AWS Startup Security Baseline

AWS Prescriptive Guidance

AWS Prescriptive Guidance: AWS Startup Security Baseline

Table of Contents

AWS Startup Security Baseline

Amazon Web Services (contributors)

April 2026 (document history)

The AWS Startup Security Baseline (AWS SSB) is a set of controls that establish a **foundational security baseline** for startups building on AWS. It is designed to reduce the most common security risks without adding significant operational overhead. The controls in this guide cover securing credentials, enabling logging and visibility, managing contact information, and implementing basic data boundaries.

The controls in this guide are designed with early-stage startups in mind. Many startups start on AWS with a single AWS account. As startups grow, they migrate to multi-account architectures. This guide is designed for single-account architectures. The controls are structured so they can be adapted as you transition to a multi-account architecture.

The AWS SSB organizes controls into two categories: account and workload. *Account* controls help keep your AWS account secure. They include recommendations for setting up user access, policies, and permissions, and include recommendations for monitoring your account for unauthorized or potentially malicious activity. *Workload* controls help secure your resources and code in the cloud, such as applications, backend processes, and data. They include recommendations such as encryption and reducing the scope of access.

> **Note**
>
> This guide does not cover all available security controls. It focuses on the foundational controls most relevant to early-stage startups. Some of the controls recommended in this guide replace the defaults configured during initial setup, while most configure new settings and policies.

Intended audience

This guide is designed for startups in the earliest stages of development, typically pre-revenue or early-revenue companies, with minimal staff and operations.

Startups or other businesses that are in later stages of operation and growth can also benefit from reviewing these controls against their current practices. If you identify any gaps, you can

implement the individual controls in this guide and evaluate them for appropriateness as a long-term solution.

> **Note**
>
> The recommended controls in this guide are foundational in nature. Startups or other companies operating at a later stage of scale or sophistication should implement additional controls beyond this baseline. For more advanced guidance, see the AWS Security Reference Architecture provided by AWS Prescriptive Guidance.

Foundational framework and security responsibilities

AWS Well-Architected provides guidance for building cloud infrastructure that meets security, reliability, performance, and cost requirements. The AWS SSB aligns to the security pillar of the AWS Well-Architected Framework. The *security pillar* provides guidance on protecting data, systems, and assets using AWS services and features.

You can assess your adherence to Well-Architected best practices by using the AWS Well-Architected Tool in your AWS account.

Security and compliance are a shared responsibility between AWS and the customer. Under the shared responsibility model, AWS is responsible for the security *of* the cloud (that is, protecting the infrastructure that runs all AWS Cloud services). You are responsible for the security *in* the cloud, as determined by the AWS services you select. The controls in this guide help you fulfill your responsibilities under the shared responsibility model.

Securing your account

Controls and recommendations in this section help keep your AWS account secure. They cover using AWS Identity and Access Management (IAM) users and roles (also known as *principals*) for both human and machine access, restricting the use of the root user, and requiring multi-factor authentication. In this section, you confirm that AWS has the contact information necessary to reach you regarding your account activity and status. You also set up monitoring services, such as AWS Trusted Advisor, AWS Identity and Access Management Access Analyzer, and AWS Budgets, so that you are notified of account activity and can respond if unauthorized or unexpected activity occurs.

This section contains the following topics:

ACCT.01 Set account-level contacts to valid email distribution lists

When setting up primary and alternate contacts for your AWS account, use an email distribution list instead of an individual's email address. Using an email distribution list makes sure that

ownership and reachability are preserved as individuals in your organization come and go. Set alternate contacts for billing, operations, and security notifications, and use appropriate email distribution lists accordingly. AWS uses these email addresses to contact you. Make sure that you retain access to them.

To update your account name, root user password, or root user email address

1. Sign in to the AWS Management Console
2. Choose your account name or number, and then choose **Account**.
3. On the Account page, next to **Account details**, choose **Actions**, and then choose the action you want to take.
4. Next to the field you want to update, choose **Edit**.
5. After you have entered your changes, choose **Save changes**.
6. If you are updating the root user password or email address, follow the verification steps that AWS displays.

To edit your contact information

1. On the Account page, under **Contact information**, choose **Edit**.
2. For the fields you want to change, enter your updated information, and then choose **Update**.

To add, update, or remove alternate contacts

1. On the Account page, under **Alternate Contacts**, choose **Edit**.
2. For the fields you want to change, enter your updated information, and then choose **Update**.

If you have an organization in AWS Organizations enabled, you can also programmatically manage the alternate contacts on your accounts through the AWS Command Line Interface (AWS CLI). For more information, see Programmatically managing alternate contacts on member accounts with AWS Organizations on the AWS Cloud Operations Blog.

ACCT.02 Restrict use of the root user

The AWS account root user is created when you sign up for an AWS account, and this user has full ownership privileges and permissions over the account that cannot be changed. Use the root user

exclusively for tasks that require root user credentials. For more information, see Tasks that require root user credentials in the IAM documentation. Perform all other actions in your account by using other types of IAM identities, such as federated users with IAM roles. For more information, see AWS security credentials in the IAM documentation.

To restrict use of the root user

1. Require multi-factor authentication (MFA) for the root user. For more information, see ACCT.05 Require multi-factor authentication (MFA) to log in.
2. Create an administrative user so that you don't use the root user for everyday tasks. For more information about configuring user access, see ACCT.03 Configure console access for each user.

ACCT.03 Configure console access for each user

AWS recommends using temporary credentials to grant access to AWS accounts and resources. *Temporary credentials* have a limited lifetime, so you do not have to rotate them or explicitly revoke them when they're no longer needed. For more information, see Temporary security credentials in the IAM documentation.

For human users, AWS recommends using federated identities from a centralized identity provider (IdP), such as AWS IAM Identity Center, Okta, Active Directory, or Ping Identity. Federating users allows you to define identities in a single, central location, and users can securely authenticate to multiple applications and websites, including AWS, by using a single set of credentials. For more information, see Identity federation in AWS and IAM Identity Center.

> **Note**
>
> Identity federation can complicate the transition from a single-account architecture to a multi-account architecture. It is common for startups to delay implementing identity federation until they have established a multi-account architecture managed in AWS Organizations.

To set up identity federation using IAM Identity Center

1. See Getting started in the IAM Identity Center documentation.
2. Make sure that your IdP enforces multi-factor authentication (MFA).

3. Apply permissions according to ACCT.04 Assign permissions.

If you are using an external or third-party IdP, see Identity providers and federation in the IAM documentation.

