

Security Best Practices for Databricks on AWS

Version 1.0 - June 16 2022



Table of Contents

Introduction	4
Databricks architecture	4
Typical security configurations	4
Most deployments	4
Highly-secure deployments	5
Databricks threat model	5
Account takeover / account compromise	6
Data exfiltration	6
Accidental insider exposure	7
Resource abuse	7
Compromise of Databricks, Inc.	8
Security configuration reference	8
Authenticate via single sign-on	8
Leverage multi-factor authentication	9
Disable local passwords	9
Set complex local passwords	9
Separate admin accounts from normal user accounts	9
Use IP access lists	9
Configure AWS PrivateLink	10
Databricks audit log delivery	10
Token management	11
Deploy with a customer-managed VPC	11
Implement network exfiltration protections	11
Use VPC endpoint policies	12
Avoid storing production data in DBFS	12
Apply bucket policies	12
Leverage data exfiltration settings within the admin console	12
Use bucket versioning	12
Backup via the Databricks migration tool	12
Store code in Git	13
Manage code run via CI/CD	13
Configure a disaster recovery (DR) site	13
Isolate sensitive workloads into different workspaces	13
SCIM synchronization of users and groups	13
Encrypt S3 buckets and restrict access	13
Use AWS Nitro instances	14
Add a customer-managed key for managed services	14

Add a customer-managed key for workspace storage	14
Cross-account IAM role configuration	14
Monitor provisioning activities in AWS CloudTrail	15
Customer-approved workspace login (CAWL)	15
AWS service quotas	15
Use clusters that support user isolation	15
Cluster policies	16
Limiting cluster creation rights	16
Controlling libraries	16
Store and use secrets securely	16
Configure AWS tagging to monitor usage and enable charge-back	16
Resources	17

1. Introduction

Databricks has worked with thousands of customers to securely deploy the Databricks platform, with the appropriate security features that meet their architecture requirements. While many organizations deploy security differently, there are guidelines and features that are commonly used by organizations who need a high level of security.

This document details typical security features that are deployed by most high-security organizations, and reviews the largest risks and the risks that customers ask about most often. It will then provide a security configuration reference linked to our documentation and finally a collection of recommended resources.

This document is focused on the multi-tenant (E2) Databricks platform on AWS and assumes at least the Premium tier. It also contains some Terraform examples.

2. Databricks architecture

Databricks is a hybrid PaaS general-purpose data-agnostic compute platform.

The phrase hybrid PaaS applies because most customers deploy a data plane (virtual network and compute) in a cloud service provider account (such as AWS, Azure, or GCP) that is owned by the customer and so is single-tenant, while the multi-tenant control plane is run within Databricks account. Customers get the benefits of a PaaS platform with the option to keep your data processing clusters locally within your environment.

The phrase general-purpose data-agnostic means that unlike a pure SaaS service, Databricks doesn't know what data your teams process with the Databricks platform. The actual code, business logic, and the datasets are provided by your teams. You won't find recommendations like "truncate user IDs" because we don't know what data you're analyzing.

If you're new to the Databricks platform, start with an overview of the architecture and review of common security questions before you hop into specific recommendations. You'll see those at our [Security and Trust Center](#) and the [security and trust overview whitepaper](#).

3. Typical security configurations

Below, you will find the typical security configurations seen by Databricks. For simplicity, we've separated these into "most deployments" and "highly-secure deployments." Most deployments are as they sound – configurations that Databricks expects to be present in most production or enterprise deployments such as Single Sign-On (SSO) protected by multi-factor authentication (MFA). The configurations for highly-secure deployments are more representative of things seen in environments with very sensitive data, intellectual property, or in regulated industries such as Healthcare, Life Sciences, or Financial Services, such as usage of PrivateLink connectivity.

Importantly, these are recommendations based on the configurations we see from our customers, and following them doesn't guarantee that you will be "secure." Please review in the context of your overall enterprise security to determine what is required to secure your deployment and your data.

Most deployments

The following typical configurations are part of most enterprise production Databricks deployments. If you are a small data science team of a few people, you may not feel the need to deploy all of these. If Databricks may become a key part of your business or if you are analyzing sensitive data, we recommend that you review these.

