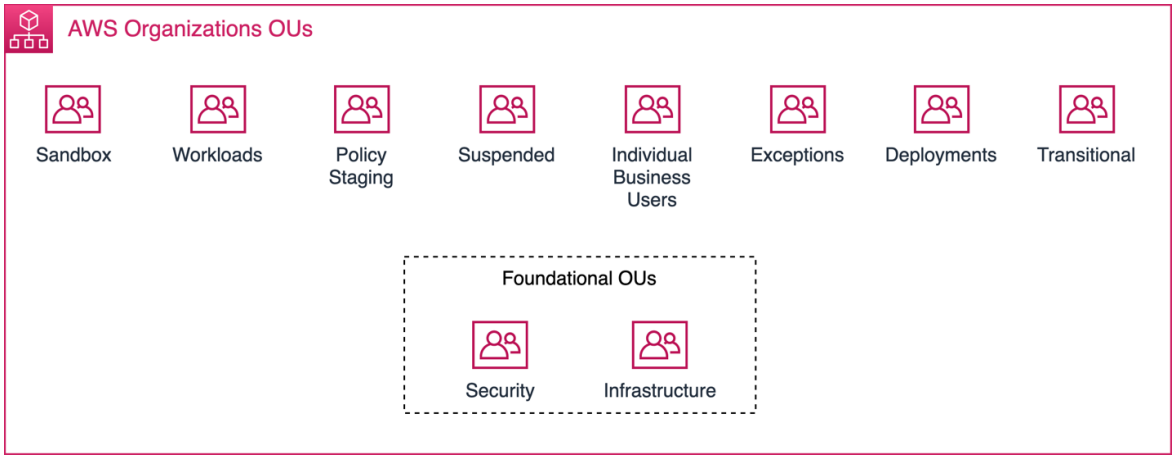
Use automation to support agility and scale

It is important to design and manage your accounts so that you can rapidly respond to business needs without the need for a corresponding linear increase in headcount. When you consider moving beyond managing just a few accounts, you must consider the work to establish processes and automation that will enable you to do so in an efficient manner.

For example, if you implement an account design in which new business initiatives call for the creation of new accounts, then you will benefit from having automation in place so that you can rapidly and reliably provision environments based on your standard configurations. Automation can also help you monitor compliance and apply updates to your baseline configurations over time.

Recommended OUs

This section provides details on the recommended OUs and, in the case of the Security OU (p. 18), a set of recommended AWS accounts.



Recommended OUs

Depending on your requirements, you might not need to establish all the recommended OUs. As you adopt AWS and learn more about your needs, you can expand the overall set of OUs. See the Patterns for organizing your AWS accounts (p. 45) for examples of how you might begin to organize your AWS accounts.

Although the recommended OUs are aligned toward common use cases, you might want to define your own OU structure that best meets your needs. This guidance is intended to meet the needs of most customers. However, it may not be a one-size-fits-all for all use cases.

The recommended OUs consists of:

Topics

We categorize the Security OU and the Infrastructure OU as foundational. The foundational OUs contain accounts, workloads, and other AWS resources that provide common security and infrastructure capabilities to secure and support your overall AWS environment.

Accounts, workloads, and data residing in the foundational OUs are typically owned by your centralized Cloud Platform or Cloud Engineering teams made up of cross-functional representatives from your Security, Infrastructure, and Operations teams.

The majority of your accounts are contained in the other OUs. These OUs are intended to contain your business-related workloads. They also contain tools and services that support the entire lifecycle of your business-related services and data.

Security OU

The Security OU is a foundational OU. Your security organization should own and manage this OU along with any child OUs and associated accounts.

We recommend that you create the following accounts in the Security OU:

- Log archive
- Security tooling
- Security read-only access
- Security break-glass

Depending on your initial requirements, you might not need to establish all of these accounts. See Patterns for organizing your AWS accounts (p. 45) for example sets of OUs and accounts that are commonly used in the early stages of adopting AWS.

Log archive account

The log archive is an account that acts as a consolidation point for log data that is gathered from all the accounts in the organization and primarily used by your security, operations, audit, and compliance teams.

For example, in this account, we recommend that you consolidate AWS API access logs recorded in AWS CloudTrail, logs of changes to AWS resources recorded in AWS Config, and other logs that have security implications. If you use VPC peering between accounts, then you might also benefit from consolidating VPC Flow Logs data in this account.

It's a common practice to integrate this consolidated log data with a security information and event management (SIEM) solution.

If you are using AWS Control Tower to manage your overall AWS environment, then CloudTrail is automatically enabled in each account, and the CloudTrail logs are consolidated in an Amazon S3 bucket in a Log archive account.

Operational log data

Operational log data used by your infrastructure, operations, and workload owning teams often overlaps with the log data used by security, audit, and compliance teams.

We recommend that you consolidate your operational log data into the log archive account. Based on your specific security and governance requirements, you may need to filter operational log data saved to this account. You may also need to specify who and what has access to the operational log data in the log archive account.

Immutable log data

Log data housed in the log archive account is considered immutable in that it is never changed.

Managing access to this account

We strongly recommend that you only house log data in this account and refrain from also including workloads in this account that act on the log data. By doing so, you can greatly limit access to this account.

Workloads and tools that need to consume the consolidated log data are typically housed in your other accounts and are granted cross-account access via IAM roles to access the log data in a read-only, least privileged manner.

Security tooling accounts

We recommend the use of one or more security tooling accounts to contain broadly applicable security-oriented workloads in the form of security services, tools, and supporting data. The number of security tooling accounts you choose to use should be based on taking into consideration the Design principles for organizing your AWS accounts (p. 14).

Common examples of AWS services

Common examples of security capabilities and AWS services that can be centrally accessed and managed via security tooling accounts include:

Detection

AWS Security Hub

We recommend that you enable AWS Security Hub across all of the accounts in your AWS organization. You can specify one of your security tooling accounts as the delegated administrator for Security Hub.

Amazon GuardDuty

We recommend that you enable Amazon GuardDuty across all of the accounts in your AWS organization. You can specify one of your security tooling accounts as the delegated administrator for GuardDuty.

AWS Config

We recommend that you configure an AWS Config aggregator in one of your security tooling accounts so that you have an aggregated view of your AWS resources, your AWS Config rules, and the AWS resources' compliance state.

Identity and Access Management

IAM Access Analyzer

We recommend that you use IAM Access Analyzer configured with the entire AWS organization as the zone of trust so that it's easier for you to quickly look across resource policies and identify resources with public or cross-account access you may not intend. We recommend that you configure this analyzer in one of your security tooling accounts.

Incident Response

Amazon Detective

We recommend that you designate one of your security tooling accounts as the delegated administrator for Amazon Detective, Amazon GuardDuty, and AWS Security Hub. By doing so, you can take advantage of the integration between these services.

Data Protection

Amazon Macie

If you intend to use Amazon Macie across your AWS organization, we recommend that you specify one of your security tooling accounts as the delegated administrator for Macie.

Infrastructure Protection

AWS Firewall Manager

If you intend to use AWS Firewall Manager across your AWS organization, we recommend that you specify one of your security tooling accounts as the delegated administrator for Firewall Manager.

Third-party cloud security monitoring tools

You can also house third-party cloud security monitoring services and tools in your security tooling accounts. For example, these accounts typically contain security information and event management (SIEM) tools and vulnerability scanners.

Automated detection and response workflows

Automated detection and response workflows that act on data collected through these types of services are normally contained in your security tooling accounts.

Incident response (IR) support

Tools to support manual incident response (IR) procedures are typically housed in your security tooling accounts.

See the AWS Security Incident Response Guide for more information.

Security team access

Security team members need access to these services and applications on a day-to-day basis to interact with and potentially configure features of the security services and tools. This access should be minimal and scoped to the needed actions (for example, viewing or acting on the security tool's console or dashboard).

Where possible, we recommend that your security team use infrastructure-as-code (IaC) techniques to automate the underlying configuration of services and tools residing in your security tooling accounts.

Security read-only access account

In support of auditing, exploratory security testing, and investigations, your security team members will need read-only access to accounts in your AWS environment when centrally located logs and other instrumentation is insufficient.

A common approach is to use federated access to provide direct read-only access to your accounts. With direct federated access to your AWS accounts, you do not need to use a security read-only access account.

However, if you prefer to use cross-account roles instead of direct federation, then we recommended that you use this security read-only account. For example, in the early stages of investigating a suspected security incident, your security team members first access this account and use a read-only IAM cross-account role to access other accounts to review and monitor the state of resources.

Typically, this account does not contain persistent workloads. Rather, team members exclusively use this account to interactively access other accounts.

If you plan to use a security read-only account, we recommend that you use federated access to this account for your security team members. Once your security team members access this account via federated access, it's recommended that cross-account IAM roles be used to provide security team members cross-account access to each account of interest.

Security break-glass account

In support of security incidents and exceptional cases in which your standard access mechanisms are not available, you should have a process by which authorized administrators can temporarily gain required access to your accounts.

Depending on your overall process for providing authorized administrators with temporary break-glass access to your accounts, you may not need a security break-glass account. However, if your overall break-glass process involves use of cross-account roles, you may benefit from using a security break-glass account.

Such an account is rarely used, but is available to members of your security and operations teams to enable extensive write access to your accounts when standard access mechanisms are not available. Special authorization is required for security and operations team members to gain access to this account at the onset of an incident, and all account access is logged in detail. Once an incident has been resolved, the temporary access to this account is revoked.

Typically, you create any supporting tools required in this account on-demand using automation and subsequently remove those supporting tools after an incident is resolved.

Example structure

The following example structure represents the recommended separation of production workloads and resources from non-production through *Prod* and *Test* child OUs.

Example account names are shown with qualifiers *-test* and *-prod*. The *-test* qualifier denotes a non-production environment. The *-prod* qualifier denotes stable, production quality environments for a given capability or workload. A *-prod* qualifier does not imply that the capability or workload environment is limited to servicing only other production quality capabilities or workloads.

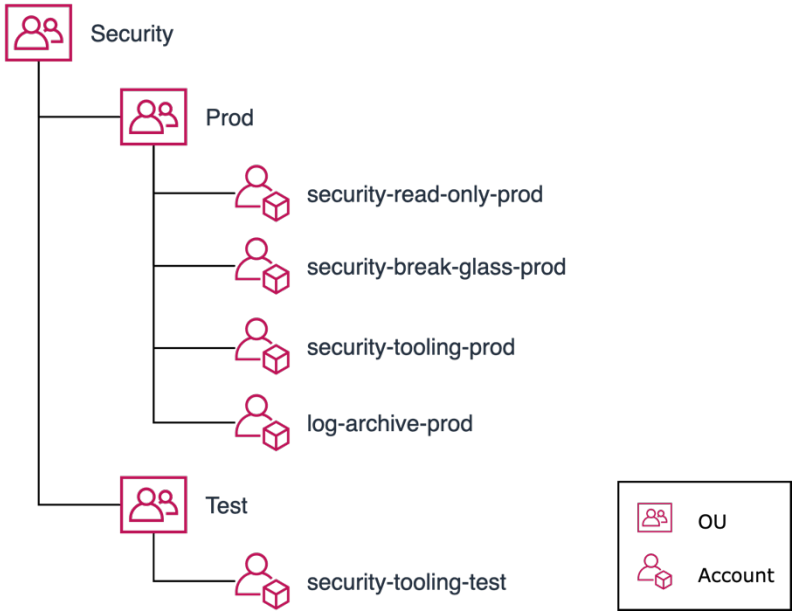
For example, the account represented by the example name *log-archive-prod* is expected to be the consolidation point for all log data across all accounts. It is simply your stable, production quality form of the log archive capability.

Similarly, use of the *-prod* qualifier on the other accounts is not intended to constrain the applicability of those accounts to production environments.

Depending on your own cloud resource naming conventions, you may choose to not apply a qualifier to the names of AWS accounts containing your stable, production quality capabilities and workloads.

The following example includes a *security-tooling-test* account or environment in which you might test and validate new and modified resource configurations before those changes are promoted to your *security-tooling-prod* account.

For general guidance on separating production and non-production workloads, see Organizing workload-oriented OUs (p. 37).



Example structure of Security OU

Infrastructure OU

The Infrastructure OU, another foundational OU, is intended to contain shared infrastructure services. Your infrastructure teams should own and manage this OU, any child OUs, and associated accounts.

Common use cases for this OU include centralized management of many networking resources. For example, AWS Site-to-Site VPN connections, AWS Direct Connect integrations, AWS Transit Gateway configurations, DNS services, Amazon VPC endpoints, and shared VPCs and subnets. More advanced use cases include VPCs and network security stacks used for centralized internet traffic inbound and outbound proxying and filtering.

Beyond shared networking services, you might also decide to manage other shared infrastructure services in this OU. For example, you might manage a shared infrastructure services VPC that includes Amazon Route 53 resolver endpoints for hybrid DNS and directory services.

For guidance on where to contain non-infrastructure shared services, see Workloads OU (p. 25).

Example structure

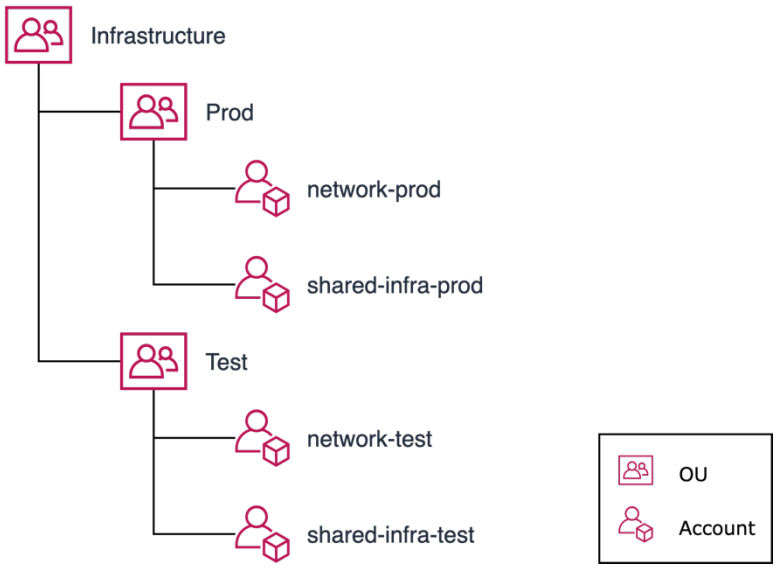
Similar to the example in the Security OU, the following example structure represents the recommended separation of production workloads and resources from non-production through *Prod* and *Test* child OUs.

In this example, the *network-prod* account contains your stable, production quality network capabilities and workloads. Depending on the nature of these capabilities and workloads, they may support both production and non-production environments.

The *network-test* and *shared-infra-test* accounts are meant to represent examples where you have separate environments in which your teams can test and validate changes to common network

capabilities and shared infrastructure services prior to promoting those changes to their respective production quality environments.

For general guidance on separating production and non-production workloads, see Organizing workload-oriented OUs (p. 37).



Example structure of Infrastructure OU

Sandbox OU

The Sandbox OU contains accounts in which your builders are generally free to explore and experiment with AWS services and other tools and services subject to your acceptable use policies. These environments are typically disconnected from your internal networks and internal services.

Sandbox per builder or team with spend limits

A common practice is to provide a sandbox account to either each builder or each small team of builders, along with cloud spend budgets to ensure that their AWS spending aligns within your policies. In more advanced scenarios, you might provide your builders and teams with the option to have multiple sandbox accounts so that they can experiment more freely with configurations that entail use of multiple accounts (for example, experimenting with cross-account IAM roles).

There is a maximum number of accounts in an organization. If you have thousands of builders and expect to allocate a sandbox environment for each builder, you might encounter the maximum quota for accounts. See Quotas for AWS Organizations for more details on the maximum number of accounts in an organization.

In cases where you need more than several thousand sandbox accounts, you might consider either creating one or more separate organizations to contain the sandbox accounts or establishing a process to recycle sandboxes when they are no longer in use.

Temporary resources and environments

Unlike more persistent development environments, it's common to set expectations with your builders that the resources they create in sandbox environments are temporary in nature. As a cost control

measure and to reinforce the temporary nature of sandbox resources, you can put automated procedures in place to periodically purge the resources created in these environments. As a further measure to reduce costs, you can also use automation to stop resources such as Amazon EC2 instances outside of normal business hours.

Wide-ranging access

Wide-ranging access is typically provided in sandbox-oriented accounts including administrative-like access within each account, full access to most AWS services, and possibly outbound and inbound access to the internet. Access to the internet might be required to connect to AWS service APIs, download externally accessible software packages, and integrate with publicly available services.