If your startup is not yet ready to configure identity federation, you can create IAM users directly as a starting point. Creating IAM users with long-term credentials is not a security best practice. Long-term credentials do not expire automatically, which increases the risk of credential exposure if they are not rotated regularly. When your startup is ready to transition to a multi-account architecture managed in AWS Organizations, migrating from IAM users to federated identities will require additional planning.

As a baseline, create an IAM user with a username, password, and multi-factor authentication (MFA) for each human operator. Do not share credentials across users, and rotate long-term credentials on a regular schedule.

To create an IAM user

1. Follow the steps in Create an IAM user in your AWS account in the IAM documentation.
2. Apply permissions according to ACCT.04 Assign permissions.

> ⚠ **Warning**
>
> IAM users have long-term credentials, which presents a security risk. To help mitigate this risk, provide these users with only the permissions they require to perform their tasks and remove these users when they are no longer needed. Avoid creating long-lived access keys for IAM users. Instead, use temporary credentials through `aws login` to access the AWS CLI and SDKs, even when using IAM user credentials. This provides the same secure authentication while eliminating the risks associated with long-lived credentials. For more information about CLI and SDK access methods, see ACCT.13 Use short-lived credentials for access to your AWS resources.

ACCT.04 Assign permissions

Configure user permissions by attaching AWS managed policies to IAM roles. *AWS managed policies* are standalone policies designed by AWS to provide permissions for many common use

cases. If you customize permissions, follow the security best practice of granting least privilege. *Least privilege* is the practice of granting the minimum set of permissions that each user needs to perform their tasks. Examples of roles for early-stage startups include administrator, developer, contractor, and finance team member. Create specialized roles as specific job functions are identified.

If you are using federated identities, users access the account by assuming an IAM role through the external identity provider. The IAM role defines the actions that users authenticated by your organization's IdP can perform. Apply custom or AWS managed policies to this role to configure permissions.

To assign permissions for federated identities using IAM Identity Center

1. See Use IAM policies in permission sets in the IAM Identity Center documentation.
2. If you are using an external or third-party IdP, see Adding IAM identity permissions in the IAM documentation.

   If you are using IAM users, configure IAM roles for the work your users perform, and have users assume those roles rather than attaching policies directly to individual IAM users. When an IAM user assumes a role, they receive temporary credentials that automatically expire. This reduces the risk of credential exposure compared to policies attached directly to IAM users, which remain in effect until explicitly removed.

ACCT.05 Require multi-factor authentication to log in

With multi-factor authentication (MFA), users have a device that generates a response to an authentication challenge. Each user's credentials and device-generated response are required to complete the sign-in process. Enable MFA for AWS account access, especially for long-term credentials such as the account root user and IAM users.

To set up MFA for the root user

1. Sign in to the AWS Management Console.
2. Choose your account name, and then choose **Security credentials**.
3. On the **Security credentials** page, under **Multi-factor authentication (MFA)**, choose **Assign MFA device**.
4. Follow the steps to configure your MFA device. For more information, see Multi-factor authentication for AWS account root user in the IAM documentation.

To set up MFA in IAM Identity Center

1. See Enable MFA in the IAM Identity Center documentation.

To set up MFA for your own IAM user

1. Sign in to the IAM console.
2. Choose your user name, and then choose **Security credentials**.
3. On the **Security credentials** tab, under **Multi-factor authentication (MFA)**, choose **Assign MFA device**.
4. Follow the steps to configure your MFA device. For more information, see AWS Multi-Factor Authentication in IAM in the IAM documentation.

To set up MFA for other IAM users

1. Sign in to the IAM console.
2. In the navigation pane, choose **Users**.
3. Choose the name of the user for whom you want to enable MFA, and then choose the **Security credentials** tab.
4. Under **Multi-factor authentication (MFA)**, choose **Assign MFA device**.
5. Follow the steps to configure the MFA device. For more information, see AWS Multi-Factor Authentication in IAM in the IAM documentation.

ACCT.06 Enforce a password policy

Users sign in to the AWS Management Console by providing sign-in credentials. AWS recommends requiring MFA for all users. Require that passwords adhere to a strong password policy to help prevent discovery through brute force or social engineering. For more information about password policy recommendations, see the Password policy guide on the Center for Internet Security (CIS) website.

> **Note**
>
> For a benchmark-aligned minimum password length, see AWS Security Hub control IAM.15, which references the CIS AWS Foundations Benchmark recommendation.

For IAM users, configure password requirements by creating a custom IAM password policy. For more information, see Set an account password policy for IAM users in the IAM documentation.

To create a custom password policy

1. Open the IAM console.
2. In the navigation pane, choose **Account settings**.
3. In the **Password policy** section, choose **Edit**.
4. Choose **Custom** to use a custom password policy.
5. Select the options that you want to apply to your password policy and choose **Save changes**.
6. Confirm that you want to set a custom password policy by choosing **Set custom**.

ACCT.07 Deliver CloudTrail logs to a protected Amazon S3 bucket

Actions taken by users, roles, and services in your AWS account are recorded as events in AWS CloudTrail. CloudTrail is enabled by default, and in the CloudTrail console, you can access 90 days of event history information. To view, search, download, archive, analyze, and respond to account activity across your AWS infrastructure, see Viewing events with CloudTrail event history in theCloudTrail documentation.

To retain CloudTrail history beyond 90 days, create a trail that delivers log files to an Amazon Simple Storage Service (Amazon S3) bucket for all event types. When you create a trail in the CloudTrail console, you create a multi-Region trail.

To create a trail that delivers logs for all AWS Regions to an Amazon S3 bucket

1. Open the CloudTrail console.
2. Follow the steps in Creating a trail in the CloudTrail documentation. On the **Choose log events** page, do the following:
   a. For **API activity**, select both **Read** and **Write**.
   b. For the **Exclude AWS KMS events** option, use the following guidance:
      - For preproduction environments, select **Exclude AWS KMS events** to exclude all AWS Key Management Service (AWS KMS) events from your trail. AWS KMS read actions such as `Encrypt`, `Decrypt`, and `GenerateDataKey` can generate a large volume of events.

- For production environments, select **Write** for management events, and clear the **Read** and **Exclude AWS KMS events** check boxes. This excludes high-volume AWS KMS read events but still logs relevant AWS KMS actions, such as `Disable`, `Delete`, and `ScheduleKey`.

c. If you do not plan to use the Amazon Relational Database Service (Amazon RDS) Data API and want to use CloudTrail for troubleshooting and data access auditing purposes, select **Exclude Amazon RDS Data API events**. The Data API can generate a high volume of CloudTrail events.

After you create the trail, it appears on the **Trails** page. CloudTrail begins publishing log files to the Amazon S3 bucket you specified within approximately 15 minutes.

> **Note**
>
> As a cost consideration, you can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail. Amazon S3 storage charges apply. For information about Amazon S3 pricing, see Amazon S3 pricing.

