Autonomous Driving Data Framework (ADDF) security and operations guide

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Autonomous Driving Data Framework (ADDF) security and operations guide

Table of Contents

Autonomous Driving Data Framework (ADDF) security and operations guide

Andreas Falkenberg, Junjie Tang, Torsten Reitemeyer, and Srinivas Reddy Cheruku, Amazon Web Services (AWS)

November 2022 (document history)

Autonomous Driving Data Framework (ADDF) is an open-source project designed to provide reusable, modular code artifacts for automotive teams who want to implement common tasks for advanced driver-assistance systems (ADAS), such as configuring centralized data storage, data processing pipelines, visualization mechanisms, search interfaces, simulation workloads, analytics interfaces, and prebuilt dashboards. Using ADDF, you can share, modify, or create fully customizable modules that reduce the amount of effort required to create and deploy these solutions.

This guide is intended to help you understand best practices for securely deploying and operating ADDF in the AWS Cloud. It discusses the following topics:

- Architecture and terminology – Review the general architecture, workflows, and important terms.
- Shared responsibility model – Understand your role and the role of AWS in securing your ADDF deployment and cloud resources.
- Security review process – Because ADDF is an open-source project, review how AWS and contributors complete security reviews.
- Built-in security features – Review how security best practices and features are built into the ADDF open-source project and its deployment framework.
- Secure setup and operation – Learn how to deploy and operate ADDF in the AWS Cloud.

Intended audience

This guide is intended for Development Operations (DevOps) teams, infrastructure engineers, administrators, IT security staff, and incident response teams who are tasked with assessing, deploying, customizing, and operating ADDF. You can apply the recommendations in this guide for proof-of-concept or production environments.

This guide assumes you have no prior knowledge of ADDF. However, we recommend that you read the ADDF readme (GitHub) before proceeding.

Targeted business outcomes

This guide is designed to help you more confidently and securely set up and operate ADDF in development and production environments.

ADDF architecture and terminology

Before you can understand the security and operational topics in this guide, it's important to have a high-level understanding of the terminology, components, and architecture of Autonomous Driving Data Framework (ADDF). This section consists of the following topics:

- ADDF terminology
- ADDF architecture

ADDF terminology

The key terminology for ADDF is as follows:

- **ADDF module** – A module is infrastructure as code (IaC) that implements a common task in an advanced driver-assistance system (ADAS). Common tasks include configuring centralized data storage, data processing pipelines, visualization mechanisms, search interfaces, simulation workloads, analytics interfaces, and prebuilt dashboards. You can create a module based on your requirements, or you can re-use or customize an existing module.

  You can use the AWS Cloud Development Kit (AWS CDK) to define ADDF modules, or you can use any common IaC framework, such as Hashicorp Terraform or AWS CloudFormation, to implement the ADDF modules. A module has a set of input parameters. Input parameters can depend on output values from other modules. An ADDF module is the smallest unit of deployment for an ADDF target AWS account.
- **ADDF deployment manifest file** – This file defines an orchestration of standalone ADDF modules. *Orchestration* refers to the deployment order of the modules. In the ADDF deployment manifest file, you can use *ADDF groups* to group related modules together. In this file, you also define the ADDF toolchain AWS account, the ADDF target AWS accounts, and the target AWS Regions.
- **ADDF deployment framework** – This framework deploys ADDF modules into the ADDF target AWS accounts based on the orchestration defined in the ADDF deployment manifest file. The ADDF deployment framework is implemented by using the following AWS open-source projects:
  - SeedFarmer (GitHub) – SeedFarmer is the CLI tool used for ADDF deployments. It manages each module state, prepares and packages the module code, creates the least-privilege policies for the ADDF deployment roles, and provides semantic instructions that CodeSeeder uses for

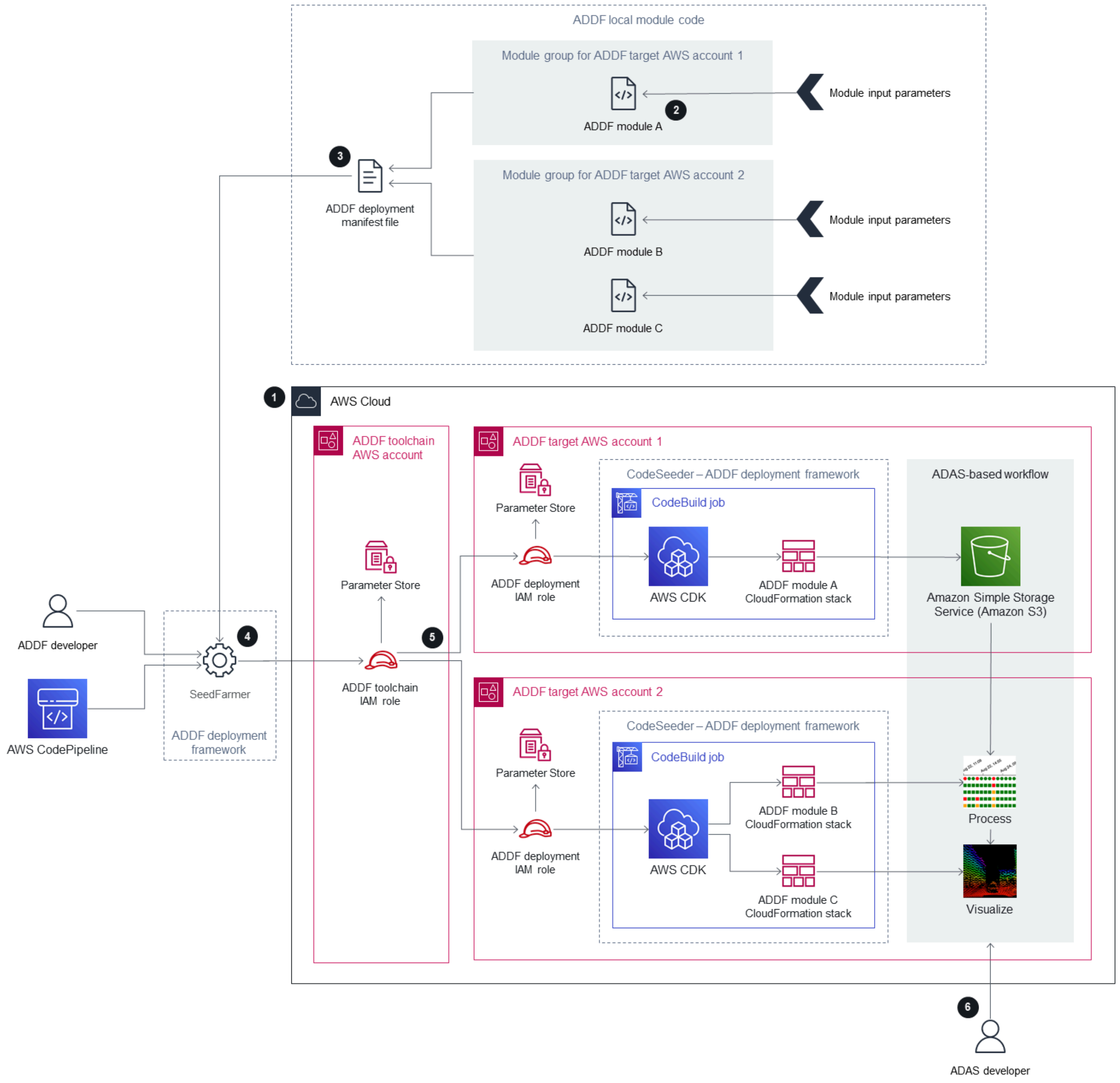
  deployment. You can interact directly with SeedFarmer to run ADDF deployments, or you can integrate it into a continuous integration and continuous deployment (CI/CD) pipeline.
  - CodeSeeder (GitHub) – CodeSeeder deploys arbitrary infrastructure as code packages through an AWS CodeBuild job. SeedFarmer automatically orchestrates and runs CodeSeeder. Only SeedFarmer directly interacts with CodeSeeder.

The ADDF deployment framework is designed to support deployments in single-account and multi-account architectures. Based on your organization's requirements, you decide whether a single-account or multi-account architecture is required.

- **ADDF toolchain AWS account –** This account orchestrates and manages the deployment of modules into the ADDF target AWS accounts, based on the definitions in the ADDF deployment manifest file. An ADDF deployment can only have one ADDF toolchain AWS account. In a single-account architecture, the ADDF toolchain AWS account is also the ADDF target AWS account. This account contains an AWS Identity and Access Management (IAM) role, called *ADDF toolchain IAM role*, which is assumed by SeedFarmer during the ADDF deployment process. In this guide, we refer to an ADDF toolchain AWS account as a *toolchain account*.
- **ADDF target AWS accounts –** These are the target accounts where you are deploying ADDF modules. You can have one or more target accounts. These accounts contain the resources and application logic described in the ADDF deployment manifest file and its mapped modules. In a single-account architecture, the ADDF target AWS account is also the ADDF toolchain AWS account. Each ADDF target account contains an IAM role, called *ADDF deployment IAM role*, that is assumed by CodeSeeder during the deployment process. In this guide, we refer to an ADDF target AWS account as a *target account*.
- **ADDF instance –** When you deploy ADDF and your modules in the cloud, as defined in your ADDF deployment manifest file, this becomes an *ADDF instance*. An ADDF instance can have a single-account or multi-account architecture, and you can deploy multiple ADDF instances. For more information about choosing the number of instances and designing an account architecture for your use case, see Defining your ADDF architecture.