No access to corporate resources and non-public data

Given the extent of access provided in sandbox environments, businesses typically employ a combination of guardrails and internal usage agreements to limit builders from accessing corporate resources and data from their sandbox accounts. Use of non-public data and intellectual property, including proprietary source code and binaries, is typically not allowed in sandbox environments.

Sandbox and development environments

Due to use of non-public data and the more formal nature of the work being performed in development environments, we recommend that you make a high-level distinction between sandbox environments and development environments. For example, in development environments your teams are performing more formal experiments, day-to-day development, and early testing work that requires access to your intellectual property and to enterprise services, such as source code and artifact management.

For more information about potential distinctions between your sandbox, development, and other environments, see the following appendices:

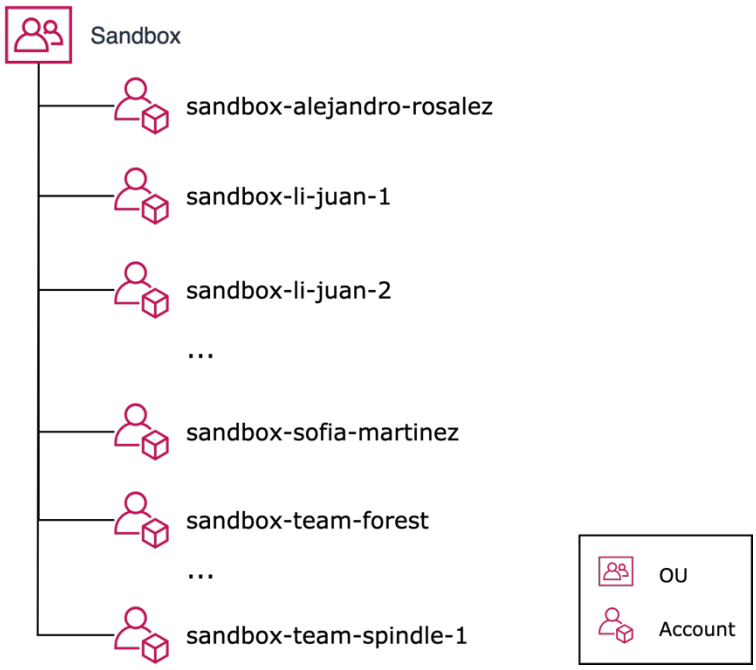
- Appendix B – Worksheet for mapping workload environment purposes to hosting environment types (p. 64)
- Appendix C – Worksheet for identifying attributes of workload hosting environments (p. 71)

Example structures

Sandbox per builder or team

In the following example, sandbox accounts are represented for individual builders and teams. One user has two sandbox accounts so that they can perform experiments that require multiple accounts.

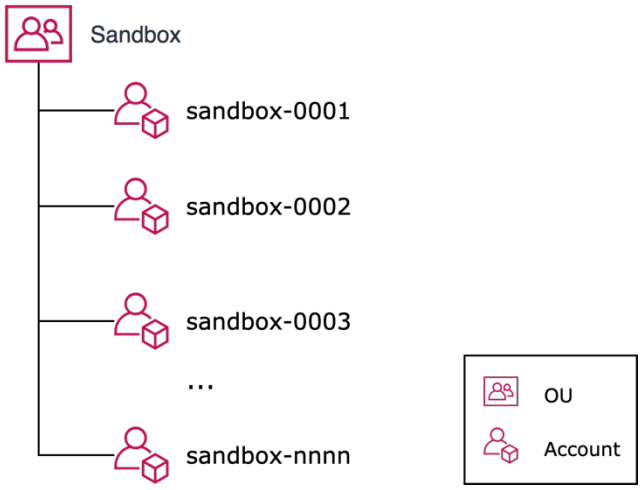
In support of hackathons and other events, you might also find value in creating transient sandbox accounts for temporary teams of people.



Example structure of Sandbox OU

Temporary recycled sandboxes

In the following example, sandbox accounts are named independently of the current user of the environment. Sandbox environments are checked out by builders and/or teams, used for a temporary period of time, and recycled for future use.



Example structure of Sandbox OU with recycled accounts

Workloads OU

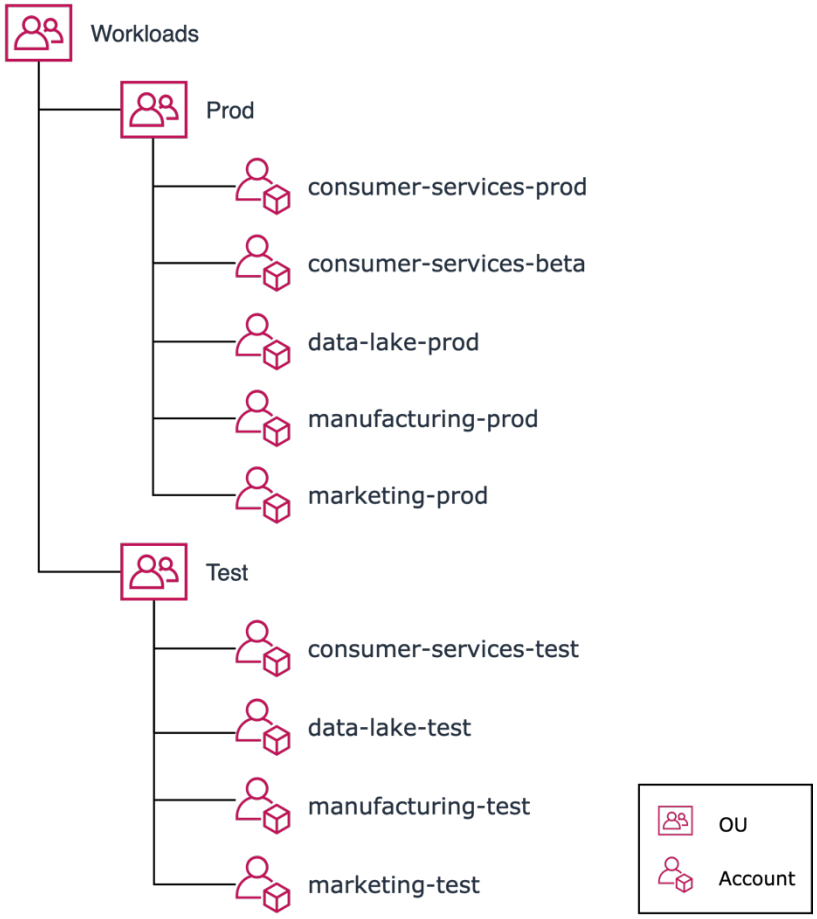
The Workloads OU is intended to house most of your business-specific workloads including both production and non-production environments. These workloads can be a mix of commercial off-the-shelf (COTS) applications and your own internally developed custom applications and data services.

Workloads in this OU often include shared application and data services that are used by other workloads.

Example structure

The following example represents a basic structure in which sets of workloads owned by diverse business units or teams reside in two child OUs: *Prod* and *Test*. In this example, a common governance and operating model applies across those areas. The *data-lake-prod* account shown in this example contains data services that are shared with other production workloads and accounts.

For general guidance on separating production and non-production workloads and resources, see Organizing workload-oriented OUs (p. 37).



Example structure of Workloads OU

Policy Staging OU

The Policy Staging OU is intended to help teams that manage overall policies for your AWS environment to safely test potentially broadly impacting policy changes before applying them to the intended OUs and/or accounts. For example, SCPs and tag policies should be tested prior to applying them to the intended OUs or accounts.

Similarly, broadly applicable account baseline IAM roles and policies should also be tested using the Policy Staging OU.

Workload-specific policies

Development and testing of workload-specific IAM roles and policies do not need to use the Policy Staging OU. Rather, workload owning teams typically develop and test these resources alongside other workload-specific resources in development and test accounts within your Security, Infrastructure, and Workloads OUs.

Recommended testing and promotion workflow

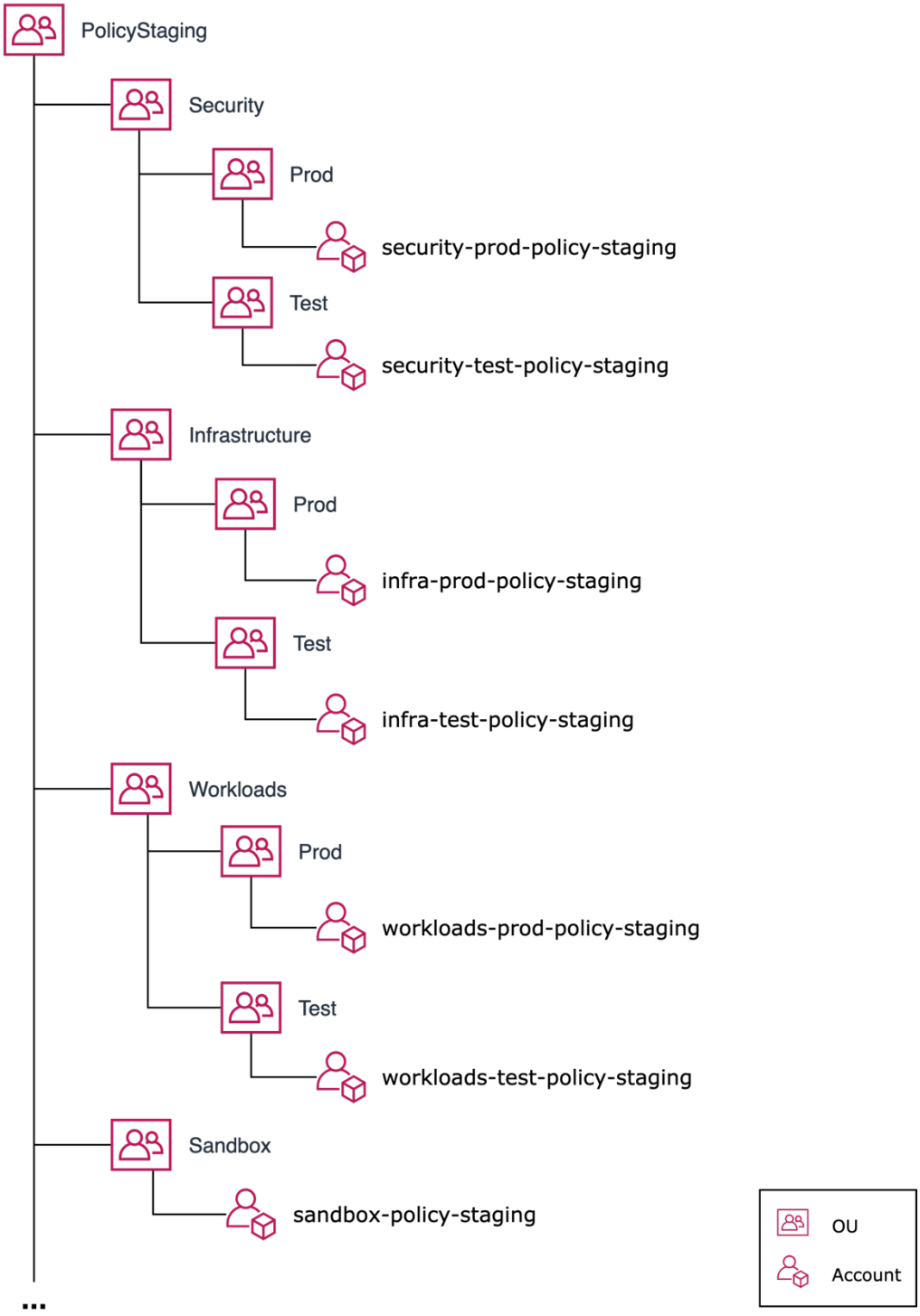
Once you have tested changes in the Policy Staging OU, we recommend that you temporarily associate the policy changes to a single account in the intended OU. If the changes are ultimately targeting an OU, apply the changes to the intended OU and remove the changes from the account only after you have validated that the changes are working as intended.

This approach enables you to validate the changes in production before more broadly applying them.

Example structure

In this example, a set of child OUs mirrors an overall OU structure. At least one test account is included under each child OU.

In support of testing SCPs and tag policies that are intended to be applied at the OU level, your teams should first apply them to one of the test child OUs. SCPs and tag policies that are applied to a specific account require creation of a test account under the appropriate test child OU.



Example structure of Policy Staging OU

Suspended OU

The Suspended OU is used as a temporary holding area for accounts that are required to have their use suspended either temporarily or permanently.

Moving an account to this OU doesn't automatically change the overall status of the account. For example, in cases where you intend to permanently stop using an account, you would follow the Closing an account process to permanently close the account.

Examples of using the Suspended OU include:

- A person's sandbox account is no longer needed due to the departure of the person from the company.
- A workload account is no longer needed due to the resources having been either retired or migrated to another account.

Constraining activity in suspended accounts

You can use service control policies (SCPs) to inhibit users other than your security and cloud platform teams from using AWS APIs in each account. Additionally, you can remove application-level access so that users can no longer access and manage application resources for each suspended account.

To reduce risk and potentially minimize costs, you can also stop any running resources and applications in each suspended account.

Resources should not be deleted from a suspended account unless the account is intended to be closed.

Tagging suspended accounts

Because you might use the Suspended OU for a variety of use cases, we recommend that you apply tags to each account to record the reason for moving the account and the OU from where the account originated. Each process that you establish to support your suspension use cases can use the tag to automatically process the suspended accounts. This tag can also aid in your internal tracing and auditing of an account's lifecycle.

Closing suspended accounts

If an account is moved to this OU prior to the start of the closure process, you can implement a policy and process to automatically start the account closing process a certain number of days after an account has been moved to this OU.

Once the account closure process has been completed, the account is longer visible in your organization.

Individual Business Users OU

The Individual Business Users OU houses accounts for individual business users and teams who need access to directly manage AWS resources outside the context of resources managed within your Workloads OU.

In some cases, you can consider a small number of AWS resources as something other than a workload. For example, a business team may require write access to Amazon S3 buckets to share marketing videos and data with a business partner. In these cases, you might choose to manage these resources in accounts within the individual business users OU rather than in accounts in the Workloads OU.

Guardrails

We recommend that you apply a combination of SCPs and IAM permissions to this OU and authorized users. This ensures that only those AWS services, resources, and actions needed are granted. Depending on the nature of the use cases, you can apply guardrails to individual accounts in this OU.

Services that do not require direct user access to accounts

The individual business users OU does not apply when users can authenticate and be authorized to interact with applications and services without requiring direct access to an account. For example, business users often need access to Amazon QuickSight for business intelligence (BI) purposes. Assuming that you consider your QuickSight-based BI capability a workload, you can position the QuickSight resources and data in a workloads account in the Workloads OU. In this case, BI users are authorized to access the QuickSight service directly without needing access at the account level.

Exceptions OU

The Exceptions OU houses accounts that require an exception to the security policies that are applied to your Workloads OU. Normally, there should be a minimal number of accounts, if any, in this OU.

Service control policies and scrutiny

Given the unique nature of the exceptions, SCPs are typically applied at the account level in this OU.

Due to the customized security controls that apply to these accounts, owners of these accounts can expect to experience greater scrutiny from security monitoring systems.

Consider Workloads OU as an alternative

If you observe a pattern in which multiple accounts require the same set of exceptions, we recommend that you examine either your existing workloads policies or an extended form of the Workloads OU structure and house the accounts under the Workloads OU. You can introduce another level of OU under the Workloads OU to represent a common set of security policies and/or operational processes that can be applied to multiple workload environments. For more information, see Organizing workload-oriented OUs (p. 37).

Deployments OU

The Deployments OU contains resources and workloads that support how you build, validate, promote, and release changes to your workloads.

You may already be using continuous integration/continuous delivery (CI/CD) capabilities to help manage and automate how changes to various types of source code are processed.

Using CI/CD capabilities residing outside of your AWS environment

If you already use on-premises and/or managed CI/CD and related capabilities that reside outside of your AWS environment and you do not expect to use and/or manage CI/CD services within your AWS

environment in the near term, you may not immediately need to establish the Deployments OU and an associated set of CI/CD oriented accounts.

In this scenario, you must work through any access and potential network connectivity dependencies between your CI/CD capabilities residing outside of your AWS environment and your workloads environments in AWS.

Separating CI/CD management capabilities from workloads

If you intend to deploy and/or manage your own CI/CD capabilities in AWS or use AWS managed CI/CD services, we recommend that you use a set of production deployment accounts within the Deployments OU to house the CI/CD management capabilities.

Reasons for separating your CI/CD management capabilities from your workload environments include:

- **Critical roles played by CI/CD capabilities** – Your CI/CD capabilities are responsible for orchestrating quality validation, security compliance checks, building and publishing production candidate artifacts, promoting artifacts, and ultimately triggering release of artifacts to production environment. Given the critical nature of these roles, it's important that you can apply appropriate policies and operational practices to your CI/CD capabilities that are different than those applied to your workload environments.

