ORACLE

Best Practices for Identity and Access Management (IAM) in Oracle Cloud Infrastructure

May 2023, version 2.1

Public

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Revision History

The following revisions have been made to this document since its initial publication.

DATE	REVISION
May 2023	Revised guidance for break glass (emergency) accounts
May 2022	Major update with latest changes in Oracle Cloud Infrastructure IAM service
August 2021	Updated to new template and edited
March 2018	Initial publication

Table of Contents

Overview **4**

IAM Service Components **4**

Compartments and Identity Domains **6**

Proof of Concept: Sandbox Compartment 7

Production Use 8

User Management **10**

Managing Users Locally Using the Console 10

Bringing Users from External Sources 11

Permission Management **11**

Policy Attachment and Inheritance 12

Enforce Least Privilege 14

Tag-Based Access Control 15

Admin Roles Versus IAM Policies 16

Admin Accounts **16**

Break Glass (Emergency) Account **17**

Enable Single Sign-On **17**

Enforce Multifactor Authentication and Adaptive Security **18**

Replication, Disaster Recovery, and High Availability **19**

Replication 19

Disaster Recovery 19

High Availability 19

Instance Principals and Dynamic Groups **20**

Federation **20**

Enable Password Management **21**

Conclusion **21**

Overview

This technical paper provides best practices for using the Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) service when you're planning, designing, and deploying solutions on OCI.

The IAM service lets you control who has access to cloud resources. You can control what type of access a group of users has and to which specific resources. The service enables you to enforce the security principle of least privilege by default. New users aren't allowed to perform actions on any resources until they're granted the appropriate permissions.

With the IAM service, you can use a single model for authentication and authorization across all OCI services. IAM makes it easy to manage access for organizations of all sizes—from one person working on a single project to large companies with many groups working on many projects at the same time—within a single account.

IAM is highly scalable, allowing you to manage hundreds of millions of users. It offers a robust identity-as-a-service solution for workforce, consumer, and developer use cases, with features such as strong multifactor authentication (MFA), adaptive security, identity lifecycle management, single sign-on (SSO) to third-party applications, and support for hybrid and on-premises environments.

IAM Service Components

The IAM service consists of several key components that help you control access to resources and configure identity domains for managing applications, SSO, and identity lifecycle management. This section provides basic definitions of the following IAM components.

- **Resource:** A cloud object that your organization's employees create and use when interacting with OCI. Resources include Compute instances, block storage volumes, virtual cloud networks (VCNs), subnets, and route tables.
- **Compartment:** A collection of related resources. Compartments are a fundamental component of OCI for organizing and isolating cloud resources. You use them to clearly separate resources for the purposes of measuring usage and billing, access (by using policies), and isolation (by separating the resources for one project or business unit from another). A common approach is to create a compartment for each major part of your organization.
- **Tenancy:** The root compartment that contains all your organization's OCI resources. Oracle automatically creates your organization's tenancy for you. Under the tenancy, you have a default identity domain, which contains users, groups, dynamic groups, MFA, and an app catalog. The tenancy also includes compartments and some policies. You can put policies into compartments inside the tenancy. You place the other types of cloud resources, such as instances, virtual networks, and block storage volumes, inside the compartments that you create.
- **Policy:** A document that specifies who can access which resources and how. Access is granted at the group level and compartment level. So, you can write a policy that gives a group a specific type of access within a specific compartment or to the tenancy itself. If you give a group access to the tenancy, the group automatically gets the same type of access to all the compartments inside the tenancy. The word *policy* can mean several things: an individual statement written in the policy language, a collection of statements in a single named policy document that has an Oracle Cloud ID (OCID) assigned to it, or the overall body of policies that your organization uses to control access to resources.

- **Identity domain**: An OCI resource that acts as container for managing users, groups, and dynamic groups; federating and provisioning users; configuring SSO for secure application integration; configuring adaptive authentication and MFA; and administering SAML and OAuth-based identity providers. It serves as an access control plane across all OCI offerings and a robust enterprise IAM for complex, hybrid IT environments.

  Resources in one identity domain are isolated from resources in other identity domains. Users always sign in to an identity domain and, based on permission, users can manage multiple domains from the Oracle Cloud Console.

- **Default domain**: Each tenancy comes with a default identity domain. Administrators (admins) in the default domain are the *super admins* of the tenancy, and they get these privileges from the seeded "tenant admin" policy, which can't be changed. So, we recommend *not* using the default domain admin, or tenant admin, for day-to-day operations. Instead, the default domain admin should create admins for managing specific resources.

- **Secondary domain**: An identity domain other than the default domain. Default and secondary domains have certain differences to consider when you're designing a solution. For more information, see the next section, "Compartments and Identity Domains."

- **Home region:** The region where IAM resources and domains reside. All IAM resources in the default domain are available across all regions, but the master set of definitions resides in a single region, the home region. You can change IAM resources in the home region only.

The following diagram illustrates these key components of the IAM service in the US East (Ashburn) (us-ashburn-1) region. The default domain resides in the root compartment and two secondary domains, ProductionDomain and ConsumerDomain, reside in the production and consumer compartments. The diagram also shows three policy statements that give access to the domain admins.