To help secure the Amazon S3 buckets where you store CloudTrail log files

1. Review the Amazon S3 bucket policy in the CloudTrail documentation for each bucket where you store log files, and adjust it as needed to remove unnecessary access.
2. Make sure to add an `aws:SourceArn` condition key to the bucket policy. For more information, see Create or update an Amazon S3 bucket for an organization trail in the CloudTrail documentation.
3. To add an additional layer of protection against accidental or unauthorized deletion of log files, see Configuring MFA delete in the Amazon S3 documentation.

ACCT.08 Prevent public access to private Amazon S3 buckets

By default, the root user of the AWS account and the IAM principal that created the bucket have permissions to read and write to Amazon S3 buckets. Additional IAM principals are granted access by using identity-based policies, and access conditions can be enforced by using a bucket policy. You can create bucket policies that grant the general public access to the bucket, creating a *public* bucket.

Buckets created on or after April 28, 2023 have the **Block Public Access** setting enabled by default. For buckets created before this date, a misconfigured bucket policy can unintentionally grant public access. You can help prevent this by enabling the **Block Public Access** setting for each bucket. If you have no current or future use cases for a public Amazon S3 bucket, enable this setting at the AWS account level.

To prevent public access to Amazon S3 buckets

1. Follow the steps in Configure block public access settings for your Amazon S3 buckets in the Amazon S3 documentation.

AWS Trusted Advisor generates a yellow finding for Amazon S3 buckets that allow list or read access to the public and generates a red finding for buckets that allow public uploads or deletes. Follow ACCT.12 Monitor for and resolve AWS Trusted Advisor high-risk items to identify and correct misconfigured buckets. In the Amazon S3 console, you can see if your bucket is publicly accessible from the **Buckets** list.

ACCT.09 Delete unused VPCs, subnets, and security groups

To reduce the opportunity for security issues, delete resources that are not being used. In a new AWS account, by default, a virtual private cloud (VPC) is created automatically in every AWS Region. This enables you to assign public IP addresses in public subnets. If these VPCs are not needed, this introduces risk of unintended exposure of resources.

If they are not in use, delete the default VPCs in each Region, including Regions where you do not plan to deploy workloads. Before you can delete a VPC, you must first delete its dependent resources in the order of their dependencies. For example, delete Amazon Elastic Compute Cloud (Amazon EC2) instances before their subnets, and delete NAT gateways and internet gateways before the VPC. Subnets and security groups are deleted when the VPC is deleted. Attempting to delete a resource that has dependent resources will result in an error.

> **Note**
>
> You can view your Regions and VPCs on the Amazon EC2 Global View console. For more information, see List and filter resources across Regions using Amazon EC2 Global View in the Amazon EC2 documentation.

To delete a default VPC and its associated resources

1. See Delete your VPC in the Amazon Virtual Private Cloud (Amazon VPC) documentation.
2. Repeat this process for each Region where default VPCs exist.

ACCT.10 Configure AWS Budgets to monitor your spending

AWS Budgets enables monitoring of monthly costs and usage with notifications when costs are forecast to exceed target thresholds. Cost forecast notifications can indicate unexpected activity, which adds a layer of monitoring alongside other services, such as Trusted Advisor. Monitoring your AWS costs helps you detect unexpected usage and avoid unplanned charges.

To set up a budget

1. See Create a cost budget in the AWS Billing and Cost Management documentation.

> **Note**
>
> AWS Budgets provides two budgets per account at no charge. For information about the cost of additional budgets, see AWS Budgets pricing.

ACCT.11 Enable IAM Access Analyzer

Enable IAM Access Analyzer in each AWS Region you use. Because IAM Access Analyzer operates on a per-Region basis, you must enable it separately in each Region to gain visibility into resource sharing across your AWS footprint. This helps prevent accidental public or cross-account access to resources, such as Amazon S3 buckets, IAM roles, and AWS KMS keys.

To enable IAM Access Analyzer

1. Open the IAM console.
2. In the left navigation pane, choose **Access Analyzer**.
3. Choose **Create analyzer**.
4. Enter a name for your analyzer.
5. For the analyzer scope, choose **Account** for a single account, or choose **Organization** if you are using AWS Organizations.

6. Choose **Create analyzer**.

Review the findings in the **Access Analyzer** console and update resource policies to remove unintended external access. For more information, see Reviewing findings for IAM Access Analyzer in the IAM documentation. Prioritize high-impact findings, such as public Amazon S3 buckets or IAM roles that are shared outside of your AWS account.

> **Note**
>
> IAM Access Analyzer pricing depends on the analyzer type and features you use. An external access analyzer is available at no additional charge. Early-stage startups should start with an external access analyzer. For more information about pricing, see IAM Access Analyzer pricing.

ACCT.12 Monitor for and resolve AWS Trusted Advisor high-risk items

AWS Trusted Advisor scans your AWS infrastructure for high-risk or high-impact issues related to security, performance, cost, and reliability. It provides detailed information about affected resources and remediation recommendations. For more information about checks and descriptions, see AWS Trusted Advisor check reference in the AWS Support documentation. Access to Trusted Advisor checks varies by AWS Support plan.

Basic Support provides access to the following:

- Checks in the Service Limits category
- Selected checks in the Security and Fault Tolerance categories, including:
  - Amazon Elastic Block Store (Amazon EBS) public snapshots
  - Amazon Relational Database Service (Amazon RDS) public snapshots
  - Amazon S3 bucket permissions
  - MFA for the root user
  - Security groups that have specific ports unrestricted
  - AWS Security Token Service (AWS STS) global endpoint usage across AWS Regions

Full access to all Trusted Advisor checks requires one of the following paid support plans**:**

- AWS Business Support+
- AWS Enterprise Support
- AWS Unified Operations

Review Trusted Advisor findings regularly and remediate issues as they are identified. If you have AWS Business Support+, AWS Enterprise Support, or AWS Unified Operations, you can subscribe to a weekly findings email. For more information, Set up notification preferences in the AWS Support documentation.

To view Trusted Advisor findings

1. See View check categories in the AWS Support documentation.
2. Start by reviewing *action recommended* issues, which are marked in red.

ACCT.13 Use short-lived credentials for access to your AWS resources

Determine how your developers access AWS services and resources through the AWS Command Line Interface (AWS CLI). To reduce security risk, avoid using IAM users with long-lived access keys for authentication when developing software or working with production data. Short-lived credentials expire automatically, which reduces the risk of credential exposure.

