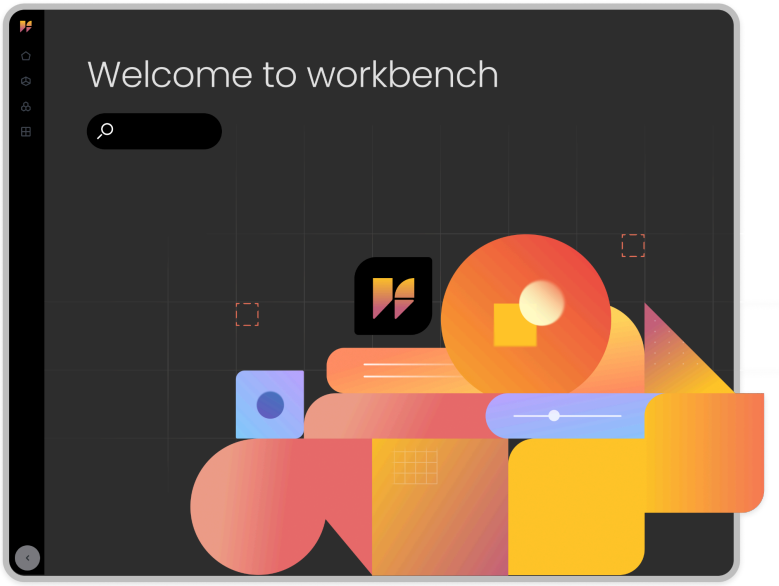
workbench

Making Workbench multicloud — now on AWS

Built on by verily

Verily Pre Workbench + AWS



Research organizations increasingly operate across multiple cloud ecosystems. Academic medical centers may use Google Cloud Platform (GCP) for enterprise infrastructure, while pharmaceutical partners run analytics on Amazon Web Services (AWS). Secure, compliant collaboration across these platforms is essential to advance biomedical research.

Verily Workbench, built on Pre, now supports both GCP and AWS. Our implementation across both platforms includes workspace isolation, granular access controls, external data integration, native tooling support, and platform-specific service access, all maintained with consistent security and governance standards.

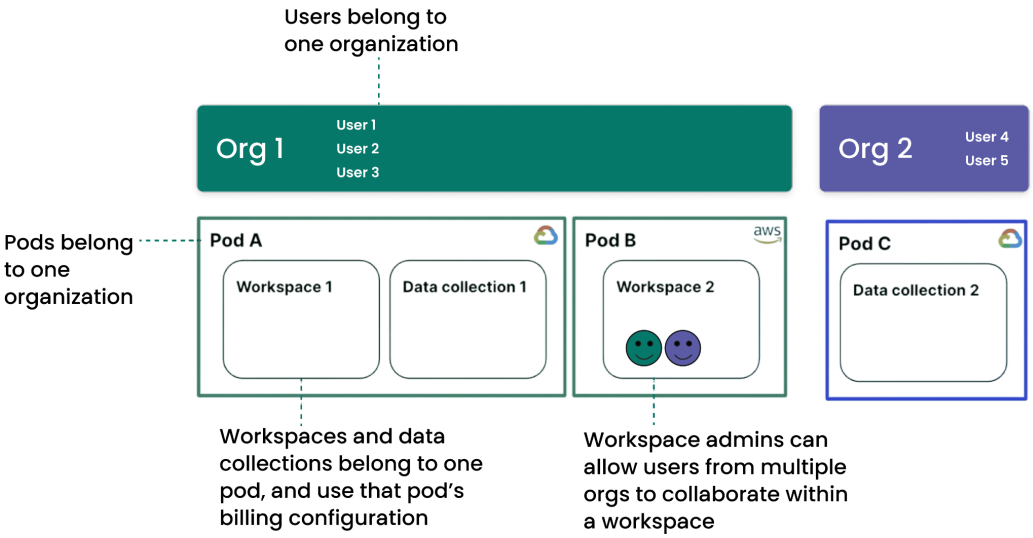
Workbench's core concepts — Organizations, Pods, Workspaces, and Resources — define tenancy, billing, and resource lifecycle independently of any cloud provider. This whitepaper details how our control plane maps these abstractions to each platform's native services, such as workspace isolation mechanisms, RBAC versus ABAC for identity management, and approaches to cross-account data sharing.

Our approach unifies foundational services such as object storage and virtual machines to support code portability, while also exposing platform-specific capabilities like BigQuery on GCP and managed genomics workflows on AWS as first-class resources. This allows researchers to use specialized tools within a governed environment, without limiting them to the lowest common denominator of features.