us-ashburn-1 region

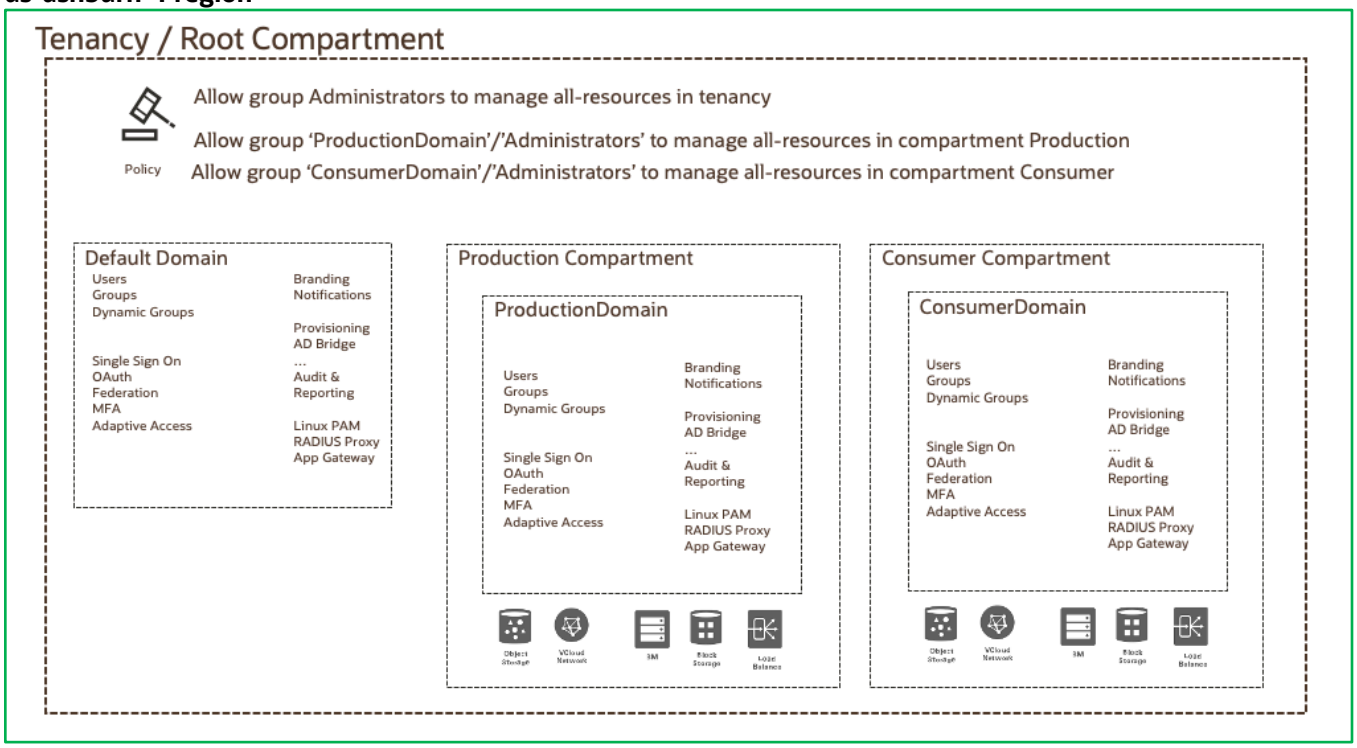

Figure 1: Oracle Cloud Infrastructure IAM Service Components

Compartments and Identity Domains

You use compartments to organize and isolate cloud resources, which makes it easier to manage and secure access to them. When you start working with Oracle Cloud Infrastructure, carefully consider the following aspects of compartments:

- When you create a resource, such as a Compute instance, block storage volume, VCN, or subnet, you must place it in a compartment.
- Compartments are tenancy-wide across regions. When you create a compartment, it's available in every region that the tenancy is subscribed to.
- Compartments are logical, not physical, so you can place related resource components in different compartments. For example, you can secure cloud network subnets with access to an internet gateway in a separate compartment from other subnets in the same cloud network.
- A resource can exist in only one compartment at a time.
- When you write a policy rule to grant a group of users access to a resource, specify the compartment to apply the access rule to. If you distribute resources across compartments, you must provide the appropriate permissions for each compartment for users who need access to those resources.
- Compartments can be nested up to six levels deep. A nested compartment inherits policies from its parent compartment. For example, CompartmentParent has a child compartment, CompartmentChild. The following policy allows groupA to manage a network in CompartmentChild:

  ```
  allow group domainA/groupA to manage virtual-network-family in compartment A
  ```

  The "Permission Management" section contains more information about this topic.
- When planning for compartments, consider how you want to aggregate usage and auditing data, which might be a consideration for your organization in the future.
- You can move resources from one compartment to another with a few exceptions (resources with dependencies). After you move a resource to another compartment, the policies that govern that compartment apply immediately and affect access to the resource. Depending on the structure of the compartment organization, metering, billing, and alarms can also be affected. Moving compartments also has implications on policies and tagging. Read more in Managing Compartments.

An identity domain represents a user population in OCI and its associated configurations and security settings. Consider the following aspects when you start working with identity domains:

- IAM with identity domains is a self-contained identity and access management service that can be used to address various IAM use cases.
- An identity domain is a resource in OCI, and you can write IAM policies to grant permission to a domain. You always sign in to a domain and, depending on permissions, one admin can manage multiple domains while signed in to one domain.
- Resources within identity domains are isolated from other identity domains. Consider creating separate identity domains for isolating developer, testing, preproduction, and production environments.
- Consider creating separate identity domains for each user population, such as separate identity domains for managing consumers and employees. Each domain can have applications and configurations that cater to consumer and employee use cases, such as different password policies and enabling self-registration for consumers.

- Each identity domain has a type associated with it, which determines the limits and features available in that identity domain. The types are Free Tier, Oracle Apps, Oracle Apps Premium, Premium, and External. You can convert identity domains from one type to another. Choose a domain type based on the use case. For more information, see IAM Identity Domain Types.
- Each tenancy comes with a free default domain in the tenancy home region. Default domain admins are super admins and can manage all resources in tenancies, including identity domains.
- The default domain always stays in the root compartment. You can't delete it.
- The home region of the default domain is selected when the tenancy is created. The home region of a secondary domain is selected when the secondary domain is created; it's the region that is selected in the Oracle Cloud Console. The home region of domains can't be changed.
- You can create multiple secondary domains with home regions that you choose. A secondary domain admin can manage resources within the secondary domain only. They need extra permission to manage resources outside of the secondary domain.
- When you subscribe a tenancy to a new region, the default domain is automatically replicated to the new region. Consider your organization's data residency requirements if you use the default domain. For a secondary domain, you need to explicitly replicate the domain to another region.

Given these considerations, we have the following recommendations:

- Don't use the default domain admin group and user with the identity domain admin role for day-to-day activities. Instead, create a separate admin for managing specific resources in OCI.
- Periodically check who is part of the default domain admin group and who has the identity domain admin role in the default domain. Users belonging to these two groups are super admins and can manage all resources in OCI.
- Use the default domain as a starter domain. Not only users from the default domain can access or manage resources in OCI; users in a secondary domain can also access and manage all OCI resources.
- Create other secondary domains for various use cases such as identities segmentation (consumer versus employees), environment separation (development, testing, production), and data residency requirement (creating a domain in a particular geographical region).

The compartment and identity domain design depends on your organization's use cases and how you want to organize and isolate resources. The following scenarios are examples.

