AWS Prescriptive Guidance

AWS Startup Security Baseline (AWS SSB)

AWS Prescriptive Guidance: AWS Startup Security Baseline (AWS SSB)

Table of Contents

AWS Startup Security Baseline (AWS SSB)

Jay Michael, Amazon Web Services (AWS)

April 2022 (last update (p. 20): May 2022)

The Amazon Web Services (AWS) Startup Security Baseline (SSB) is a set of controls that create a minimum foundation for businesses to build securely on AWS without decreasing their agility. The controls in this guide are designed with early startups in mind, mitigating the most common security risks without requiring significant effort. As the organization grows or to address the needs of larger enterprises, you can scale and build upon these controls. They form the basis of your security posture and are focused on securing credentials, enabling logging and visibility, managing contact information, and implementing basic data boundaries.

The controls in the AWS SSB are separated into two categories, account and workload. Account controls help keep your AWS account secure. It includes recommendations for setting up user access, policies, and permissions, and it includes recommendations for how to monitor 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. It includes recommendations such as encryption and reducing the scope of access.

> **Note**
> Some of the controls recommended in this guide replace the defaults configured during initial setup, while most configure new settings and policies. This document should in no way be considered comprehensive of all available controls.

Intended audience

This guide is best suited for startups that are in the very beginning stages of development, with minimal staff and operations.

Startups or other businesses that are in later stages of operation and growth can still derive significant value from reviewing these controls against their current practices. If you identify any gaps, you can implement the individual controls in this guide and then 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 add additional controls as applicable.

Foundational framework and security responsibilities

AWS Well-Architected helps cloud architects build a secure, high-performing, resilient, and efficient infrastructure for their applications and workloads. The AWS Startup Security Baseline aligns to the security pillar of the AWS Well-Architected Framework. The *security pillar* describes how to take

advantage of cloud technologies to protect data, systems, and assets in a way that can improve your security posture. This helps you meet your business and regulatory requirements by following current AWS recommendations.

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. The shared responsibility model is often described by saying that AWS is responsible for the security of the cloud (that is, for protecting the infrastructure that runs all the services offered in the AWS Cloud), and you are responsible for the security in the cloud (as determined by the AWS Cloud services that you select). In the shared responsibility model, implementing the security controls in this document is part of your responsibility as a customer.

Securing your account

Controls and recommendations in this section help keep your AWS account secure. It emphasizes using AWS Identity and Access Management (IAM) users, user groups, 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, Amazon GuardDuty, and AWS Budgets, so that you are notified of activity in your account and can respond quickly if the activity is unauthorized or unexpected.

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, so it is important you retain access to them.

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

1. Sign in to the **Account Settings** page in the Billing and Cost Management console at https://console.aws.amazon.com/billing/home?#/account.
2. On the **Account Settings** page, next to **Account Settings**, choose **Edit**.
3. Next to the field you want to update, choose **Edit**.
4. After you have entered your changes, choose **Save changes**.
5. After you have made all of your changes, choose **Done**.

To edit your contact information

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

ACCT.02 – Restrict use of the root user

The 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. Only use the root user for the specific tasks that require it. For more information, see Tasks that require root user credentials (AWS General Reference). Perform all other actions in your account by using IAM users or federated users with IAM roles.

To restrict use of the root user

1. Require multi-factor authentication (MFA) for the root user as described in ACCT.05 – Require multi-factor authentication (MFA) to log in (p. 5).
2. Create your first IAM admin user and user group (IAM documentation).

ACCT.03 – Configure console access for each user

As a baseline, create an IAM user for each person who needs to access the AWS Management Console. Do not share credentials across users.

To create an IAM user

1. Create IAM users (IAM documentation).
2. Apply permissions according to ACCT.04 – Assign permissions by using IAM user groups (p. 4).

While IAM users are satisfactory for initial development on AWS, we recommend that you use federated users from a centralized identity provider (IdP), such as AWS Single Sign-On (AWS SSO), 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 just one set of credentials. For more information, see Identity federation in AWS (AWS website).

To set up identity federation

1. If you are using AWS SSO, see Getting started (AWS SSO documentation).
2. Make sure that your IdP enforces multi-factor authentication (MFA).