ADDF architecture

The following diagram shows a high-level architecture for an ADDF instance in the AWS Cloud. It shows a multi-account architecture, including a dedicated toolchain account and two target accounts. This guide discusses the end-to-end process of using ADDF to deploy resources to the target accounts.


ADDF local module code
Module group for ADDF target AWS account 1
ADDF module A
Module input parameters
2
3
ADDF deployment manifest file
Module group for ADDF target AWS account 2
ADDF module B
Module input parameters
ADDF module C
Module input parameters
1
AWS Cloud
ADDF toolchain AWS account
Parameter Store
ADDF toolchain IAM role
5
ADDF target AWS account 1
Parameter Store
ADDF deployment IAM role
CodeSeeder – ADDF deployment framework
CodeBuild job
AWS CDK
ADDF module A CloudFormation stack
ADAS-based workflow
Amazon Simple Storage Service (Amazon S3)
ADDF target AWS account 2
Parameter Store
ADDF deployment IAM role
CodeSeeder – ADDF deployment framework
CodeBuild job
AWS CDK
ADDF module B CloudFormation stack
ADDF module C CloudFormation stack
Process
Visualize
ADDF developer
AWS CodePipeline
4
SeedFarmer
ADDF deployment framework
6
ADAS developer

1. **Create and bootstrap the ADDF AWS accounts.**

   To function properly, each account must be bootstrapped to ADDF and to AWS CDK. If this a new ADDF deployment or you are adding new target accounts, do the following:

   a. Bootstrap AWS CDK in the toolchain account and each target account. For instructions, see Bootstrapping (AWS CDK documentation). ADDF uses AWS CDK to deploy its infrastructure.

b. Bootstrap ADDF in the toolchain account and each target account. For instructions, see *Bootstrap AWS account(s)* in the ADDF Deployment Guide. This sets up all ADDF-specific IAM roles required by SeedFarmer and CodeSeeder.

> **Note**
>
> You need to perform this step only if you're initially deploying ADDF or adding new target accounts. This step is not part of reoccurring ADDF deployments to already established ADDF instances.

2. **Create or customize the ADDF modules.**

   Create or customize ADDF modules based on the specific problem you are trying to solve. Your module should represent an isolated task or group of tasks. Define the input parameters for the module as needed, and use the module output values as input parameters for other modules.

3. **Define the module orchestration in the ADDF deployment manifest file.**

   In the ADDF manifest file, organize modules into groups and define the deployment order and dependencies between them. In this file, you also specify the single toolchain account and the target accounts (including AWS Regions) for each ADDF group and its modules.

4. **Evaluate the ADDF deployment manifest file and establish the deployment scope.**

   The ADDF developer or a CI/CD pipeline, such as AWS CodePipeline, starts an evaluation of the ADDF deployment manifest file by calling the CLI tool, SeedFarmer. To start the evaluation:

   - SeedFarmer uses the ADDF deployment manifest file as an input parameter for the evaluation.
   - To assume the ADDF toolchain IAM role, SeedFarmer expects the same, valid IAM role or user credentials that was defined during the ADDF bootstrap process, in step 1.

   If SeedFarmer does not have the correct credentials to assume the ADDF toolchain IAM role or can't access the ADDF deployment manifest file, the evaluation does not start.

   If SeedFarmer can start the evaluation, it assumes the ADDF toolchain IAM role in the toolchain account. From there, SeedFarmer can access any target account, by assuming the ADDF deployment IAM role in that account. SeedFarmer then tries to read any ADDF metadata in the toolchain account and target accounts. One of the following happens:

- If there is no ADDF metadata to read, that indicates that this is a new ADDF instance. SeedFarmer determines that the deployment scope is the entire the ADDF deployment manifest file and its contents.
- If ADDF metadata exists, SeedFarmer compares the ADDF deployment manifest file and its contents to the MD5 hashes of the existing deployed artifacts in the target accounts. If deployable changes are detected, this process continues. If no deployable changes are detected, the process is complete.

5. **Deploy the in-scope ADDF modules to the target accounts.**

   CodeSeeder now has an ordered list of deployments to run, according to the ADDF deployment manifest file and the evaluation results from the previous step. Based on that ordered list, CodeSeeder assumes the ADDF deployment IAM role in each associated target account. It then runs CodeSeeder in an AWS CodeBuild job to create or update the individual IaC deployments, such as AWS CDK applications, for the ADDF module. By default, ADDF uses AWS CDK as its IaC framework, but other common IaC frameworks are also supported. After the process is complete for each target account, you have a fully deployed, cross-account, end-to-end ADAS-based workflow, as you defined in the ADDF deployment manifest file.

   If you use a single-account architecture, the toolchain account and the target accounts are the same account, and the one account has all of the described functionality.

6. **Use the ADDF-deployed infrastructure.**

   An ADAS developer can use the deployed ADAS-based workflow, as defined by your use case.

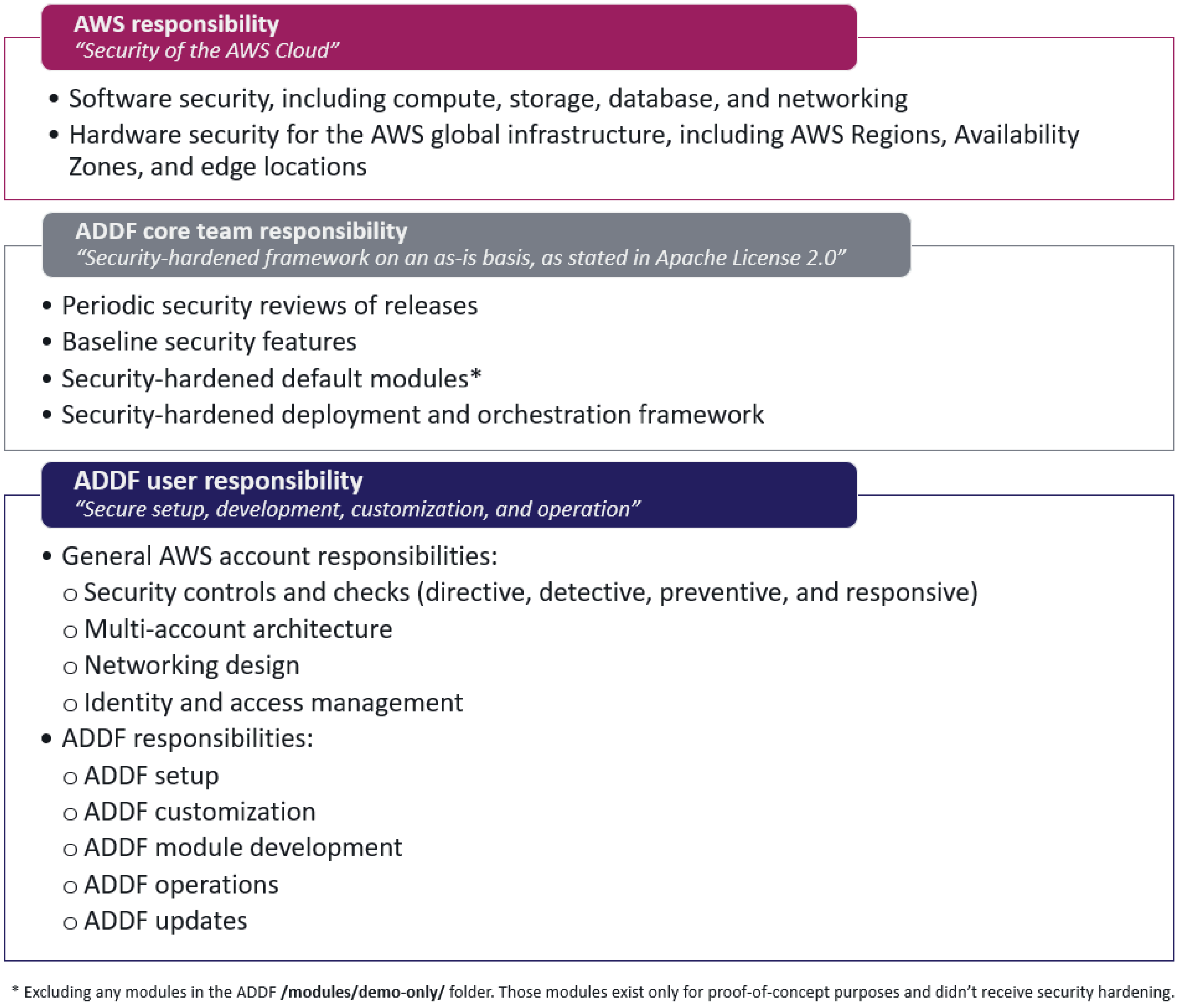
   This workflow describes the architecture of a single instance of an ADDF multi-account environment. Depending on your development, deployment, and operations model, we recommend that you run multiple ADDF instances in a multi-stage environment. A typical setup might include a dedicated ADDF instance with dedicated AWS accounts for each deployment stage, such as branches for development, testing, and production. You can also run multiple ADDF instances in the same single-account or multi-account environment in the same AWS Region, assuming that you created a unique resource namespace for each ADDF instance. For more information, see Defining your ADDF architecture.

ADDF shared responsibility model

The shared responsibility model that applies to AWS services also applies to Autonomous Driving Data Framework (ADDF). The following entities share the responsibility to secure ADDF as set out in the following diagram:

- **AWS –** The cloud infrastructure provider offering AWS services.
- **ADDF core team –** The ADDF core team is the entity that publishes ADDF releases in the ADDF repository (GitHub).
- **ADDF user –** ADDF users include, but aren't limited to:
  - **ADDF developer –** Anyone that changes, customizes, or creates new ADDF module code.
  - **ADDF operator –** Anyone that sets up and operates an ADDF instance.
  - **ADAS developer –** The end-user or consumer of the resources deployed by ADDF. For example, an ADAS developer can query a visualization frontend that was created as part of the ADDF deployment.