Proof of Concept: Sandbox Compartment

If your organization is small or if you're still in the proof-of-concept stage of evaluating OCI, consider placing all the resources in the root compartment or tenancy. This approach makes it easy for you to quickly view and manage all the resources. You can still write policies and create groups to restrict permissions on specific resources to only the users who need access.

You can use the default domain for creating users and groups, and managing applications and MFA. We recommend setting up a separate "sandbox" compartment to give users a dedicated space to try out features. In the sandbox compartment, you can create a sandbox domain and grant users permissions to create and manage resources. So, admins of the sandbox domain have more flexibility in allowing users to try out various features of identity domains, such as MFA, SSO, and OAuth, while maintaining stricter permissions on the resources in the tenancy (root) compartment and default domain.

As shown in the following diagram, the admins of the sandbox domain can manage all the resources in the Sandbox compartment only, and the default domain admin has permission to manage all resources in the tenancy.

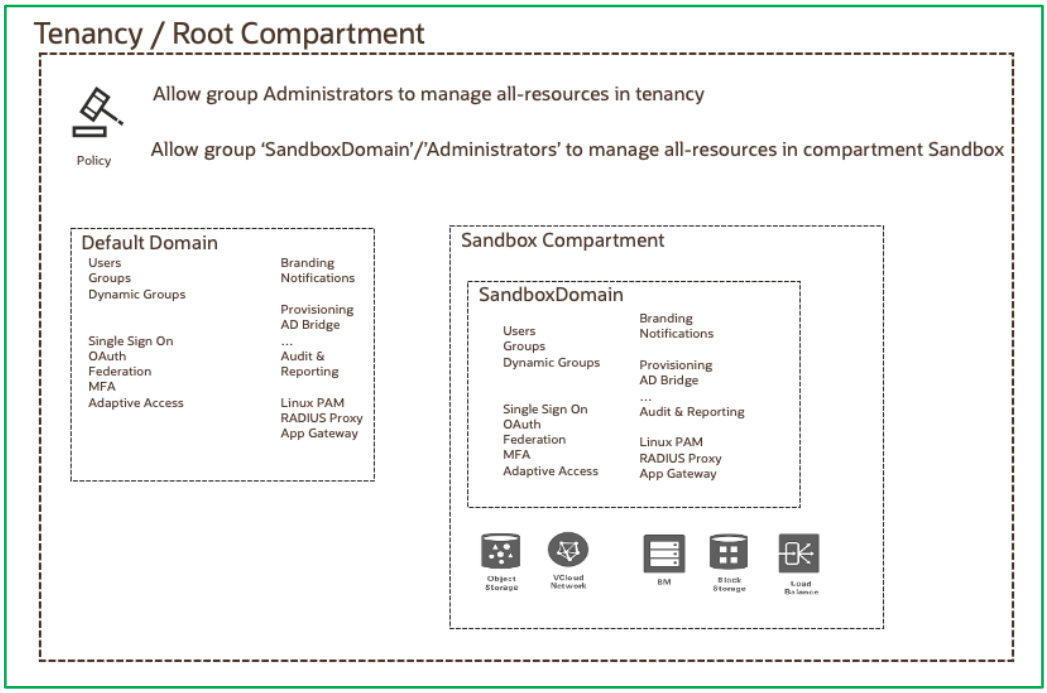

Figure 2: Sandbox Compartment with Sandbox Domain

Production Use

For production, restrict access to resources and consider how to organize the resources into compartments. Create a plan for the tenancy and compartments before you add users, groups, applications, and resources. In the plan, include the compartment hierarchy for organizing resources and the definitions of the user groups that need access to the resources. Users, groups, and applications are configured within the identity domains. Compartment hierarchy and identity domains impact how you write policies to manage access, so consider them together.

If your organization has multiple departments that you want to manage separately or several distinct projects that are easier to manage separately, we recommend aligning the compartment structure with these different departments or projects. With this approach, you can add a dedicated admin group for each compartment or project who can set the access policies for just that project. Users, groups, dynamic groups, and applications are managed from identity domains. You can give one group control over all their resources while not allowing them admin rights to the root compartment or any other projects. This way, you can enable different groups in your organization to set up their own subclouds for their own resources and administer them independently.

The following scenario illustrates how to design compartments and define related policies.

The company ACME has three major departments: A, B, and C. ACME also has multiple types of admins: database, network, storage, and security. Each department has a database admin who manages that department's database. Network, storage, and security admins need to access and manage corresponding network, storage, and security-related resources for all three departments.

ACME also has a consumer-facing application, MuShop, and wants to separate consumer identities and employee identities. For example, their production domain hosts employee identities and applications, such as Fidelity and Jira. Their consumer domain has consumer identities and the consumer retail application, MuShop.

The admin of the consumer domain resides in the employee's identity store and needs to manage consumer identities and applications.

To accommodate these needs, create five compartments to align with ACME's department structure. Then, create two domains: one for production and one for consumer. Define groups that map to each type of admin. Finally, define policies to control who can access which resources.

The following diagram illustrates a possible compartment, identity domain, and policy design for this scenario:

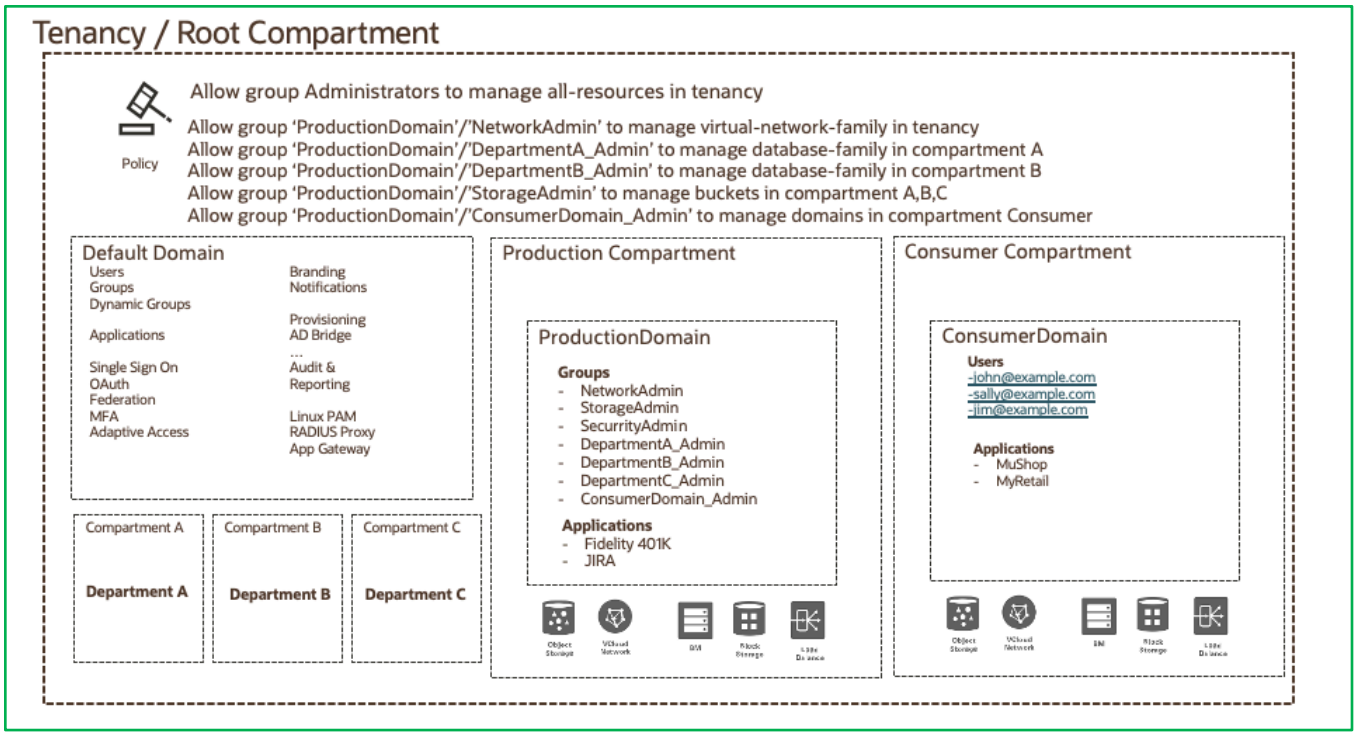

Figure 3: Production Example

Let's describe some of the policies mentioned in the diagram.

- `Allow group Administrators to manage all-resources in tenancy`

  This policy is the standard tenant admin policy, which can't be changed. The Administrators group is not prefixed with a domain name, which means that it belongs to the default domain. So, any user who belongs to the Administrators group in the default domain has permission to manage all resources in tenancy.

  We don't recommend using default domain admin for day-to-day operations. So, we created more admin groups in the production domain for managing various resources in OCI, as shown in the diagram.

- `Allow group 'ProductionDomain'/'NetworkAdmin' to manage virtual-network-family in tenancy`

  This policy allows users who are part of the production domain's NetworkAdmin group to manage networks in the tenancy.

- `Allow group 'ProductionDomain'/'DepartmentA_Admin' to manage database-family in compartment A`

  This policy allows users who are part of the production domain's DepartmentA_Admin group to manage databases in compartment A.

- `Allow group 'ProductionDomain'/'ConsumerDomain_Admin' to manage domains in compartment Consumer where request.domain.name='ConsumerDomain'`

  This policy allows users who are part of the production domain's ConsumerDomain_Admin group to manage the consumer domain. This ability is possible because the domain is a resource in OCI, and any other domain users can manage other domains when given the appropriate permission in policies.

User Management

You can manage users and groups in an identity domain in multiple ways. Consider the following aspects when managing users in the identity domains.

CONSIDERATION	RECOMMENDATION AND OPTIONS
What type of users you are managing? Are they employees or consumer users?	Consumer users typically require self-service capabilities. Use IAM self-registration capabilities or the System for Cross-domain Identity Management (SCIM) API for building custom solutions. Employee users can come from various systems, such as Fusion Human Capital Management (HCM) or Active Directory. See the "Bringing Users from External Sources" section.
Is user management automated or manual?	Use the Oracle Cloud Console to manually create users for quick testing and sandbox use cases. Use the Oracle Cloud Console to import large number of users and groups by using the export and import CSV option.
Do you want to synchronize passwords while bringing users into IAM?	Use REST APIs to import hashed passwords into IAM using a CSV import job.
Do you want to create users in inactive states? Do you want to activate them later?	Use REST APIs or CSV import with the active state as False.

Managing Users Locally Using the Console

To manage users, you must have one of the following permissions:

- You're part of an admin group in the identity domain. An admin group in the identity domain can manage all resources in the domain including users and groups.
- You have permission to manage users. The policy admin can further delegate user and group management by writing policies, such as the following example:

  ```
  allow group <domain1>/<group1> to manage users in tenancy where request.domain.name=domain
  ```

- You're part of the user admin or user manager admin roles.

You can use IAM policies instead of admin roles for permission management of users and groups. For more information, see the "Admin Roles Versus IAM Policies" section.

A new user has no permissions until you place the user in a group that has at least one policy that gives that group permission to either the tenancy or a compartment.

We recommend clearly categorizing the roles of new users first and then placing them in groups with the appropriate governing policies. For example, if a user is a database admin, you can place them in a database admin group that has policies that grant permissions to manage the `database-family` resource type for the corresponding compartments.

Bringing Users from External Sources

You can bring users into IAM from external sources by using automation in several ways, depending on the use case. IAM supports the following methods:

- **App Catalog**: We have a rich provisioning App Catalog to bring in users from applications such as Oracle Human Capital Management (HCM), Oracle Unified Directory, and Oracle Internet Directory.
- **SCIM APIs**: Use the IAM SCIM API to manage the user lifecycle.
- **Bridge component**: IAM provides the Active Directory Bridge and Provisioning bridge software component that customers install locally to sync and provision users.
- **Just-in-Time (JIT)**: Import users as part of federation authentication using just-in-time provisioning.

Consider the following aspects when managing users in the identity domains:

- The bridge component is software that you need to download, install, and configure to use with IAM.
- Active Directory Bridge works with Active Directory only and is also used for delegated authentication where users can sign in to IAM using Active Directory credentials directly.
- The provisioning bridge is used by certain application templates such as Oracle Unified Directory and Oracle E-Business Suite. The bridge is not required for all provisioning templates; for example, the Fusion Applications template doesn't need a bridge.
- Consider which groups and users you want to import using filters available in the provisioning template.
- Consider using CSV import through a REST API for synchronizing passwords into IAM.