Workbench concepts

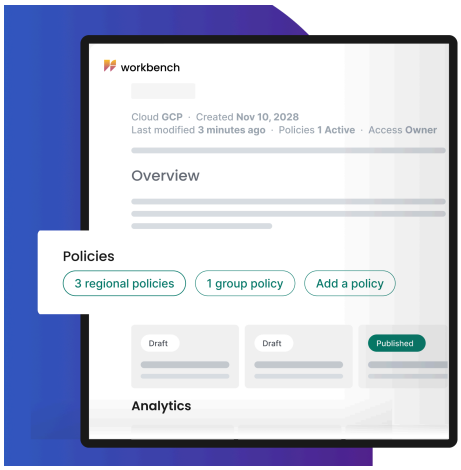
Verily Workbench is a Trusted Research Environment (TRE). This infrastructure enables biomedical researchers to securely manage sensitive government-funded datasets while maintaining mandatory compliance.

Its architecture is built around a few key hierarchical concepts that manage tenancy, billing, and the lifecycle of cloud resources. Crucially, these foundational concepts are designed to be entirely independent of any specific cloud provider at a conceptual level; their practical implementation will naturally vary across various cloud platforms.



Organizations

The highest level of tenancy is the organization, which provides a clear boundary for ownership and administration. Each Workbench user belongs to a single organization. An organization's administrators are responsible for managing users, defining groups, setting governance policies, and configuring the billing structure for all the work conducted under its purview.



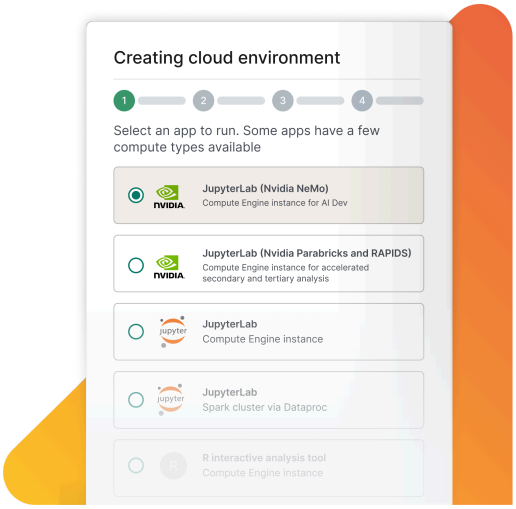
Pods

To manage cloud costs, Workbench uses a concept called pods. A pod is the essential link between the work being done on the platform and a specific cloud billing profile. Within an organization, administrators create one or more pods, each tied to a different billing profile. This structure allows an institution to organize resources and people, ensuring that the cloud costs generated by a particular project or team are charged to the correct budget. Users must select a pod when creating a new

workspace, which dictates which billing account will be used for all cloud costs associated with that workspace. Organization administrators control which users are allowed to create workspaces in a pod, either directly or by delegating pod management capability to specific users.

Workspaces

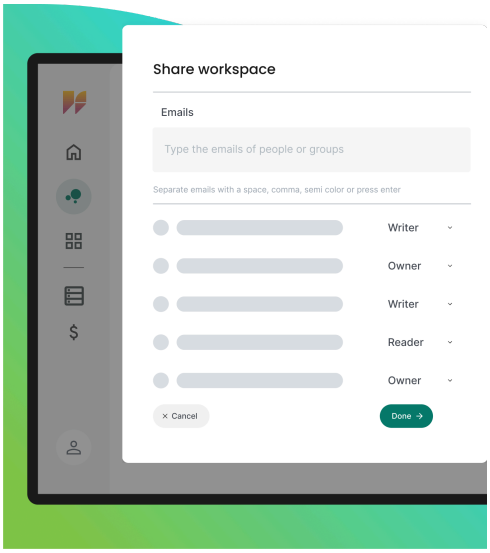
The workspace is the collaborative environment where research and analysis take place. It serves as a container that organizes all the elements of a research project, including data, code, analysis tools (like JupyterLab or Visual Studio Code), and documentation. Workspaces are designed to be shared securely with collaborators, allowing teams — which can include members from multiple organizations — to work together with consistent tools and well-defined access controls.



Resources

Within a workspace, data and tools are organized as resources. These resources are primarily categorized in two ways.