- Evaluate whether [multiple workspaces](#) are required for segmentation
- Check that your [S3 buckets are encrypted and that public access is blocked](#)
- Deploy Databricks into a [customer-managed VPC](#) for increased control over the network environment. Even if you do not need this now, this option increases the chances for future success with your initial workspace
- [Authenticate via single sign-on](#)
- Use [multi-factor authentication](#)
- [Separate accounts with admin privileges](#) from day-to-day user accounts
- Configure [Databricks audit log](#) delivery
- Configure maximum token lifetimes for future tokens using [token management](#)
- Configure [admin console settings](#) according to your organization's needs
- Apply bucket policies or other mitigations to [avoid storing production datasets in DBFS](#)
- [Backup your notebooks stored in the control plane](#) or store your notebooks in [git repos](#)
- [Store and use secrets securely](#) in Databricks or using a third-party service
- Consider whether to [implement network protections for data loss](#)

Highly-secure deployments

In addition to the configurations typical to all deployments, the following configurations are often used in highly-secure Databricks deployments. While these are common, not all highly-secure environments use all of these settings. We recommend incorporating these items and the threat model in the following section alongside your existing security practices.

- Evaluate whether customer-managed encryption keys are needed on the [control plane](#) or [data plane](#) for control over data at rest (Requires Enterprise tie)
- Keep an up-to-date user list by using [SCIM](#)
- [Set complex local passwords](#) or [disable local passwords](#)
- Use either [IP access lists](#) or front-end [PrivateLink](#)
- Configure back-end (data plane to control plane) [PrivateLink](#) connectivity
- [Implement network protections for data exfiltration](#)
- Evaluate whether your datasets require [bucket versioning](#)
- Evaluate whether your workflow requires using [git repos](#) or [CI/CD](#)
- Plan for and deploy [a disaster recovery site](#) if you have strong continuity requirements
- Consider requiring AWS [Nitro instances](#) that provide encryption for ephemeral storage at rest and between instances
- Encourage the use of [clusters that support user isolation](#)
- Configure [cluster policies](#) to enforce data access patterns and control costs
- Evaluate [tagging](#) to monitor and manage chargeback and cost control

4. Databricks threat model

Databricks can at first appear to be an unusual architecture, but it aligns with typical PaaS security concerns. The most common threat categories that come up in customer conversations are, in priority order:

1. Account takeover or account compromise

2. Data exfiltration
3. Accidental insider exposure
4. Resource abuse such as crypto mining
5. Potential compromise of Databricks Inc.

This section addresses common questions about these risks, discusses probabilities, and provides mitigation strategies.

[Account takeover / account compromise](#)

Risk description

Databricks is a general-purpose compute platform that can sometimes be set-up by customers to access critical data sources. If the credentials belonging to a user at one of our customers were compromised by phishing, brute force, or similar, an attacker might get access to all of the data accessible from the environment.

Probability

Without proper protections, account takeover would be a good strategy for an attacker. Fortunately, it is easy to apply protection strategies that dramatically reduce the risk.

Mitigation strategies

1. Strongly authenticate users. The best defense to account takeover is strong authentication. Databricks recommends the following best practices:
 - [Authenticate using single sign-on](#) (SSO)
 - [Leverage multi-factor authentication](#) (MFA)
2. Protect administrative accounts with local passwords. When setting up Databricks, the workspace admin needs a local password (non-SSO) within Databricks. There are two options for protecting these accounts:
 - [Disable local passwords](#)
 - [Set complex local passwords](#)
3. Implement automation that removes or disables old employee accounts when they leave your company, reinforcing the controls mentioned above and terminating active sessions:
 - [SCIM](#) for User Deprovisioning
4. Restrict network access. Just like other SaaS or PaaS services, Databricks does not require that users log in from a specific network (like your office or VPN) unless you enable that configuration. If an account were compromised, having network access would make it easier for an attacker to Databricks. Mitigate this risk by either of the following steps:
 - Use [IP access lists](#)
 - [Configure private network connectivity between users and the control plane](#)
5. Monitor user activities. Monitor user activities to detect anomalies (such as unusual time of login, or simultaneous remote logins):
 - Configure [Databricks audit log](#) delivery
6. Manage personal access tokens for REST API authentication. Personal access tokens are a proxy for the user who generates it, allowing full privileges. Tokens should be controlled and protected as closely as you would protect a user's credential. This is possible with:
 - [Token management](#)

[Data exfiltration](#)

Risk description

If a valid user or an attacker can log into the environment, a common action on objective is to exfiltrate sensitive data from the environment, after which the attacker might store it, sell it, or ransom it.