Choose the approach that matches your current AWS access pattern

- Sign in with console credentials (Recommended) – If you use root, IAM users, or federation with IAM for AWS account access, use `aws login` to obtain temporary credentials for AWS CLI or AWS SDK access.
- Sign in with IAM Identity Center credentials – If you use IAM Identity Center for AWS account access, this approach provides centralized identity management and automatic credential rotation.
- **Federated access through your corporate identity provider** – Use your organization's existing identity provider, such as Okta, Active Directory, or Ping Identity, with MFA enforcement.

To obtain temporary AWS CLI credentials using the `aws` login

1. Install or update the AWS CLI. For more information, see Installing or updating to the latest version of the AWS CLI in the AWS CLI documentation.
2. Enter `aws login` and follow the authentication prompts.
3. Authenticate using your IAM user credentials and MFA.

After you authenticate, the AWS CLI manages temporary credentials for your session. When your session expires, enter `aws login` again to re-authenticate. For information about session duration settings, see IAM role session duration in the IAM documentation.

For AWS Partner integrations and third-party solutions, use short-lived credentials where possible. IAM temporary delegation for AWS Partners allows you integrate AWS Partner products by using short-lived credentials instead of long-lived access keys. IAM Outbound Identity Federation allows AWS workloads to authenticate to external solutions by using short-lived tokens instead of long-lived API keys.

Securing your workloads

A workload is a collection of resources and code that delivers business value, such as a customer-facing application or a backend process. As you build and deploy workloads on AWS, the controls in this section help you protect your data, limit exposure of sensitive resources, and establish secure defaults. The controls cover managing application secrets, restricting access scope, minimizing access routes to private resources, and encrypting data in transit and at rest.

This section contains the following topics:

WKLD.01 Use IAM roles for compute environment permissions

In AWS Identity and Access Management (IAM), a *role* represents a set of permissions that can be assumed by an IAM user, an AWS service, or a federated identity for a configurable period of time. Using roles removes the need to store or manage long-term credentials, which reduces the chance of unintended use. Assign an IAM role directly to Amazon Elastic Compute Cloud (Amazon

EC2) instances, AWS Fargate tasks and services, AWS Lambda functions, and other AWS compute services that support IAM roles. Applications that use an AWS SDK and run in these compute environments automatically use the IAM role credentials for authentication.

For instructions on using IAM roles with services, see the following documentation:

- IAM roles for Amazon EC2 in the Amazon EC2 documentation
- IAM roles for tasks in the Amazon Elastic Container Service (Amazon ECS) documentation
- Lambda execution role in the AWS Lambda documentation
- For other AWS compute services, refer to the *Security* section of the AWS service documentation.

WKLD.02 Restrict credential usage scope with resource-based policies

Policies define permissions or specify access conditions for AWS resources. There are two primary types of policies:

- *Identity-based policies* are attached to principals and define what the principal's permissions are in the AWS environment.
- *Resource-based policies* are attached to a resource, such as an Amazon Simple Storage Service (Amazon S3) bucket, or virtual private cloud (VPC) endpoint. These policies specify which principals are allowed access, supported actions, and any other conditions that must be met.

For a principal to access a resource, the principal must have permission in its identity-based policy and meet the conditions of the resource-based policy. For more information, see Identity-based policies and resource-based policies in the IAM documentation.

The following conditions help restrict access to trusted sources and reduce the risk of unintended access:

- Restrict access to principals in a specified organization (defined in AWS Organizations) by using the `aws:PrincipalOrgID` condition.
- Restrict access to traffic that originates from a specific VPC or VPC endpoint by using the `aws:SourceVpc` or `aws:SourceVpce` condition, respectively.
- Allow or deny traffic based on the source IP address by using an `aws:SourceIp` condition.

The following example shows a resource-based policy that uses the `aws:PrincipalOrgID` condition to allow only principals in your organization to access an Amazon S3 bucket. Replace `o-xxxxxxxxxxx` with your organization ID and `bucket-name` with your bucket name:

```
{
    "Version": "2012-10-17",
    "Statement":[
      {
        "Sid":"AllowFromOrganization",
        "Effect":"Allow",
        "Principal":"*",
        "Action":"s3:*",
        "Resource":"arn:aws:s3:::bucket-name/*",
        "Condition": {
          "StringEquals": {"aws:PrincipalOrgID":"<o-xxxxxxxxxxx>"}
        }
      }
    ]
 }
```

WKLD.03 Use ephemeral secrets or a secrets-management service

Application secrets include credentials, such as key pairs, access tokens, digital certificates, and sign-in credentials. The application uses these secrets to gain access to other services it depends upon, such as a database. To help protect these secrets, we recommend they are either *ephemeral* (generated at the time of request and short-lived, such as with IAM roles) or retrieved from a secrets management service. This reduces the risk of secrets being accidentally stored in static configuration files, environment variables, or source code. Centralizing secrets management also makes it straightforward to move application code between development and production environments without reconfiguring credentials.

For a secrets management, use a combination of Parameter Store (a capability of AWS Systems Manager) and AWS Secrets Manager:

- Use Parameter Store to manage secrets and other parameters that are individual key-value pairs, string-based, short in overall length, and accessed frequently. You use an AWS Key Management Service (AWS KMS) key to encrypt the secret. There is no charge to store parameters in the

standard tier of Parameter Store. For more information about parameter tiers, see Managing parameter tiers in the Systems Manager documentation.

- Use Secrets Manager to store secrets that are in document form (such as multiple, related key-value pairs), that are larger than 4 KB (such as digital certificates), or that would benefit from automated rotation.

You can use Parameter Store APIs to retrieve secrets stored in Secrets Manager. With this approach, you can standardize the code in your application when using a combination of both services.

To manage secrets in Parameter Store

1. Create a symmetric AWS KMS key. For more information, see Create a symmetric encryption KMS key in the AWS KMS documentation.
2. Create a `SecureString` parameter. For more information, see Create a SecureString parameter in the Systems Manager documentation. Secrets in Parameter Store use the `SecureString` data type.
3. In your application, retrieve a parameter from Parameter Store by using the AWS SDK for your programming language. For code examples, see GetParameter in the Systems Manager documentation.

To manage secrets in Secrets Manager

1. Create a secret. For more information, see Create a secret in the Secrets Manager documentation.
2. Retrieve secrets from Secrets Manager in code. For more information, see Get secrets from AWS Secrets Manager in the Secrets Manager documentation.

For information about improving the availability and latency of secret retrieval, see Use AWS Secrets Manager client-side caching libraries to improve the availability and latency of using your secrets on the AWS Security Blog. The client-side caching SDK reduces the number of API calls your application makes to Secrets Manager and can improve secret retrieval performance.