ACCT.04 – Assign permissions by using IAM user groups

You can use IAM user groups to configure permissions for multiple IAM users. Assign users to user groups based on their job functions. Examples of user groups include application, data, networking, and

Development Operations (DevOps) engineers. You can also divide the user types into smaller user groups based on decision-making authority, such as for senior or non-senior engineers.

When assigning permissions to an IAM user group, you can customize the permissions, or you can attach AWS managed policies, which 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.

To create user groups and assign permissions

1. Create IAM user groups (IAM documentation).
2. Attach an AWS managed policy to an IAM user group (IAM documentation).

ACCT.05 – Require multi-factor authentication (MFA) to log in

Enable MFA for the AWS account root user and any principal users with elevated permissions, such as those who can create, update, or delete resources or access data that is critical to business operations.

To enable MFA for the root user

1. Sign in to the AWS Management Console at https://console.aws.amazon.com/.
2. On the right side of the navigation bar, choose your account name, and then choose **My Security Credentials**.
3. If necessary, choose **Continue to Security Credentials**.
4. Expand the **Multi-Factor Authentication (MFA)** section.
5. Choose **Activate MFA**.
6. Follow the wizard instructions to configure your MFA devices accordingly. For more information, see Enabling MFA devices for users in AWS (IAM documentation).

To set up MFA for your own IAM user account

1. Use your AWS account ID or account alias, your IAM user name, and your password to sign in to the IAM console at https://console.aws.amazon.com/iam.
2. In the navigation bar on the upper right, choose your user name, and then choose **My Security Credentials**.
3. On the **AWS IAM credentials** tab, in the **Multi-factor authentication** section, choose **Manage MFA device**.

To enable MFA for other IAM users

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam.
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. Next to **Assigned MFA device**, choose **Manage**.

5. Follow the wizard instructions to configure your MFA devices accordingly. For more information, see Enabling MFA devices for users in AWS (IAM documentation).

ACCT.06 – Enforce a password policy

IAM users log in to the AWS Management Console by using a combination of username and password, with MFA recommended. Require passwords to adhere to a strong password policy to help prevent discovery through brute force or social engineering.

For more information about the latest recommendations for strong passwords, see Password Policy Guide on the Center for Internet Security (CIS) website.

You can configure password requirements in a custom IAM password policy. For more information, see Setting an account password policy for IAM users (IAM documentation).

To create a custom password policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam.
2. In the navigation pane, choose **Account settings**.
3. In the **Password policy** section, choose **Change password policy**.
4. Select the options that you want to apply to your password policy, and then choose **Save changes**.

ACCT.07 – Deliver CloudTrail logs to a protected 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 (CloudTrail documentation).

To retain CloudTrail history beyond 90 days with additional data, you create a new 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 S3 bucket

1. Create a trail (CloudTrail documentation). On the **Choose log events** page, do the following:
   a. For **API activity**, choose both **Read** and **Write**.
   b. For preproduction environments, choose **Exclude AWS KMS events**. This excludes 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, choose to log **Write** management events, and clear the check box for **Exclude AWS KMS events**. This excludes high-volume AWS KMS read events but still logs relevant write events, such as `Disable`, `Delete`, and `ScheduleKey`. These are the minimum recommended AWS KMS logging settings for a production environment.
2. The new trail appears on the **Trails** page. In about 15 minutes, CloudTrail publishes log files that show the AWS application programming interface (API) calls made in your account. You can see the log files in the S3 bucket that you specified.

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

1. Review the Amazon S3 bucket policy (CloudTrail documentation) for any and all buckets where you store log files and adjust it as needed to remove any unnecessary access.
2. As a security best practice, be sure to manually add an `aws:SourceArn` condition key to the bucket policy. For more information, see Create or update an Amazon S3 bucket to use to store the log files for an organization trail (CloudTrail documentation).
3. Enable MFA Delete (Amazon S3 documentation).

ACCT.08 – Prevent public access to private S3 buckets

By default, only the root user of the AWS account and the IAM principal, if used, have permissions to read and write to Amazon S3 buckets created by that principal. 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 access to the bucket to the general public, a *public bucket*.

Users could misconfigure the bucket policy and unintentionally grant access to the public. You can prevent this misconfiguration by enabling the **Block Public Access** setting for each bucket. If you have no current or future use cases for a public S3 bucket, enable this setting at the AWS account level. This setting prevents policies that allow public access.