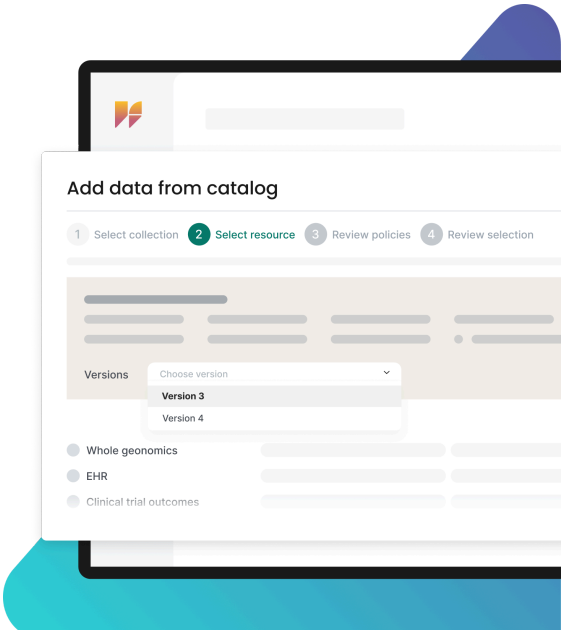
Controlled resources are data or tools that are managed directly within the workspace, such as a BigQuery dataset or a Cloud Storage bucket created specifically for the project. When a workspace is deleted, its controlled resources are permanently deleted along with it.



In contrast, **referenced resources** are pointers to data that exists outside the workspace, such as a publicly available dataset or a Workbench-curated data collection. These references allow researchers to analyze data without copying it, and if a workspace is deleted, the original data pointed to by the reference remains unaffected.

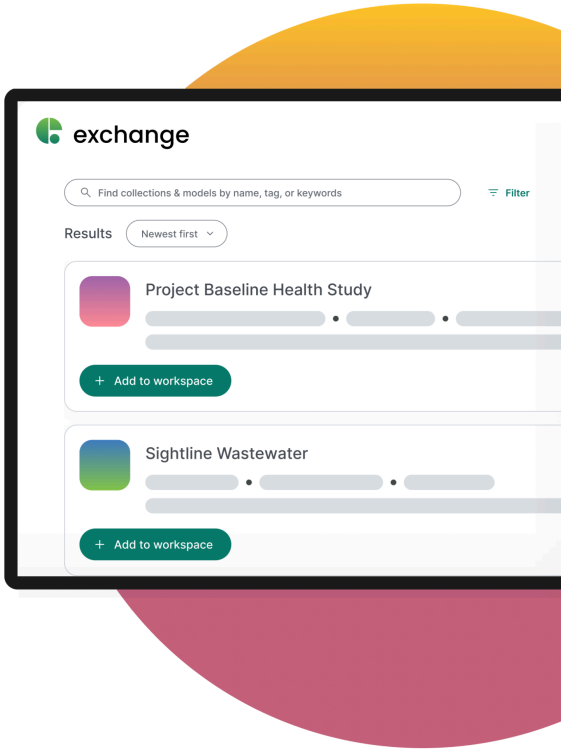
Data collections

While controlled and referenced resources are the technical building blocks for data within a workspace, data collections unlock the true potential of Workbench by allowing data owners to make large-scale, multimodal datasets discoverable and usable.



At the organization level, data owners can curate groups of Workbench resources related to a specific project, study, or purpose into a data collection. Data owners control the accessibility and governance of these collections through permissions and policies. These policies are critical, as they propagate from the collection and are automatically attached to any workspace that references the data, ensuring compliance.

Researchers can then discover these collections through the Verily Exchange, built on Pre, and create references to some or all of the resources within them for analysis. In addition to raw data, a collection's resources may also represent other supporting content, such as markdown documentation describing the data or code to help with analysis.