Probability

Generally, the probability of this attack category is low because it presumes either a malicious insider or account compromise. However, if there is a malicious insider or compromised account, attackers would likely attempt to exfiltrate data.

Mitigation strategies

1. Network protections
 - [Implement network exfiltration protections](#) to limit where data can be sent from Databricks
 - [Use VPC endpoint policies](#) to limit where data can be written from Databricks
2. Control data access
 - [Avoid storing production data in DBFS](#) as it is accessible via API and CLI
 - [Apply bucket policies](#) to control access to your data from outside of Databricks
3. Use [data exfiltration settings within the Admin Console](#) to prevent simple methods for exfiltration

Accidental insider exposure

Risk description

High-performing engineers will generally find the best or fastest way to complete their tasks, but sometimes that way may create security impacts to organizations. One user may think that their job would be much easier if they didn't have to deal with security controls, or another might helpfully copy some data to a public S3 bucket to simplify sharing of data. We can provide education for these users, but companies should also consider providing guardrails.

Probability

There is significant variability in the likelihood and the impact of individual errors in this category, but most security professionals identify this as a significant potential risk.

Mitigation strategies

1. Backup data and code
 - [Use S3 bucket versioning](#) so that highly-sensitive deleted data can be recovered
 - [Backup notebooks in your environment](#) using the `migrate` tool
2. Use software development lifecycle (SDLC) processes to control what code is executed
 - [Store production code in Git](#) that you can access via the Databricks Repos feature
 - Use [a CI/CD process](#) that pushes only authorized code to production
3. Make sure your users have the access necessary
 - [Use SCIM](#) for user de-provisioning and group management
 - Monitor the [Databricks audit log](#) to identify what users are logging in, and what types of clusters they're configuring
 - Use clusters that [support user isolation](#)
4. Deploy [data exfiltration protections](#), as the protection they provide against accidental insider exposure is similar to that provided against a malicious attacker

Resource abuse

Risk description

Databricks can deploy large amounts of compute power. As such, it could be a valuable target for crypto mining if a customer's user account were compromised.

Probability

This has not been a prominent activity in practice, but customers will sometimes bring up this concern.

Mitigation strategies

1. Native AWS protections
 - Apply the restricted [cross-account IAM role](#) provided in our documentation to limit the risk of IAM role compromise
 - Use [AWS service quotas](#) to limit the resources that can be deployed
 - Regularly monitor usage data in AWS and in Databricks
 - Regularly [monitor CloudTrail](#) to identify which AMIs are launched
2. Databricks protections
 - Use [cluster policies](#) to limit the maximum size and type of a cluster
 - Limit which users can [create clusters](#)
 - [Control the libraries](#) that can be used in the environment to limit the risk of compromised libraries
3. Monitoring controls
 - Monitor utilization of your deployment with [Overwatch](#)
 - Monitor the [Databricks audit log](#) to identify what users are logging in, and what types of clusters they're configuring

Compromise of Databricks, Inc.

Risk description

Security-minded customers sometimes voice a concern that Databricks itself might be compromised, which could result in the compromise of their environment.

Probability

Databricks has an extremely strong security program which manages the risk of such an incident – see our [Security and Trust Center](#) for an overview on the program and the security features in the Databricks product. However, the risk for any company is never completely eliminated.