Cost considerations

The cost of secrets management depends on which service you use and how your application accesses secrets:

- For Parameter Store, standard tier parameters are available at no additional charge for values up to 4 KB. The advanced tier applies additional charges for larger parameters or higher throughput.
- AWS Secrets Manager charges for each secret stored on a monthly basis and charges for each API call made to retrieve secrets. Using the Secrets Manager client-side caching SDK reduces the number of API calls your application makes to Secrets Manager, which can reduce costs.
- Encrypting secrets with an AWS managed KMS key is available at no additional charge. Customer-managed keys incur a monthly charge for each key and a charge for each API call.

For most early-stage startups, a cost effective starting point is to use Parameter Store with an AWS managed KMS key for frequently accessed secrets and use Secrets Manager for secrets that benefit from automated rotation.

For current pricing, see AWS Systems Manager pricing and AWS Key Management Service pricing.

WKLD.04 Prevent application secrets from being exposed

During local development, application secrets can be stored in local configuration or code files and accidentally checked in to source code repositories. If a repository hosted on a public service provider is unsecured, unauthorized users can access it and discover exposed secrets. Use available tools to prevent secrets from being committed to your repository. During code reviews, check for hardcoded credentials, API keys, and other secrets before merging changes.

The following open-source tools can help prevent application secrets from being checked in to source code repositories:

- Gitleaks on GitHub
- detect-secrets on GitHub
- git-secrets on GitHub
- TruffleHog on GitHub

> **Note**
>
> These tools are open source and available at no charge.

For guidance on detecting and remediating secrets that have already been exposed, see WKLD.05 Detect and remediate when secrets are exposed.

WKLD.05 Detect and remediate exposed secrets

In WKLD.03 Use ephemeral secrets or a secrets-management service and WKLD.04 Prevent application secrets from being exposed, you put measures in place to protect secrets. In this control, you set up tooling to detect secrets that were accidentally committed or exposed, and take action to revoke or rotate them.

An exposed secret can be exploited and risks unauthorized access to your AWS resources and data. Rotate or revoke it immediately after detection.

Scan code repositories regularly for accidentally committed secrets. Use Kiro CLI or the open-source tools listed in WKLD.04 and integrate the tool into your local development or CI/CD pipeline. If you identify an exposed secret, remediate it immediately. Rotate or revoke the exposed credential to prevent further use, and remove it from source control history.

To detect exposed secrets using Kiro CLI

1. Install Kiro CLI in your development environment. For more information, see Kiro CLI in the Kiro documentation.
2. Configure Kiro CLI to scan your code repositories, focusing on high-risk repositories such as production or public-facing code.
3. Schedule regular scans. Consider daily scans for production repositories and weekly scans for development repositories.
4. Review scan results and identify any exposed secrets.

To remediate exposed secrets

1. Rotate or revoke the exposed secret immediately in the originating service (for example, regenerate an API key or reset a password).
2. Create a new secret in AWS Secrets Manager or AWS Systems Manager Parameter Store.
3. Update your applications to retrieve the new secret from the secure storage service.
4. Remove the exposed secret from your code repository history by using `git filter-repo`.

The open-source tools listed in WKLD.04 can also detect secrets that are already present in your repository.

> **Note**
>
> Kiro CLI is available at no charge under the Free tier. For more information, see Kiro pricing.

WKLD.06 Use Systems Manager instead of SSH or RDP

Public subnets, which have a default route pointing to an internet gateway, present a greater security risk than *private subnets*, which have no route to the internet. You can run Amazon EC2 instances in private subnets and use the Session Manager capability of AWS Systems Manager to remotely access the instances through either the AWS Command Line Interface (AWS CLI) or AWS Management Console. You can then use the AWS CLI or console to start a session that connects into the instance through a secure tunnel, which removes the need to manage credentials for Secure Shell (SSH) or Windows remote desktop protocol (RDP).

Use Session Manager instead of running Amazon EC2 instances in public subnets or running bastion hosts.

To set up Session Manager

1. Verify that the Amazon EC2 instance uses a supported operating system Amazon Machine Image (AMI), such as Amazon Linux or Ubuntu, with the AWS Systems Manager Agent (SSM Agent) pre-installed.
2. Confirm that the instance has connectivity, either through an internet gateway or through VPC endpoints, to the following endpoints (replacing `<Region>` with the appropriate AWS Region):
   - `ec2messages.<Region>.amazonaws.com`
   - `ssm.<Region>.amazonaws.com`
   - `ssmmessages.<Region>.amazonaws.com`
3. Attach the `AmazonSSMManagedInstanceCore` AWS managed policy to the IAM role associated with your instances.

For more information, see Setting up Session Manager in the *AWS Systems Manager User Guide*.

To start a session

1. See Starting a session in the Systems Manager documentation.

> **Note**
>
> Session Manager is available at no additional charge for Amazon EC2 instances. If you use VPC endpoints for Session Manager connectivity, interface endpoints incur an hourly charge and a per-GB data-processing charge. For more information, see Systems Manager pricing.

WKLD.07 Log data events for S3 buckets with sensitive data

By default, AWS CloudTrail captures *management events*, which are events that create, modify, or delete resources in your account. This does not include read or write operations on individual objects in Amazon S3 buckets. To support investigation during a security event, for detection and auditing purposes, log data events for Amazon S3 buckets that store sensitive or business-critical data.

To log data events for trails

1. Open the CloudTrail console.
2. In the navigation pane, choose **Trails**, and then choose a trail name.
3. In **General details**, choose **Edit** to change the following settings (you cannot change the name of a trail).
   a. In **Data events**, choose **Edit**.
   b. For **Data event source**, choose **S3**.
   c. For **All current and future S3 buckets**, clear **Read** and **Write** to deselect the default selection.
   d. In **Individual bucket selection**, choose the bucket on which to log data events. To add more buckets, choose **Add bucket**.
   e. Choose to log **Read** events (such as `GetObject`), **Write** events (such as `PutObject`), or both.
   f. Choose **Update trail**.

> **Note**
>
> Additional charges apply for logging CloudTrail data events. For more information, see AWS CloudTrail pricing.

WKLD.08 Encrypt Amazon EBS volumes

Verify that encryption by default is enabled for Amazon Elastic Block Store (Amazon EBS) volumes in your AWS account. Enabling encryption by default ensures that new Amazon EBS volumes and snapshots are encrypted automatically, removing the need to configure encryption for each volume individually. Encrypted volumes have the same input/output operations per second (IOPS) performance as unencrypted volumes with a minimal effect on latency. For more information, see Must-know best practices for Amazon EBS encryption on the AWS Compute Blog.

To enable encryption by default for Amazon EBS volumes, see Enable encryption by default in the Amazon EBS documentation. Enabling encryption by default does not encrypt existing unencrypted volumes. To encrypt an existing unencrypted Amazon EBS volume, create an encrypted snapshot copy of the volume and then create a new encrypted volume from that snapshot. For step-by-step instructions, see Create an Amazon EBS volume in the Amazon EBS documentation.