The following diagram summarizes the shared responsibility between AWS, the ADDF core team, and the ADDF user.



AWS responsibility
"Security of the AWS Cloud"

- Software security, including compute, storage, database, and networking
- Hardware security for the AWS global infrastructure, including AWS Regions, Availability Zones, and edge locations

ADDF core team responsibility
"Security-hardened framework on an as-is basis, as stated in Apache License 2.0"

- Periodic security reviews of releases
- Baseline security features
- Security-hardened default modules*
- Security-hardened deployment and orchestration framework

ADDF user responsibility
"Secure setup, development, customization, and operation"

- General AWS account responsibilities:
  - Security controls and checks (directive, detective, preventive, and responsive)
  - Multi-account architecture
  - Networking design
  - Identity and access management
- ADDF responsibilities:
  - ADDF setup
  - ADDF customization
  - ADDF module development
  - ADDF operations
  - ADDF updates

* Excluding any modules in the ADDF **/modules/demo-only/** folder. Those modules exist only for proof-of-concept purposes and didn't receive security hardening.

AWS responsibility

AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud, as defined in the AWS shared responsibility model. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services.

ADDF core team responsibility

The ADDF core team provides a framework that is secure in itself, on a best-effort basis, according to Apache License 2.0 (GitHub). The ADDF core team is responsible for the following:

- Periodic security reviews of releases
- Baseline security features
- Security-hardened default modules (This excludes any modules in the `/modules/demo-only/` folder. Those modules are only for proof-of-concept purposes and don't receive security hardening.)
- Security-hardened deployment and orchestration framework

These security responsibilities extend only to the framework, as provided in the GitHub repository, without any modifications or customization. This includes all ADDF modules, except ADDF modules in the `modules/demo-only/` folder. ADDF modules in this folder aren't security hardened and shouldn't be deployed in production environments or in any environment with sensitive or protected data. These modules are included to showcase system capabilities, and you can use them as the base for creating your own customized, security-hardened modules.

Note

ADDF as a framework is delivered on an as-is basis. It doesn't come with any liability and warranty, as stated in the Apache License 2.0 (GitHub). You should conduct your own security assessment of ADDF and verify it's compliant with your organization's specific security requirements.

ADDF user responsibility

ADDF and its modules are secure only if ADDF is set up, customized, and operated in a secure manner. The ADDF user is fully responsible for the security of the following:

- General AWS account responsibilities:
  - Security controls and checks (directive, detective, preventive, and responsive)
  - Multi-account architecture
  - Networking design

  - Identity and access management
- ADDF-specific responsibilities:
  - ADDF setup
  - ADDF customization
  - ADDF module development
  - ADDF operations
  - ADDF updates

General AWS account responsibilities

Before you deploy any ADDF-related resources into AWS accounts, your AWS accounts should be configured according to the best practices in the AWS Well-Architected Framework. This includes directive, detective, preventive, and responsive security controls. You should have detailed mitigation processes in place, in case of any security violations or incidents. Your organization's policy should include requirements for centrally managing identity and access and networking. Commonly, these requirements and services are handled by a dedicated landing zone team.

ADDF-specific responsibilities

Secure ADDF setup

An ADDF user's responsibility starts with the secure setup of ADDF according to the ADDF documentation. We highly recommend that you follow the instructions in the ADDF Deployment Guide (GitHub). For more information about securely setting up ADDF, see Defining your ADDF architecture and Initial setup.

Secure ADDF customization

In case of any customization of ADDF core functionality, such as CodeSeeder, SeedFarmer, and ADDF core modules, the ADDF user assumes full responsibility for those changes. For more information, see Customizing the ADDF deployment framework code.

Secure ADDF module development

The ADDF user is fully responsible for any custom module that is deployed using ADDF. Furthermore, the ADDF user is responsible for any code changes to ADDF-supplied modules. For more information, see Writing custom modules in ADDF.

Secure ADDF updates and operations

As the framework evolves, ADDF receives feature and security updates. It is the ADDF user's responsibility to regularly check for updates published to the GitHub repository and to operate ADDF securely over the long-term. For more information, see Reoccurring ADDF deployments, Reoccurring security audits, ADDF updates, and Decommissioning.

ADDF security review process

Autonomous Driving Data Framework (ADDF) was built with security in mind. Before release to the public, AWS performed an initial, internal security review of ADDF and resolved any identified security issues. Both AWS and the open-source community contribute to ongoing security reviews of the framework.

Regular security reviews by AWS

ADDF is published under the awslabs GitHub organization that is owned by AWS. AWS performs regular automatic and manual security reviews of the code in this organization, to verify security on a best-effort basis. According to AWS policy, AWS doesn't disclose information about the security review frequency, approach, or tools used. Furthermore, AWS doesn't publish any internal audit reports about ADDF. However, any identified security findings are fixed and published through pull request, with high urgency.

Note

ADDF as a framework is delivered on an 'AS-IS' BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including without limitation, any warranties or conditions of title, non-infringement, merchantabiity, or fitness for a particulary purpose, as stated in the Apache License 2.0 (GitHub). You should conduct your own security assessment of ADDF and verify whether it's compliant with your organization's specific security requirements and, as set forth in Apache License 2.0, you are solely responsible for determining the appropriateness of using or redistributing ADDF and assume any risks associated with your exercise or permissions under such license.

Open-source security reviews and contributions

ADDF is an open-source project that welcomes contributions. We invite all users to conduct their own security review of the framework and contribute by reporting any security-related findings. If you find an issue in the code, please follow the guidelines in Security issue notifications (ADDF documentation).

ADDF built-in security features

Autonomous Driving Data Framework (ADDF) has various built-in security features. By default, these features are designed to help you set up a secure framework and help your organization meet common enterprise security requirements.

The following are the built-in security features:

- Least privilege for ADDF module code
- Infrastructure as code
- Automated security checks for IaC
- Custom least-privilege policy for the AWS CDK deployment role
- Least-privilege policy for the module deployspec file
- Data encryption
- Credential storage using Secrets Manager
- Security reviews of SeedFarmer and CodeSeeder
- Permissions boundary support for the AWS CodeBuild role for CodeSeeder
- AWS multi-account architecture
- Least-privilege permissions for multi-account deployments

Least privilege for ADDF module code

Least privilege is the security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions. ADDF-provided modules strictly follow the principle of least privilege in their code and deployed resources, as follows:

- All AWS Identity and Access Management (IAM) policies generated for an ADDF module have the minimum permissions needed for the use case.
- AWS services are configured and deployed according to the principle of least privilege. ADDF-provided modules use only the services and service features that are required for the specific use case.

Infrastructure as code

ADDF, as a framework, is designed to deploy ADDF modules as infrastructure as code (IaC). IaC eliminates manual deployment processes and helps prevent errors and misconfigurations, which can result from manual processes.

ADDF is designed to orchestrate and deploy modules by using any common IaC framework. This includes, but isn't limited to:

- AWS Cloud Development Kit (AWS CDK)
- AWS CloudFormation
- Hashicorp Terraform

You can use different IaC frameworks to write different modules, and then you use ADDF to deploy them.

The default IaC framework used by ADDF modules is AWS CDK. AWS CDK is a high-level object-oriented abstraction that you can use to define AWS resources imperatively. AWS CDK already enforces security best practices by default for various services and scenarios. By using AWS CDK, the risk of security misconfigurations is reduced.

Automated security checks for IaC

The open-source cdk-nag utility (GitHub) is integrated into ADDF. This utility automatically checks ADDF modules that are based on AWS CDK for adherence to general and security best practices. The cdk-nag utility uses rules and rule packs to detect and report code that violates best practices. For more information about the rules and a comprehensive list, see cdk-nag rules (GitHub).

Custom least-privilege policy for the AWS CDK deployment role

ADDF makes extensive use of AWS CDK v2. It is required that you bootstrap all ADDF AWS accounts to AWS CDK. For more information, see Bootstrapping (AWS CDK documentation).

By default, AWS CDK assigns the permissive AdministratorAccess AWS managed policy to the AWS CDK deployment role created in bootstrapped accounts. The complete name of this role is `cdk-[CDK_QUALIFIER]-cfn-exec-role-[AWS_ACCOUNT_ID]-[REGION]`. AWS CDK uses this role to deploy resources into the bootstrapped AWS account as part of the AWS CDK deployment process.

Depending on your organization's security requirements, the `AdministratorAccess` policy might be too permissive. As part of the AWS CDK bootstrap process, you can customize the policy and permissions according to your needs. You can change the policy can by re-bootstrapping the account with a newly defined policy by using the `--cloudformation-execution-policies` parameter. For more information, see Customizing bootstrapping (AWS CDK documentation).

Note

Although this security feature isn't specific to ADDF, it is listed in this section because it can increase the overall security of your ADDF deployment.

Least-privilege policy for the module deployspec file

Each module contains a deployment specifications file that is called **deployspec.yaml**. This file defines the deployment instructions for the module. CodeSeeder uses it to deploy the defined module in the target account by using AWS CodeBuild. CodeSeeder assigns a default service role to CodeBuild to deploy the resources, as instructed in the deployment specifications file. This service role is designed according to the least-privilege principle. It includes all required permissions to deploy AWS CDK applications, because all ADDF-provided modules are created as AWS CDK applications.