Mitigation strategies

1. Databricks controls
 - Monitor the [Databricks audit log](#) to identify the activities of Databricks employees who provide support to your deployment
 - Consider enabling [customer-approved workspace login](#)
2. AWS controls
 - Apply the restricted [cross-account IAM role](#) provided in our documentation to limit the risk of IAM role compromise
 - [Monitor CloudTrail](#) to identify abnormal provisioning activity
3. Process controls
 - Review the [Databricks security documentation](#)
4. Prepare “break-glass” customer-controls in the event of an active compromise
 - Understand how to disable the ability of Databricks to deploy resources within your account by removing the [cross-account IAM role](#)
 - Disable a Databricks workspace to render applicable data unreadable by revoking a [customer-managed key for managed services](#) (not guaranteed to be a reversible operation)

5. Security configuration reference

Here are the security configurations referenced in this document.

Authenticate via single sign-on

Databricks [supports SAML 2.0](#) to connect with your identity provider (IdP), which can enforce lockout and password policies.

Leverage multi-factor authentication

Most identity providers (IdPs) either directly provide or integrate with multi-factor authentication (MFA) validations. The most highly-secure Databricks customers require an MFA prompt during user login, either at the Databricks prompt or through a VPN requirement.

For the highest security environments, Databricks also advocates where possible for the use of physical authentication tokens such as FIDO2 keys, that augment traditional Multi-Factor authentication by requiring interaction with a physical token that cannot be compromised.

Disable local passwords

Customers can [configure password access control](#), which restricts who is allowed to log in with a local password. By default, admins can use local passwords, but that ability can be restricted to allow authentication only through SSO.

Set complex local passwords

Either in place of or in addition to disabling local passwords, Databricks recommends that the local passwords configured for admin accounts be very long and complex, and securely stored.

Separate admin accounts from normal user accounts

It is a general best practice across all of security that an administrator should not use their privileged accounts to perform day-to-day tasks. This best practice applies to Databricks as well. If you have Databricks administrators who are also normal users of the Databricks platform (for example, there's a lead data engineer who administers the platform and also does data engineering work), Databricks recommends creating a separate account for administrative tasks.

Use IP access lists

[Configure IP access lists](#) that restrict the IP addresses that can authenticate to Databricks by checking if the user or API client is coming from a known good IP address range such as a VPN or office network. Established user sessions do not work if the user moves to a bad IP address, such as when disconnecting from the VPN.

IP access lists can be configured by using Terraform:

```
data "http" "my" {
  url = "https://ifconfig.me"
}

resource "databricks_workspace_conf" "this" {
  custom_config = {
    "enableIpAccessLists": true
  }
}

resource "databricks_ip_access_list" "only_me" {
  label = "only ${data.http.my.body} is allowed to access workspace"
  list_type = "ALLOW"
  ip_addresses = ["${data.http.my.body}/32"]
  depends_on = [databricks_workspace_conf.this]
}
```

This feature requires the Enterprise pricing tier.

Configure AWS PrivateLink

AWS [PrivateLink](#) provides a private network route from one AWS environment to another. PrivateLink can be configured both between Databricks users and the control plane, and also between the control plane and the data plane.

Between Databricks users and the control plane, PrivateLink provides strong controls that limit the source for inbound requests. If a company already routes traffic through an AWS environment, they can use PrivateLink so that the communication between users and the Databricks control plane does not traverse public IP addresses.

This feature requires the Enterprise pricing tier.

Databricks audit log delivery

[Configure Databricks to store the audit log](#) in an S3 bucket under your control. You can analyze the log by using a Databricks notebook or through a third-party security analytics product. Monitor user activities to detect anomalies such as unusual time of login or simultaneous remote logins.

Databricks also recommends that customers review the blog post [monitoring your Databricks Lakehouse platform with audit logs](#). This post shows common detections and provides a working data pipeline to monitor the Databricks audit log.