> **Note**
>
> Encrypting Amazon EBS volumes with an AWS managed AWS KMS key is available at no additional charge. Customer managed keys incur a monthly charge per key and a charge per API call. For more information, see AWS Key Management Service pricing.

WKLD.09 Encrypt Amazon RDS databases

Enable encryption for Amazon Relational Database Service (Amazon RDS) databases to protect data at rest. Amazon RDS encrypts data at the underlying volume level and delivers the same IOPS performance as unencrypted volumes with a minimal effect on latency. For more information, see Overview of encrypting Amazon RDS resources in the Amazon RDS documentation.

To encrypt a new Amazon RDS database instance, see Encrypt a database instance in the Amazon RDS documentation.

Note

Encryption must be enabled when creating the database. You cannot enable encryption on an existing unencrypted Amazon RDS database instance. If you need to encrypt an existing unencrypted database, you must create a new encrypted database and migrate your data. For more information, see Copying a DB snapshot for Amazon RDS in the Amazon RDS documentation.

Note

Encrypting Amazon RDS databases with an AWS managed AWS KMS key is available at no additional charge. Customer-managed keys incur a monthly charge per key and a charge per API call. For more information, see AWS Key Management Service pricing.

WKLD.10 Deploy private resources into private subnets

Deploy resources that don't require direct internet access (such as Amazon EC2 instances, databases, queues, caching, or other infrastructure) into a VPC private subnet. Private subnets don't have a route declared in their route table to an attached internet gateway and cannot receive internet traffic. Traffic from a private subnet that is destined for the internet must go through network address translation (NAT). You can use a managed AWS NAT Gateway or an Amazon EC2 instance running NAT processes in a public subnet. For more information about network isolation, see Infrastructure security in Amazon VPC in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

Use the following practices when creating private resources and subnets:

- When creating a private subnet, disable **Auto-assign public IPv4 address**.
- When creating private Amazon EC2 instances, disable **Auto-assign Public IP**. This prevents a public IP address from being assigned if the instance is unintentionally deployed into a public subnet due to misconfiguration.
- When creating AWS Fargate tasks and services, deploy them into private subnets and set **Assign public IP** to **TURNED OFF**. Fargate tasks deployed in a public subnet can be assigned a public IP address, which exposes them directly to the internet. For more information, see AWS Fargate task networking in the Amazon Elastic Container Service (Amazon ECS) documentation.

When deploying a resource, specify the private subnet in the resource's network configuration.

> **Note**
>
> Private subnets are available at no additional charge. If your private resources require outbound internet access, AWS NAT Gateway incurs an hourly charge and a per-GB data-processing charge. For more information, see Amazon VPC pricing.

WKLD.11 Restrict network access by using security groups

Use security groups to control traffic to Amazon EC2 instances, containers, Amazon RDS databases, and other supported resources. *Security groups* act as a virtual firewall that can be applied to a group of related resources to consistently define rules for allowing inbound and outbound traffic. In addition to rules based on IP addresses and ports, security groups support rules to allow traffic from resources associated with other security groups. For example, a database security group can have rules to allow only traffic from an application server security group.

Security groups apply to AWS Fargate tasks in the same way they apply to Amazon EC2 instances. When you create an Amazon ECS service or run a Fargate task, you assign one or more security groups to the task's Elastic Network Interface. For more information, see AWS Fargate task networking in the Amazon Elastic Container Service documentation.

By default, security groups allow all outbound traffic but don't allow inbound traffic. You can remove the outbound traffic rule, or configure additional rules to restrict outbound traffic and allow inbound traffic. If the security group has no outbound rules, outbound traffic from your instance is blocked. For more information, see Control traffic to resources using security groups in the Amazon VPC documentation.

The following example shows three security groups that control traffic from an Application Load Balancer to containers (Amazon EC2 instances or Fargate tasks) that connect to an Amazon RDS for PostgreSQL database.

Security group	Inbound rules	Outbound rules
Application Load Balancer security group	**Description:** Allow HTTPS traffic from anywhere **Type:** HTTPS	**Description:** Allow all traffic to anywhere **Type:** All traffic

	Source: Anywhere-IPv4 (0.0.0.0/0)	**Destination:** Anywhere-IPv4 (0.0.0.0/0)
Container security group (Amazon EC2 or Fargate task)	**Description:** Allow HTTP traffic from the Application Load Balancer **Type:** HTTP **Source:** Application Load Balancer security group	**Description:** Allow all traffic to anywhere **Type:** All traffic **Destination:** Anywhere-IPv4 (0.0.0.0/0)
Amazon RDS database security group	**Description:** Allow PostgreSQL traffic from container **Type:** PostgreSQL **Source:** Container security group	None

> **Note**
>
> Security groups are available at no additional charge.

WKLD.12 Use VPC endpoints to access supported services

In VPCs, resources that need to access AWS or other external services require either a route to the internet (`0.0.0.0/0`) or to the public IP address of the target service. Use VPC endpoints to enable a private IP route from your VPC to supported AWS or other services, removing the need for an internet gateway, NAT device, virtual private network (VPN) connection, or AWS Direct Connect connection.

You can attach policies and security groups to VPC endpoints to control access to a service. For example, you can write a VPC endpoint policy for Amazon DynamoDB to allow only item-level actions and prevent table-level actions for resources in the VPC, regardless of their own permission policy. You can also write an Amazon S3 bucket policy to allow only requests originating from a specific VPC endpoint, denying other external access. A VPC endpoint can also have a security

group rule that, for example, restricts access to Amazon EC2 instances associated with an application-specific security group, such as the business-logic tier of a web application.

VPC endpoints come in two types: *interface* endpoints and *gateway* endpoints. You access most services by using a VPC interface endpoint. DynamoDB is accessed using a gateway endpoint. Amazon S3 supports both interface and gateway endpoints. We recommend gateway endpoints for workloads that are contained within a single AWS account and Region. Gateway endpoints come at no additional charge. We recommend interface endpoints when you need more extensible access, such as to an Amazon S3 bucket from other VPCs, from on-premises networks, or from different AWS Regions.

For more information about using VPC endpoints, see the following resources:

- For more information about selecting between gateway and interface endpoints for Amazon S3, see Choosing your VPC endpoint strategy for Amazon S3 on the AWS Architecture Blog.
- Access an AWS service using an interface VPC endpoint in the Amazon VPC documentation.
- Gateway endpoints in the Amazon VPC documentation.
- For example Amazon S3 bucket policies that restrict access to a specific VPC or VPC endpoint, see Restricting access to a specific VPC in the Amazon S3 documentation.
- For example DynamoDB endpoint policies that restrict actions, see Endpoint policies for DynamoDB in the Amazon VPC documentation.