THE MICHAEL J. FOX FOUNDATION FOR PARKINSON'S RESEARCH

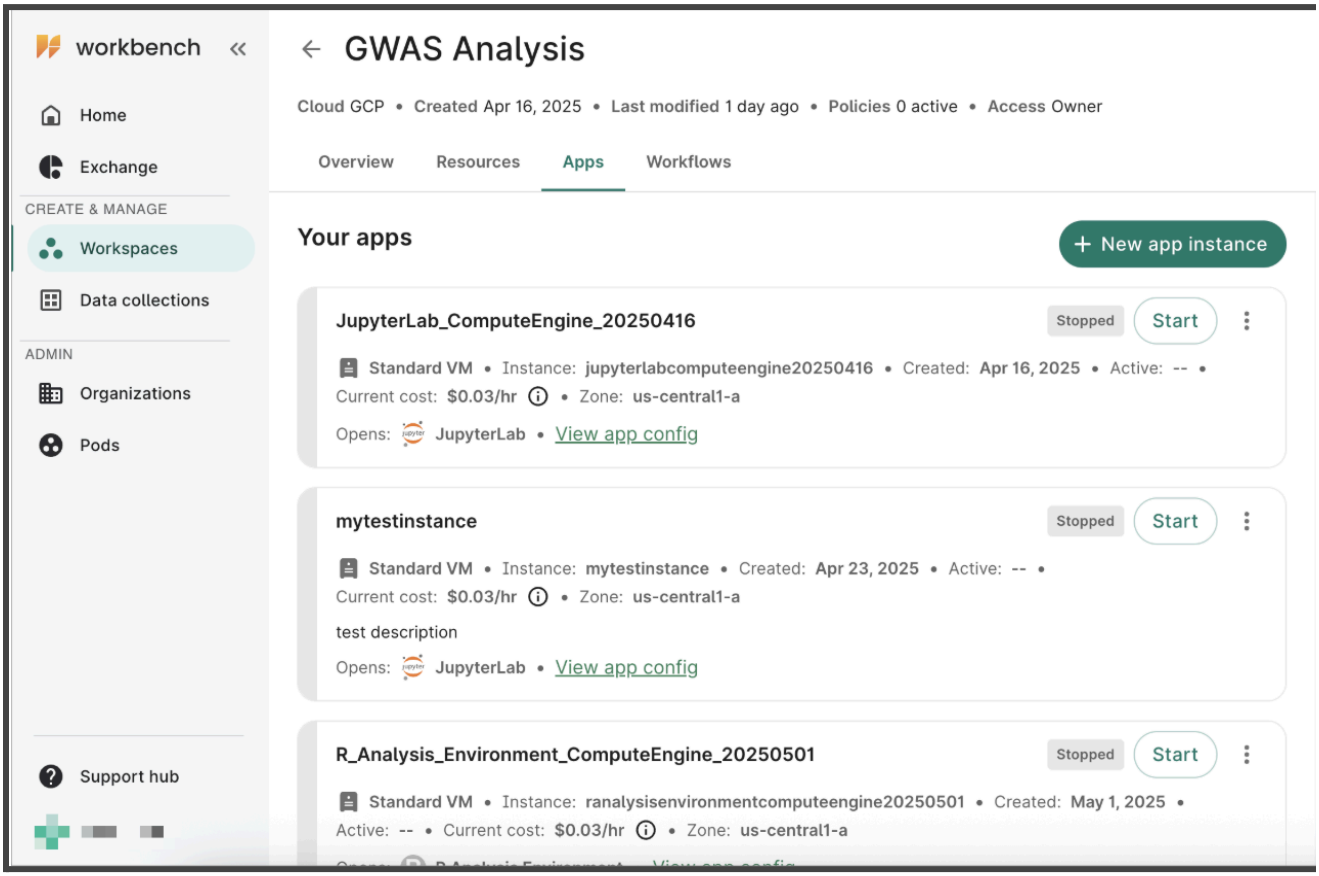
An excellent example of how this capability manages access to data can be found in this case study of how The Michael J. Fox Foundation for Parkinson's Research (MJFF) and the Global Parkinson's Genetics Program (GP2) have partnered with Verily to host and secure their latest data release on Verily Workbench by leveraging data collections.

Technical deep dive: Adding AWS support to Verily Workbench

Projects vs. Landing Zones

This is where the abstract Workbench concepts meet cloud-native reality. Because Verily Workbench was originally built as a GCP-native platform, its foundational concepts had a natural, direct implementation using corresponding GCP services. A Workbench pod mapped cleanly to a GCP Billing Account, and a workspace mapped directly to a GCP Project.

The core challenge of adding AWS support was to map these same abstract concepts to a completely different set of foundational services and philosophies, while keeping the user experience cloud-agnostic. From a user's perspective, the Verily Workbench experience is consistent across cloud platforms. A user creates a workspace within a pod, and then uses the Workbench API to create and manage controlled resources (like storage buckets or cloud app virtual machines) within that workspace.



Behind the scenes, our control plane implements this abstraction on foundational concepts like billing and access control that vary significantly across different cloud platforms. The specific control plane service implementing this abstraction is called the Workspace Manager service.

On Google Cloud, a GCP Project provides a 1:1 mapping to a Workbench Workspace. When a new workspace is created, the Workspace Manager creates a new, dedicated GCP Project for it. This project serves as a self-contained unit that provides:

- Resource Encapsulation: All controlled resources created for that workspace (like GCS buckets) live inside this single, dedicated project.
- Atomic Lifecycle: When a workspace is deleted, the Workspace Manager deletes the entire GCP Project. This provides a single operation to remove all associated resources and permissions.
- Cost Attribution: The Pod maps directly to a GCP Billing Account, and each new project is attached to it, providing direct cost attribution per workspace.