For example, your CI jobs and CD pipelines typically need write access to publish and promote candidate artifacts to an artifact management service. However, your production workload environments should only require read access to artifact management services in order to obtain the already built and promoted artifacts.

- **CD pipelines affect non-production and production workload environments** – When CD pipelines orchestrate the validation of changes and ultimately trigger the release of changes to production, the pipelines often need to access workloads residing in both non-production test and production workload environments.

For example, if you manage your CI/CD capabilities in your production workload environments, then you must allow the production workload environments to access your non-production environments. By centralizing your CI/CD capabilities in your CI/CD accounts, you can avoid enabling your production workload environments access to non-production environments.

- **CI/CD capabilities depend on unique tooling** – Your CI/CD management capabilities, CI jobs, and CD pipelines often depends on tools that are different from those required to run and operate your workloads. Limiting the use of these tools to your CI/CD accounts can help you reduce the complexity and attack surface of your workload environments.

Running CI jobs and CD build stages in deployment accounts

Because CI jobs and CD pipeline build stages are responsible for generating formal candidate artifacts, we recommend that you perform these activities in a production environment. Rather than perform these activities in your production workload environments, we recommend that you run them in your production CI/CD accounts.

Aligning CI/CD accounts with groups of workloads

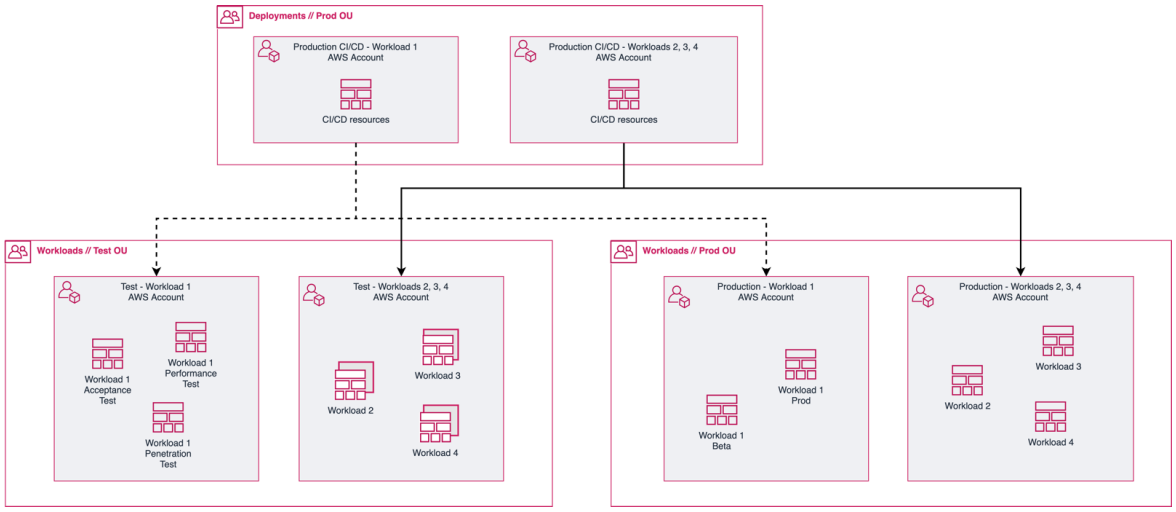
We recommend that you define CI/CD accounts in the Deployments OU that are aligned with how you group related workloads in your workload-oriented OUs. By doing so, you can more easily align the security policies and operational requirements of each group of workloads with their companion CI/CD account.

This approach helps limit the scope of impact of an issue in a CI/CD account to a single workload or group of workloads. Any access issues and resource conflicts that arise in one CI/CD account will most likely impact only the associated group of workloads and the accounts in which the workloads reside.

In the following diagram, **Workload 1** represents a workload that is dedicated to its own production account. **Workloads 2, 3, and 4** are configured as a group of related workloads and are managed in a separate production account.

A companion set of test accounts contain test instances of the workloads. In each test account, there are likely be multiple workload environments for a given workload so that various forms of testing can be supported at the same time.

A CI/CD account has been created to support Workload 1 and a second CI/CD account has been created to support the set of related workloads 2, 3, and 4. The CI/CD resources in each production CI/CD account may need to interact with the target workload environments in both the test and production accounts.



Example alignment of CI/CD accounts to workload accounts

Considering use of multi-tenant shared CI/CD services

We don't recommend using a common CI/CD account in support of a diverse set of workloads due to the inherent difficulties in managing and securing the environment. In this approach, a wide array of teams likely needs access to the common CI/CD services. Separating access between teams within a common CI/CD account and limiting adverse impacts across many workload accounts is more complicated and prone to error than using a distinct CI/CD account for each group of workloads.

Enabling team access to production CI/CD services

Team members who own certain workloads require some degree of access to the production CI/CD services associated with those workloads. At minimum, the ability to monitor execution of CI jobs and CD pipelines is required.

Depending on the process by which CI jobs and CD pipelines are promoted to your production CI/CD accounts, some limited degree of write access to CI jobs and CD pipelines may be necessary for designated administrators within the teams that own the CI jobs and CD pipelines.

Whether team members require direct access to the accounts in which the CI/CD service resides depends on the type of CI/CD tooling that you're using. For example, if you're managing your own CI/CD tooling, then it's likely that access controls are managed within the CI/CD tooling and that authentication occurs outside the context of the account in which the CI/CD tooling resides. In this example, team members likely won't need direct access to the CI/CD accounts.

If you're using AWS managed CI/CD services, then team members require at least read access to the CI/CD accounts to monitor execution of CI jobs and CD pipelines.

In the following diagram, teams who own the workloads associated with each production CI/CD account are shown accessing the CI/CD resources in their respective accounts.

Teams accessing workload-specific CI/CD accounts

Providing CD pipeline access to workload accounts

If your CD pipelines use deployment methods and tools to push changes to workload environments, then either your CD pipelines or the deployment tools to which deployment tasks are delegated require sufficient write access to the target workload environments.

An alternative method of deploying changes to workload environments is the pull style of deployment that intentionally avoids requiring CD pipelines to have write access to target workload environments. In the pull model, tools within the target workload environments have the permissions necessary to both detect changes of interest (for example, newly promoted artifacts or configuration changes) and to deploy those changes locally within the workload environment.

Testing changes to your CI/CD capabilities

We recommend that you establish non-production test accounts in your Deployments OU. These accounts can be used to test changes to how you use and manage your CI/CD capabilities before promoting those changes to your production CI/CD environments.

Establishing a set of test accounts for your CI/CD is most important when you're planning to manage CI/CD tools in AWS. To support and evolve your use of these tools, you should have a test environment separate from production.

If you use AWS managed CI/CD related services including AWS CodePipeline, AWS CodeBuild, AWS Proton, and/or AWS CodeDeploy, you can still benefit from having a test environment so that you can test changes to how you secure and use those managed services.

As a general practice, your test CI/CD environments should not have access to any of your production CI/CD and production workload environments. Instead, your test CI/CD environment access should be constrained to non-production workload environments.

Developing and testing CI jobs and CD pipelines

The Deployments OU is not intended to be used as a means for teams to develop and test CI jobs and CD pipelines. Rather, just like the development and early testing of any other workload, we recommend that you develop and test CI jobs and CD pipelines in your development AWS environments.

As with other types of code, you can promote source for CI jobs and CD pipelines to your production CI/CD environments using processes tailored for the promotion of CI jobs and CD pipelines. This approach enables you to minimize the access required in your production CI/CD environments and constrain your test CI/CD environments in your Deployments OU to testing of the CI/CD capabilities themselves.

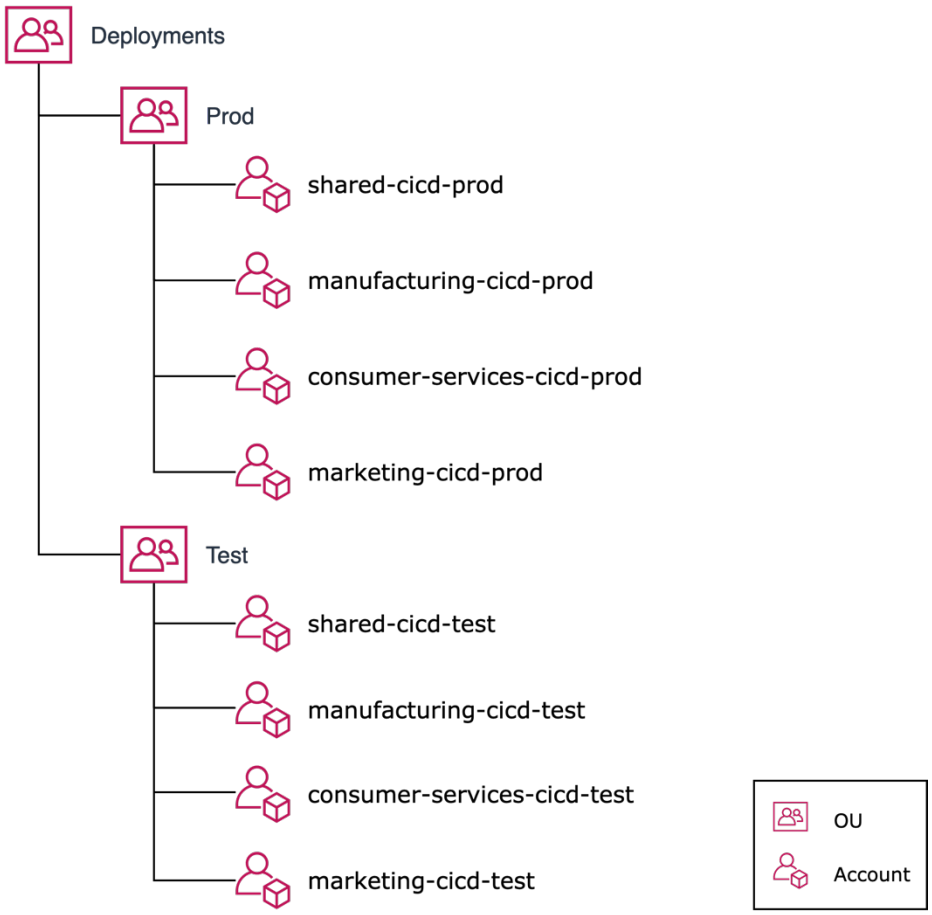
Example structure

In the following example, a series of production accounts represent centrally-managed, shared CI/CD services and a set of federated, business unit-managed CI/CD resources.

In a typical scenario, the stable, production-quality CI/CD capabilities in the accounts qualified with *-prod* are expected to effect changes across both production and non-production workload environments. The *-prod* qualifier is simply denoting that these are your stable, production-quality CI/CD environments.

The example accounts in the Test OU are qualified with *-test*. These accounts are intended to represent environments in which you might test changes to your CI/CD platforms or how you use managed CI/CD services before you apply those changes to your stable, production-quality CI/CD environments.

For general guidance on separating production and non-production workloads and resources, see Organizing workload-oriented OUs (p. 37).



Example structure of Deployments OU

Transitional OU

The Transitional OU is intended as a temporary holding area for existing accounts and workloads that you move to your organization before you formally integrate them into your more standardized areas of your AWS environment structure.

Common reasons for moving accounts into your organization

Common reasons for moving accounts into your organization include:

- Acquisition of a company that is already using AWS and has a set of accounts
- Existence of your own accounts that were created before you established your newer AWS environment structure
- Movement of accounts that have previously been managed by a third party

Benefits of moving accounts into your AWS organization

By moving existing accounts into your organization, you can begin to gain some of the benefits of using AWS Organizations including:

- Centralized visibility
- Option to begin applying common policies
- Consolidated billing, cost, and asset management
- Simplified use of AWS Organizations-enabled AWS security services
- Integration with existing federated access capabilities

Considerations for moving accounts into your organization

If you plan to move an account from an existing organization, you must first remove the account from the organization. For more information, see Removing a member account from your organization. Once an account is removed from an organization, it is referred to as a standalone account.

Moving a standalone account that does not have dependencies on other accounts is a straightforward process. In this case, there's generally no need to migrate or modify the existing workloads in the account to be moved. For more information, see Inviting an account to join your organization.

If the standalone account to be moved has dependencies on other accounts, then you should evaluate those dependencies to determine if they should be addressed before moving the account.

In your target organization, we recommend that you review SCPs in the organization's root to ensure that those SCPs won't adversely impact the accounts to be moved.

If you're moving a set of related accounts to your organization, you can create a child OU under the Transitional OU for the related set of accounts.

After moving accounts

Over time, as you better understand the direction for these accounts and the workloads contained in them, you can either move the accounts to your Workloads OU as is, invest in migrating the workloads to other accounts, or decommission either the workloads or accounts.

Organizing workload-oriented OUs

The recommended Security OU (p. 18), Infrastructure OU (p. 22), Workloads OU (p. 25), and Deployments OU (p. 30) are top-level OUs that contain workloads. This section outlines considerations for organizing these workload-oriented OUs.

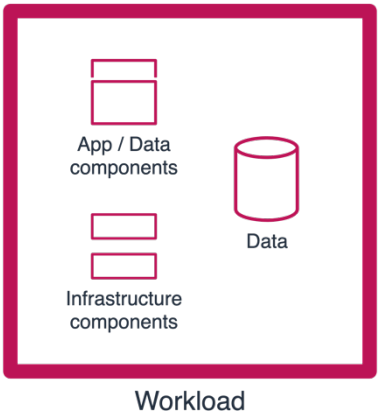
Topics

Workloads and environments

This section defines basic terms and concepts related to workloads. Becoming aware of these concepts helps you understand our recommendations for organizing your workload-oriented OUs.

Workloads

Many of your top-level OUs will house collections of applications, cloud resources, and data in the form of workloads. A *workload* is a discrete collection of components and data that you manage. A workload can be a commercial off-the-shelf (COTS) application or your own custom application and data service.



Composition of a workload

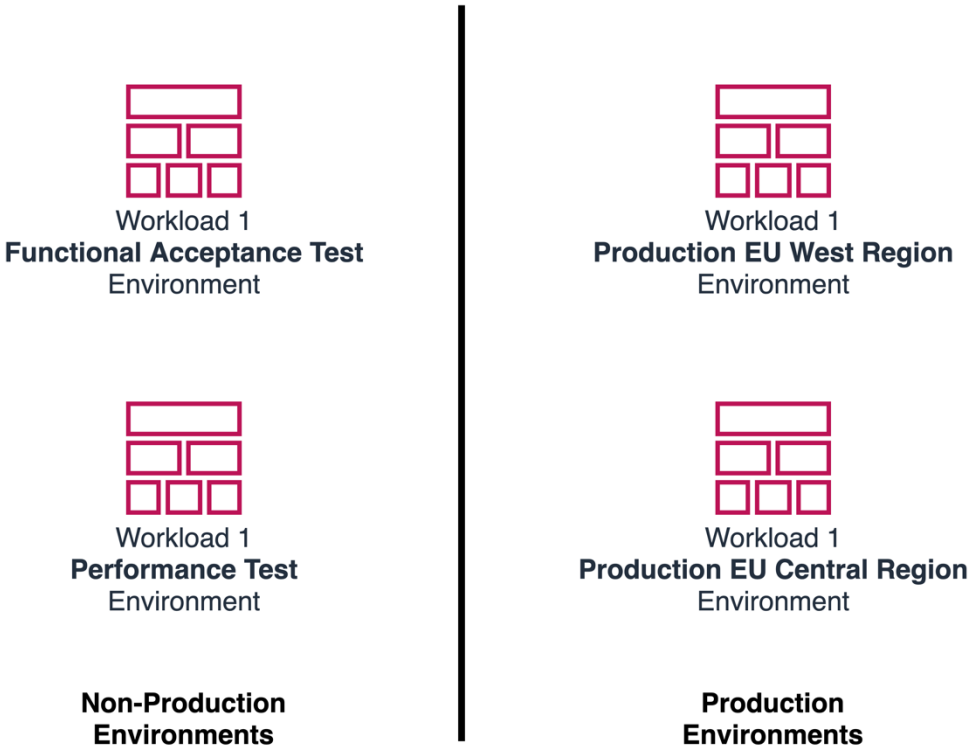
Workload environments

For a given type of workload, you typically have multiple instances. This setup means that you can experiment, develop, and test changes to the workload before you promote and deploy those changes to the production instances of the workload. A given instance of a workload is a *workload environment*.

Whether your workload is a COTS application, a custom application, a custom data service, or a foundational security or infrastructure capability, you often need separate non-production workload

environments to support your software development lifecycle (SDLC) processes. You can have multiple SDLC processes depending on the diversity of your workload portfolio and your company organization.