> **Note**
>
> Gateway endpoints are available at no additional charge. Interface endpoints incur an hourly charge and a per-GB data-processing charge. These charges are lower than the equivalent charges for routing traffic through AWS NAT Gateway. For more information, see Amazon VPC pricing.

WKLD.13 Require HTTPS for public web endpoints

Require HTTPS so that your endpoints can use certificates to prove their identity and so that traffic between your endpoint and clients is encrypted. For public websites, HTTPS also improves search engine ranking.

Many AWS services provide public web endpoints for your resources, such as AWS Elastic Beanstalk, Amazon CloudFront, Amazon API Gateway, Elastic Load Balancing, and AWS Amplify. For instructions about how to require HTTPS for each of these services, see the following:

- Configuring HTTPS for your Elastic Beanstalk environment in the AWS Elastic Beanstalk documentation
- Requiring HTTPS for communication between viewers and CloudFront in the Amazon CloudFront documentation
- How can I use an Application Load Balancer to redirect HTTP requests to HTTPS? on AWS re:Post
- How do I redirect HTTP requests to HTTPS on a Classic Load Balancer? on AWS re:Post

> **Note**
>
> Classic Load Balancer is a legacy option. For new deployments, we recommend using an Application Load Balancer.

- Connecting a custom domain in the AWS Amplify documentation

Static websites hosted on Amazon S3 do not support HTTPS. To require HTTPS for these websites, you can use CloudFront. When you use CloudFront to serve content from an Amazon S3 bucket, you don't need to enable public access on the bucket. Use an origin access control (OAC) to allow CloudFront to access the private bucket.

For instructions on setting up CloudFront to serve a static website hosted on Amazon S3, see How do I use CloudFront to serve a static website hosted on Amazon S3? on AWS re:Post.

To configure HTTPS for a static website hosted on Amazon S3

1. If you are configuring access to a public Amazon S3 bucket, require HTTPS between viewers and CloudFront. For more information, see Require HTTPS for communication between viewers and CloudFront in the Amazon CloudFront documentation.
2. If you are configuring access to a private Amazon S3 bucket, restrict access to Amazon S3 content by using an origin access control (OAC). For more information, see Restricting access to an Amazon S3 origin in the Amazon CloudFront documentation.

Configure HTTPS endpoints to require modern Transport Layer Security (TLS) protocols and ciphers, unless compatibility with older protocols is needed. For example, use the

`ELBSecurityPolicy-TLS13-1-0-PQ-2025-09` policy or the most recent policy available for Application Load Balancer HTTPS listeners. The most current policies require TLS 1.3 at a minimum, forward secrecy, and strong ciphers that are compatible with modern web browsers.

For more information about the available security policies for HTTPS public endpoints, see the following:

- Predefined SSL security policies for Classic Load Balancers in the Elastic Load Balancing documentation
- Security policies for your Application Load Balancer in the Elastic Load Balancing documentation
- Supported protocols and ciphers between viewers and CloudFront in the Amazon CloudFront documentation

WKLD.14 Use edge-protection services for public endpoints

Rather than serve traffic directly from compute services such as Amazon EC2 instances or containers, use an edge protection service. An edge protection service sits between internet traffic and your backend resources, filtering unwanted requests, enforcing encryption, and applying rules such as load balancing before traffic reaches your workloads.

AWS services that can provide public endpoint protection include AWS WAF, Amazon CloudFront, Elastic Load Balancing, Amazon API Gateway, and AWS Amplify Hosting. Deploy VPC-based services, such as Elastic Load Balancing, in a public subnet to receive internet traffic and forward it to your workloads running in a private subnet.

Amazon CloudFront, Amazon API Gateway, and Amazon Route 53 provide protection from Layer 3 and 4 distributed denial of service (DDoS) attacks at no additional charge. AWS WAF provides protection against Layer 7 attacks and incurs additional charges.

For instructions on getting started with each of these services, see the following:

- Getting started with AWS WAF
- Getting started with Amazon CloudFront
- Getting started with Elastic Load Balancing
- Getting started with Amazon API Gateway
- Getting started with AWS Amplify Hosting

WKLD.15 Define security controls in templates and deploy them by using CI/CD practices

Infrastructure as code (IaC) is the practice of defining your AWS resources and configurations in templates and code that you deploy by using continuous integration and continuous delivery (CI/CD) pipelines, the same pipelines used to deploy software applications. IaC tools, such as AWS CloudFormation and the AWS Cloud Development Kit (AWS CDK), support IAM identity-based and resource-based policies and integrate with AWS services, such as AWS WAF and Amazon VPC. Define your IAM policies, resource-based policies, and security service configurations as IaC templates. Commit the templates to a source code repository and deploy them by using CI/CD pipelines.

Commit application permission policies with application code in the same repository. Manage general resource policies and security service configurations in separate repositories and deployment pipelines. This separation reduces the risk of a single compromised repository affecting both application code and security configurations.