This 1:1 project-per-workspace model is specific to GCP. When our team designed the Workbench implementation for AWS, we adjusted to a different foundational structure. AWS does not offer a direct equivalent to the GCP Project. An AWS Account is the analogue for billing, but accounts are designed as more persistent management units, not for dynamic, per-workspace lifecycles. As a result, on AWS, Workbench's 1:1 mapping is at a higher level: one Workbench pod maps to one AWS account.

This single account must now contain the resources for all workspaces within that pod. This required a different architectural approach: if workspaces are not in isolated units, how do we provide the shared, foundational infrastructure (like networking) while maintaining strict isolation between them?

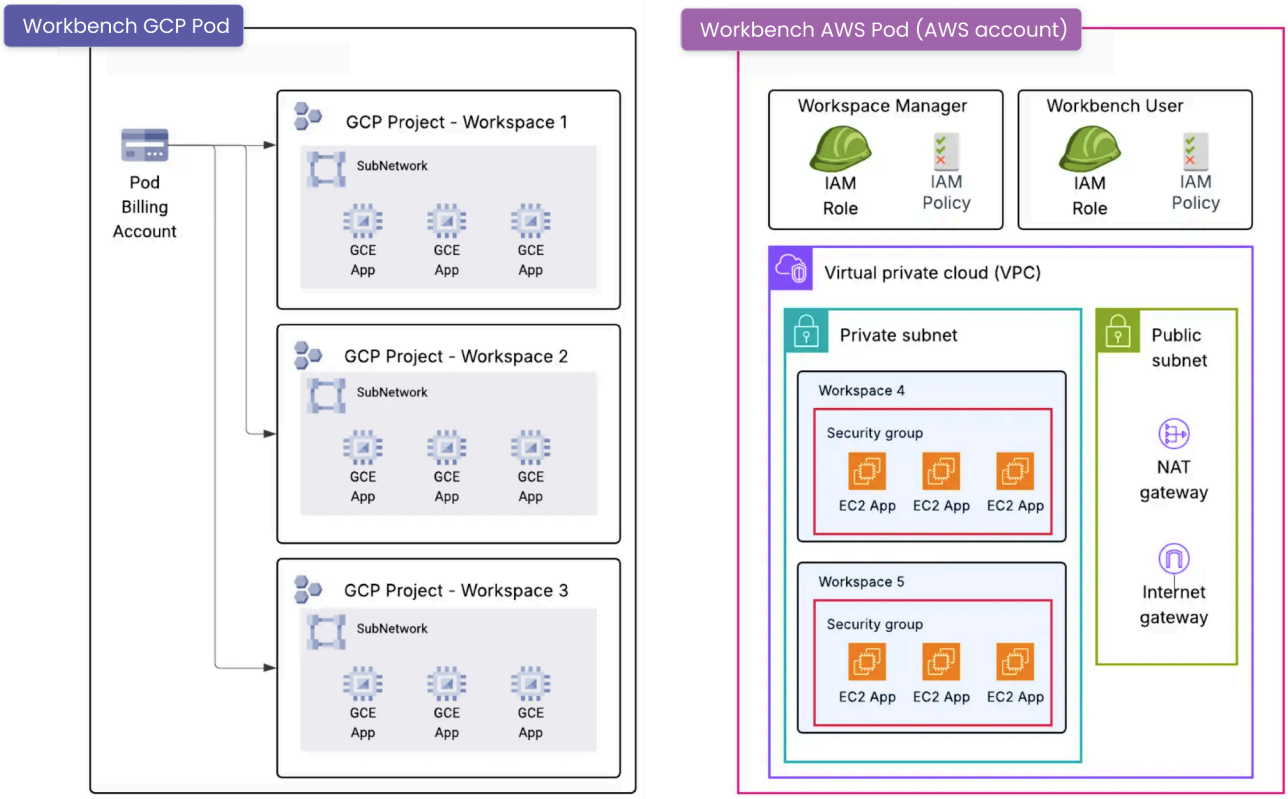
Our solution was to create the concept of a **landing zone**.

A landing zone is a set of shared, regional infrastructure that we provision inside the pod's AWS account. This infrastructure is defined and managed as "Infrastructure as Code" (IaC) using Workbench-authored Terraform modules and actuated by Verily's continuous deployment (CD) systems. These modules reproducibly provision the necessary "support resources" — VPCs, subnets, NAT gateways, and shared IAM policies — for each AWS region a Workbench pod supports.

When a user creates a new workspace, the Workspace Manager doesn't create a new account. Instead, it creates that workspace's controlled resources within the pod's shared account, and these new resources utilize the pre-existing landing zone infrastructure. For instance, if a user creates a Workbench cloud app in the AWS us-east-1 region, the EC2 virtual machine (VM) resource created by the Workspace Manager is placed in the landing zone's shared VPC subnet and attached to an instance profile. The VPC, subnet, and instance profile are all landing zone support resources created to facilitate workspace resource creation.