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

- Configure block public access settings for your S3 buckets (Amazon S3 documentation).

AWS Trusted Advisor generates a yellow finding for S3 buckets that allow list or read access to the public and generates a red finding for buckets that allow public uploads or deletes. As a baseline, follow the control ACCT.12 – Monitor for and resolve high-risk issues by using Trusted Advisor (p. 8) to identify and correct misconfigured buckets. Publicly accessible S3 buckets are also indicated in the Amazon S3 console.

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

To reduce the opportunity for security issues, delete or turn off any 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, which enables you to assign public IP addresses in public subnets. However, 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 all Regions, not just those in the Regions where you might deploy workloads. Deleting a VPC also deletes its components, such as subnets and security groups.

> **Note**
> You can view all Regions and VPCs on the Amazon EC2 Global View console at https://console.aws.amazon.com/ec2globalview/home. For more information, see List and filter resources across Regions using Amazon EC2 Global View (Amazon EC2 documentation).

To delete unused default VPCs

1. Delete your VPC (Amazon VPC documentation).

2. Repeat as needed for VPCs in other Regions.

ACCT.10 – Configure AWS Budgets to monitor your spending

AWS Budgets enable monitoring of monthly costs and usage with notifications when costs are forecasted to exceed target thresholds. Forecasted cost notifications can provide an indication of unexpected activity, providing extra defense in addition to other monitoring systems, such as AWS Trusted Advisor and Amazon GuardDuty. Monitoring and understanding your AWS costs is also part of good operational hygiene.

To set up a budget in AWS Budgets

- Create a cost budget (AWS Budgets documentation).

ACCT.11 – Enable and respond to GuardDuty notifications

Amazon GuardDuty is a threat-detection service that continuously monitors for malicious or unauthorized behavior to help protect your AWS accounts, workloads, and data. When it detects unexpected and potentially malicious activity, GuardDuty delivers detailed security findings for visibility and remediation. GuardDuty can detect threats such as cryptocurrency mining activity, access from Tor clients and relays, unexpected behavior, and compromised IAM credentials. Enable GuardDuty and respond to findings to stop potentially malicious or unauthorized behavior in your AWS environment. For more information about findings in GuardDuty, see Finding types (GuardDuty documentation).

You can use Amazon CloudWatch Events to set up automated notifications when GuardDuty creates a finding or the finding changes. First, you set up an Amazon Simple Notification Service (Amazon SNS) topic and add endpoints, or email addresses, to the topic. Then, you set up a CloudWatch event for GuardDuty findings, and the event rule notifies the endpoints in the Amazon SNS topic.

To enable GuardDuty and GuardDuty notifications

1. Enable Amazon GuardDuty (GuardDuty documentation).
2. Create a CloudWatch Events rule to notify you of GuardDuty findings (GuardDuty documentation).

ACCT.12 – Monitor for and resolve high-risk issues by using Trusted Advisor

AWS Trusted Advisor passively 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 a complete list of checks and descriptions, see AWS Trusted Advisor check reference (Trusted Advisor documentation).

Review Trusted Advisor findings on a recurring basis, and remediate issues as necessary. If you have the AWS Business Support or Enterprise Support plans, you can subscribe to a weekly findings email. For more information, see Set up notification preferences (AWS Support documentation).

To view issues in Trusted Advisor

- Review each check category according to the instructions in View check categories (AWS Support documentation). At a minimum, we recommend reviewing the **action recommended** issues, which are red.

Securing your workloads

Controls and recommendations in this section help you secure your workloads running in AWS, while you are building them. They emphasize secure practices for managing application secrets and scope of access, minimizing access routes to private resources, and using encryption to protect 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 a person or service for a configurable period of time. Using roles eliminates the need to store or manage long-term credentials, significantly reducing 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 whenever supported. Applications that use an AWS SDK and run in these compute environments automatically use the IAM role credentials for authentication.

For an example, see Using IAM roles with Amazon Elastic Compute Cloud instances (YouTube video). The approach for using IAM roles for other AWS services is similar, and instructions can be found in the AWS Documentation for the service.

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

Policies are objects that can define permissions or specify access conditions. There are two primary types of policies:

- *Identity-based policies* are attached to principals and define what the principal's permissions 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 be allowed access to perform an action against a resource, it must have permission granted 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 (IAM documentation).