The following example shows multiple environments of a workload across non-production test and production workload environments.



Example of multiple environments of a workload

With COTS applications, you may not perform custom development, apart from implementing custom integrations with your own systems. However, you can experiment with and formally test new versions of the COTS applications in non-production environments before deploying them to production.

For detailed examples of common purposes of workload environments in relation to an SDLC, see Appendix B – Worksheet: Mapping workload environment purposes to hosting environment types (p. 64).

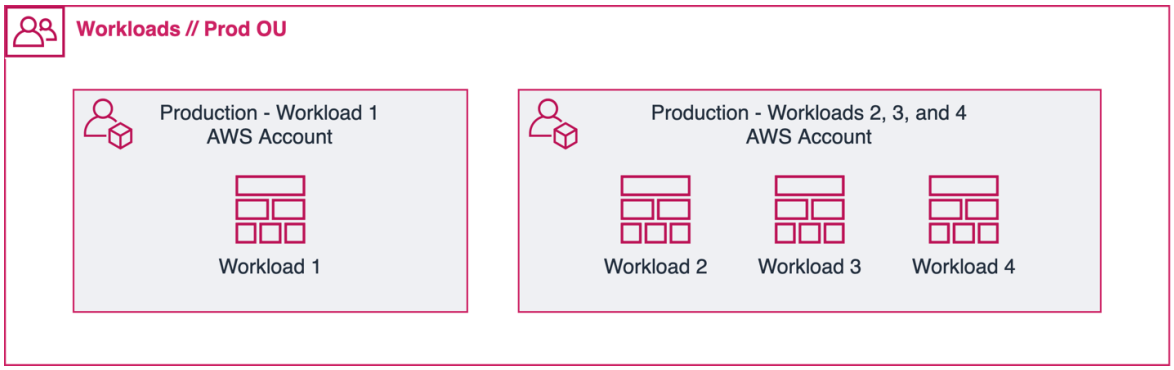
Workload accounts

In your on-premises context, you can refer to the places where your workloads reside as *hosting environments*. For example, you might have a dedicated production hosting environment in which your production workload environments reside.

In AWS, each of your workload environments is typically contained in an account, where each account is similar to a distinct hosting environment.

Depending on how you choose to scope your workload accounts, you might have a single workload environment per account. Or, you might have multiple workload environments and perhaps multiple workload types in the same workloads account.

The following diagram shows two degrees of scoping production workload accounts. In one example, a workload account is dedicated to a single workload environment. In the other example, multiple workload types reside in a single production workload account.



Example workload accounts with different degrees of scoping

Production and non-production workload environments

We recommend that you isolate production workload environments and data in production accounts housed within production OUs, under your top-level workload-oriented OUs. Apart from production OUs, we recommend that you define one or more non-production OUs that contain accounts and workload environments that are used to develop and test workloads.

Workload dependencies across environments

When you consider the structure of your workload-oriented OUs, you should decide on the extent to which you expect access between production and non-production environments.

Production environments accessing non-production

Generally, workloads deployed to your production environments should not depend on workloads contained in your non-production environments.

Non-production environments accessing dependencies

In non-production environments, it is common for workloads to depend on stable shared application, data, and infrastructure services. Where feasible, we recommend that these shared services be non-production test instances. These non-production test instances should use test data so that your non-production workloads do not depend on access to your production environments and data.

For example, you can configure workloads in a non-production test environment that depend on integrating with a data service to use a stable, shared test instance of the service that is populated with test data.

However, in some cases non-production environments may need access to production shared services. For example, it's typical for non-production development and test environments to require read-only access to shared source code and artifact management services. Providing access to these shared services enables you to deploy candidate and promoted changes and artifacts to your non-production environments in support of development and testing activities.

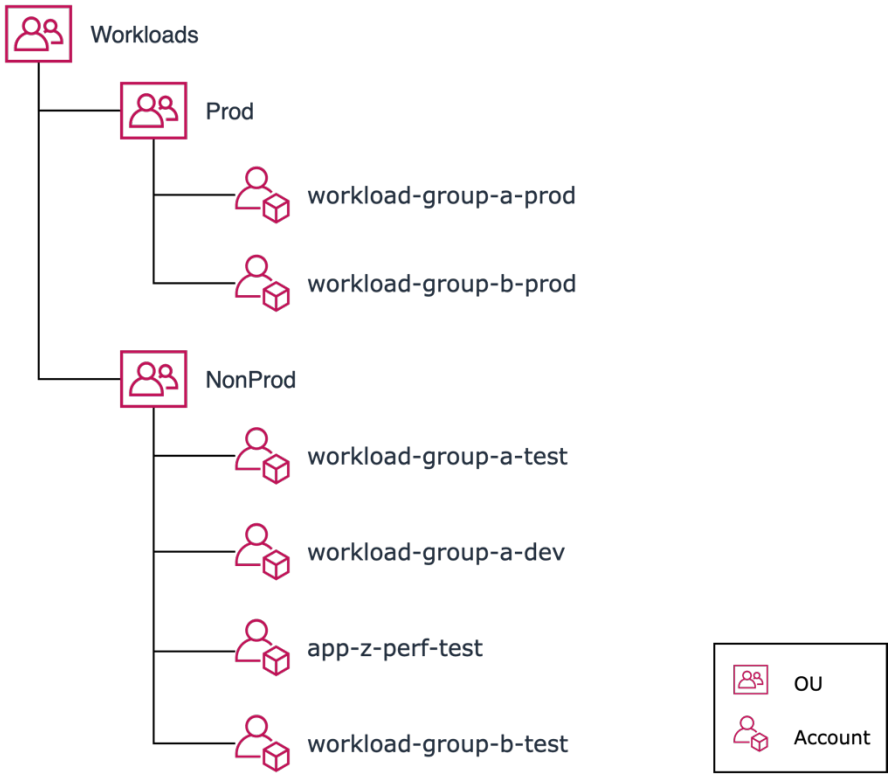
OU structure for non-production environments

You can use OUs to organize your non-production environments in a couple ways.

Option A: Common guardrails across non-production environments

When non-production workloads require the same set of overall access policies or benefit from being operationally managed together, you can define a single *NonProd* OU to contain all the accounts that support non-production forms of your workloads.

The following example shows the Workloads OU where a *Prod* child OU contains production accounts and workloads, and a *NonProd* child OU combines both development and test accounts and workloads.



Example Workloads OU with common policies across a NonProd child OU

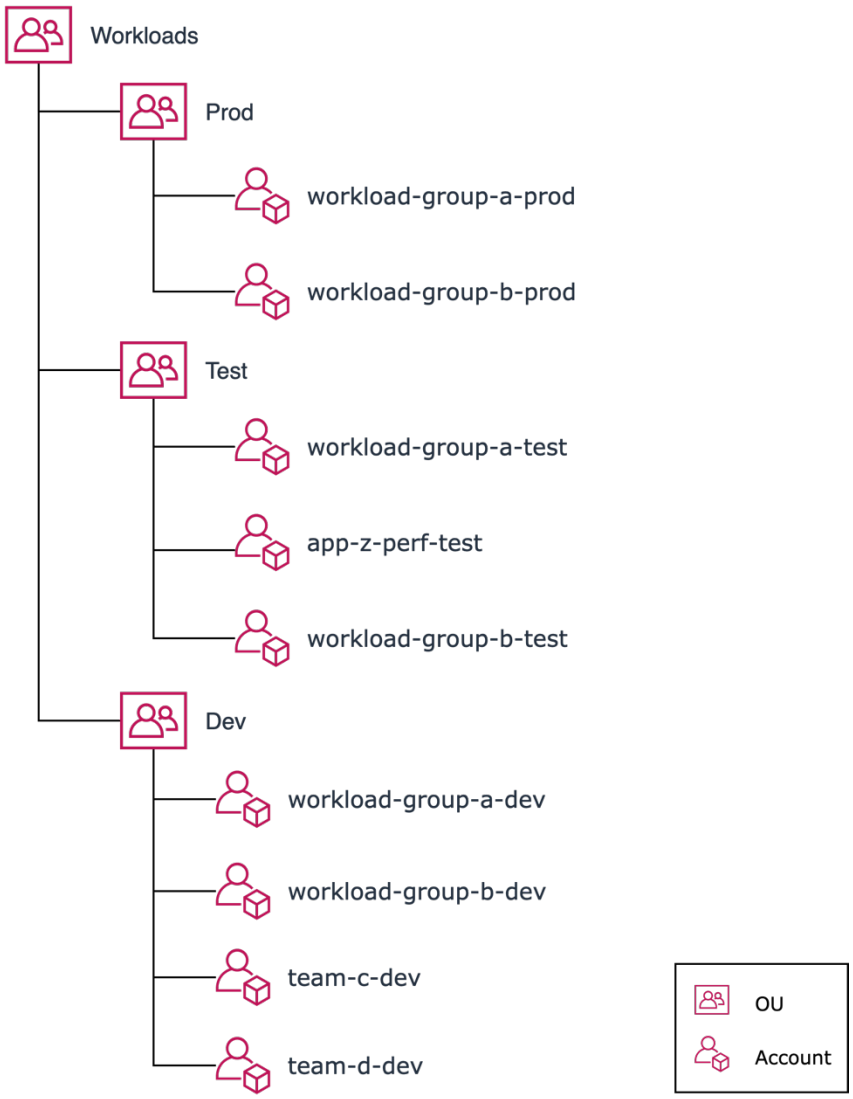
Option B: Different guardrails across non-production environments

Sometimes your process for developing and testing changes involves workload environments that have fundamentally different access policies or ways in which you manage and apply foundational resources. In these cases, it makes sense to create distinct OUs to support these diverse requirements.

For example, you want to support development environments that provide teams with more freedom to experiment, iterate, and develop largely on their own (rather than more formally managed and controlled production-like test environments). In this case, overall access policies and management of baseline resources for the development environments is significantly different than those used to

support test environments. It makes sense for you to create a distinct OU for development work and another OU for your test workloads.

The following example represents a simple form of this structure where *Test* and *Dev* OUs reside adjacent to the recommended *Prod* OU.



Example Workloads OU with different policies for Test and Dev child OUs

The preceding example shows two different approaches to scoping development environment accounts. One approach is where development environments are aligned with the same groupings of workloads as used in test and production OUs. The other approach is one in which development environments are aligned based on teams.

Worksheets to help decide on workload-oriented OUs

The following appendices include a set of worksheets and example considerations for identifying your overall types of workload environments and supporting OUs:

- Appendix B – Worksheet for mapping workload environment purposes to hosting environment types (p. 64)

- Appendix C – Worksheet for identifying attributes of workload hosting environments (p. 71)

Appendix B (p. 64) helps you identify the overall types of work you perform from design through production and helps you identify the corresponding workload environments in which you expect to perform work and house workloads.

Appendix C (p. 71) helps you further refine the overall types of workload environments by identifying key distinguishing access and management attributes of each overall type of workload environment.

By understanding commonalities of, and contrasts between, your overall types of workload environments, you can make an informed decision about the set of child OUs that can best support your workload-oriented OUs.

Extended workload-oriented OU structure

An extended form of the workload-oriented OU structure can be used to support cases in which you need to either organize workloads for visibility and management purposes or apply different security and operational policies to either a workload or group of related workloads.

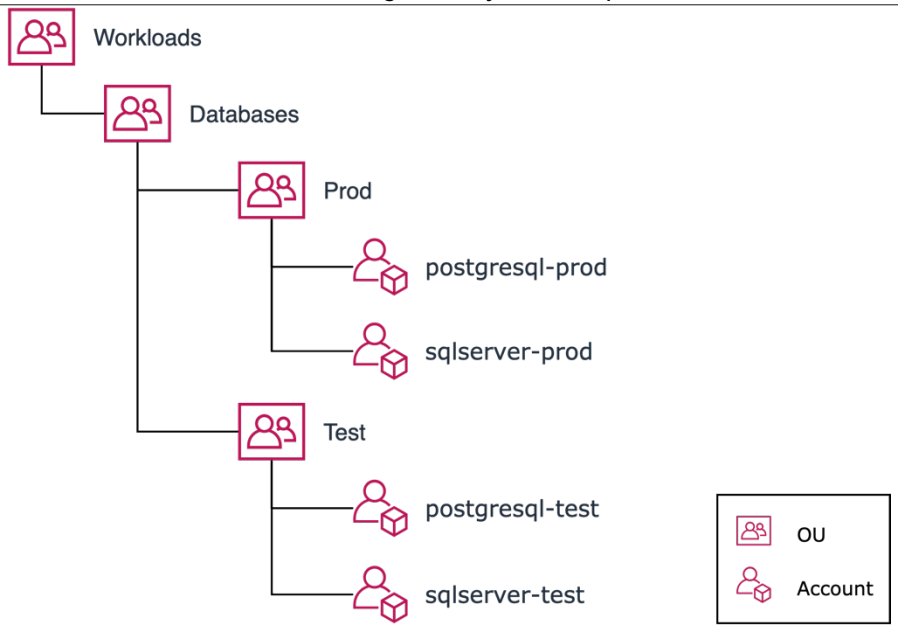
When workloads have diverse security and operational policy requirements, you cannot effectively manage guardrails and other controls at the level of the workload-oriented OU. By adding child OUs to a workload-oriented OU, you can group related workloads in the same child OU. You can then apply distinct security and operational policies to the child OUs.

For example, a workload or a group of related workloads may benefit from having a distinct allow list of AWS services that is implemented via a service control policy (SCP). This policy may be different than the requirements associated with other workloads. Rather than applying the SCP to each of the related workload accounts, it is recommended that you apply the SCP to an OU that groups the related accounts.

Grouping related workloads

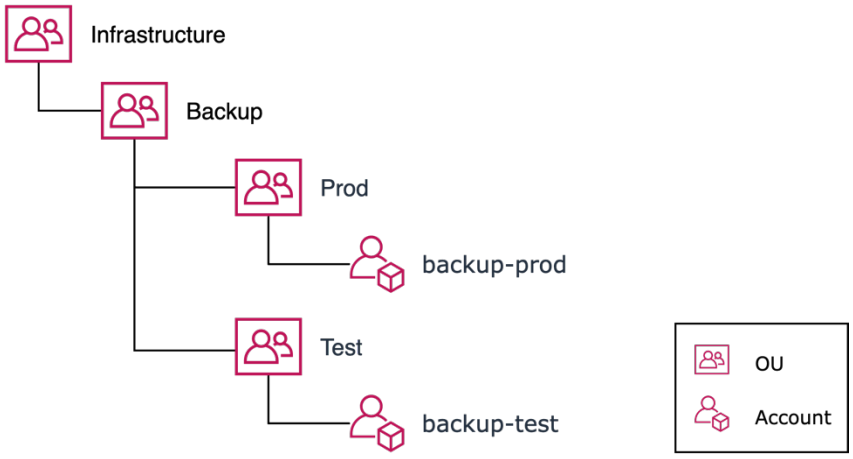
When you have groups of related workloads that require the same overall set of security and operational policies, you can create a child OU for each group of workloads.

For example, if you manage a series of database services that are shared across your organizations and have common security and operational policy requirements, you might find value in grouping those data services under a common child OU.



Group of workloads with distinct policy requirements

The following example represents a shared backup capability you can provide across your AWS environment. If this capability requires a set of security and operational policies that are distinct from other infrastructure workloads, then you can allocate a distinct OU for this workload.