However, if you need to run any stage commands outside of AWS CDK, you need to create a custom IAM policy instead of using the default service role for CodeBuild. For example, if you are using an IaC deployment framework other than AWS CDK, such as Terraform, you need to create an IAM policy that grants sufficient permissions for that specific framework to function. Another scenario that requires a dedicated IAM policy is when you include direct AWS Command Line Interface (AWS CLI) calls to other AWS services in the `install`, `pre_build`, `build`, or `post_build` stage commands. For example, you need a custom policy if your module includes an Amazon Simple Storage Service (Amazon S3) command to upload files to an S3 bucket. The custom IAM policy provides fine-grained control for any AWS command outside of the AWS CDK deployment. For an example custom IAM policy, see ModuleStack (SeedFarmer documentation). When creating a custom IAM policy for your ADDF module, make sure that you apply least-privilege permissions.

Data encryption

ADDF stores and processes potentially sensitive data. To help protect this data, SeedFarmer, CodeSeeder, and ADDF-provided modules encrypt data at rest and in transit for all used AWS services (unless explicitly stated otherwise for modules in the `demo-only` folder).

Credential storage using Secrets Manager

ADDF handles various secrets for different services, such as Docker Hub, JupyterHub, and Amazon Redshift. ADDF uses AWS Secrets Manager to store any ADDF-related secrets. This helps you remove sensitive data from the source code.

Secrets Manager secrets are stored only in the target accounts, as needed for that account to function properly. By default, the toolchain account doesn't contain any secrets.

Security reviews of SeedFarmer and CodeSeeder

SeedFarmer and CodeSeeder (GitHub repositories) are used to deploy ADDF and its ADDF modules. These open-source projects undergo the same regular AWS internal security review process as ADDF, as described in ADDF security review process.

Permissions boundary support for the AWS CodeBuild role for CodeSeeder

IAM *permissions boundaries* are a common security mechanism that defines the maximum permissions that an identity-based policy can grant to an IAM entity. SeedFarmer and CodeSeeder support an IAM permissions boundary attachment for each target account. The permissions boundary limits the maximum permissions of any service role used by CodeBuild when CodeSeeder deploys modules. IAM permissions boundaries must be created outside of ADDF by a security team. IAM permissions boundary policy attachments are accepted as an attribute within the ADDF deployment manifest file, **deployment.yaml**. For more information, see Permissions boundary support (SeedFarmer documentation).

The workflow is as follows:

1. Your security team defines and creates an IAM permissions boundary according to your security requirements. The IAM permissions boundary must be individually created in each ADDF AWS account. The output is a permissions boundary policy Amazon Resource Name (ARN) list.
2. The security team shares the policy ARN list with your ADDF developer team.
3. The ADDF developer team integrates the policy ARN list into the manifest file. For an example of this integration, see sample-permissionboundary.yaml (GitHub) and Deployment manifest (SeedFarmer documentation).
4. After successful deployment, the permissions boundary is attached to all service roles that CodeBuild uses to deploy modules.
5. The security team monitors that the permissions boundaries are applied as needed.

AWS multi-account architecture

As defined in the security pillar of the AWS Well-Architected Framework, it is considered best practice to separate resources and workloads into multiple AWS accounts, based on your organization's requirements. This is because an AWS account acts as an isolation boundary. For more information, see AWS account management and separation. The implementation of this concept is called a *multi-account architecture*. A properly designed AWS multi-account architecture provides workload categorization and reduces the scope of impact in the event of a security breach, as compared to a single-account architecture.

ADDF natively supports AWS multi-account architectures. You can distribute your ADDF modules across as many AWS accounts as needed for your organization's security and separation-of-duties requirements. You can deploy ADDF into a single AWS account, combining the toolchain and target account functions. Alternatively, you can create individual target accounts for ADDF modules or module groups.

The only restriction that you need to consider is that an ADDF module represents the smallest unit of deployment for each AWS account.

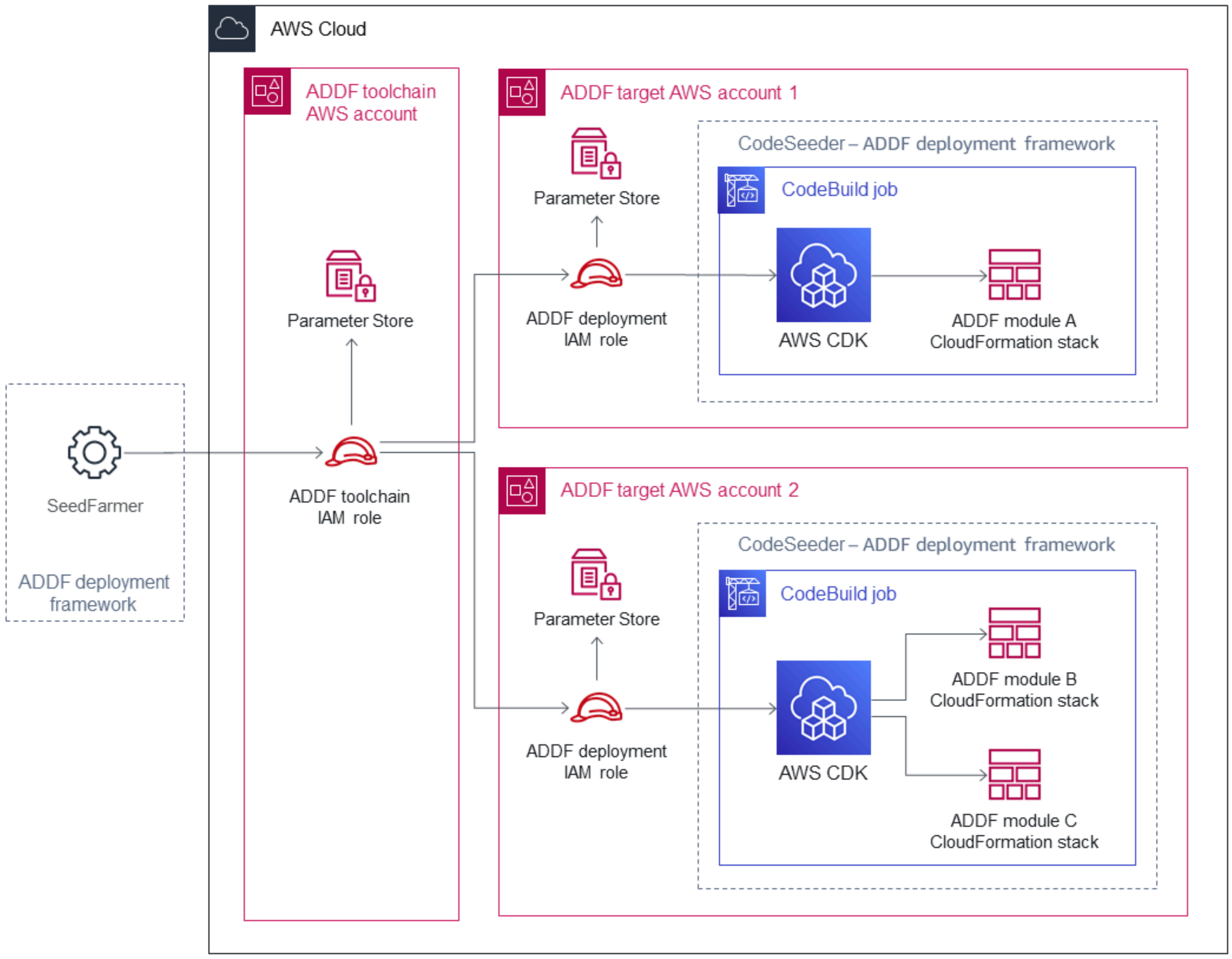
For production environments, it is recommended that you use a multi-account architecture consisting of a toolchain account and at least one target account. For more information, see ADDF architecture.

Least-privilege permissions for multi-account deployments

If you use a multi-account architecture, SeedFarmer needs to access the target accounts to perform the following three actions:

1. Write the ADDF module metadata to the toolchain account and target accounts.
2. Read the current ADDF module metadata from the toolchain account and the target accounts.
3. Initiate AWS CodeBuild jobs in the target accounts, for the purpose of deploying or updating modules.

The following figure shows the cross-account relationships, including operations for assuming ADDF-specific AWS Identity and Access Management (IAM) roles.


AWS Cloud
ADDF toolchain AWS account
Parameter Store
ADDF toolchain IAM role
SeedFarmer
ADDF deployment framework
ADDF target AWS account 1
Parameter Store
ADDF deployment IAM role
CodeSeeder – ADDF deployment framework
CodeBuild job
AWS CDK
ADDF module A CloudFormation stack
ADDF target AWS account 2
Parameter Store
ADDF deployment IAM role
CodeSeeder – ADDF deployment framework
CodeBuild job
AWS CDK
ADDF module B CloudFormation stack
ADDF module C CloudFormation stack

These cross-account actions are achieved by using well-defined assume-role operations.

- The ADDF toolchain IAM role is deployed in the toolchain account. SeedFarmer assumes this role. This role has permissions to perform an `iam:AssumeRole` action and can assume the ADDF deployment IAM role in each target account. In addition, the ADDF toolchain IAM role can run local AWS Systems Manager Parameter Store operations.
- The ADDF deployment IAM role is deployed in each target account. This role can be assumed only from the toolchain account by using the ADDF toolchain IAM role. This role has permissions to run local AWS Systems Manager Parameter Store operations and has permissions to run AWS CodeBuild actions that initiate and describe CodeBuild jobs through CodeSeeder.