Recommended conditions for resource-based policies include:

- Restrict access to only 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 is an example of a resource-based policy that uses the `aws:PrincipalOrgID` condition to allow only principals in the **<o-xxxxxxxxxxx>** organization to access the **<bucket-name>** S3 bucket:

```
{
   "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 consist largely of credentials, such as key pairs, access tokens, digital certificates, and user name and password combinations . 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 prevents accidental exposure through less secure mechanisms, such as persisting in static configuration files. This also makes it easier to promote application code from development to production environments.

For a secrets-management service, we recommend using 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 (Systems Manager documentation).

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

You can use Parameter Store APIs to retrieve secrets stored in Secrets Manager. This allows you to 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 (AWS KMS documentation).
2. Create a SecureString parameter (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 an example in Java, see GetParameter.java (AWS Code Sample Catalog).

To manage secrets in Secrets Manager

1. Create a secret (Secrets Manager documentation).
2. Retrieve secrets from AWS Secrets Manager in code (Secrets Manager documentation).

   It is important to read Use AWS Secrets Manager client-side caching libraries to improve the availability and latency of using your secrets (AWS blog post). Using client-side SDKs, which already have best practices implemented, should accelerate and simplify the use and integration of Secrets Manager.

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. Unsecured repositories hosted on public service providers can be subject to unauthorized access and subsequent discovery of these secrets. Use available tools to prevent secrets from being checked in. Incorporate checks for exposed secrets as part of your manual code review processes.

Some common tools that can prevent application secrets from being checked-in to source code repositories are:

- Gitleaks (GitHub repository)
- Whispers (GitHub repository)
- detect-secrets (GitHub repository)
- git-secrets (GitHub repository)
- TruffleHog (GitHub repository)

WKLD.05 – Detect and remediate exposed secrets

In WKLD.03 – Use ephemeral secrets or a secrets-management service (p. 11) and WKLD.04 – Prevent application secrets from being exposed (p. 12), you put measures in place to protect secrets. In this

control, you deploy a solution that can detect if secrets have bypassed these prevention measures, and you can remediate accordingly.

Amazon CodeGuru Reviewer detects application secrets in source code and provides a mechanism to remediate and publish the detected secrets in Secrets Manager. Application code for retrieving the secret from Secrets Manager is also provided. Conduct a cost-benefit analysis to determine if this solution is right for your business. As an alternative, some of the open-source solutions in WKLD.04 – Prevent application secrets from being exposed (p. 12) provide detection capability for existing secrets.

To set up CodeGuru Reviewer integration with Secrets Manager

- Use CodeGuru Reviewer to identify hardcoded secrets and AWS Secrets Manager to secure them (AWS blog post and guided walkthrough).

WKLD.06 – Use Systems Manager instead of SSH or RDP

Public subnets, which have a default route pointing to an internet gateway, are inherently a greater security risk than *private subnets*, those with no route to the internet. You can run 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, preventing the need to manage additional credentials used for Secure Shell (SSH) or Windows remote desktop protocol (RDP).

Use Session Manager instead of running EC2 instances in public subnets, running jump boxes, or running bastion hosts.

To set up Session Manager

1. Make sure the EC2 instance is using the latest operating system Amazon Machine Images (AMIs), such as Amazon Linux 2 or Ubuntu. The AWS Systems Manager Agent (SSM Agent) is pre-installed on the AMI.
2. Make sure the instance has connectivity, either through an internet gateway or through VPC endpoints, to these addresses (replacing **`<region>`** with the appropriate AWS Region):
    a. Ec2messages.<region>.amazonaws.com
    b. ssm.<region>.amazonaws.com
    c. ssmmessages.<region>.amazonaws.com
3. Attach the AWS managed policy `AmazonSSMManagedInstanceCore` to the IAM role that is associated to your instances.

For more information, see Setting up Session Manager (Systems Manager documentation).

To start a session

- Start a session (Systems Manager documentation).

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

By default, AWS CloudTrail captures management events, events that create, modify, or delete resources in your account. These management events do not capture read or write operations to individual objects in Amazon Simple Storage Service buckets. During a security event, it is important to capture unauthorized data access or use at an individual record or object level. Use CloudTrail to log data events for any S3 buckets that store sensitive or business-critical data, for detection and auditing purposes.