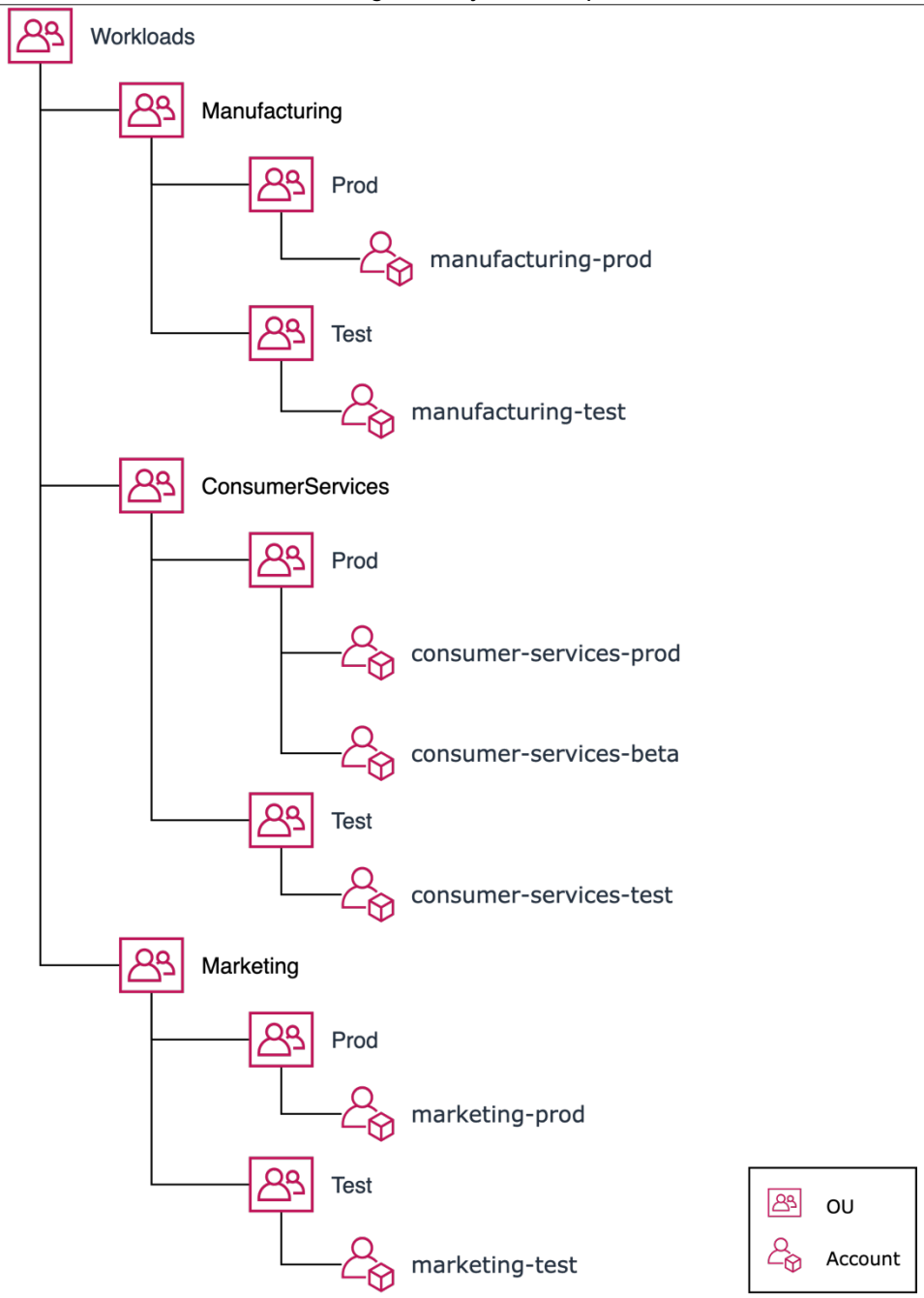


Single workload with distinct policy requirements

Separating business units with significantly different policies

If you have largely autonomous business units (BUs) that manage workloads in your common AWS organization and the BUs have significantly different security and operational policies, you can create a child OU under your Workloads OU for each BU.

In the following example, each BU is provided with its own OU so that different SCPs and/or operational policies can be applied independently form the other OUs.



Example business unit separation

Patterns for organizing your AWS accounts

This section introduces a series of example AWS account structures based on the principles and best practices contained in this document.

It's common for customers to start with a basic structure and incrementally expand it as their needs evolve and their experience with AWS grows. Accordingly, the examples start simple and progressively grow to represent this typical evolution.

More detailed descriptions of the OUs represented in these examples are addressed in Recommended OUs (p. 17) and Organizing workload-oriented OUs (p. 37).

Single AWS account

If you have experimented with AWS, you may have used a single AWS account in which to perform some of your initial non-production work. You may have even started to manage some of your early workloads as production resources in the single AWS account.

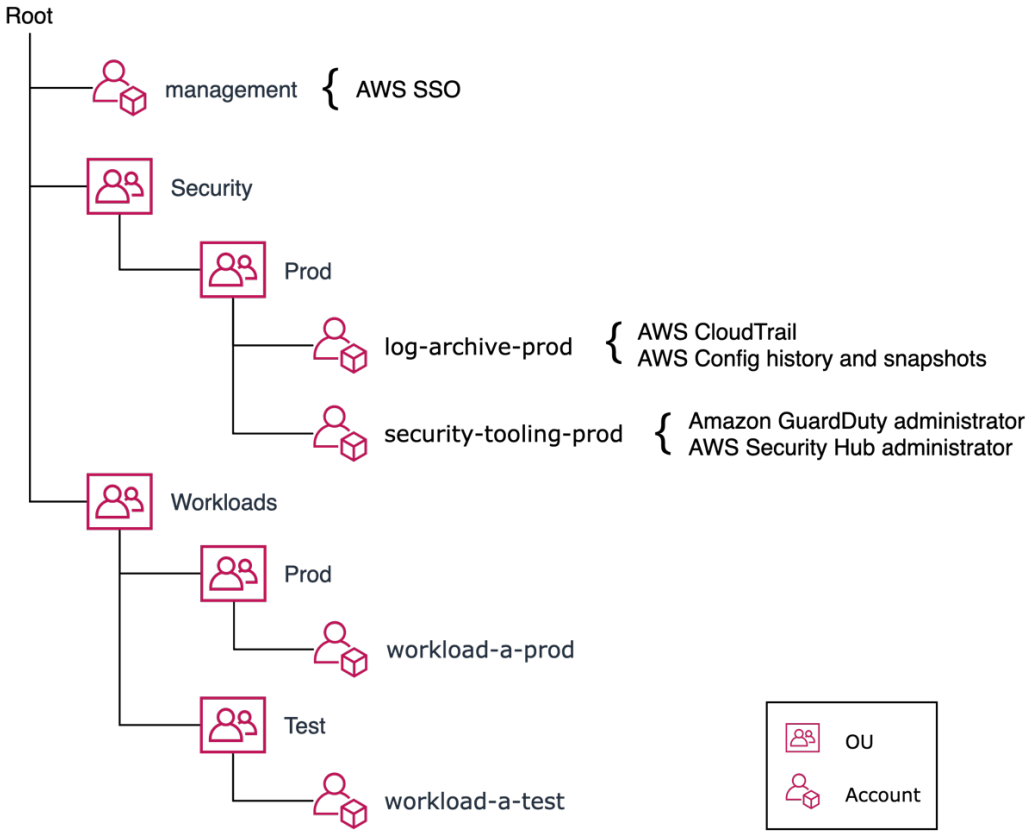
If you have a single account today and either have production workloads in place or are considering deploying production workloads, we recommend that you transition to the use of multiple accounts so that you can gain the Benefits of using multiple AWS accounts (p. 5). See the Transitional OU (p. 35) for information on the considerations for moving accounts into a new overall AWS environment.

Production starter organization

This pattern represents a minimal starter environment in which the primary focus is on supporting a workload in a production environment.

For example, you may have an Amazon S3-backed on-premises backup solution that you simply need to test and deploy to production. Similarly, you may have a static web site that depends on Amazon CloudFront as a content delivery network (CDN) and uses a private bucket in Amazon S3 to manage the web content.

In these scenarios, you might not need sandbox and development environments. The following figure shows an example of this type of minimal starter production environment.



Example production starter organization

In this example, the organization's management account uses AWS Single Sign-on (AWS SSO) to help provide your human users with federated access to the AWS accounts in your organization.

The Security OU (p. 18) contains a *log-archive-prod* account to act as the consolidation point in the organization for log data that is gathered from all of the accounts—not just other production environments—and primarily used by your security, audit, and compliance teams.

The Security OU also contains a *security-tooling-prod* account where you manage recommended security tools and service resources.

Since the capabilities provided in the *log-archive-prod* and *security-tooling-prod* accounts are expected to be of production quality, these accounts are contained in a Prod OU under the Security OU. The *-prod* suffix in these example account names emphasizes the production quality of their resources and workloads. The suffix is not intended to suggest that these accounts and their resources apply only to production accounts.

In future configurations, you can introduce non-production or test OUs and accounts associated with your Security OU. Where it's feasible to test changes inside the same organization, these non-production environments can help you develop and test changes for your production quality capabilities before promoting those changes to your production environments.

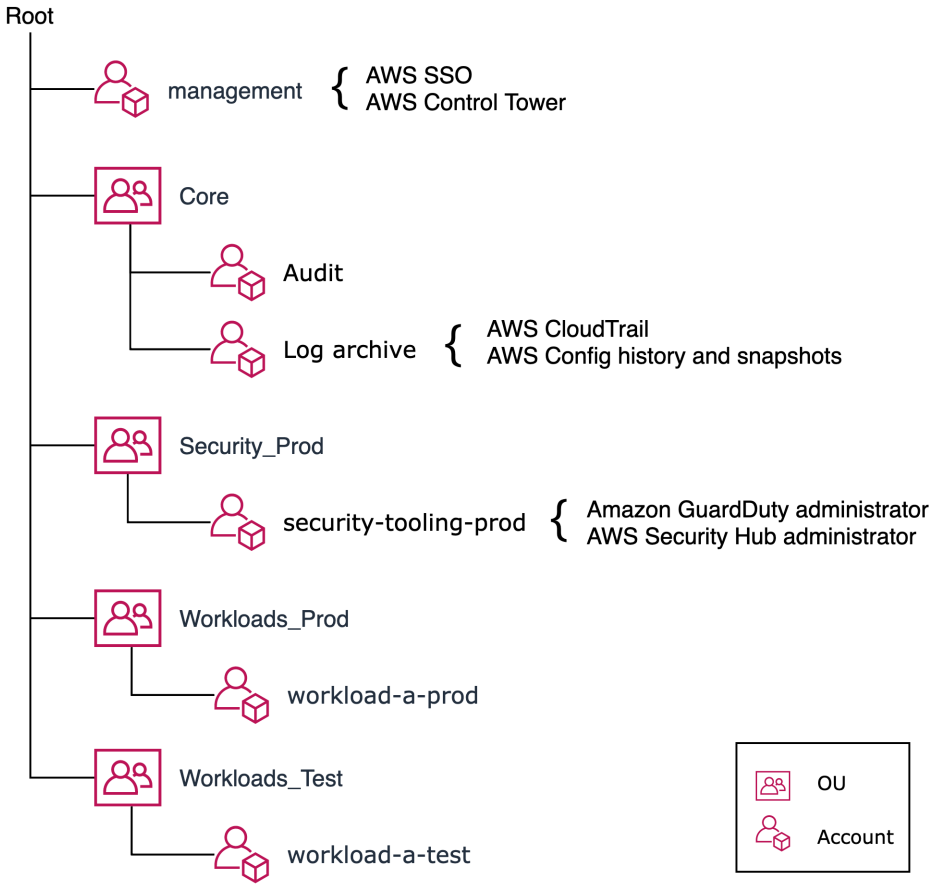
In cases where you cannot easily test certain changes that are foundational to your AWS environment in the same AWS organization, you may benefit from using a separate AWS organization to test such foundational changes. See Multiple AWS organizations (p. 12) for more information.

A Workloads OU (p. 25) along with Prod and Test OUs contain the *workload-a-test* and *workload-a-prod* accounts.

Production starter organization with AWS Control Tower

When you use AWS Control Tower to establish your AWS environment, it automatically creates the Audit and Log archive accounts under a Core OU. The Log archive account plays the same role as the Log archive account described in the Security OU (p. 18). The Audit account is intended to provide your security team with cross-account access to other member accounts in your organization.

AWS Control Tower also automatically sets up AWS SSO in the organization's management account. The following figure shows this configuration.



Example production starter organization with AWS Control Tower

In this example, a Security OU (p. 18) is created by the AWS Control Tower Account Factory feature to contain a *security-tooling-prod* account where recommended security tools and service resources are managed.

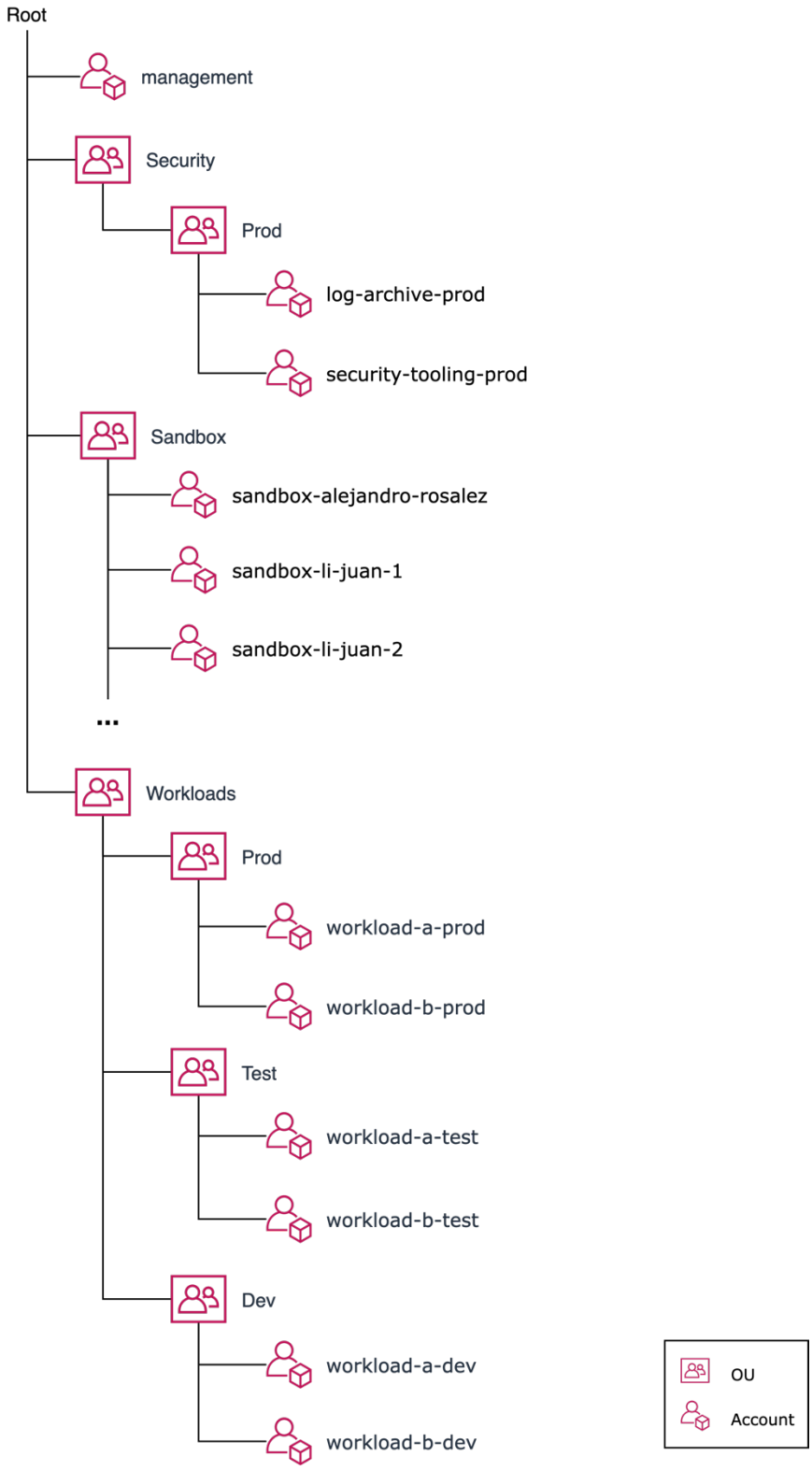
Two Workloads OUs (p. 25) house the production and test environments for a workload.

Since AWS Control Tower currently supports a single level of OUs, the names of the security and workloads OUs include underscores to represent the desired hierarchy. For example: Security_Prod, Workloads_Prod, and Workloads_Test.