These ADDF-specific IAM roles are created as part of the ADDF-bootstrapping process. For more information, see *Bootstrap AWS account(s)* in the ADDF Deployment Guide (GitHub).

All cross-account permissions are set up according to the principle of least privilege. If one target account is compromised, there is minimal or no impact to the other ADDF AWS accounts.

In the case of a single-account architecture for ADDF, the role relationships remain the same. They just collapse into a single AWS account.

ADDF secure setup and operation

Autonomous Driving Data Framework (ADDF) should be treated as a custom piece of software that requires ongoing maintenance and care by a dedicated DevOps and security team in your organization. This section describes security-related common tasks that help you set up and operate ADDF throughout its lifecycle.

This section includes the following tasks:

- Defining your ADDF architecture
- Initial setup
- Customizing the ADDF deployment framework code
- Writing custom modules in ADDF
- Reoccurring ADDF deployments
- Reoccurring security audits
- ADDF updates
- Decommissioning

Defining your ADDF architecture

An ADDF instance is only as secure as the AWS account environment it's deployed in. This AWS account environment must be designed to meet the security and operational needs of your specific use case. For example, the security and operations-related tasks and considerations for setting up an ADDF instance in a proof-of-concept (PoC) environment are different than those for setting up ADDF in a production environment.

Running ADDF in a PoC environment

If you intend to use ADDF in a PoC environment, we recommend that you create a dedicated AWS account for ADDF that doesn't contain any other workloads. This helps keep your account secure while you explore ADDF and its features. The following are the benefits of this approach:

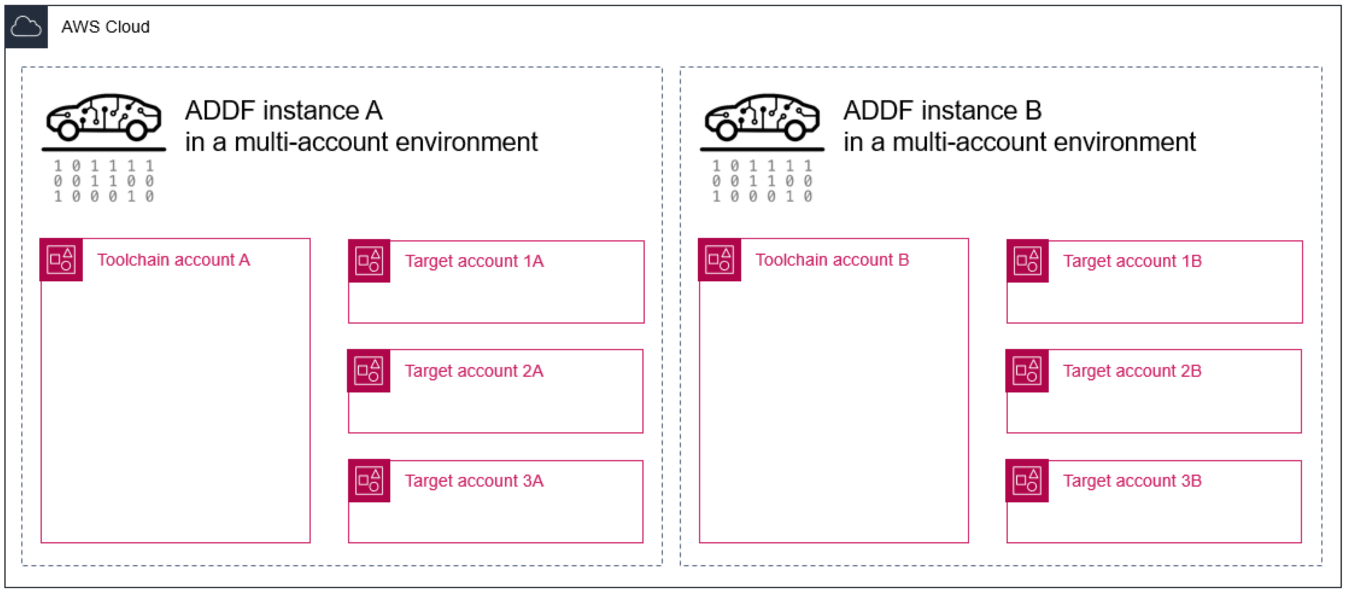
- In case of a severe ADDF misconfiguration, no other workloads would be adversely affected.
- There is no risk of any other workload misconfiguration that could adversely affect the setup of ADDF.

Even for a PoC environment, we still recommend that you follow as many of the best practices listed in Running ADDF in a production environment as possible.

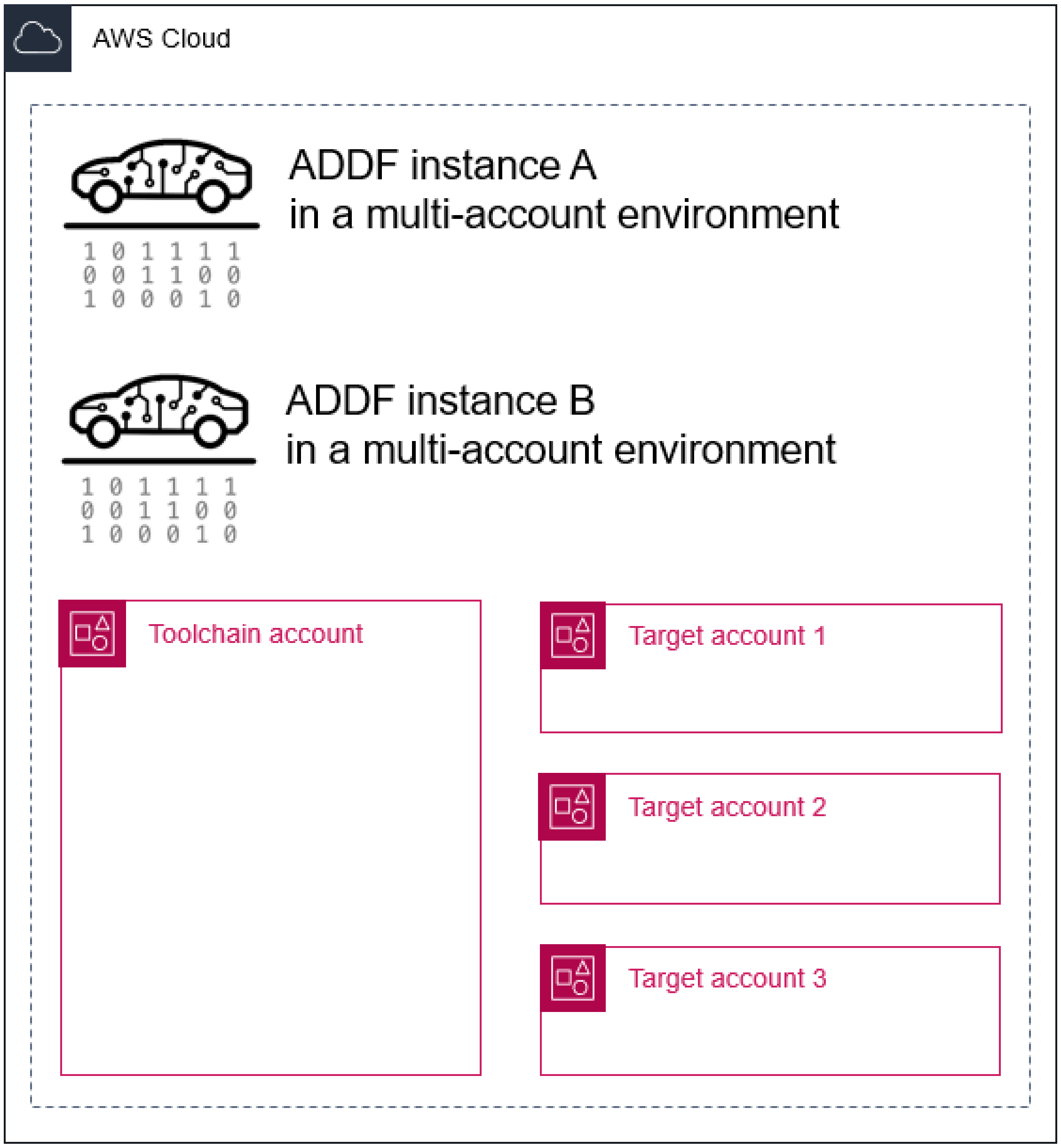
Running ADDF in a production environment

If you intend to use ADDF in an enterprise production environment, we highly recommend that you consider your organization's security best practices and implement ADDF accordingly. In addition to your organization's security best practices, we recommend that you implement the following:

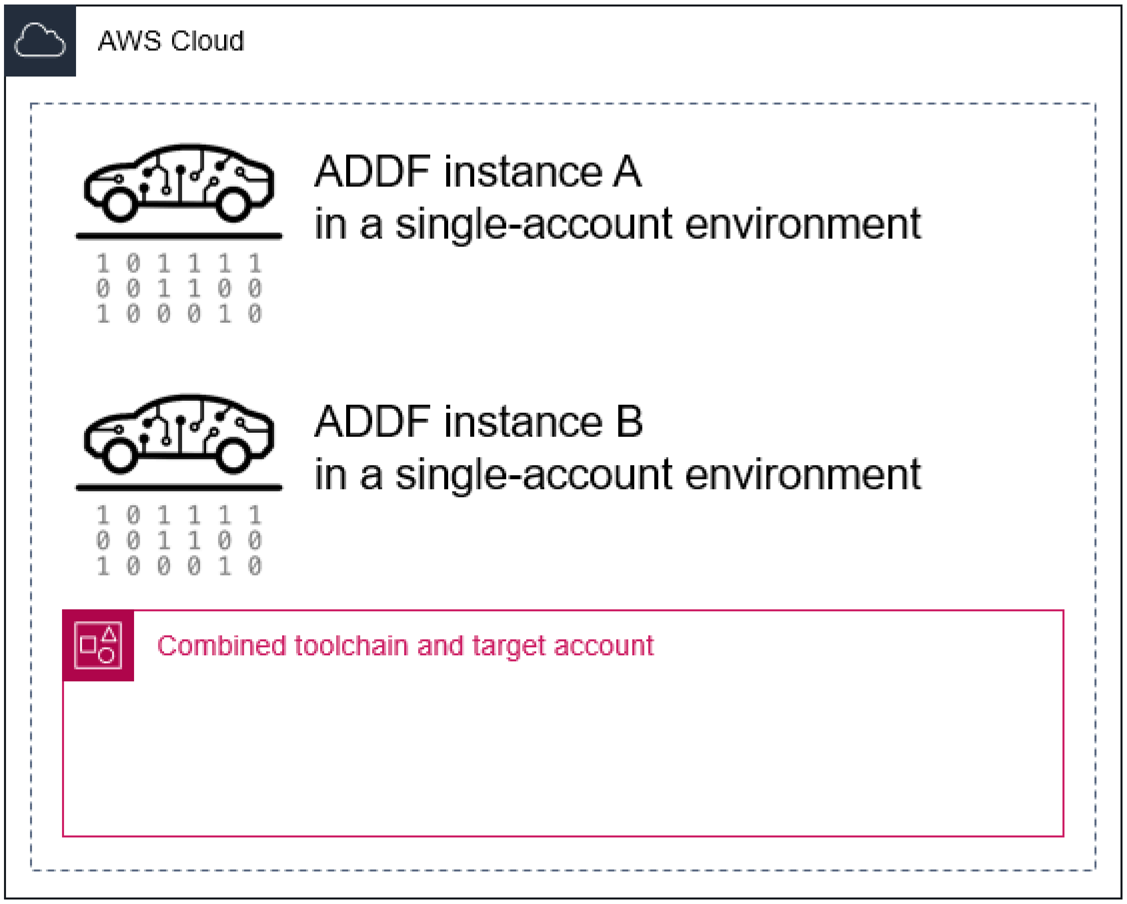
- **Create a long-term, committed ADDF DevOps team** – ADDF needs to be treated as a custom piece of software. It requires ongoing maintenance and care by a dedicated DevOps team. Before starting to run ADDF in a production environment, a DevOps team of sufficient size and capabilities should be defined with a full resource commitment, until the end-of-life of the ADDF deployment.
- **Use a multi-account architecture** – Each ADDF instance should be deployed in its own dedicated AWS multi-account environment, without any other unrelated workloads. As defined in the AWS account management and separation (AWS Well-Architected Framework), it is considered best practice to separate resources and workloads into multiple AWS accounts, based on your organization's requirements. This is because an AWS account acts as an isolation boundary. A properly designed AWS multi-account architecture provides workload categorization and reduces the scope of impact in the event of a security breach, as compared to a single-account architecture. Using a multi-account architecture also helps your accounts remain within their AWS service quotas. Distribute your ADDF modules across as many AWS accounts as needed to meet your organization's security and separation-of-duties requirements.
- **Deploy multiple ADDF instances** – Set up as many separate ADDF instances as you need in order to properly develop, test, and deploy ADDF modules according to your organization's software development processes. When setting up multiple ADDF instances, you can use one of the following approaches:
  - **Multiple ADDF instances in different AWS multi-account environments** – You can use separate AWS accounts to isolate different ADDF instances. For example, if your organization has dedicated development, testing, and production stages, you can create separate ADDF instances and dedicated accounts for each stage. This provides many benefits, such as reducing the risk of any error propagating across stages, helping you implement an approval process, and restricting user access to only certain environments. The following image shows two ADDF instances deployed in separate, multi-account environments.



- **Multiple ADDF instances in the same AWS multi-account environment –** You can create multiple ADDF instances that share the same AWS multi-account environment. This effectively creates isolated branches in the same AWS accounts. For example, if different developers are working in parallel, a developer can create a dedicated ADDF instance in the same AWS accounts. This helps developers work in isolated branches for development and testing purposes. If you use this approach, for each ADDF instance, your ADDF resources must have unique resource names. This is supported in ADDF pre-supplied modules by default. You can use this approach as long as you do not exceed the AWS service quotas. The following image shows two ADDF instances deployed in a shared, multi-account environment.



- **Multiple ADDF instances in the same AWS single-account environment –** This architecture is very similar to the previous example. The difference is that the multiple ADDF instances are deployed in a single-account environment instead of a multi-account environment. This architecture can fit very simple ADDF use cases that have a very limited scope and multiple developers working on different branches at the same time.


AWS Cloud
ADDF instance A in a single-account environment
ADDF instance B in a single-account environment
Combined toolchain and target account

Because SeedFarmer is the single tool that controls deployments for an ADDF instance, you can build any environment and account architecture that fits your organization's deployment strategy and CI/CD processes.

- **Customize the AWS Cloud Development Kit (AWS CDK) bootstrap process according to your organization's security requirements** – By default, AWS CDK assigns the AdministratorAccess AWS managed policy during the bootstrapping process. This policy grants full administrative privileges. If this policy is too permissive for your organization's security requirements, you can customize which policies are applied. For more information, see Custom least-privilege policy for the AWS CDK deployment role.
- **Adhere to best practices when setting up access in IAM** – Establish a structured AWS Identity and Access Management (IAM) access solution that allows your users to access the ADDF AWS accounts. The framework of ADDF is designed to adhere to the principle of least privilege. Your IAM access pattern should also follow the principle of least privilege, should be compliant with your organization's requirements and should adhere to the Security best practices in IAM (IAM documentation).

- **Set up networking according to your organization's best practices** – ADDF includes an optional networking AWS CloudFormation stack that creates a basic public or private virtual private cloud (VPC). Depending on your organization's configuration, this VPC might expose resources directly to the internet. We recommend that you follow your organization's networking best practices and create a custom security-hardened network module.
- **Deploy security prevention, detection, and mitigation measures at the AWS account level** – AWS offers various security services, such as Amazon GuardDuty, AWS Security Hub, Amazon Detective, and AWS Config. Enable those services in your ADDF AWS account and integrate your organization's security prevention, detection, mitigation, and incident-handling processes. We recommend that you follow Best Practices for Security, Identity, & Compliance (AWS Architecture Center) and any service-specific recommendations contained in the documentation for that service. For more information, see AWS Security Documentation.

ADDF doesn't address any of these topics because the implementation and configuration details heavily depend on the requirements and processes that are specific to your organization. Instead, it's the core responsibility of your organization to address these topics. Commonly, the team that manages your AWS landing zone helps you plan and implement your ADDF environment.

Initial setup

Set up ADDF according to the ADDF Deployment Guide (GitHub). The starting point for any deployment is the `/manifest` folder in the autonomous-driving-data-framework Git Hub repository. The `/manifest/example-dev` folder contains a sample deployment for demo purposes. Use this sample as a starting point for designing your own deployment. In that directory, there is an ADDF deployment manifest file called **deployment.yaml**. It contains all the information for SeedFarmer to manage, deploy, or delete ADDF and its resources in the AWS Cloud. You can create groups of ADDF modules in dedicated files. The **core-modules.yaml** is an example of the core module group, and it includes all core modules provided by ADDF. To summarize, the **deployment.yaml** file contains all references to the groups and modules that will be deployed to their target accounts and specifies the deployment order.

For a secure and compliant configuration, especially in an environment that isn't for proof of concept, we recommend that you review the source code of each module that you intend to deploy. According to security hardening best practices, you should deploy only modules that are required for your intended use case.

Note

ADDF modules in the `modules/demo-only/` folder aren't security hardened and shouldn't be deployed in production environments or in any environment with sensitive or protected data. These modules are included to showcase system capabilities, and you can use them as the base for creating your own customized, security-hardened modules.

Customizing the ADDF deployment framework code

The ADDF deployment framework and its orchestration and deployment logic can be fully customized to meet any requirements. However, we suggest you either refrain from customizing or minimize your changes for the following reasons:

- **Keep upstream compatibility** – Upstream compatibility makes it easier to update ADDF for the latest features and security updates. Changing the framework breaks native backwards compatibility with SeedFarmer, CodeSeeder, and any ADDF core modules.
- **Security consequences** – Changing the ADDF deployment framework can be a complex task that can have unintended security consequences. In the worst-case scenario, framework changes can create security vulnerabilities.

When possible, build and customize your own module code instead of modifying the ADDF deployment framework and ADDF core module code.

Note

If you feel that specific parts of the ADDF deployment framework need improvement or further security hardening, please contribute your changes to the ADDF repository through a pull request. For more information, see Open-source security reviews and contributions.

Writing custom modules in ADDF

Creating a new ADDF module or extending an existing module is a core concept of ADDF. When creating or customizing modules, we suggest that you follow general AWS security best-practices and your organization's best practices for secure coding. Furthermore, we recommend that you conduct initial and periodic internal or external technical security reviews, based on your organization's security requirements, to further reduce the risk of security issues.

Reoccurring ADDF deployments

Deploy ADDF and its modules as described in the ADDF Deployment Guide (GitHub). To support reoccurring ADDF deployments that add, update, or remove resources in your target accounts, SeedFarmer uses MD5 hashes, stored in the Parameter Store of your toolchain and target acccounts, to compare the currently deployed infrastructure against the infrastructure defined in the manifest files in your local code base.