Audit log delivery can be configured by using Terraform:

```
variable "databricks_account_id" {
  description = "Account ID. Find your ID in the the bottom left corner of the account console at https://accounts.cloud.databricks.com/"
}

resource "aws_s3_bucket" "logdelivery" {
  bucket = "${var.prefix}-logdelivery"
  acl    = "private"
  versioning {
    enabled = false
  }
  force_destroy = true
  tags = merge(var.tags, {
    Name = "${var.prefix}-logdelivery"
  })
}

resource "aws_s3_bucket_public_access_block" "logdelivery" {
  bucket                = aws_s3_bucket.logdelivery.id
  ignore_public_acls    = true
}

data "databricks_aws_assume_role_policy" "logdelivery" {
  external_id = var.databricks_account_id
  for_log_delivery = true
}

resource "aws_iam_role" "logdelivery" {
  name           = "${var.prefix}-logdelivery"
  description    = "(${var.prefix}) UsageDelivery role"
```

```

    assume_role_policy = data.databricks_aws_assume_role_policy.logdelivery.json
    tags                = var.tags
  }

data "databricks_aws_bucket_policy" "logdelivery" {
  full_access_role = aws_iam_role.logdelivery.arn
  bucket           = aws_s3_bucket.logdelivery.bucket
}

resource "aws_s3_bucket_policy" "logdelivery" {
  bucket = aws_s3_bucket.logdelivery.id
  policy = data.databricks_aws_bucket_policy.logdelivery.json
}

resource "databricks_mws_credentials" "log_writer" {
  account_id      = var.databricks_account_id
  credentials_name = "Usage Delivery"
  role_arn        = aws_iam_role.logdelivery.arn
}

resource "databricks_mws_storage_configurations" "log_bucket" {
  account_id                = var.databricks_account_id
  storage_configuration_name = "Usage Logs"
  bucket_name               = aws_s3_bucket.logdelivery.bucket
}

resource "databricks_mws_log_delivery" "audit_logs" {
  account_id = var.databricks_account_id
  credentials_id = databricks_mws_credentials.log_writer.credentials_id
  storage_configuration_id =
databricks_mws_storage_configurations.log_bucket.storage_configuration_id
  delivery_path_prefix = "audit-logs"
  config_name = "Audit Logs"
  log_type = "AUDIT_LOGS"
  output_format = "JSON"
}

```

Token management

Customers can use the [Token Management](#) API or UI controls to enable or disable personal access tokens (PATs) for REST API authentication, limit the users who are allowed to use PATs, set the maximum lifetime for new tokens, and manage existing tokens. Highly-secure customers typically provision a maximum token lifetime for new tokens for a workspace.

This feature requires the Enterprise pricing tier.

Deploy with a customer-managed VPC

By default, Databricks VPCs are managed by Databricks, which does not allow customers to implement Databricks into their existing network architecture. You can deploy Databricks into a [customer-managed VPC](#) that enables customers to route traffic through network enforcement points (such as firewalls or proxies) and to control data access using VPC endpoints.

Important note: while it is possible to deploy Databricks with public IP addresses, Databricks recommends private IP addresses and it is very unusual for a security-conscious customer to prefer public IPs.

Implement network exfiltration protections

By default, Databricks data plane hosts within your AWS environment have unlimited outbound network access. If you use a [customer-managed VPC](#), you can lock down outbound access. Databricks has created a [blog post](#) that discusses how to do this using the AWS firewall, but it can be generalized to other network security tools [using details in Databricks documentation](#).

The TLS connections between the control plane and the data plane cannot be broken, and so it's not possible to use a technology like SSL or TLS inspection. The custom TLS certificate that would be needed cannot be pre-loaded on the Databricks AMI that is built for all customers.

Use VPC endpoint policies

Apply network configurations that limit the resources that are accessible via VPC endpoints. For example, you can limit the S3 buckets where data can be written, which would prevent attackers from writing to unauthorized S3 buckets. Databricks can share examples as a part of our *Data Exfiltration Protections for Databricks on AWS* whitepaper.

Avoid storing production data in DBFS

By default, DBFS is a filesystem that is accessible to all users of the given workspace and can be accessed via API. This is not necessarily a major data exfiltration concern as you can limit access to accessing data via the DBFS API or the Databricks cli using IP access lists or private network access. However, as use of Databricks grows and more users join a workspace, those users would have access to any data stored in DBFS, creating the potential for undesired information sharing. Databricks recommends that our customers do not store production data in DBFS, and Databricks staff can share a bucket policy for AWS that limits access to DBFS.

Apply bucket policies

Control where [buckets can be accessed from](#) by using [S3 bucket policies](#), which prevent direct access to storage buckets from outside of Databricks. This blocks an authorized user who connects from an uncontrolled EC2 instance either through misconfiguration or an intentionally flexible policy.