Basic organization

The following example incorporates a security tooling environment for common security services, a second workload, and support for sandbox and development environments. Additions include:

- A Sandbox OU (p. 23) to contain a series of disconnected sandbox environment accounts
- An additional workload in the form of *workload-b-prod*, *workload-b-test*, and *workload-b-dev* accounts
- A Dev OU under the Workloads OU (p. 25) to contain development environment accounts associated with the workloads

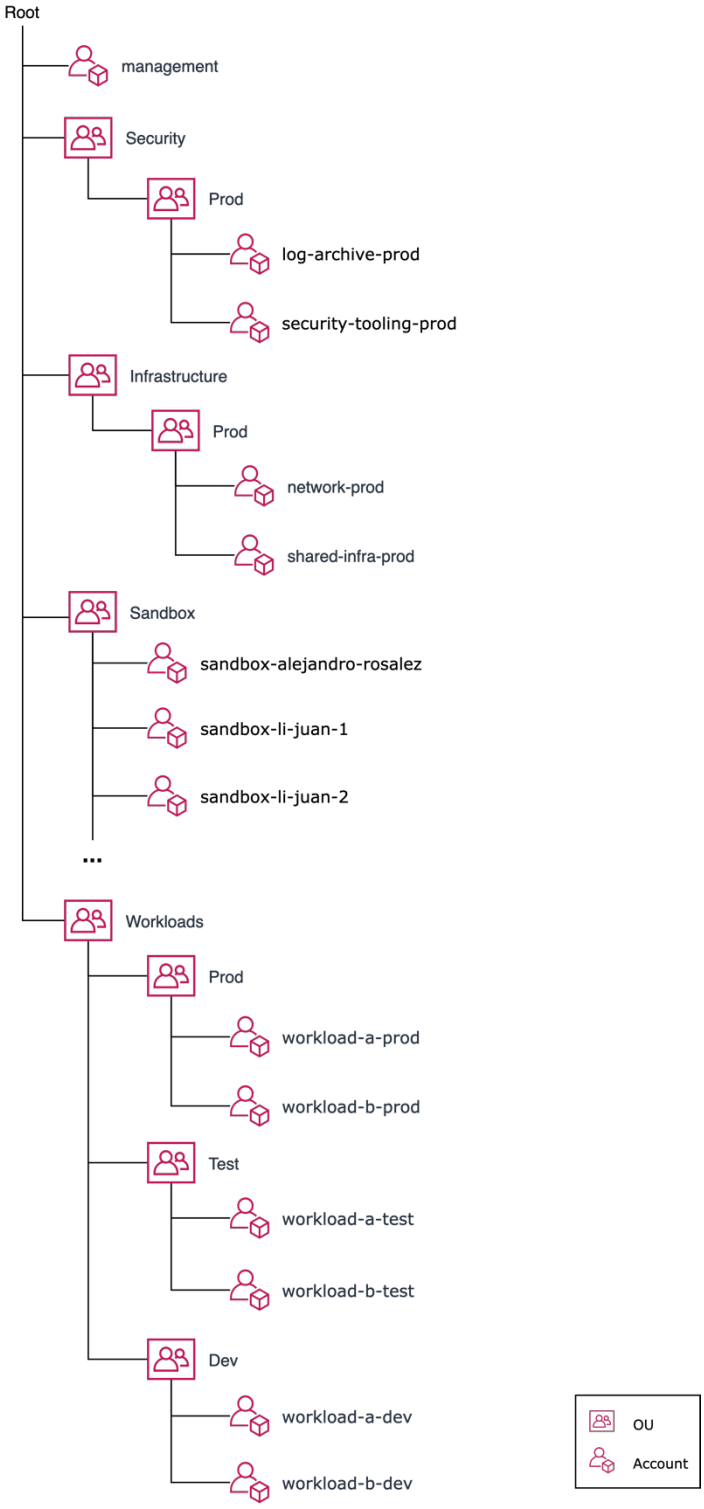


Example basic organization with multiple workloads

Basic organization with infrastructure services

This example includes support for common infrastructure resources. The additions include:

- An Infrastructure OU (p. 22) containing a Prod OU
- A *network-prod* account to contain resources required to connect VPCs in the workload accounts to your on-premises network. For example, AWS Transit Gateway, AWS Site-to-Site VPN, and AWS Direct Connect resources
- A *shared-infra-prod* account to contain common shared infrastructure services to be used by other accounts. For example, Amazon Route 53 resolver endpoints.

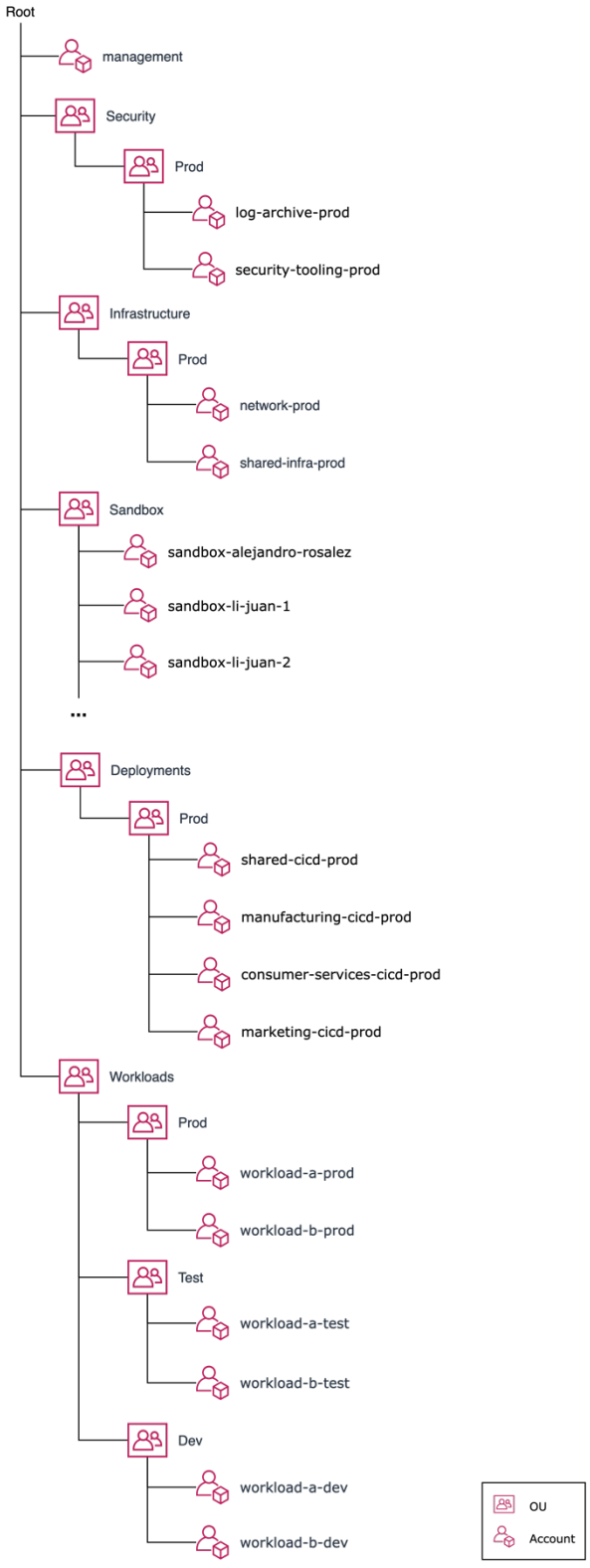


Example basic organization with infrastructure services

Basic organization with CI/CD as a separate function

This example incorporates support for CI/CD resources that are used to validate changes to their respective workloads and automate deployments to the test and production workload environments. The additions include:

- A Deployments OU (p. 30) and a child Prod OU
- A set of *workload-a-cicd-prod* and *workload-b-cid-prod* accounts to contain the production quality CI/CD resources for each of the respective workloads

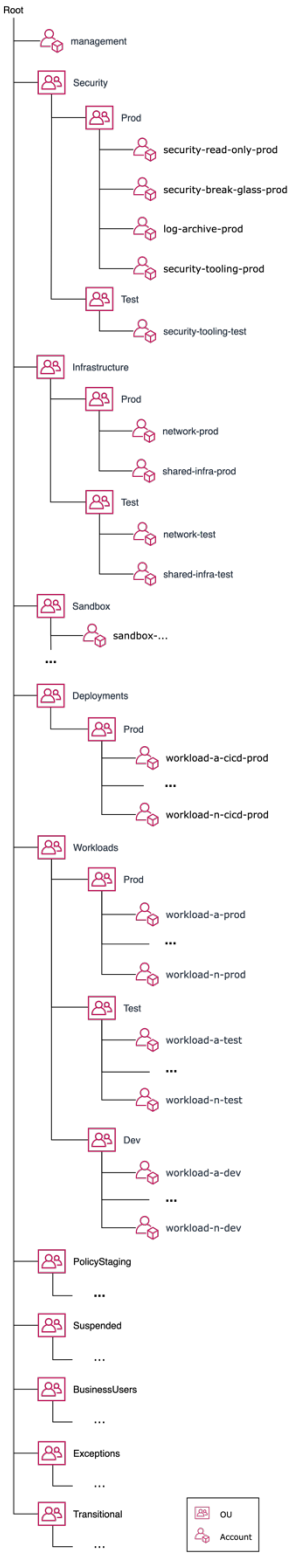


Example basic organization with CI/CD

Advanced organization

This example represents all of the recommended OUs and a greater number and diversity of workloads.

Additionally, Test child OUs are represented under the Security and Infrastructure OUs to support test environments for the security and infrastructure workloads.



Example advanced organization

Conclusion

If you are in the early stages of adopting AWS, you can use these best practices to start implementing an AWS environment structure that is sufficient to meet your initial needs. As your adoption of AWS expands and your requirements increase, you can be confident that your AWS environment can be expanded to meet those needs without requiring significant restructuring.

If you already have an AWS environment in place, you can use these best practices to assess its current state. By doing so, you can determine if you're fully realizing the benefits of using multiple OUs and AWS accounts and, if necessary, you can make plans to enhance your current environment.

In either case, your AWS sales team and AWS Partner Network (APN) Partners are ready to help you apply these best practices to meet your business needs.

Implementation

The earlier sections of this guide covered the benefits of using multiple accounts to organize your AWS environment, this section dives deeper into some implementation options you can use to build your multi-account environment.

Your environment may start with a small number of accounts, and as the number of accounts in your environment increase, you will need automations to manage your cloud environment. These services offer features to help you manage your multi-account environment at scale.

Topics

Getting started with your multi-account Environment

New Customers

Managing your own operations

When starting on AWS you should evaluate your business goals and determine your future operating model, and make a decision on which technology to use when building your multi-account environment. If you are planning on managing your own environment, start by evaluating AWS Control Tower.

AWS Control Tower is a service built based on the guidance included in this paper, making it the preferred solution for new-to-the-cloud customers looking for a service-managed environment. AWS Control Tower offers a simplified experience and automatically deploys your initial environment, helping you manage the multi-account environment efficiently, leveraging other AWS services. For a complete list of these services, see the AWS Control Tower user guide.

Operations managed by AWS or AWS Partners

AWS and Partners can help you operate, update, monitor, or deploy your environment and your workload infrastructure, with AWS Managed Services and AWS Partners have offerings that can help with your operations.

Hybrid operations (mix of customer and AWS managed)

If your portfolio includes different operating models, you will need to consider a combination of the approaches described above. Proving the flexibility to decide what type of operations you will manage on your environment, and which operations you would like to be overseen. For example, you could manage the account vending process and the environment set up using AWS Control Tower, and leverage Amazon Managed Services for managing the operations related to moving to the cloud.

Existing Customers

If you are currently operating your multi-account environment effectively with AWS Organizations, we recommend you continue using your current approach. Monitor the guidance on the paper, the

documentation for the services you are currently using. Implementation and operational effectiveness should continue to operate their multi-account environment with AWS Organizations. Monitor new feature releases for additional benefits to your customers, as AWS Organizations continues to make it easier for customers to centrally manage and govern their AWS resources.

AWS offers services to manage your multi-account environment, AWS Control Tower can be deployed on your current organization. AWS Control Tower will add automations to manage your multi-account environment. Review the AWS Control Tower documentation for extending governance. If you are looking for a fully managed environment, including operational capabilities, consider using AWS Managed Services.

Services to help you implement your multi-account environment

AWS Organizations

AWS Organizations provides the underlying infrastructure and capabilities for customers to build and manage their multi-account environments. Using AWS Organizations, customers can automate AWS account creation and management; govern access within the organization to AWS services, resources, and region using preventative guardrails; centrally manage policies across multiple AWS accounts (SCPs, Tag Policies, AWS Backup, ML opt-out); configure multi-account capabilities for AWS services (such as AWS Config, AWS CloudTrail, AWS CloudFormation, GuardDuty, Amazon Macie, and SSO); share resources across accounts; and consolidate their bill.

AWS has the following resources available for help you establish your multi-account environment using AWS Organizations:

- AWS Organizations features
- Organizations supported multi-account services
- Organization Quotas

AWS Control Tower

AWS Control Tower provides a simplified way to set up and govern a secure, multi-account AWS environment based on the guidance in this paper. AWS Control Tower automates the creation of your multi-account environment using AWS Organizations, instantiating a set of initial accounts and with some default guardrails and configurations for the environment. Although AWS Control Tower reduces flexibility, it also provides automations to manage your cloud environment efficiently.

> **Note**
> The OU structure that AWS Control Tower initially deploys is slightly different from the guidance in this paper, please review AWS multi-account environment with Control Tower (p. 83) for a detailed implementation and mapping between the Control Tower implementation and the guidance offered in this paper.

AWS has the following resources available for help you establish your multi-account environment using AWS Control Tower:

- Appendix E: How does AWS Control Tower establish your multi-account environment? (p. 83)
- Getting started with Control Tower
- Control Tower Quotas

AWS Managed Services

AWS Managed Services (AMS) leverages AWS services and a growing library of automations, configurations, and run books, to provide an end-to-end operational solution for both new and existing AWS environments. AMS covers the people element of operating the technology that AWS services provide.

For addtional information, see AWS Managed Services features .

Contributors

Contributors to this document include:

- Paul Bayer, Principal Consultant, Amazon Web Services
- Sam Elmalak, Principal Solutions Architect, Amazon Web Services
- Ilya Epshteyn, Senior Manager, Identity Solutions, Amazon Web Services
- Christopher Kampmeier, Senior Solutions Architect, Amazon Web Services
- Tomas Riha, Senior Solutions Architect, Amazon Web Services
- Dave Walker, Principal Solutions Architect, Security and Compliance, Amazon Web Services
- Brandon Wu, Senior Security Solutions Architect, Amazon Web Services
- Nathan Case, Security Strategist, Amazon Web Services
- Alex Torres, Solutions Developer, Amazon Web Services

Additional resources

For additional information, see:

- Management and Governance on AWS
- Security, Identity, and Compliance on AWS
- AWS Best Practices for Security, Identity, and Compliance
- AWS Well-Architected (p. 62)
- AWS IAM Permissions Guardrails

Appendix A: Relation to AWS Well-Architected

AWS Well-Architected helps cloud architects build secure, high-performing, resilient, and efficient infrastructure for their applications and workloads. Based on five pillars — operational excellence, security, reliability, performance efficiency, and cost optimization — AWS Well-Architected provides a consistent approach for customers and partners to evaluate architectures, and implement designs that can scale over time.

The best practices for organizing your AWS environment addressed in this guide augment and support the best practices represented in the following sections of the Well-Architected pillars.

Topics

Operational Excellence Pillar

- Organization - Operating Model – Use of multiple AWS accounts and OUs enable you to support multiple operating models within a common overall AWS environment.
- Prepare – Mitigate Deployment Risks – Use of separate AWS accounts for deployment-oriented operations help secure those operations and isolate them from the targeted test and production workload environments.

Security Pillar

- Security Foundations - AWS Account Management and Separation – Use of separate accounts for your test and production workloads helps maintain isolation between those environments.
- Identity and Access Management – Identity Management – Use of a centralized identity provider and federated access helps you more efficiently manage human access across your accounts.
- Identity and Access Management – Permissions Management – You can define permission guardrails for your AWS environment and provide least privilege access to identities that need access to your accounts.
- Detection - Configure – Your security operations team can benefit from the centralization of logs generated across your accounts in support of analysis and detection requirements.

Reliability Pillar

- Foundations - Manage Service Quotas and Constraints – By isolating workloads in their own accounts, you can more easily manage service quotas for those workloads.

Cost Optimization Pillar

- Practice Cloud Financial Management – Cost-Aware Processes – You should build in cost awareness from the start of your cloud adoption journey.
- Expenditure and Usage Awareness – Governance – Your account structure can help you isolate different workloads for fiscal and billing purposes.
- Expenditure and Usage Awareness – Monitor Cost and Usage – Since costs are allocated by default at the account level, you can distinguish costs across accounts.

Appendix B: Worksheet for mapping workload environment purposes to hosting environment types

You can use the worksheet in this appendix to help you determine where workload environments for a given purpose will be deployed in your overall AWS environment. This worksheet, the worksheet in Appendix C (p. 71), and the Organizing workload-oriented OUs (p. 37) section are intended to be used together.

Combined, these worksheets and guidance can help your team quickly iterate on and arrive at an informed view of your own approach to organizing your workload-oriented OUs. They can also help inform how you might identify and scope your workload-oriented accounts.

For example, as you complete the worksheets in Appendix B and Appendix C, you should gain a better understanding of:

- How you expect to position AWS sandbox environments in relation to work done on your corporate desktops and in your AWS development environments.
- How you expect to generally divide and position your set of AWS development, test, and production environments in relation to your SDLC. Coupled with these best practices, this knowledge should help you refine your set of workload-oriented OUs and help inform the overall scoping of your workload-oriented accounts.