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

To log data events for trails

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/
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**.
    d. In Individual bucket selection, browse for the bucket on which to log data events. You can select multiple buckets in this window. Choose **Add bucket** to log data events for more buckets. Choose to log **Read** events, such as `GetObject`, **Write** events, such as `PutObject`, or both.
    e. Choose **Update trail**.

WKLD.08 – Encrypt Amazon EBS volumes

Enforce encryption of Amazon Elastic Block Store (Amazon EBS) volumes as the default behavior in your AWS account. Encrypted volumes have the same input/output operations per second (IOPS) performance as unencrypted volumes with a minimal effect on latency. This prevents rebuilding volumes at a later date for compliance or other reasons. For more information, see Must-know best practices for Amazon EBS encryption (AWS blog post).

To encrypt Amazon EBS volumes

- Enable encryption by default (Amazon EC2 documentation).

WKLD.09 – Encrypt Amazon RDS databases

Similar to WKLD.08 – Encrypt Amazon EBS volumes (p. 14), enable encryption of Amazon Relational Database Service (Amazon RDS) databases. This encryption is performed at the underlying volume level and has the same IOPS performance as unencrypted volumes with a minimal effect on latency. For more information, see Overview of encrypting Amazon RDS resources (Amazon RDS documentation).

To encrypt an RDS database instance

- Encrypt a database instance (Amazon RDS documentation).

WKLD.10 – Deploy private resources into private subnets

Deploy resources that don't require direct internet access, such as 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 originating from a private subnet that is destined for the internet must undergo network address translation (NAT) through either a managed AWS NAT Gateway or an EC2 instance running NAT processes in a public subnet. For more information about network isolation, see Infrastructure security in Amazon VPC (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 EC2 instances, disable **Auto-assign Public IP**. This prevents a public IP from being assigned if the instance is unintentionally deployed into a public subnet via misconfiguration.

You specify the subnet for a resource as part of its configuration, when required. You can deploy a VPC that follows best practices using the Modular and Scalable VPC Architecture Quick Start (AWS Quick Starts).

WKLD.11 – Restrict network access by using security groups

Use security groups to control traffic to EC2 instances, RDS databases, and other supported resources. Security groups act as a virtual firewall that can be applied to any group of related resources in order 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 to other security groups. For example, a database security group can have rules to allow only traffic from an application server security group.

By default, security groups allow all outbound traffic but don't allow inbound traffic. The outbound traffic rule can be removed, or you can configure additional rules added to restrict outbound traffic and allow inbound traffic. If the security group has no outbound rules, no outbound traffic originating from your instance is allowed. For more information, see Control traffic to resources using security groups (Amazon VPC documentation).

In the following example, there are three security groups that control traffic from an Application Load Balancer to EC2 instances that connect to an Amazon RDS for MySQL database.

Security group	Inbound rules	Outbound rules
Application Load Balancer security group	**Description:** Allow HTTPS traffic from anywhere **Type:** HTTPS **Source:** Anywhere-IPv4 (0.0.0.0/0)	**Description:** Allow all traffic to anywhere **Type:** All traffic **Destination:** Anywhere-IPv4 (0.0.0.0/0)

Security group	Inbound rules	Outbound rules
EC2 instance security group	**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)
RDS database security group	**Description:** Allow MySQL traffic from EC2 instance **Type:** MySQL **Source:** EC2 instance security group	No outbound rules

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, preventing the need to use an internet gateway, NAT device, virtual private network (VPN) connection, or AWS Direct Connect connection.

VPC endpoints support attaching policies and security groups to further 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 all resources in the VPC, regardless of their own permission policy. You can also write an S3 bucket policy to allow only requests originating from a specific VPC endpoint, denying all other external access. A VPC endpoint can also have a security group rule that, for example, restricts access to only EC2 instances that are associated to an application-specific security group, such as the business-logic tier of a web application.

There are different kinds of VPC 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. Gateway endpoints are recommended for workloads contained within a single AWS account and Region, and come at no additional charge. Interface endpoints are recommended if more extensible access is required, such as to an S3 bucket from other VPCs, from on-premises networks, or from different AWS Regions. Interface endpoints incur an hourly uptime charge and a per-GB data-processing charge, both of which are lower than the respective charges for sending the data to `0.0.0.0/0` through AWS NAT Gateway.