Leverage data exfiltration settings within the admin console

The [admin console](#) contains [a variety of settings](#) that provide protection. Most admin console controls are simple enable/disable buttons. Some of the most important ones are:

- Export notebooks or cells containing code and partial interactive query results
- Download notebook results
- Block notebook clipboard features
- Download MLflow run artifacts
- Block application attacks via iFrames and cross-site scripting

Use bucket versioning

AWS [S3 bucket versioning](#) stores older versions of bucket data. If through a malicious insider (or operator error) your datasets in S3 were to be accidentally deleted, bucket versioning would allow you to recover that data.

Because of how Databricks stores data in DBFS, S3 bucket versioning is not recommended for the bucket you configure for DBFS.

Monitor workspace using Overwatch

[Overwatch](#) was built to enable Databricks' customers, employees, and partners to quickly / easily understand operations within Databricks deployments. As enterprise adoption increases there's an ever-growing need for strong governance. Overwatch means to enable users to quickly answer questions and then drill down to make effective operational changes.

Most use cases are not strictly security-focused, but improved visibility strengthens all security teams. For example, suppose a PyPi library you incorporate is compromised by crypto miners, you would be grateful to have a tool that excels at troubleshooting heavy utilization within your Databricks deployment.

Backup via the Databricks migration tool

The [Databricks migration tool](#) allows admins to export most portions of their workspace and then import them into a new workspace. It is often used as a part of [disaster recovery](#) strategies to perform batch analytics of notebook code (such as secret hunting) or as a general-purpose backup of code stored in the Databricks control plane.

Guidance around these tools:

- [migrate](#) is a tool to migrate a workspace one time. It uses the Databricks CLI/API in the background.
- [databricks-sync](#) is a tool that has been used for multi cloud migrations, as well as disaster recovery synchronization of workspaces. It uses the Terraform provider to synchronize incremental changes.
- You can run either tool from a command line or from a notebook.

Store code in Git

The [Databricks Repos](#) feature allows you to move away from data stored just in the Databricks control plane, and instead store data in a Git repo. You can check code in/out and switch branches. Use Repos for better code visibility and tracking.

Manage code run via CI/CD

Mature organizations often build [production workloads by using CI/CD](#) to integrate code scanning, better control permissions, perform linting, and more. When there is highly sensitive data analyzed, a CI/CD process can also allow scanning for known scenarios such as hard coded secrets.

Configure a disaster recovery (DR) site

While Databricks doesn't offer disaster recovery services, many customers use Databricks capabilities including the Account API to create a cold (on standby) workspace in another region. We have [documentation](#) to guide customers through this process.

Isolate sensitive workloads into different workspaces

While Databricks has numerous capabilities for isolating different workloads, such as table ACLs and IAM passthrough for very sensitive workloads, the primary isolation method is to move sensitive workloads to a different workspace. This sometimes happens when a customer has very different teams (for example, a security team and a marketing team) who must both analyze different data in Databricks.

SCIM synchronization of users and groups

SCIM (System for Cross-domain Identity Management) allows you to sync groups and user status between your SAML 2.0 identity provider (IdP) and Databricks. There are four major benefits of this system:

1. When you remove a user, the user is automatically removed from Databricks. Any active sessions are terminated

2. Users can also be disabled temporarily via SCIM. Customers have used this capability for scenarios where customers believe that an account may be compromised and need to investigate
3. Groups are automatically synchronized
4. Setting up SCIM enables SCIM-based Identity Passthrough, which can be an effective way to provision data access for some organizations

Encrypt S3 buckets and restrict access

S3 buckets are used for two roles within a Databricks deployment: the root S3 bucket that you provide when you configure Databricks initially and additional buckets where you store your data. For these buckets, it is your responsibility to verify that the buckets are encrypted (either [by default](#), or [for each bucket individually](#)) and that [public access is not allowed](#). As you are responsible for the S3 buckets, you must ensure that these buckets are correctly configured and that you control access to them.