The following TypeScript AWS CDK stack demonstrates three foundational security controls from this document: an Amazon S3 bucket with `BlockPublicAccess` and server-side encryption (ACCT.08), a CloudTrail trail with log file validation (ACCT.07), and IAM Access Analyzer (ACCT.11).

```
import * as cdk from "aws-cdk-lib";
import {
  aws_s3 as s3,
  aws_cloudtrail as cloudtrail,
  aws_accessanalyzer as accessanalyzer,
  aws_iam as iam,
  RemovalPolicy,
} from "aws-cdk-lib";
import { Construct } from "constructs";

export class SecurityBaselineStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const accountId = cdk.Stack.of(this).account;
    const region = cdk.Stack.of(this).region;
    const trailName = "audit-trail";
    const trailArn = `arn:aws:cloudtrail:${region}:${accountId}:trail/${trailName}`;
```

```
    // ------------------------------------------------------
    // Tagging — applied to every resource in the stack
    // ------------------------------------------------------

    cdk.Tags.of(this).add("Environment", "production");
    cdk.Tags.of(this).add("Team", "platform");
    cdk.Tags.of(this).add("ManagedBy", "cdk");

    // ------------------------------------------------------
    // ACCT.08 — Block Public Access on S3
    // WKLD.08 — Encrypt data at rest (SSE-S3, AWS-managed)
    // ------------------------------------------------------
    const loggingBucket = new s3.Bucket(this, 'AccessLogsBucket', {
      blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
      encryption: s3.BucketEncryption.S3_MANAGED,
      enforceSSL: true,
      versioned: true,
      accessControl: s3.BucketAccessControl.LOG_DELIVERY_WRITE,
      lifecycleRules: [
        {
          id: "ArchiveAfter90Days",
          transitions: [
            {
              storageClass: s3.StorageClass.GLACIER,
              transitionAfter: cdk.Duration.days(90),
            },
          ],
        },
      ],
      removalPolicy: RemovalPolicy.RETAIN,
    });

    const auditLogsBucket = new s3.Bucket(this, "AuditLogsBucket", {
      blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL,
      encryption: s3.BucketEncryption.S3_MANAGED,
      enforceSSL: true,
      versioned: true,
      serverAccessLogsBucket: loggingBucket,
      serverAccessLogsPrefix: 'access-logs',
      lifecycleRules: [
        {
          id: "ArchiveAfter90Days",
          transitions: [
            {
```

```
            storageClass: s3.StorageClass.GLACIER,
            transitionAfter: cdk.Duration.days(90),
          },
        ],
      },
    ],
    removalPolicy: RemovalPolicy.RETAIN,
  });

  // ------------------------------------------------------
  // Bucket policy — CloudTrail access + account boundary
  //
  // Per AWS docs, CloudTrail needs two permissions:
  //   1. GetBucketAcl to verify bucket ownership
  //   2. PutObject to write log files
  // Both are scoped to this specific trail via aws:SourceArn.
  // ------------------------------------------------------

  auditLogsBucket.addToResourcePolicy(
    new iam.PolicyStatement({
      sid: "AWSCloudTrailAclCheck",
      effect: iam.Effect.ALLOW,
      principals: [new iam.ServicePrincipal("cloudtrail.amazonaws.com")],
      actions: ["s3:GetBucketAcl"],
      resources: [auditLogsBucket.bucketArn],
      conditions: {
        StringEquals: {
          "aws:SourceArn": trailArn,
        },
      },
    })
  );

  auditLogsBucket.addToResourcePolicy(
    new iam.PolicyStatement({
      sid: "AWSCloudTrailWrite",
      effect: iam.Effect.ALLOW,
      principals: [new iam.ServicePrincipal("cloudtrail.amazonaws.com")],
      actions: ["s3:PutObject"],
      resources: [
        `${auditLogsBucket.bucketArn}/cloudtrail/AWSLogs/${accountId}/*`,
      ],
      conditions: {
        StringEquals: {
```

```
            "s3:x-amz-acl": "bucket-owner-full-control",
            "aws:SourceArn": trailArn,
          },
        },
      })
    );

    auditLogsBucket.addToResourcePolicy(
      new iam.PolicyStatement({
        sid: "DenyExternalAccess",
        effect: iam.Effect.DENY,
        principals: [new iam.AnyPrincipal()],
        actions: ["s3:*"],
        resources: [
          auditLogsBucket.bucketArn,
          `${auditLogsBucket.bucketArn}/*`,
        ],
        conditions: {
          StringNotEquals: {
            "aws:PrincipalAccount": accountId,
          },
          Bool: {
          "aws:PrincipalIsAWSService": "false",
          },
        },
      })
    );

    // ------------------------------------------------------
    // ACCT.07 — Deliver CloudTrail logs to a protected S3 bucket
    // ------------------------------------------------------

    const trail = new cloudtrail.Trail(this, "AuditTrail", {
      trailName: trailName,
      bucket: auditLogsBucket,
      s3KeyPrefix: "cloudtrail",
      isMultiRegionTrail: true,
      isOrganizationTrail: false,
      includeGlobalServiceEvents: true,
      enableFileValidation: true,

      // ACCT.07: Captures both read and write management events.
      // For production environments, consider filtering high-volume events
      // per the guidance in ACCT.07.
```

```
      managementEvents: cloudtrail.ReadWriteType.ALL,
    });

    // ------------------------------------------------------
    // ACCT.11 — Enable IAM Access Analyzer (account scope)
    // ------------------------------------------------------

    const analyzer = new accessanalyzer.CfnAnalyzer(this, "AccessAnalyzer", {
      analyzerName: "account-analyzer",
      type: "ACCOUNT",
    });

    // ------------------------------------------------------
    // Outputs
    // ------------------------------------------------------

    new cdk.CfnOutput(this, "AuditLogsBucketArn", {
      description: "ARN of the audit logs S3 bucket",
      value: auditLogsBucket.bucketArn,
    });

    new cdk.CfnOutput(this, "AuditTrailArn", {
      description: "ARN of the CloudTrail trail",
      value: trail.trailArn,
    });

    new cdk.CfnOutput(this, "AccessAnalyzerArn", {
      description: "ARN of the IAM Access Analyzer",
      value: analyzer.attrArn,
    });
  }
}
```

For more information about getting started with IaC on AWS, see Getting started with the AWS CDK in the AWS CDK documentation.

> **Note**
>
> AWS CloudFormation is available at no additional charge, and the AWS CDK is open source and also available at no charge. You pay only for the AWS resources that your stacks create.

Contributors

Authoring

- Andrew Watkins, Principal Solutions Architect
- Cole Calistra, Principal Solutions Architect
- Faisal Farooq, Solutions Architect Manager
- Israel Lopez Moriano, Sr. Solutions Architect
- Jacob Beard, Solutions Architect
- Jay Michael, Principal Solutions Architect
- Justin Plock, Principal Solutions Architect
- Matthew de Anda, Sr. Solutions Architect
- Michael Nguyen, Sr. Solutions Architect
- Paul Hawkins, Principal, Office of the CISO
- Ritik Khatwani, Sr. Solutions Architect
- Swara Gandhi, Sr. Solutions Architect

Reviewing

- Abhishek Agawane, Delivery consultant
- Robert Put
- Mike Sullivan
- Bob Lee III
- Marcus Watson

Technical writing

- Lilly AbouHarb, Sr. Technical Writer

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed.

Change	Description	Date
Republished	Republished guide with significant updates throughout, including adding cost considerations.	April 6, 2026
Amazon S3 bucket settings	We updated the ACCT.08 Prevent public access to private Amazon S3 buckets section to reflect that Amazon S3 buckets created after April 28, 2023 have the **Block Public Access** setting enabled by default.	May 18, 2023
IAM security best practices	We updated this guide for alignment with the latest AWS Identity and Access Management (IAM) best practices. For more information, see Security best practices in the IAM documentation.	February 1, 2023
IAM roles	We provided additional links to AWS service documentation in the WKLD.01 Use IAM roles for compute environment permissions section.	September 22, 2022
Password policy	We updated the recommendations for strong passwords	May 10, 2022

	to use the latest guidance from the Center for Internet Security (CIS).	
Initial publication	—	April 13, 2022