Permission Management

Permission management in Oracle Cloud Infrastructure is done through policies. A policy allows a group to work in certain ways with specific types of resources in a particular compartment or tenancy. Policies give access to groups of users, not to individual users. Users gain access by being in groups.

Policies only *allow* access. They can't explicitly deny it. If you need to restrict a particular user's access, you can remove the user from a particular group of interest or delete the user from the IAM service entirely.

Each policy consists of one or more policy statements that follow this basic syntax:

```
Allow group <domain name>/<group name> to <verb> <resource-type> in compartment <compartment name>
```

- `<domain name>` is the domain where the group resides.
- `<verb>` denotes the type of access: `inspect`, `read`, `use`, or `manage`. Each successive type of access includes the access in the preceding types. For example, `inspect` gives users in the group the ability to list resources without access to confidential information or user-specified metadata in the resource. `read` includes `inspect` plus the ability to get user-specified metadata and the actual resource itself.
- `<resource-type>` can be an aggregate (family) resource or an individual resource. For example, `database-family` is an aggregate resource-type, and `db-systems` and `db-nodes` are individual resource-types in that family.

We recommend starting with more granular policy definitions and then updating them for different use cases. For more information, see How Policies Work in the IAM service documentation.

If you don't prefix a domain name, as shown in the following example, then the default domain is implied. The following examples are equivalent:

```
Allow group NetworkAdmin to manage virtual-network-family in compartment AcmeCorp

Allow group 'Default'/'NetworkAdmin' to manage virtual-network-family in compartment AcmeCorp
```

If you're writing policies for groups in the default domain, we recommend prefixing `default` to the group name to make the policy statement more readable. Use the Policy Builder tool to author policies quickly and easily, as shown in the following image.

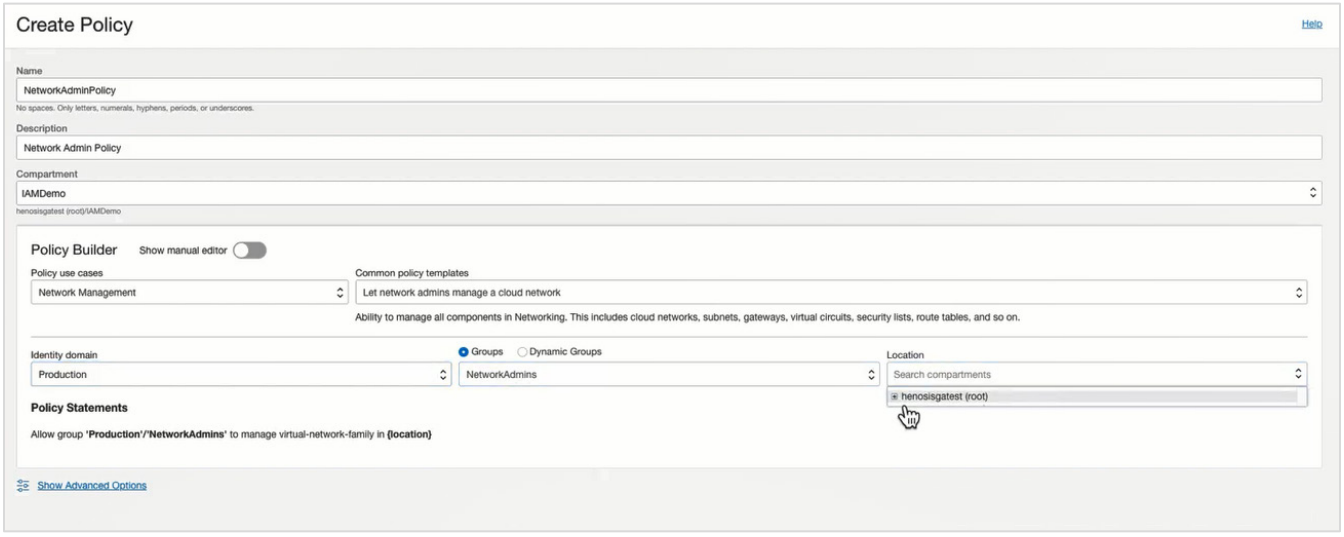

Figure 4: Creating a Policy with the Policy Builder

Policy Attachment and Inheritance

Before you create any policies, you must understand policy attachment and inheritance. When you create a policy, you create it under a compartment, and the policy is attached to that compartment. Where you attach a policy controls who can then modify it or delete it. Also, compartments inherit any policies from their parent compartment.

In the following example, policy 1 is attached to the root compartment, policy 2 is attached to the parent compartment, and policy 3 is attached to the child compartment. So, any user who has permission to manage policies in these compartments can update or delete these policies.

Policy 1 allows the admin of the default domain to manage all resources in the tenancy (root compartment). In the example, the hierarchy of compartments is root > parent > child. So, using policy inheritance, policy1 allows the admin to manage all resources in the parent and child compartments, too.

Similarly, policy 2 allows Domain1 and Group1 to manage the network in the parent and child compartments because of the parent > child compartment hierarchy.

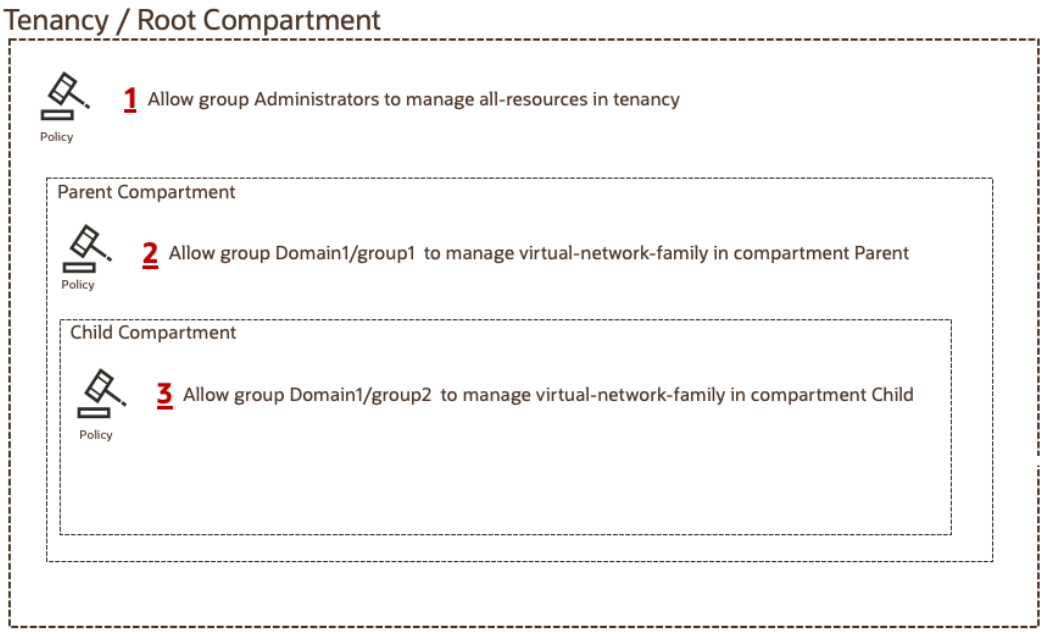

Figure 5: Policy Inheritance of Child Compartments from Parent Compartments

We have the following recommendations:

- Based on the project, consider who are the policy admins and decide on a compartment where you want to attach policies.
- Limit the number of admins who have access to author policies.
- In the following policy, `manage all-resources` means managing policies, too:

  ```
  allow group domain/group to manage all-resources in compartment A
  ```

  We recommend using specific permission instead, such as `manage policies` or `manage virtual-network-families`.
- If you need to make more policy admins, restrict them to a minimum scope by making them policy admin for a particular compartment. This restriction allows them to create policies to manage resources, but they can write policies to manage resources within that compartment only.
- To prevent the accidental deletion of a policy, don't give policy delete permission. For example, in the following policy, the `manage` keyword allows all create, read, update, and delete (CRUD) operations:

  ```
  allow group domain1/group1 to manage policies in compartment
  ```

  Instead, use the following policy:

  ```
  allow group domain1/group1 to manage policies in compartment demo
  where request.permission = any {'POLICY_CREATE', 'POLICY_UPDATE'}
  ```

Enforce Least Privilege

We recommend writing policies to enforce least privilege and gradually adding permission when required. Let's understand how to write these policies with some examples.

POLICY DEFINITION	EXPLANATION

Policy A
`Allow dynamic-group AuditDG to manage objects in compartment AcmeCorp`
Policy B
`Allow dynamic-group AuditDG to manage objects in compartment AcmeCorp where all {target.bucket.name = 'audit_logs_bucket', request.permission='OBJECT_CREATE'}`
	Policy A doesn't enforce least privilege. It allows managing objects in all buckets in compartment AcmeCorp.
	Policy B restricts the manage objects access in the following ways:
	• Restricts access to only one bucket by specifying the bucket name:
	`target.bucket.name='audit_logs_bucket'`
	• Restricts access to update and delete by giving only create permission:
	`request.permission='OBJECT_CREATE'`

`Allow dynamic-group AuditDG to read secret-family in compartment AcmeCorp where target.secret.name = 'audit-secret'`	This policy specifies read-only access to the group because the group needs read access to consume it.

Policy A
`Allow group XYZ to manage groups in tenancy where request.permission != 'GROUP_DELETE'`
Policy B
`Allow group XYZ to manage groups in tenancy where any {request.permission='GROUP_INSPECT', request.permission='GROUP_CREATE', request.permission='GROUP_UPDATE'}`
Policy C
`Allow group XYZ to manage groups in tenancy where any {request.operation='ListGroups', request.operation='GetGroup', request.operation='CreateGroup', request.operation='UpdateGroup'}`
	Group XYZ needs to be able to list, get, create, and update groups, but not delete them.
	Policies A, B, and C achieve the same result, but consider the following differences:
	• **Policy A**: Any new permissions the service might add in the future are automatically granted to group XYZ, omitting only GROUP_DELETE.
	• **Policy B**: The allowed permissions are explicitly stated. We recommend this option.
	• **Policy C**: This option writes a condition based on the specific API operation.

`Allow group DomainA/GroupA to manage object-family in tenancy where request.networkSource.name='corpnet'`	This policy restricts access based on a network source.
`Allow DomainA/Contractors to use instances in compartment contractors where all {request.utc-timestamp after '<TIME>', request.utc-timestamp before '<TIME>'}`	This policy uses time-based variables to restrict the access granted in the policy to only certain time frames.

Tag-Based Access Control

Oracle Cloud Infrastructure Tagging lets you add metadata to resources, which enables you to define keys and values and associate them with resources. You can use the tags to organize and list resources based on your organization's business needs.

By using tags in policy statements, you can control access of resources in OCI. Tag-based access control (TBAC) provides more flexibility to policies by letting you define access policies with tags that span compartments, groups, and resources. You can control access based on a tag that exists on the requesting resource, such as a group, dynamic group, or compartment, or on the target of the request, such as a resource or compartment.

Avoid entering confidential information when assigning descriptions, tags, or friendly names to cloud resources through the Oracle Cloud Console, API, or CLI.

Using the following example, let's explore some key consideration while using TBAC. The example shows two Compute instances tagged with various values:

Figure 6: Example Compute Instances with Tagged Values

With tags in place, we recommend the following best practices:

- Control who can apply a tag to the resources. For example, you can restrict the user part of GroupA to applying the tag "Operations.Environment" by using the following policy:

  ```
  Allow group DomainA/GroupA to use tag-namespaces in tenancy where all{target.tag-namespace.name='Operations'}
  ```

- The policy admin who is writing policies using tags must know all the resources and requestors that carry the tag and must consider the impact of writing policies using tags. For example, an admin is writing the following policy:

  ```
  Allow group DomainA/ProductionAdmins to manage instance in compartment Production where target.resource.tag.Operations.Project= 'Alpha'
  ```

  Before writing policies, the policy admin must know all the instances that are tagged with `project = alpha` or know who has access to apply the `Operations.Project` tag so that they know who has access using the policy that they're writing.

Read more about TBAC in the OCI documentation.

Admin Roles Versus IAM Policies

Each identity domain has admin roles that allow identity domain admins to have different levels of access to various tasks and resources. This section describes the privileges of each admin role.

Roles apply only to users within the domain and allow granular access to apps in the domain or data planes integrated with domains, without the ability to scope to compartments. Policies can apply more broadly but only to the control planes and allow scoping to compartments.