This fundamental difference in resource models, with a dedicated project per workspace on GCP versus a multi-workspace, shared account on AWS, is the core architectural distinction. It allows us to present a unified UX, but it dictates how we must implement isolation.

On GCP, the project provides a clear resource boundary. On AWS, we built and enforce that resource boundary in software.



Next, we'll explore how we handle the significant differences in each cloud provider's Identity and Access Management (IAM) strategies.

IAM: RBAC vs. ABAC

A core challenge for any multicloud platform is reconciling the unique, and often conflicting, Identity and Access Management (IAM) models of each provider.

It's important to distinguish between two separate layers of security. The first is API-level AuthNZ (Authorization and Authentication), which controls who can call the Workbench API itself. This layer is unified across all cloud platforms, using Auth0 for Customer Identity and Access Management (CIAM). When a user makes any API request (like creating a workspace), they must present a valid Auth0 access token. This system is identical for all cloud platforms.

The second and more complex layer is cloud resource IAM. Once our API has authenticated a user via Auth0 and authorized them through Workbench's internal authorization mechanisms, it must then enforce that user's permissions on the actual, native cloud resources (like GCS buckets or S3 buckets).

In Workbench we call this translation process "cloud sync." Authorization at the Workbench API level is uniform, but cloud sync is the specific mechanism that manifests Workbench's internal permission model (e.g., "this user is a 'writer' on this workspace") into native cloud permissions.

The how of this cloud sync process is completely different for each provider. For GCP, it's via manipulating Google Groups to implement Role-Based Access Control (RBAC); for AWS, it's via Attribute-Based Access Control (ABAC) and short-lived credentials.

Cloud sync on GCP

On Google Cloud, our cloud sync process implements an RBAC model using Google Groups as the bridge to native GCP permissions. When a new workspace is created in a GCP pod, the Workspace Manager creates a set of "policy groups" (e.g., `ws-12345-writers, ws-12345-readers`) as Google Groups and manages membership in these groups to match users' Workspace roles. When a user then creates a controlled resource, such as a GCS bucket, in the workspace, the Workspace Manager programmatically applies a native GCP IAM policy binding directly to that bucket. This binding grants the appropriate policy group a specific role.

Cloud sync on AWS

While our GCP strategy relies on Google Groups, we couldn't apply the same RBAC model to AWS without fighting the platform's core IAM philosophy. Google Groups act as a global identity provider (IdP), making them a natural principal for GCP. In contrast, AWS's recommended approach strongly favors using IAM Roles and temporary credentials over managing static permissions for long-lived, account-scoped IAM Groups.

To build an idiomatic and secure solution, Workbench's cloud sync for AWS was designed around this IAM Role-centric model. This choice led us to implement Attribute-Based Access Control (ABAC).

Our ABAC model works by ensuring tags on a user's session match the tags on the resource they're trying to access. First, whenever Workspace Manager creates a resource, it applies a tag identifying its workspace (e.g., `vwb-workspace-id: "12345"`).