Topics

Use the results for internal documentation

Later, as you establish and expand your standards documentation for using AWS, you might find value in publishing the resulting customized table and descriptions internally. Your teams might find value in being able to quickly understand where workload environments for a given purpose are meant to be positioned in your AWS environment.

How to use this worksheet

1. Review Descriptions of example purposes of workload environments (p. 65) and Description of example types of workload hosting environments (p. 67). Note how your current on-premises

standards and your initial expectations for working in AWS are similar to and different from the examples.

2. Review the Example worksheet (p. 68) to gain a sense of typical mappings of workload environment purposes to workload hosting environment types. Note where your expectations align with and differ from the examples. Mark where you expect to position workloads of a given purpose.
3. Make a copy of the Empty worksheet (p. 69), expand and modify the table by adding, removing, and/or changing rows and columns based on your needs. You can revisit and refine your initial assignments after you complete the worksheet in Appendix C.
4. Modify the example descriptions to suit your needs.
5. Use Appendix C (p. 71) to complete a related worksheet to help you document the key attributes of each of your own list of workload hosting environment types.
6. Update and refine the table and descriptions, based on your initial review of both this worksheet and the worksheet in Appendix C (p. 71).

Descriptions of example purposes of workload environments

This section provides descriptions of each of the example purposes of workload environments shown in the Example worksheet (p. 68). These examples and descriptions are meant to act as a reference and starting point for you to modify and extend to suit your needs.

Self-paced learning and experimentation workload environments

Builders learning and experimenting on their own, ideally with access to a wide variety of technologies.

Development workload environments

Work done by builder teams up and down the stack to develop and support their workloads and services. Examples of code includes proprietary application code, test scripts, test data, data models, machine learning models, security roles and policies, and Infrastructure as Code (IaC). Generally, it's most efficient for builders to carry out development tasks locally on their corporate desktops.

Static analysis, build, and unit testing workload environments

Teams naturally develop, test, and carry out static analysis, builds of draft artifacts, and unit tests in their corporate desktop and/or cloud development environments. More formally controlled executions of these processes including builds of formal candidate artifacts typically occur through either continuous integration (CI) jobs or in early CI stages of continuous delivery (CD) pipelines.

CI jobs and CD pipelines workload environments

CI jobs and CD pipelines are developed and tested by teams in their development environments before they are potentially formally validated in a test environment. Given the common requirements to formally control creation of candidate artifacts and orchestration of validation stages, formal execution of CI jobs and CD pipelines typically occurs in production environments.

Smoke testing workload environments

Smoke testing is often used to quickly determine the overall success or failure of either deploying or releasing a change to any workload environment. A smoke test is an initial indication of the success or failure of a deployment or release. Unlike many other types of system level testing, smoke testing is also typically used against production workloads to initially validate that release of a change was generally successful.

Development integration testing workload environments

Development integration testing is the early integration testing of code and configurations that typically occur prior to formal types of system testing. This is done so that basic integration issues are detected early in the lifecycle.

Production-like system testing workload environments

System level testing is the end-to-end testing of services and applications that occur in production-like environments prior to changes being promoted for use in production. In the spirit of *shifting left* so that issues are found earlier in the lifecycle, builder teams can take advantage of automation and the on-demand and elastic nature of the cloud to perform early system level testing in their development environments.

Common examples of system level testing include, but are not limited to:

- **Functional acceptance testing** – Verification that the service meets the business requirements. In mature situations, this testing is often largely automated.
- **User acceptance testing** – Validation from a user's perspective or through a proxy, such as a product owner, that the service meets their expectations.
- **Exploratory testing** – Functionality testing by humans in an ad hoc manner.
- **Performance testing** – Systems level testing to ensure a service is able to meet the expected performance goals with the supporting capacity.
- **Workload-recovery testing** – Testing the ability of your workloads to recover from component-level and workload-level failures in the face of sustained failures.
- **Penetration testing** – Also known as *pen* testing, testing services to identify security vulnerabilities.
- **Multi-region testing** – Testing that a workload is deployed in an active-active mode across multiple AWS Regions.
- **Disaster recovery / business continuity testing** – Business testing to ensure that services can be restored when in the event of a large-scale disaster.

Stable shared test workload environments

Teams who own shared services are sometimes responsible for managing stable test instances of their workloads that are loaded with test data in support of development and integration testing.

Resiliency testing workload environments

Also known as *chaos engineering*, this scenario is where failures are purposely introduced into the environment to ensure that the workloads respond in the expected manner. More advanced scenarios include running such tests against production workloads.

Demo workload environments

Internal teams demonstrating services and capabilities to internal and/or external customers throughout the lifecycle.

External pre-release workload environments

Sometimes referred to as *alpha*, *beta*, or *early access*, these workloads are typically accessed by external customers. Given the external access nature of these workloads, it's common for these workloads to be contained in either production or test environments.

Production workload environments

Formally managed and monitored workloads that are deployed to distinct production environments.

Descriptions of example workload hosting environment types

The following descriptions provide more detail for the example workload hosting environment types. These examples and descriptions are meant to act as a reference and starting point for you to modify and extend depending on your needs.

Each workload hosting environment type is not intended to represent a single account in your overall AWS environment. Rather, a workload hosting environment type is meant to help you think about the overall categorization of and distinction between the accounts in which your workloads reside. You may end up having numerous accounts of any given hosting environment type.

For example, if there are significantly different security and operations needs of the environment types that you identify, then those environments might help you refine your overall OU structure for workloads.

For a set of example attributes for each of the example hosting environment types, see Appendix C (p. 71).

Corporate desktop environments

Depending on the degree of access your builders have on their corporate desktops, they might perform much of their day-to-day, non-cloud work locally on their desktops. As you adopt AWS, many of your builder teams will start creating and managing AWS resources and workloads in their sandbox and development AWS environments.

Sandbox environments

Sandbox environments are environments allocated to individuals in which they have significant degrees of administrative access so that they can learn and dive deep into a wide variety of AWS services and other technologies. Typically, sandbox environments do not have access to your internal networks, services, and data.

Foundational guardrails are used to mitigate the risk of this extensive access. For more information, see Sandbox OU (p. 23).

Development environments

An extension of your corporate desktops that enable your builder teams to carry out formal development tasks both with and in AWS that cannot occur only on their local corporate desktops.

Development environments are often allocated on a team and/or individual basis. Depending on the permissions granted to your builder teams, varying degrees of self-paced learning, experimentation, and prototyping can also occur in development environments.

Foundational guardrails and shared infrastructure services that are treated as production stable resources are typically used to ensure the proper degrees of access control and stability of development environments.

Data-oriented development environments

A hybrid of the attributes of development and production environments. Data scientists and data engineers need an environment in which they can develop and perform early forms of testing machine learning and other algorithms that require access to production data.

Given the hybrid nature of this environment type, the guardrails and controls for data development are typically a blend of those used to support development and production environments.

Test environments

A type of workload hosting environment in which changes are formally validated before they are promoted for release to production environments. Depending on how you view this overall stage of your SDLC and the distinct security and operational requirements of it, you may end up dividing this overall type into multiple types of environments.

Typically, test environments are secured and managed in much the same manner as production environments so that validation efforts occur in production-like environments and issues are detected prior to release to production.

Production environments

Highly secured, formally managed hosting environments in which production data and workloads reside.

Example worksheet

The following example worksheet represents a typical mapping of workload purposes to types of workload hosting environments. This example is meant to spur discussion of and comparison to both your current on-premises environments and your expectations for working on AWS.

Workload Environment Purpose	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Self-paced Learning / Early Experimentation	•	•	•	•		
Development	•		•	•		

Workload Environment Purpose	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Static Analysis, Building, Unit Testing	•		•	•		
CI Jobs and CD Pipelines			•	•	•	•
Smoke Testing			•	•	•	•
Development Integration Testing			•	•	•	
Production Like System Testing			•	•	•	
Stable Shared Test			•	•	•	
Resiliency Testing			•	•	•	•
Demo	•	•	•	•	•	•
External Pre-Release (Alpha, Beta)					•	•
Production						•

Empty worksheet

Use this worksheet to help you describe your expectations for where certain purposes of workloads are expected to occur across your AWS workload environments. Copy and modify the rows and column headers in the following worksheet as needed.

Workload Environment Purpose	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Self-paced Learning/ Early Experimentation						
Development						
Static Analysis,						

Workload Environment Purpose	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Building, Unit Testing						
CI Jobs and CD Pipelines						
Smoke Testing						
Development Integration Testing						
Production Like System Testing						
Stable Shared Test						
Resiliency Testing						
Demo						
External Pre-Release (Alpha, Beta)						
Production						

Appendix C: Worksheet for identifying attributes of workload hosting environments

Use the following worksheet to help you identify and refine your expectations for common security and operational attributes across your set of AWS environments. This worksheet helps you plan how to structure your overall AWS environment based on your requirements.

Topics

How to use this worksheet

1. Follow the instructions in Appendix B: Worksheet for mapping workload environment purposes to hosting environment types (p. 64).
2. Review Descriptions of example attributes (p. 71) located after the worksheets below. Note how your current on-premises standards and your initial expectations for working in AWS are similar to and different from the example attributes.
3. Review the Example worksheet (p. 76) example attributes to get a sense of typical attributes of workload hosting environment types. Note where your expectations align with and differ from the examples. Modify the example attributes in each cell based on your expectations.
4. Make a copy of the Empty worksheet (p. 79). Expand and modify the table by adding, removing, and/or changing rows and columns based on your needs.
5. Mark where you expect to position workloads of a given purpose. Include learnings from the completion of the worksheet in Appendix B (p. 64).
6. Modify the example attribute descriptions to suit your needs.
7. Iterate on the worksheet from Appendix B (p. 64) based on the outcome of completing the following worksheet.
8. Publish and share internally a draft of the resulting tables and descriptions so that your cross-functional teams can further evolve them.

Descriptions of example attributes

The following descriptions elaborate on the meaning of each of the example attributes of workload hosting environment types. These examples and descriptions are meant to act as a reference and starting point for you to modify and extend to suit your needs.

Note: In the table of example attributes of workload hosting environment types, you'll see a close alignment between test and production environment types. This is a typical approach used by organizations to help ensure that testing occurs in production-like environments prior to changes being promoted to production environments.

If this degree of alignment is important to you, then it can factor into how closely you align your production OU to test environment OUs, both in terms of SCPs and overall security baselines.

Owners/Tenants

These are the teams or individuals accountable for the environment and the workloads deployed to it.

Sandbox, Development, and Data-oriented Development

Although sandbox environments are typically allocated on an individual basis and development environments are often allocated on a team basis, your requirements might vary. For example, in support of short duration events, such as company hackathons, you might need to allocate sandbox environments to teams. Similarly, you might have the need for individual owned development environments if team owned environments don't meet your needs.

Regardless of whether a team or an individual owns an environment, you should consider applying virtually the same attributes across a given type of environment.

Test and Production

The ownership and tenancy model typically depends on the operational model you use. For example, in a more traditional centralized operational model, your operations team might own the production and test environments and possibly the workloads residing in them.

In a more federated DevOps style operating mode, individual application and data services teams might own their own production and test environments and the workloads residing in them.

Tolerance for extended outages

This refers to the tolerance of the business for extended outages of a given type of environment. For example, if access to a particular type of environment is down for several hours, what's the impact to the business?

Generally, organizations treat the environments in which they perform their formal day-to-day work as needing to have production qualities in terms of stability and access.

Corporate Desktops

An inability to use corporate desktops for any appreciable length of time is considered to be a significant impact in most organizations.

Sandbox

In our example definition of sandbox environments, they are not typically viewed as having a great deal of importance to everyday formal tasks.

Development and Data Oriented Development

To the extent that development and early testing of code for a variety of foundation and business workloads occurs through development environments, extended outages of development environments can have a significant impact on your business.

Test

Similar to development, your ability to validate important changes will be severely impacted due to extended outages of your test environments.

Internet access

This refers to the extent to which access to and from the internet is allowed.

Sandbox and Development outbound access

Typically, users of sandbox and development environments benefit from the ability to download packages from the internet. For example, access to public language platform and OS-native package repositories, including npms, rpms, PyPi modules, etc. Additionally, sandbox and development environments typically benefit from having access to public web services to support experimentation, development, and early testing.

In either case, you might implement filters on outbound requests to the internet to reduce the risk of accessing malicious or unauthorized software packages and services.

Sandbox and Development inbound access

Sandbox environments might have the ability to contain workloads that are accessible from the internet. Given that they contain internal data and Intellectual Property (IP), inbound access from the internet is not typically allowed for development environments. In this respect, your AWS development environments may be similar to your corporate desktops.

Test and Production internet access

Outbound and inbound internet access with test and production environments is typically more controlled than in lower environments. For example, you may require defined connectivity for each workload that performs outbound requests to the internet or receives inbound requests.

Internal network access

This refers to the extent to which workloads in an environment are allowed to connect to other internal services whether they reside in your on-premises or on AWS environments.

Sandbox

Given the broad administrative access and bidirectional connectivity with the internet, sandbox environments are typically inhibited from accessing internal data and services.

Development

Development environments, like corporate desktops, are typically allowed to connect to your source code management, artifact management, CI/CD, and deployment/release automation services. These environments also typically have connectivity to shared infrastructure services, such as DNS resolution and user authentication and identity management instances populated with test data.

Data

This refers to the overall type of data allowed in each environment.

Sandbox

Unlike corporate desktops and development environments, sandbox environments are typically intended to house only public data. Due to the risk of data loss and unintended exposure, intellectual property (IP) and internal data are typically not allowed in sandbox environments.

IP often includes proprietary source code, configuration data, artifacts (such as applications and binary packages), non-public test data, business, and operational data.

Development and Test

Use of sensitive production data is typically prohibited in these environments.

Data Oriented Development

A variation of a development environment is one that supports the needs of data scientists and data engineers. These teams typically need direct human access to cloud environments to develop and iterate on machine learning (ML) models. Because these workloads need access to production sensitive data, additional guardrails and controls are typically applied to these forms of development environments.

Third-party software and cloud services

This refers to the extent to which third-party and overall cloud services are allowed in each environment. Typically, your existing standards for acceptable use of third-party software and cloud services should extend to your AWS environments.

Degree of access

This refers to the extent to which people interacting with an environment have access to resources in the environment.

Foundation team development

Given the nature of their development work, your cloud foundation or cloud platform team of cross-functional network, security, and other responsibilities typically needs greater write access to foundational resources in their development environments.

Lifespan of resources

This refers to how long workloads and supporting resources are expected to live in a given environment. Even in development environments, teams should be encouraged to automate important configurations as much as feasible. Automation enables teams to recreate resources and workloads on demand so that they have less dependency on hand-built configurations.

Direct human write access to workload resources

This refers to whether human users have the ability to directly create, update, and delete workload resources.

Corporate Desktop, Sandbox, and Development

Across corporate desktop, sandbox, and development environments, individuals and team often need to perform direct manual manipulation of cloud resources in early stages of experimentation. Later, as certain cloud resource and workload configurations mature, investment in automation often makes sense.

Production

In support of workloads in production environments, as a general recommendation, changes made to production should be automated. In these cases, human user write access to workload resources should not typically be necessary and not allowed.

Direct read access to production resources and data could be allowed on a least privileged basis to support audit and operational troubleshooting scenarios.

Break-glass scenarios should be supported to enable temporary write access with strict auditing under exceptional conditions.

Automated workload provisioning

This refers to the extent to which automated workload provisioning applies to a type of environment.

Corporate Desktops

A modest degree of automated workload provisioning can be developed and tested on local corporate desktops. To the extent that type 2 hypervisors for running virtual machines locally and/or containers are supported on corporate desktops, builders can carry out some degrees of environment automation testing on their corporate desktops. However, unless an AWS service API is available locally, workload automation that depends on AWS service APIs needs to occur in their AWS development environments.

Sandbox

Because sandbox environments are not typically connected to your source code management systems and don't usually contain internal data, there might be limited value in developing workload automation in sandbox environments. Reuse of open source or other third-party automation is typical in sandbox environments.

Development

Teams typically use a mix of manual and automated means to initially experiment and perform early iterations on their cloud resource configurations. Once teams mature their desired configurations, they typically invest in infrastructure as code (IaC) to automate workload configurations. Your standard source code management systems are typically used to house the IaC automation files.

Formal change management for workloads

This refers to whether formal change management processes are required to apply changes to workloads.

Corporate Desktop, Sandbox, and Development

Formal change management does not typically apply to workloads contained in these environments.

Test

Minimally, we recommend that you at least test your change management processes in test environments.

Production

Typically, some form of change management process applies to all changes made to workload in your production environments.

Degree of centrally managed foundation

This refers the extent to which there's a centrally managed set of foundation resources in a given environment.