This approach follows the GitOps paradigm, where your source repository (the local code base where you operate SeedFarmer) is the source of truth, and the infrastructure declared explicitly in it is the desired outcome of your deployment. For more information about GitOps, see What is GitOps (GitLab website).

Reoccurring security audits

Just like any other software in your organization, integrate ADDF and your custom ADDF module code into your security risk management, security review, and security audit cycle.

ADDF updates

ADDF receives regular updates as part of its ongoing development effort. This includes feature updates, and security-related improvements and fixes. We recommend that you regularly check for new framework releases and apply updates in a timely manner. For more information, see Steps to update ADDF (ADDF documentation).

Decommissioning

If ADDF is no longer needed, delete ADDF and all its related resources from your AWS accounts. Any unattended and unused infrastructure incurs unnecessary costs and poses a potential security risk. For more information, see Steps to destroy ADDF (ADDF documentation).

Next steps

This guide reviewed the security and operations best practices and considerations when deploying the Autonomous Driving Data Framework (ADDF) in your AWS Cloud environment. This guide reviews the shared responsibility model between the ADDF user, the ADDF core team, and AWS so that you understand your role and responsibilities for setting up and operating ADDF securely. It also includes recommendations for operating ADDF securely through its lifecycle, including environment-specific recommendations.

We recommend that you familiarize yourself with the resources in the Resources section. When you are ready, you can set up ADDF according to the instructions in the ADDF Deployment Guide (GitHub).

As you set up and operate ADDF, if you think that deployment framework needs improvement or further security hardening, please contribute your changes to the ADDF repository through a pull request. For more information, see Open-source security reviews and contributions.

Resources

AWS documentation

- Develop and deploy a customized workflow using ADDF on AWS (AWS blog post)
- AWS security service documentation
- Security best practices in IAM
- AWS account management and separation
- Bootstrapping for AWS CDK
- AWS shared responsibility model
- AWS Well-Architected Framework

Open-source resources

- ADDF repository (GitHub)
- ADDF Deployment Guide (GitHub)
- CodeSeeder repository (GitHub)
- SeedFarmer repository (GitHub)

Notices

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
Initial publication	—	November 15, 2022

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Numbers

7 Rs

Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following:

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. You migrate servers from an on-premises platform to a cloud service for the same platform. Example: Migrate a Microsoft Hyper-V application to AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there’s no business justification for migrating them.

- Retire – Decommission or remove applications that are no longer needed in your source environment.

A

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

A database migration method in which the source and target databases are kept in sync (by using a bidirectional replication tool or dual write operations), and both databases handle transactions from connecting applications during migration. This method supports migration in small, controlled batches instead of requiring a one-time cutover. It's more flexible but requires more work than active-passive migration.

active-passive migration

A database migration method in which in which the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include `SUM` and `MAX`.

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.

artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see What is Artificial Intelligence?

artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.

AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

B

bad bot

A bot that is intended to disrupt or cause harm to individuals or organizations.

BCP

See business continuity planning.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

big-endian system

A system that stores the most significant byte first. See also endianness.

binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"

bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

blue/green deployment

A deployment strategy where you create two separate but identical environments. You run the current application version in one environment (blue) and the new application version in the other environment (green). This strategy helps you quickly roll back with minimal impact.

bot

A software application that runs automated tasks over the internet and simulates human activity or interaction. Some bots are useful or beneficial, such as web crawlers that index information on the internet. Some other bots, known as *bad bots*, are intended to disrupt or cause harm to individuals or organizations.

botnet

Networks of bots that are infected by malware and are under the control of a single party, known as a *bot herder* or *bot operator*. Botnets are the best-known mechanism to scale bots and their impact.

branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see About branches (GitHub documentation).

break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the Implement break-glass procedures indicator in the AWS Well-Architected guidance.

brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

buffer cache

The memory area where the most frequently accessed data is stored.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

The slow and incremental release of a version to end users. When you are confident, you deploy the new version and replace the current version in its entirety.

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use AWS Fault Injection Service (AWS FIS) to perform experiments that stress your AWS workloads and evaluate their response.

CI/CD

See continuous integration and continuous delivery.

classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.

cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to edge computing technology.

cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see Building your Cloud Operating Model.

cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or AWS CodeCommit. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

A buffer cache that is empty, not well populated, or contains stale or irrelevant data. This affects performance because the database instance must read from the main memory or disk, which is slower than reading from the buffer cache.

cold data

Data that is rarely accessed and is typically historical. When querying this kind of data, slow queries are typically acceptable. Moving this data to lower-performing and less expensive storage tiers or classes can reduce costs.

computer vision (CV)

A field of AI that uses machine learning to analyze and extract information from visual formats such as digital images and videos. For example, AWS Panorama offers devices that add CV to on-premises camera networks, and Amazon SageMaker provides image processing algorithms for CV.

configuration drift

For a workload, a configuration change from the expected state. It might cause the workload to become noncompliant, and it's typically gradual and unintentional.

configuration management database (CMDB)

A repository that stores and manages information about a database and its IT environment, including both hardware and software components and their configurations. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

The process of automating the source, build, test, staging, and production stages of the software release process. CI/CD is commonly described as a pipeline. CI/CD can help you automate processes, improve productivity, improve code quality, and deliver faster. For more information, see Benefits of continuous delivery. CD can also stand for *continuous deployment*. For more information, see Continuous Delivery vs. Continuous Deployment.

CV

See computer vision.

D

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data drift

A meaningful variation between the production data and the data that was used to train an ML model, or a meaningful change in the input data over time. Data drift can reduce the overall quality, accuracy, and fairness in ML model predictions.

data in transit

Data that is actively moving through your network, such as between network resources.

data mesh

An architectural framework that provides distributed, decentralized data ownership with centralized management and governance.

data minimization

The principle of collecting and processing only the data that is strictly necessary. Practicing data minimization in the AWS Cloud can reduce privacy risks, costs, and your analytics carbon footprint.

data perimeter

A set of preventive guardrails in your AWS environment that help make sure that only trusted identities are accessing trusted resources from expected networks. For more information, see Building a data perimeter on AWS.

data preprocessing

To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.

data provenance

The process of tracking the origin and history of data throughout its lifecycle, such as how the data was generated, transmitted, and stored.

data subject

An individual whose data is being collected and processed.

data warehouse

A data management system that supports business intelligence, such as analytics. Data warehouses commonly contain large amounts of historical data, and they are typically used for queries and analysis.

database definition language (DDL)

Statements or commands for creating or modifying the structure of tables and objects in a database.

database manipulation language (DML)

Statements or commands for modifying (inserting, updating, and deleting) information in a database.

DDL

See database definition language.

deep ensemble

To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.

deep learning

An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources. For example, a defense-in-depth approach might combine multi-factor authentication, network segmentation, and encryption.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

deployment

The process of making an application, new features, or code fixes available in the target environment. Deployment involves implementing changes in a code base and then building and running that code base in the application's environments.

development environment

See environment.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

development value stream mapping (DVSM)

A process used to identify and prioritize constraints that adversely affect speed and quality in a software development lifecycle. DVSM extends the value stream mapping process originally designed for lean manufacturing practices. It focuses on the steps and teams required to create and move value through the software development process.

digital twin

A virtual representation of a real-world system, such as a building, factory, industrial equipment, or production line. Digital twins support predictive maintenance, remote monitoring, and production optimization.

dimension table

In a star schema, a smaller table that contains data attributes about quantitative data in a fact table. Dimension table attributes are typically text fields or discrete numbers that behave like text. These attributes are commonly used for query constraining, filtering, and result set labeling.

disaster

An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location. These events can be natural disasters, technical failures, or the result of human actions, such as unintentional misconfiguration or a malware attack.

disaster recovery (DR)

The strategy and process you use to minimize downtime and data loss caused by a disaster. For more information, see Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

See database manipulation language.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

See disaster recovery.

drift detection

Tracking deviations from a baselined configuration. For example, you can use AWS CloudFormation to detect drift in system resources, or you can use AWS Control Tower to detect changes in your landing zone that might affect compliance with governance requirements.

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

edge computing

The technology that increases the computing power for smart devices at the edges of an IoT network. When compared with cloud computing, edge computing can reduce communication latency and improve response time.

encryption

A computing process that transforms plaintext data, which is human-readable, into ciphertext.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endianness

The order in which bytes are stored in computer memory. Big-endian systems store the most significant byte first. Little-endian systems store the least significant byte first.

endpoint

See service endpoint.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

enterprise resource planning (ERP)

A system that automates and manages key business processes (such as accounting, MES, and project management) for an enterprise.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

environment

An instance of a running application. The following are common types of environments in cloud computing:

- development environment – An instance of a running application that is available only to the core team responsible for maintaining the application. Development environments are used to test changes before promoting them to upper environments. This type of environment is sometimes referred to as a *test environment*.
- lower environments – All development environments for an application, such as those used for initial builds and tests.
- production environment – An instance of a running application that end users can access. In a CI/CD pipeline, the production environment is the last deployment environment.
- upper environments – All environments that can be accessed by users other than the core development team. This can include a production environment, preproduction environments, and environments for user acceptance testing.

epic

In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.