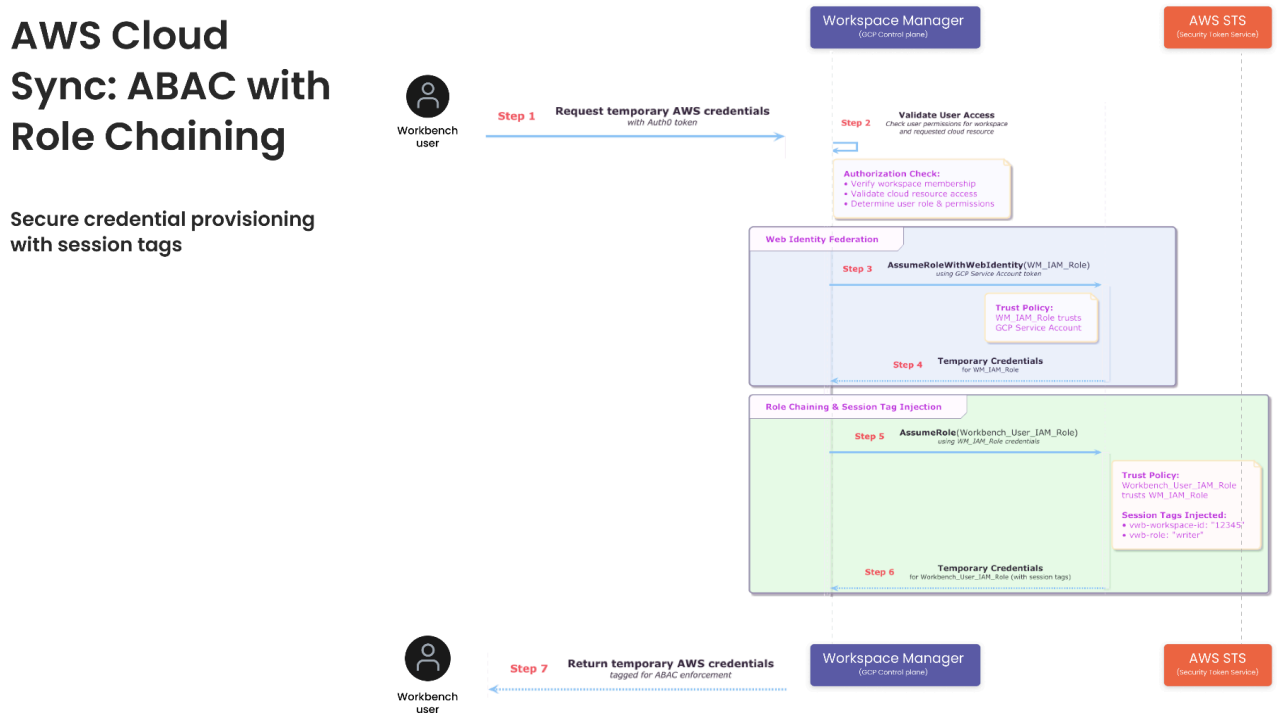
The second aspect involves creating the user's tagged session. A user, already authenticated via Auth0, calls a Workspace Manager endpoint to get temporary AWS credentials. This action initiates a secure, multi-step role-chaining process.

This chain begins when the Workspace Manager service (running in GKE as a GCP Service Account) uses Web Identity Federation to assume the Workspace Manager IAM Role in AWS. This IAM role is a support resource in the pod's landing zone, and its trust policy is locked down to only permit the Workspace Manager GCPS Service Account to assume it.

Next, this Workspace Manager IAM Role chains to (assumes) a second role, the Workbench User IAM Role (also a support resource in the landing zone). This second role's trust policy, in turn, only trusts the Workspace Manager IAM Role. This secure chain is critical, as it guarantees that only Workspace Manager has the power to generate these tagged user credentials, after having performed its own authentication and authorization logic.

As part of this final `AssumeRole` call, Workspace Manager injects the user's context (like `vwb-workspace-id: "12345"` and `vwb-role: "writer"`) as session tags into the temporary credentials it returns to the user.

The result is a simple, powerful, and highly scalable enforcement model. We don't need to create a new policy or group every time a user creates a workspace or resource on AWS. Instead, a single, central IAM policy (provided by the landing zone) enforces the rules for everyone, based purely on matching tags; in plain English: ***"Allow a user to perform action s3:ListObject on a bucket in this account, if and only if the bucket's vwb-workspace-id tag matches the user's vwb-workspace-id session tag."***



Tying it together

From the perspective of a user interacting with the Workbench UI or CLI, the experience is unified. They are a "writer" on workspace "`12345`" regardless of the cloud. Behind the scenes, the Workspace Manager does the heavy lifting: it validates their single, unified Auth0 identity, then uses the appropriate "cloud sync" process to translate that abstract fact to cloud-level truth. This helps bridge the gap between two completely different native permission models: GCP's group-based RBAC (with permissions set on resources persistently at create time) and AWS's tag-based ABAC (with permissions granted on-demand at access time).

We have not touched on the user experience of interacting with their Workbench controlled resources in the cloud directly; we will talk in depth about this in a later section of this post.
But first, this "cloud sync" model has one more major challenge to address. It perfectly handles permissions for resources that Workbench creates and controls, but what about data that already exists in the cloud, often in another account entirely? Next, we'll explore the challenge of providing secure access to these external data sources.

Accessing external data

The "cloud sync" model we've described is perfect for permissions on controlled resources — the ones Workbench creates and manages. Research, however, often relies heavily on referenced resources: data that already exists in the cloud, often in a different account or project owned by another team or institution.
Providing secure access to this external data is a critical function of Workbench. Once again, while the user's goal is the same ("add this external data to my workspace"), Workbench's implementation must adapt to the distinct cloud-native patterns of each platform.

External data on GCP

On Google Cloud, this process aligns with the same RBAC model used for controlled resources. Because Google Groups are global, first-class principals, sharing is straightforward. The owner of an external GCS bucket or BigQuery dataset simply updates their resource's IAM policy to grant the Workbench workspace's "policy group" (e.g., `ws-12345-writers`) the desired role.