Consider the following recommendations and options for using admin roles and policies:

ADMIN ROLES	PRIVILEGES	RECOMMENDATIONS AND OPTIONS
Identity domain admin	Has superuser privileges for an identity domain in IAM. Identity domain admins have the following abilities: • Manage users, groups, applications, system configuration, and security settings • Perform delegated administration by assigning users to different administrative roles • Enable and disable MFA, configure MFA settings, and configure authentication factors • Create self-registration profiles to manage different sets of users, approval policies, and applications	When you specify a domain admin while creating a domain, IAM assigns an identity domain admin role to that user. Assigning a domain admin while creating a domain isn't mandatory. You can create a domain without specifying an admin to it and later write policies to give permission to users to manage a domain, such as the following policy: `Allow group 'Domain1'/'Group1' to manage domains in compartment A where request.domain.name=Domain2` So, you can use policies to manage domains instead of assigning the identity domain admin role to the individual user.
User admin, user manager, and helpdesk admin	Manage users, groups, and group memberships for an identity domain.	Instead of using this admin role, you can write policies to give permission to manage user groups and group membership

Admin Accounts

Admin accounts have privileged access that can be misused if the accounts are compromised. Ensure the following requisites for admin accounts:

- Enable MFA for all admin accounts with more secure factors, such as FIDO2 security keys or Mobile App push notification. Mobile App requires a configurable unlock using phone biometrics before responding to the notification, making it more secure than a one-time password (OTP) sent through SMS or email.
- Create more admin accounts instead of sharing admin credentials among users, which allows audit logs to capture who changed what.
- Periodically monitor all admin account access.
- Check applications in the identity domains that have been granted the identity domain admin (IDA) role. The IDA role is the domain admin and can manage all resources in the domain. An app with the IDA role can invoke APIs and change what an admin can do from the Console.
- Consider enabling adaptive access for admin accounts so that you can step up authentication or deny access if the risk score is high.

Break Glass (Emergency) Account

Create a “break glass” account for emergency use when no one can access the Oracle Cloud Console. Consider the following factors when creating a break glass account:

- All users who are part of the default domain admin group are global admins and can manage all resources in OCI.
- After the tenant admin creates more admins for managing specific resources in OCI and creates the break glass account, they should lower their privileges by removing themselves from the default domain admin group and identity domain admin role.
- While creating accounts in the Console, you must specify an email address. For account setup, you can choose the email address of the admin creating the account. After the admin has set the password for the break glass account, they should remove the email address and update the profile with an invalid email address so that reset password attempts don’t work.
- Enable monitoring of the break glass account and notify other admins whenever this account is used.
- Because break glass accounts are highly privileged accounts, enable MFA for them, and ensure that they’re not connected to an individual or a device that might not be available during an emergency. Consider having at least one break glass account with an MFA factor that’s not asynchronous, for example, Fast ID Online (FIDO) security keys that are stored securely in a known place.
- Ensure that this account is in the default domain and only part of the admin group or has permission to manage all resources in the tenancy.
- Ensure that there is no explicit `deny` in the default domain sign-in policy, which can prevent the break glass account from signing in.

Enable Single Sign-On

Accessing applications and services can require users to track multiple URLs, user names, and passwords. We recommend enabling IAM’s single sign-on (SSO) capability across both on-premises and cloud applications and services.

- Use the rich IAM app catalog, composed of SAML, OpenID, and provisioning apps.
- Use a SAML, OAuth, or OpenID app wizard for apps that aren’t available in the catalog.
- Use standards-based integration by using SAML, OpenID, or OAuth protocol.
- Use the IAM app gateway’s header-based authentication for apps that don’t support open standard protocol, and eventually upgrade such apps to support SAML, OpenID, or OAuth.
- IAM identity provider policies control which identity provider (IdP) is used for authentication. IdP policies are applied before authentication.
- Use rules, conditions, and the order of execution to orchestrate a complex sign-in requirement using IdP policies. For example, an admin can configure a passwordless rule that’s triggered for any user who is part of the passwordless group or a rule to always authenticate using an external IdP if the username ends with `exampledomain.com`.

Enforce Multifactor Authentication and Adaptive Security

Multifactor authentication (MFA) gives organizations a crucial layer of security. It secures end-user credentials and admin access to on-premises and software-as-a-service (SaaS) applications.

In IAM, you enforce MFA by using sign-on policies. Sign-in policies allow the admin to configure rules for application access, step up authentication, MFA, and adaptive risk scoring to challenge or deny access.

We recommend creating a sign-in experience based on context. For example, if a user is accessing a financial application, you can decide to prompt for MFA. If a user is accessing a virtual video conference application to change profile settings, you can decide not to prompt for MFA.

Consider the following available conditions, risk events, and actions to design authentication and MFA based on the users of the application, how the application is used, application compliance requirements, and the type of user experience that users expect to have.

CONDITIONS	RISK EVENTS	ACTIONS
User Groups Applications Authentication methods Trusted device Network perimeter Risk score	Too many unsuccessful MFA events Impossible travel between locations Access from a suspicious IP address Access from an unknown device Too many unsuccessful sign-ins Access from an unfamiliar location	Allow Deny Reauthenticate MFA • Required or optional • Every *n* hours or days • Specific factors, such as FIDO2 • Skip for trusted devices

The risk events aren't used directly in the conditions. The risk score is used in the policy condition, and risk scoring is configured in the adaptive security screens. Define the weight for individual risk events that influence how the system generates risk scores, as shown in the following image.