See the following resources for additional information about using VPC endpoints:

- For more information about selecting between gateway and interface endpoints for Amazon S3, see Choosing Your VPC Endpoint Strategy for Amazon S3 (AWS blog post).
- Create an interface endpoint (Amazon VPC documentation).
- Create a gateway endpoint (Amazon VPC documentation).
- For example S3 bucket policies that restrict access to a specific VPC or VPC endpoint, see Restricting access to a specific VPC (Amazon S3 documentation).
- For example DynamoDB endpoint policies that restrict actions, see Endpoint policies for DynamoDB (Amazon VPC documentation).

WKLD.13 – Require HTTPS for all public web endpoints

Require HTTPS to provide additional credibility to your web endpoints, allow your endpoints to use certificates to prove their identity, and confirm that all traffic between your endpoint and connected clients is encrypted. For public websites, this provides the additional benefit of higher 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 require HTTPS for each of these services, see the following:

- Elastic Beanstalk (Elastic Beanstalk documentation)
- CloudFront (CloudFront documentation)
- Application Load Balancer (AWS Knowledge Center)
- Classic Load Balancer (AWS Knowledge Center)
- Amplify (Amplify documentation)

Static websites hosted on Amazon S3 do not support HTTPS. To require HTTPS for these websites, you can use CloudFront. Public access to S3 buckets that are serving content through CloudFront is not required.

To use CloudFront to serve a static website hosted on Amazon S3

1. Use CloudFront to serve a static website hosted on Amazon S3 (AWS Knowledge Center).
2. If you are configuring access to a public S3 bucket, require HTTPS between viewers and CloudFront (CloudFront documentation).

   If you are configuring access to a private S3 bucket, restrict access to Amazon S3 content by using an origin access identity (CloudFront documentation).

In addition, 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-FS-1-2-Res-2020-10` or the most recent policy available for Application Load Balancer HTTPS listeners, instead of the default `ELBSecurityPolicy-2016-08`. The most current policies require TLS 1.2 at 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:

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

WKLD.14 – Use edge-protection services for public endpoints

Rather than serve traffic direct from compute services such as EC2 instances or containers, use an edge-protection service. This provides an additional layer of security between incoming traffic from the

internet and your resources that serve that traffic. These services can filter unwanted traffic, enforce encryption, and apply routing or other rules, such as load balancing, before traffic reaches your internal resources.

AWS services that can provide public endpoint protection include the AWS WAF, CloudFront, Elastic Load Balancing, API Gateway, and Amplify Hosting. Run VPC-based services, such as Elastic Load Balancing, in a public subnet as a proxy to web service resources running in a private subnet.

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

Instructions for getting started with each of these services can be found here:

- Getting Started with AWS WAF (AWS website)
- Getting started with Amazon CloudFront (CloudFront documentation)
- Getting started with Elastic Load Balancing (Elastic Load Balancing documentation)
- Getting started with API Gateway (API Gateway documentation)
- Getting started with Amplify Hosting (Amplify documentation)

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

Infrastructure as code (IaC) is the practice of defining all of your AWS service 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 services, such as AWS CloudFormation, support both IAM identity-based and resource-based policies and support AWS security services, such as Amazon GuardDuty, AWS WAF, and Amazon VPC. Capture these artifacts as IaC templates, commit the templates to a source code repository, and then deploy them by using CI/CD pipelines.

Unless required otherwise, commit application permission policies with application code in the same repository, and manage general resource policies and security service configurations in separate code repositories and deployment pipelines.

For more information about getting started with IaC on AWS, see the AWS Cloud Development Kit (CDK) documentation.

Contributors

Contributors to this document include:

- Jay Michael, Principal Solutions Architect
- Cole Calistra, Principal Solutions Architect
- Justin Plock, Principal Solutions Architect
- Faisal Farooq, Solutions Architect
- Michael Nguyen, Sr. Solutions Architect
- Ritik Khatwani, Sr. Solutions Architect
- Paul Hawkins, Principal, Office of the Chief Information Security Officer (CISO)

A special thank you to the following people who also helped with guidance and review:

- Robert Put
- Mike Sullivan
- Bob Lee III

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.

update-history-change	update-history-description	update-history-date
Password policy (p. 20)	We updated the recommendations for strong passwords to use the latest guidance from the Center for Internet Security (CIS).	May 10, 2022
Initial publication (p. 20)	—	April 13, 2022