The user then completes the connection by adding a "referenced resource" to their Workbench workspace, which points to the now-accessible external data. In this model, the responsibility rests almost entirely with the external data owner; Workbench's job is to provide a stable, group-based identity that the owner can discover and trust.

External data on AWS

On AWS, this simple, one-sided grant is not possible. The AWS IAM model requires a coordinated, two-way trust relationship for cross-account access. This means that for a user to access an external resource, permissions must be set on both sides: the external resource's policy must trust the Workbench account, and the user's IAM role within Workbench must be granted permission to access the resource.

To orchestrate our side of this handshake, the Workbench Landing Zone includes a dedicated "Workbench External Access" IAM Role. When a user needs to access an external resource, such as an S3 bucket (which they will add as a referenced resource), they assume this role. The policies attached to this role use ABAC, but with a different focus: instead of matching workspace tags, they use session tags that reference the account ID of the external resource and other key resource identifiers as appropriate.

On the other side, the owner of the external resource (like an S3 bucket) must update their resource-based policy to Allow access from the ARN of our Workbench pod's AWS account. This opens the first part of the trust.

This is where Workbench's ABAC model provides the crucial, final layer of security. A data owner doesn't want to grant access to everyone in the Workbench pod account; they want to grant it only to a specific workspace (or workspaces).

To enable this, when a user from workspace `"12345"` with role "writer" assumes the "External Access" role, Workbench injects a workspace-specific session tag where the tag key identifies the workspace and the value identifies their role: `vwb-12345: "writer"`

This tag structure is incredibly flexible, as it allows external resource owners to grant access to multiple workspaces. Their bucket policy can be fine-grained and secure, using a condition to check for the presence and value of these tags. In plain English, their policy can say: ***"I will Allow actions from the Workbench account, but only if the user's session has a tag with the key vwb-12345 (and a value of reader or writer), OR a tag with the key vwb-67890...".***

Unifying the experience

Our core philosophy is to deliver a unified Workbench experience that enables scientific workflows to run efficiently and reliably, maximizing value for researchers. We intentionally diverge from strict uniformity to highlight each provider's unique, specialized services. This approach avoids the "least common denominator" issue, where multicloud platforms support only features common to all providers.

We focus on unifying the experience for foundational services — the common building blocks like object storage (GCS, S3) and virtual machines (GCE, EC2). For example, from the Workbench UI, a user has a consistent way to preview object contents or get a link to the resource in its native console. When they launch a cloud analysis app (like JupyterLab), we can FUSE-mount their workspace's storage buckets, allowing their code to see a simple file path regardless of whether the underlying provider is GCP or AWS. This removes friction for a large fraction of the tasks that all researchers perform.

This hybrid "unify-and-diverge" philosophy also extends to how users interact with native cloud tooling. On GCP, the native experience is seamless. Because our "cloud sync" model provisions a Google Group for the workspace, a user can log in to the GCP console with their Google-backed identity (which is linked to their Auth0 account) and have the exact same permissions on their workspace projects that they have in Workbench.

On AWS, the native experience is different, by design. A user doesn't have a permanent IAM User. Instead, to interact with the AWS CLI or SDKs, they configure their tools to use the process credential provider. This delegates credential fetching to the Workbench CLI, which calls the Workspace Manager's credential endpoints to retrieve temporary, ABAC-tagged credentials on the user's behalf. Crucially, Workbench automates this entire configuration setup within its cloud analysis app environments (like JupyterLab), meaning the user's tools work out-of-the-box with zero friction. For console access to a service like S3, Workbench uses these same credentials to generate pre-signed URLs, allowing a user to access a bucket directly in their browser without ever "logging in" to the AWS console.

This aligns perfectly with each platform's idiomatic security model: one is based on a persistent, federated user identity, while the other is based on temporary, role-assumed credentials.

Conclusion

Verily Workbench's multicloud architecture is a significant technical milestone. It delivers a robust Trusted Research Environment that supports both GCP and AWS while maintaining security, governance, and platform-specific capabilities.

This foundation unlocks new possibilities for biomedical research collaboration. Academic institutions on GCP can now securely share data with pharmaceutical partners on AWS in a single governed workspace. Research consortia can reference datasets across cloud boundaries without migration or duplication. Organizations can adopt Workbench without re-platforming their existing infrastructure. This accelerates time to insight while maintaining compliance with data governance requirements.

By solving the challenges of multicloud identity management, resource isolation, and cross-account data access, we've created an architecture that scales beyond two cloud providers. The same abstraction layer that maps Workbench concepts to GCP Projects and AWS Landing Zones can extend to additional platforms. This ensures research organizations maintain flexibility as cloud ecosystems continue to evolve.

Learn more at verily.com/workbench-request