Use AWS Nitro instances

AWS Nitro instances can provide two major security benefits:

1. AWS Nitro instances use NVMe disks that automatically encrypt data at rest. From [AWS docs](#) as of July 19 2021: The data on NVMe instance storage is encrypted using an XTS-AES-256 block cipher implemented in a hardware module on the instance. The encryption keys are generated using the hardware module and are unique to each NVMe instance storage device. All encryption keys are destroyed when the instance is stopped or terminated and cannot be recovered. You cannot disable this encryption and you cannot provide your own encryption key.
2. Many AWS Nitro instances also automatically encrypt data in transit between hosts. You can configure instance types included in the [Encryption in Transit section](#) of the AWS Nitro documentation so that intra-cluster (inter-host) traffic will be encrypted in-transit.

Databricks cannot authoritatively provide detail on capabilities in AWS, and the information above is provided on a best-effort basis as a convenience to Databricks customers.

Add a customer-managed key for managed services

Add a [managed services customer-managed key](#) for select encrypted data stored within the Databricks control plane, such as notebooks, secrets, Databricks SQL queries, and Databricks SQL query history.

Databricks requires direct access to this key through an AWS IAM role for ongoing operations. You can revoke access to the key to prevent Databricks from accessing encrypted data within the control plane (or in our backups). This is like a “nuclear option” where the workspace ceases to function, but it provides an emergency control for extreme situations.

This feature requires the Enterprise pricing tier.

Add a customer-managed key for workspace storage

Add a [workspace storage customer-managed key](#) for select encrypted data stored within the data plane, such as EBS volumes and the S3 bucket used as the root bucket for DBFS. Databricks requires direct access to this key via an AWS IAM role for ongoing operations, but a customer-managed key helps meet compliance requirements and allows you to revoke access when required.

This feature is only available at the Enterprise pricing tier.

Cross-account IAM role configuration

Databricks on AWS uses a [cross-account IAM role](#) so that the Databricks account can take actions inside of your AWS account. Customers often ask if it is possible to restrict permissions provided in the default role.

The first step when restricting permissions is to check that you have the right starting point. By default, [our documentation](#) shows an IAM Policy with a simplified set of permissions. However, you can get a much longer and more restrictive policy by clicking on the “Your VPC, custom” tab. This policy typically addresses the needs of customers without any additional adjustments.

If you want to further tweak permissions, the next key resource is the AWS E2 Enterprise Security Guide, which is available from your Databricks account team. In this document, there is a list of the permissions that are required in order to use Databricks, with a description for why Databricks needs each capability. The right column lists when we require each role:

- Launch: for initial workspace creation
- Repair: to fix some issues either automatically or manually through support
- Termination: during workspace termination
- Operational: for day-to-day usage

Some capabilities are required only at launch and can be removed later for day-to-day operational usage. It's important to note that deviating from the standard set of IAM roles described in our docs can create support issues. Changing these permissions are not supported, and could create undefined behavior. If you run into issues, revert to the documented policy.

Monitor provisioning activities in AWS CloudTrail

A core notion in security is that you cannot purely trust the system to tell you when it is compromised. You must be able to observe the system from the outside. The [Databricks audit log](#) is an extremely valuable feature for monitoring what users do, but many customers want an outside resource to help monitor that Databricks itself doesn't do something wrong.

Cloud provider audit logs such as CloudTrail provide a great mechanism for observing the actions of Databricks (and users) in the data plane. It provides visibility into:

- Instance creation, to help identify bitcoin mining and also control for billing.
- Outbound network connections, to help identify data exfiltration.
- APIs called by the AWS account, to help identify account/key compromise.

Most customers have favorite tools in place to analyze cloud provider log data, but you can also analyze this in Databricks. Databricks can also share a document detailing recommendations for AWS CloudTrail monitoring and has a blog post covering [building ETL pipelines for the cybersecurity lakehouse with Delta Live Tables](#).

Customer-approved workspace login (CAWL)

Enabling customer-approved workspace login for a Databricks account, by default, disables workspace access to all the relevant workspaces from Databricks engineers and support staff. Workspace admins can temporarily enable support access to a restricted workspace within the account for up to 48 hours if required. Find details about CAWL [in our documentation](#).

AWS service quotas

While a very coarse control, [AWS service quotas](#) provide an overarching control to prevent excessive resource consumption.