Sandbox

In sandbox environments, given the experimental and wide-ranging administrative access, there may be little if any centrally managed foundation resources other than common enterprise guardrails (see next section).

Development

In development environments, you can provide centrally-managed foundation services to teams. By doing so, you can remove undifferentiated heavy lifting from their day-to-day work.

Test and Production

Similar to development, you can relieve individual teams from needing to manage underlying foundation services.

Common enterprise guardrails

Across all of your AWS environments, you'll benefit from applying a set of enterprise guardrails. The makeup and depth of these guardrails may vary across your types of environments depending on the level of risk to the business and the degree of access allowed to owners of the environments.

For example, AWS API logging via AWS CloudTrail and cloud resource configuration recording via AWS Config is typically a common guardrail applied across all environments.

Example worksheet

This example worksheet represents a typical mapping of attributes to types of workload hosting environments. This example is meant to spur discussion and comparison to both your current on-premises conventions and your expectations for working on AWS.

Example Worksheet - Attributes of Workload Hosting Environment Types

Attribute	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Owners / Tenants	Individual	Individual	Team	Team	Same as Production	Depends on operating model
Tolerance to Extended Outages	Low	High	Low to Medium	Low to Medium	Low	Extremely Low
Internet Access	Outbound requests subject to proxying and filtering controls No inbound requests	Outbound and inbound requests	Outbound requests subject to proxying and filtering controls No inbound requests	Given the presence of production data, outbound requests will likely be more controlled	Same as Production	Workload-specific outbound and inbound requests Proxy-based access to

Attribute	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
				than development environments. No inbound requests		external services
Internal Network Access	Shared development and infrastructure services Other development workloads Corporate services	No connectivity to corporate and data center services No connectivity to shared development and infrastructure services	Shared development and infrastructure services Other development workloads No access to business production services	Shared development and infrastructure services Access to defined production data sources	Shared development and infrastructure services Other test services	Shared infrastructure services Other production services
Data	Intellectual property (IP) Test data	Public data only Public test data (no Intellectual property)	Intellectual property (IP) Test data No access to production data	Intellectual property (IP) Production data	Intellectual property (IP) Test data Typically avoid use of production sensitive data unless sanitized	Intellectual property (IP) Production data
Third-Party Software and Cloud Services	Installation of approved software	Access to broad set of AWS services Installation of approved software	Access to enterprise standardized AWS services Controlled access to AWS services undergoing standardization for the purposes of testing Installation of approved software	Access to enterprise standardized AWS services Controlled access to AWS services undergoing standardization for the purposes of testing Installation of approved software	Access to enterprise standardized AWS services Access to AWS services undergoing standardization Installation of approved software	Access to enterprise standardized AWS services Installation of approved software.

Attribute	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Degree of Access	Limited OS configuration	Wide ranging administrative cloud resource write access Some limits of modifying foundation resources	Wide ranging access including write access to develop and test workload-specific IAM service roles and policies Some limits of modifying foundation resources	Given the presence of production data, likely more limited access to cloud resource write access than in development	Same as Production	Least privileged access Strictly controlled access based on operating model that is in effect Service-to-service access based on authorization.
Lifespan of Resources	Up to builder to manage	Temporary	Up to owning teams to manage	Up to owning teams to manage	Same as Production	Depends on business need
Direct Human Write Access to Workload Resources	Yes	Yes	Yes	Yes	Same as Production	No
Automated Workload Provisioning	Limited	Limited	Mix of manual and automated	Mix of manual and automated	Same as Production	Yes
Formal Change Management for Workloads	No	No	No	No	Same as Production	Yes
Degree of Centrally Managed Foundation	As appropriate for corporate desktops	Sufficient to ensure overall security	Typical foundation resources centrally managed	Typical foundation resources centrally managed	Same as Production	Typical foundation resources centrally managed

Attribute	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Common Enterprise Guardrails	Desktop specific	Yes Guardrails to prevent write access to baseline security monitoring services and configuration	Yes Guardrails to prevent write access to foundation resources	Yes Guardrails may be a hybrid of those used for development and production environments	Same as Production	Yes

Empty worksheet

Copy and modify the rows and column headers in the following worksheet and note how each attribute relates to a given environment type.

Attributes of Workload Hosting Environment Types

Attribute	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Owners / Tenants						
Tolerance to Extended Outages						
Internet Access						
Internal Network Access						
Data						
Third-Party Software and Cloud Services						
Degree of Access						
Lifespan of Resources						
Direct Human						

Attribute	Corporate Desktops	Sandbox	Development	Data-oriented Development	Test	Production
Write Access to Workload Resources						
Automated Workload Provisioning						
Formal Change Management for Workloads						
Degree of Centrally Managed Foundation						
Common Enterprise Guardrails						

Appendix D: Multiple AWS Regions

If you plan to use multiple AWS Regions, keep the following considerations in mind as you design your overall AWS environment.

Topics

Geographic scopes of data protection

If you use different AWS Regions that are in the same geographic scope defined by the data protection requirements applicable to your workloads, you can use the same IAM IdP(s) to federate to all accounts in live, disaster recovery, or load balanced live environments. You can replicate databases between environments using appropriate mechanisms, such as Amazon DynamoDB global tables or Amazon RDS read replicas. In such circumstances, it is also possible for you to distribute core elements of your foundational AWS environment such that the log archive bucket is in one Region and assets in other accounts in other Regions log cross-Region to it.

You should carefully consider whether the data protection requirements applicable to your workload differ across countries, or are subject to data sovereignty requirements or export control. This may impact your ability to make cross-Region data transfers. (Note that cross-Region data transfers incur networking costs.)

Performance considerations

There are also performance considerations to keep in mind for certain workloads. Some services are by their nature per-Region, which makes it more sensible for you to deploy such workloads with all assets in the same Region. For example, AWS KMS keys cannot be exported from a Region, and use of a KMS key in another Region is likely going to add latency to an application. We therefore recommend using AWS KMS in the same Region, unless specific governance policies, regulatory or corporate, mandate otherwise.

Close collaboration between your security and architecture teams and your workload owning teams is important to properly using KMS. Your design of how Amazon S3 objects, EBS volumes, and other data are encrypted and potentially replicated across Regions should factor in low latency when required.

Where cross-account replication of these assets is required, Amazon S3 Cross-Region Replication (CRR) enables on-the-fly re-encryption of an object with an AWS KMS key in the destination Region. Multi-Region duplication of AWS KMS keys for the decryption of cross-Region copied EBS volumes can be achieved using the techniques covered in Busy Engineer's Document Bucket.

Log management

When logs are generated, we recommend that you implement secondary controls to filter them before they are passed outside a compliance scope boundary associated with an account, or are passed cross-

Region. If your logs contain sensitive data, this approach helps ensure that such sensitive data cannot escape your defined compliance scope boundary using AWS logging capabilities.

Although AWS CloudTrail has built-in cross-account logging capability and AWS Config can aggregate configuration and compliance data across accounts and Regions, it might be more appropriate for you to aggregate logs in an account. You can use AWS Lambda functions or similar to filter the logs before sending them to another Region for aggregation into a multi-Region logging archive.

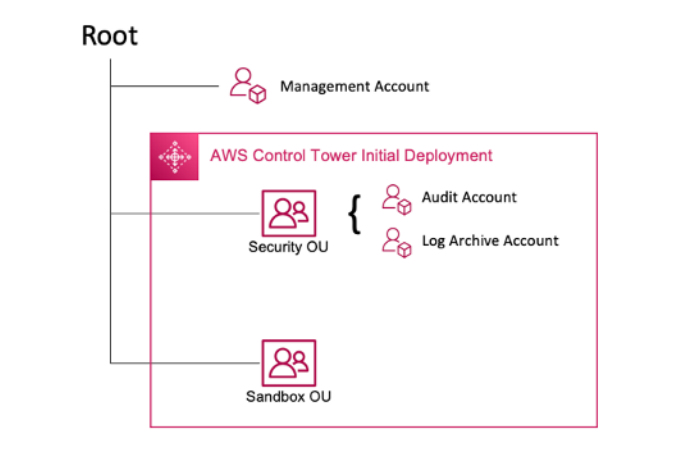
Appendix E: How does AWS Control Tower establish your multi-account environment?

Topics

Establish your multi-account environment with AWS Control Tower

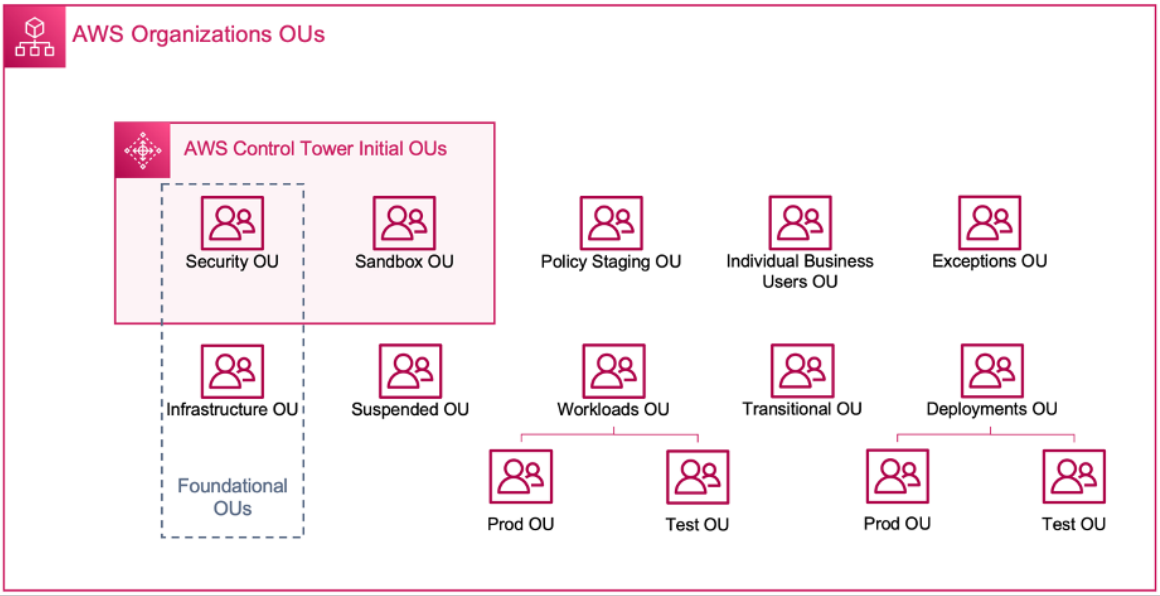
When you set up your multi-account environment using AWS Control Tower, it creates two OUs.

- **Security OU** - Within this OU, AWS Control Tower creates two accounts:
  - Log Archive
  - Audit *(This account corresponds to the Security Tooling account discussed previously in the guidance.)*
- **Sandbox OU** - This OU is the default destination for accounts created within AWS Control Tower. It contains accounts in which your builders can explore and experiment with AWS services, and other tools and services, subject to your team's acceptable use policies.



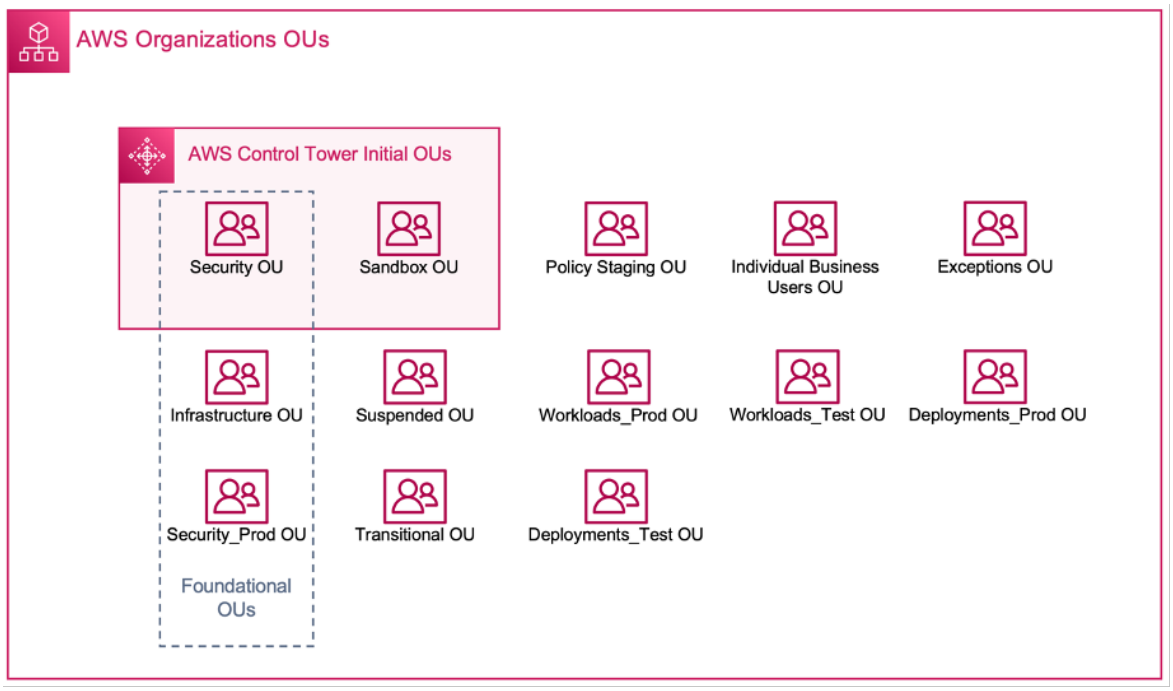
AWS Control Tower allows you to create, register, and manage additional OUs to expand the initial environment to implement the guidance.

The following diagram shows the OUs initially deployed by AWS Control Tower. You can expand your AWS environment to implement any of the recommended OUs included in the diagram, to meet your requirements.



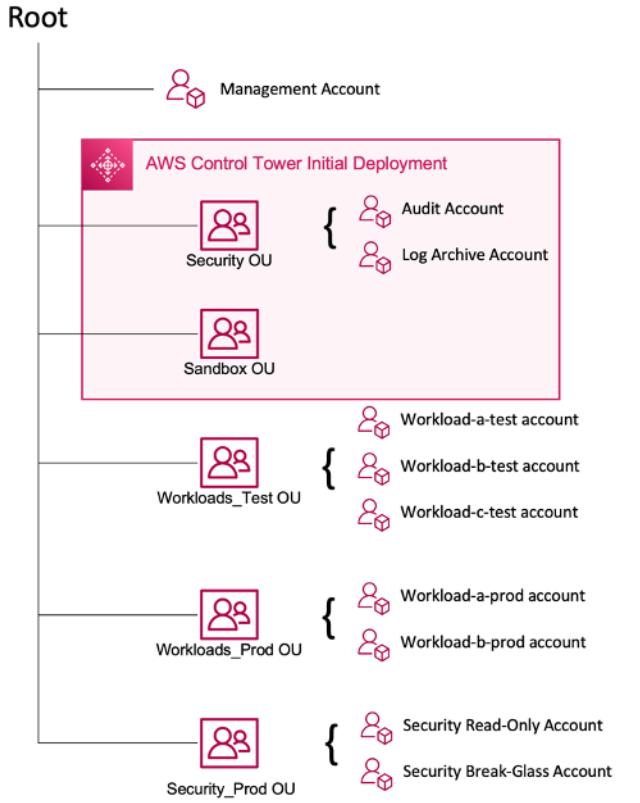
Example: Workloads in a flat OU structure

AWS Control Tower supports a flat OU structure. To implement the OU structure described earlier in the guidance, we recommend the naming convention of including an underscore ("_") in the names of OUs that are nested. Here is a possible implementation of a flat OU structure.



In AWS Control Tower, the Security Read-Only account and the Security Break-Glass account should be placed in a secondary Security OU (Security_Prod), which must be registered with AWS Control Tower. A secondary Security OU (Security_Prod) is needed because AWS Control Tower does not permit additional accounts to be added to the primary Security OU.

In the following diagram, you will see how AWS recommends the categorization of accounts within the Workloads_Prod and Workloads_Test OUs, as well as in the Security_Prod OU, to accommodate the flat OU structure in AWS Control Tower.



Next Steps for setting up your multi-account environment

To get started with AWS Control Tower, visit the Getting Started with AWS Control Tower documentation page. We recommend that you review the pre-requisites and next steps required to establish your multi-account environment on AWS.

For complete guidance on establishing your multi-account environment, you can review the guidance included in this whitepaper (p. 1).

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

update-history-change	update-history-description	update-history-date
Updated (p. 86)	Updated guidance for establishing a multi-account environment.	July 19, 2021
Initial release (p. 86)	Whitepaper first published.	March 18, 2021

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.