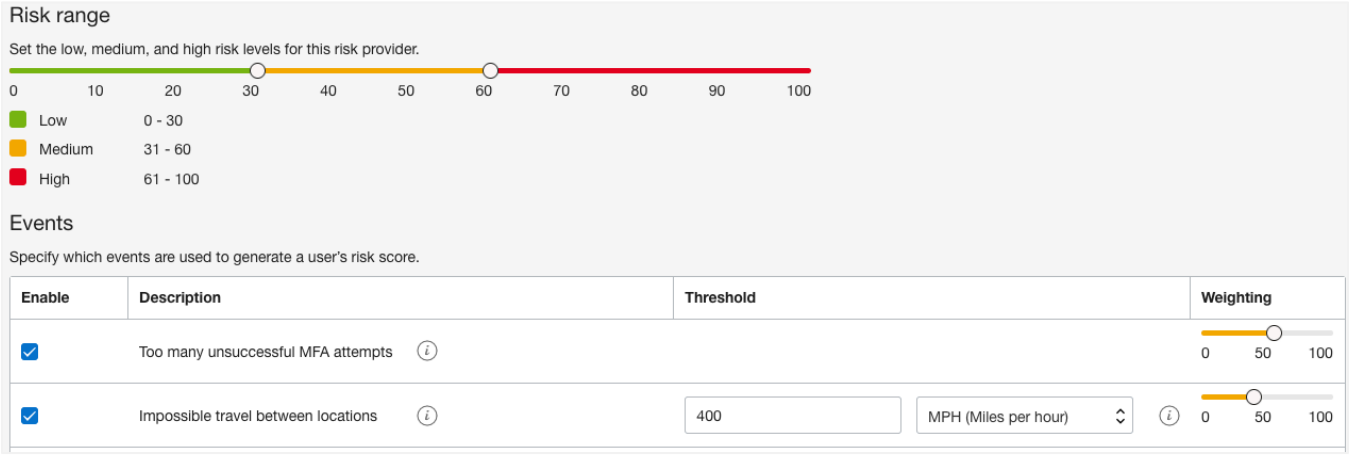

Figure 7: Risk Range Chart, Including Events and Weights

Replication, Disaster Recovery, and High Availability

This section clarifies ownership for replication, disaster recovery, and high availability in IAM between the customer and Oracle Cloud Infrastructure.

Replication

Each identity domain has a home region associated with it, and you can further replicate domains to other tenancy-subscribed regions. When you replicate a domain to another region, all the identity resources are replicated to that region, and IAM can now authorize users against resources in that region.

All authentication and editing of identity resources always happen in the home region of the domain. So, this replication is not a high-availability or disaster-recovery solution. However, we're planning to add support for high-availability read-only and high-availability read/write using replication.

Replication is a factor only when a tenancy is subscribed to more than one region, the default domain is automatically subscribed to all the tenancy-subscribed regions, and the automatic subscription can't be changed. Customers can choose to replicate secondary domains to any of the tenancy-subscribed regions.

We recommend using a secondary domain for data residency requirements. With secondary domains, you can choose the geographical regions where you want to replicate a domain.

Ownership for replication is shared between OCI and customers.

Disaster Recovery

Disaster recovery refers to backing up the domain's data in another region in preparation for a disaster. This independent operation has nothing to do with replication or the high availability of IAM.

If an entire OCI region becomes unavailable, traffic is routed to the disaster recovery region to speed service recovery and retain as much data as possible. Oracle pairs regions with disaster recovery regions for you. Customers can't choose the disaster recovery region. The regional mappings are obtained from the OCI Compliance team, who decides based on country laws and regulations.

For more information, see Disaster Recovery Region Pairings and Disaster Recovery and Identity Domains.

OCI has ownership for disaster recovery. Customers might need to update firewalls to allow traffic from the disaster recovery region.

High Availability

IAM offers high availability within the home region. Inside an OCI region, we have either fault domains in a single availability domain or multiple availability domains. They act the same way, but fault domains are physically closer together than availability domains. IAM is deployed with redundant installations in each region (two across the availability or fault domains), which provides high availability within the region.

OCI has ownership for high availability. Customers own making applications, such as custom sign-in pages, ready for high availability.

Instance Principals and Dynamic Groups

The instance principals feature of IAM allows users to call IAM-protected APIs from an Oracle Cloud Infrastructure Compute instance (virtual machine or bare metal) without needing to create IAM users or manage credentials for each instance.

For example, you have an application running on a Compute instance that needs to access the Object Storage service. Without instance principals, you need to create a specific user and then create a policy to grant permission to read and write to a bucket in the Object Storage service and then assign this policy to that user. The application uses the credentials for the user to access the object bucket. The problem with this approach is that the private key for the user must be accessible to the application, probably by storing it in a configuration file. The process for obtaining the private key and storing it in the configuration file can be complicated and create security risks.

When you use instance principals, you create dynamic groups. Dynamic groups allow you to group Compute instances as principal actors, like user groups. You can then create policies to permit instances to make API calls against OCI services. When creating a dynamic group, provide an unchangeable name for the dynamic group that's unique across all groups within the tenancy.

Note: Any user who has access to the instance automatically inherits the privileges granted to the instance. Before you use this feature to grant permissions to an instance, find out who can access the instance and determine whether they need authorization with the permissions that you're granting to the instance.

Federation

IAM supports federation using SAML 2.0 and OpenID Connect. Federation is done at the identity domain level. Each tenancy can have more than one domain, and you federate an identity domain to any of the external identity providers (IdPs) that support SAML2.0 or OpenID Connect.

Consider the following factors when setting up federation:

- An identity domain can act as an IdP and a service provider.
- Use the identity domain application catalog to add a service provider. If you don't find an application in the template, use the application template wizard.
- All the common social IdP templates are available. OpenID Connect is available from the "Add Social IDP" menu.
- The IdP policies allow you to select an IdP at runtime, based on various conditions. A single IdP in the policy takes you to that IdP automatically without showing the IAM sign-in screen. For example, you can create an IdP policy to authenticate against a social IdP if a user belongs to consumer user group.
- For federation to work, you need to sync users between IAM and the external IdP or use SAML JIT provisioning.

Enable Password Management

Self-service password reset reduces user frustration and help-desk overload. IAM allows users to reset their password and enroll MFA devices using self-service.

IAM password policies have three variants: simple, standard, and custom with comprehensive password rules. We recommend the following best practices for creating password policies:

- Don't rely on the default password policy. Instead, create a password policy by using the custom policy template so that you can tailor it to your organization's compliance requirement.
- Identify groups that need separate password policies, and enforce strong password policies for admin accounts.
- Depending on the use case, consider automatic account unlocks and different lock threshold for groups that have least privileges.
- While setting password policy priorities, consider this scenario: If a user has more than one group assigned to them, the password policy with the highest priority is the password policy assigned to the user.

Conclusion

This paper recommends best practices for using the Oracle Cloud Infrastructure IAM service to securely manage and control access to cloud resources. The following highlights summarize these best practices:

- Plan the tenancy and compartments before you add users and resources.
- Align compartment design with the structure of your organization's departments or projects.
- Categorize the roles of users first, and then place them into groups with the appropriate governing policies.
- Grant least privilege to users and gradually grant more permissions as needed.
- Enforce strong password policies and update passwords regularly.
- Use instance principals and dynamic groups when calling OCI services from Compute instances.

Oracle Cloud Infrastructure is continuously evolving with new features. Stay updated through online documents and training at oracle.com/cloud/.