Use clusters that support user isolation

When Databricks started, standard mode clusters were the default because our customers were small engineering teams who had the same access to the same datasets. But over the years, we built out new user isolation capabilities that meet the security needs of our customers today, and so recommend utilizing clusters that support user isolation if possible.

The following types of clusters will enforce user isolation so that users with different privilege levels can coexist on the same cluster:

- High concurrency clusters with [table access control lists](#) (Table ACLs clusters for short)
- High concurrency clusters with [credential passthrough](#)
- SQL endpoints
- For workspaces with Unity Catalog enabled, any cluster that does not have [cluster security mode](#) set to “None”

Excluding SQL Endpoints which operates in a specialized manner for SQL queries, the above cluster types include enforcement such that each user leverages a different non-privileged user account on the cluster host, languages are limited to those that can be implemented in an isolated manner (SQL and Python), and Spark APIs accessible to code must be on an allowlist believed to be safe for isolation.

Customers with more stringent security requirements can enforce [cluster policies](#) that do not allow standard clusters to be created within the environment.

If you need standard clusters (for example, for certain machine learning capabilities), customers whose workspaces have Unity Catalog enabled can allow users to create single-user clusters. Administrators for workspaces without unity catalog can create clusters and use Cluster ACLs to control the users permitted to attach notebooks.

Cluster policies

Databricks admins can control many aspects of the clusters that are spun up, including available instance types, Databricks versions, and the size of instances by using [cluster policies](#). Admins can enforce some Spark configuration settings. Admins can configure multiple cluster policies, allowing certain groups of users to create small clusters, some groups of users to create large clusters, and other groups to only use existing clusters.

Limiting cluster creation rights

Using either [cluster policies](#) or the older [cluster ACLs](#), admins can define what users or groups within the organization are able to create clusters.

Cluster ACLs allow you to specify which users can attach a notebook to a given cluster. Note that if a user shares a notebook already attached to a standard mode cluster, the recipient will also be able to execute code on that cluster. This does not apply to clusters that enforce user isolation: SQL endpoints, high concurrency with table ACLs clusters, and high concurrency with credential passthrough clusters. Customers who use the preview of Unity Catalog can also enable single-user clusters to enforce isolation clusters.

Controlling libraries

By default, Databricks allows customers to install Python, R, or scala libraries from the standard public repositories, such as pypi, CRAN, or maven.

Those who are concerned about supply-chain attacks, can host their own repositories and then configure Databricks to use those instead. You can block access to other sources of libraries. Documentation for doing so is outside the scope of this document, but reach out to your Databricks team for assistance as required.

Store and use secrets securely

Integrating with heterogeneous systems requires managing a potentially large set of credentials and safely distributing them across an organization. Instead of directly entering your credentials into a notebook, use Databricks secrets to store your credentials and reference them in notebooks and jobs. [Databricks secret management](#) allows users to use and share credentials within Databricks securely. You can also choose to use a third party secret management service, such as AWS Secrets Manager or a third party secret manager.

Configure AWS tagging to monitor usage and enable charge-back

To track Databricks usage through to AWS resource billing can [configure tagging](#) on clusters or pools. These can also be enforced via [cluster policies](#) for different groups within your organization.

6. Resources

Many different capabilities have been discussed in this document, with documentation links where possible. Organizations who prioritize high security can learn more than what is in this document. Here are additional resources to help you learn more:

1. Request the *Enterprise Security Guide* and compliance documentation from your Databricks account team.
2. Review the security features in the [Security and Trust Center](#), along with the overall documentation about the Databricks security and compliance programs.
3. The [Security and Trust Overview Whitepaper](#) provides an overview of the Databricks architecture and platform security practices.
4. Documentation article: [E2 Security Overview](#)
5. Blog: [Security Best Practice for AWS on Databricks. \(Azure equivalent\)](#)
6. Whitepaper: Data Exfiltration Protections for Databricks on AWS (request from your Databricks Account Team)
 - a. See also our blog on [Network Protections for Databricks on AWS](#)
7. Whitepaper: Data Plane Host Security Summary (request from your Databricks Account Team)
8. Documentation article: [Customer support access](#), including customer-approved workspace login.