F

fact table

The central table in a star schema. It stores quantitative data about business operations. Typically, a fact table contains two types of columns: those that contain measures and those that contain a foreign key to a dimension table.

fail fast

A philosophy that uses frequent and incremental testing to reduce the development lifecycle. It is a critical part of an agile approach.

fault isolation boundary

In the AWS Cloud, a boundary such as an Availability Zone, AWS Region, control plane, or data plane that limits the effect of a failure and helps improve the resilience of workloads. For more information, see AWS Fault Isolation Boundaries.

feature branch

See branch.

features

The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.

feature importance

How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations

(SHAP) and integrated gradients. For more information, see Machine learning model interpretability with :AWS.

feature transformation

To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the “2021-05-27 00:15:37” date into “2021”, “May”, “Thu”, and “15”, you can help the learning algorithm learn nuanced patterns associated with different data components.

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

The use of multiple conditions to allow or deny an access request.

flash-cut migration

A database migration method that uses continuous data replication through change data capture to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

G

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

Gitflow workflow

An approach in which lower and upper environments use different branches in a source code repository. The Gitflow workflow is considered legacy, and the trunk-based workflow is the modern, preferred approach.

greenfield strategy

The absence of existing infrastructure in a new environment. When adopting a greenfield strategy for a system architecture, you can select all new technologies without the restriction of compatibility with existing infrastructure, also known as brownfield. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries. *Detective guardrails* detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

H

HA

See high availability.

heterogeneous database migration

Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A *historian* is a type of database that is used to collect and store data from various sources in a factory.

homogeneous database migration

Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.

hot data

Data that is frequently accessed, such as real-time data or recent translational data. This data typically requires a high-performance storage tier or class to provide fast query responses.

hotfix

An urgent fix for a critical issue in a production environment. Due to its urgency, a hotfix is usually made outside of the typical DevOps release workflow.

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

I

IaC

See infrastructure as code.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

idle application

An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.

IIoT

See industrial Internet of Things.

immutable infrastructure

A model that deploys new infrastructure for production workloads instead of updating, patching, or modifying the existing infrastructure. Immutable infrastructures are inherently more consistent, reliable, and predictable than mutable infrastructure. For more information, see the Deploy using immutable infrastructure best practice in the AWS Well-Architected Framework.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

Industry 4.0

A term that was introduced by Klaus Schwab in 2016 to refer to the modernization of manufacturing processes through advances in connectivity, real-time data, automation, analytics, and AI/ML.

infrastructure

All of the resources and assets contained within an application's environment.

infrastructure as code (IaC)

The process of provisioning and managing an application's infrastructure through a set of configuration files. IaC is designed to help you centralize infrastructure management, standardize resources, and scale quickly so that new environments are repeatable, reliable, and consistent.

industrial Internet of Things (IIoT)

The use of internet-connected sensors and devices in the industrial sectors, such as manufacturing, energy, automotive, healthcare, life sciences, and agriculture. For more information, see Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

Internet of Things (IoT)

The network of connected physical objects with embedded sensors or processors that communicate with other devices and systems through the internet or over a local communication network. For more information, see What is IoT?

interpretability

A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

IoT

See Internet of Things.

IT information library (ITIL)

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)

Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

An implementation of mandatory access control (MAC) where the users and the data itself are each explicitly assigned a security label value. The intersection between the user security label and data security label determines which rows and columns can be seen by the user.

landing zone

A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

lift and shift

See 7 Rs.

little-endian system

A system that stores the least significant byte first. See also endianness.

lower environments

See environment.

M

machine learning (ML)

A type of artificial intelligence that uses algorithms and techniques for pattern recognition and learning. ML analyzes and learns from recorded data, such as Internet of Things (IoT) data, to generate a statistical model based on patterns. For more information, see Machine Learning.

main branch

See branch.

malware

Software that is designed to compromise computer security or privacy. Malware might disrupt computer systems, leak sensitive information, or gain unauthorized access. Examples of malware include viruses, worms, ransomware, Trojan horses, spyware, and keyloggers.

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

manufacturing execution system (MES)

A software system for tracking, monitoring, documenting, and controlling production processes that convert raw materials to finished products on the shop floor.

MAP

See Migration Acceleration Program.

mechanism

A complete process in which you create a tool, drive adoption of the tool, and then inspect the results in order to make adjustments. A mechanism is a cycle that reinforces and improves itself as it operates. For more information, see Building mechanisms in the AWS Well-Architected Framework.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

A lightweight, machine-to-machine (M2M) communication protocol, based on the publish/subscribe pattern, for resource-constrained IoT devices.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed, and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

Migration Acceleration Program (MAP)

An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.

migration at scale

The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.

migration pattern

A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.

Migration Readiness Assessment (MRA)

The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.

migration strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

ML

See machine learning.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can use a microservices architecture. For more information, see Decomposing monoliths into microservices.

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"

mutable infrastructure

A model that updates and modifies the existing infrastructure for production workloads. For improved consistency, reliability, and predictability, the AWS Well-Architected Framework recommends the use of immutable infrastructure as a best practice.

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

A migration method in which the source workload is taken down during the migration process. This method involves extended downtime and is typically used for small, non-critical workloads.

OI

See operations integration.

OLA

See operational-level agreement.

online migration

A migration method in which the source workload is copied to the target system without being taken offline. Applications that are connected to the workload can continue to function during the migration. This method involves zero to minimal downtime and is typically used for critical production workloads.

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

A machine-to-machine (M2M) communication protocol for industrial automation. OPC-UA provides an interoperability standard with data encryption, authentication, and authorization schemes.

operational-level agreement (OLA)

An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).

operational readiness review (ORR)

A checklist of questions and associated best practices that help you understand, evaluate, prevent, or reduce the scope of incidents and possible failures. For more information, see Operational Readiness Reviews (ORR) in the AWS Well-Architected Framework.

operational technology (OT)

Hardware and software systems that work with the physical environment to control industrial operations, equipment, and infrastructure. In manufacturing, the integration of OT and information technology (IT) systems is a key focus for Industry 4.0 transformations.

operations integration (OI)

The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the operations integration guide.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

organizational change management (OCM)

A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called *people acceleration*, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic `PUT` and `DELETE` requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC, which provides more granular and enhanced access control.

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

P

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

personally identifiable information (PII)

Information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual. Examples of PII include names, addresses, and contact information.

PII

See personally identifiable information.

playbook

A set of predefined steps that capture the work associated with migrations, such as delivering core operations functions in the cloud. A playbook can take the form of scripts, automated runbooks, or a summary of processes or steps required to operate your modernized environment.

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

An object that can define permissions (see identity-based policy), specify access conditions (see resource-based policy), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy).

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store best adapted to their requirements. For more information, see Enabling data persistence in microservices.

portfolio assessment

A process of discovering, analyzing, and prioritizing the application portfolio in order to plan the migration. For more information, see Evaluating migration readiness.

predicate

A query condition that returns `true` or `false`, commonly located in a `WHERE` clause.

predicate pushdown

A database query optimization technique that filters the data in the query before transfer. This reduces the amount of data that must be retrieved and processed from the relational database, and it improves query performance.

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

Privacy by Design

An approach in system engineering that takes privacy into account throughout the whole engineering process.

private hosted zones

A container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs. For more information, see Working with private hosted zones in the Route 53 documentation.

proactive control

A security control designed to prevent the deployment of noncompliant resources. These controls scan resources before they are provisioned. If the resource is not compliant with the control, then it isn't provisioned. For more information, see the Controls reference guide in the AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

product lifecycle management (PLM)

The management of data and processes for a product throughout its entire lifecycle, from design, development, and launch, through growth and maturity, to decline and removal.

production environment

See environment.

programmable logic controller (PLC)

In manufacturing, a highly reliable, adaptable computer that monitors machines and automates manufacturing processes.

pseudonymization

The process of replacing personal identifiers in a dataset with placeholder values. Pseudonymization can help protect personal privacy. Pseudonymized data is still considered to be personal data.

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based MES, a microservice can publish event messages to a channel that other microservices can subscribe to. The system can add new microservices without changing the publishing service.

Q

query plan

A series of steps, like instructions, that are used to access the data in a SQL relational database system.

query plan regression

When a database service optimizer chooses a less optimal plan than it did before a given change to the database environment. This can be caused by changes to statistics, constraints, environment settings, query parameter bindings, and updates to the database engine.

R

RACI matrix

See responsible, accountable, consulted, informed (RACI).

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

RASCI matrix

See responsible, accountable, consulted, informed (RACI).

RCAC

See row and column access control.

read replica

A copy of a database that's used for read-only purposes. You can route queries to the read replica to reduce the load on your primary database.

re-architect

See 7 Rs.

recovery point objective (RPO)

The maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

recovery time objective (RTO)

The maximum acceptable delay between the interruption of service and restoration of service.

refactor

See 7 Rs.

Region

A collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the others to provide fault tolerance, stability, and resilience. For more information, see Specify which AWS Regions your account can use.

regression

An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

rehost

See 7 Rs.

release

In a deployment process, the act of promoting changes to a production environment.

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

An application's ability to resist or recover from disruptions. High availability and disaster recovery are common considerations when planning for resiliency in the AWS Cloud. For more information, see AWS Cloud Resilience.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

rotation

The process of periodically updating a secret to make it more difficult for an attacker to access the credentials.

row and column access control (RCAC)

The use of basic, flexible SQL expressions that have defined access rules. RCAC consists of row permissions and column masks.

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

A set of manual or automated procedures required to perform a specific task. These are typically built to streamline repetitive operations or procedures with high error rates.

S

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata. The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are four primary types of security controls: preventative, detective, responsive, and proactive.

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers, networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

security response automation

A predefined and programmed action that is designed to automatically respond to or remediate a security event. These automations serve as detective or responsive security controls that help you implement AWS security best practices. Examples of automated response actions include modifying a VPC security group, patching an Amazon EC2 instance, or rotating credentials.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

The URL of the entry point for an AWS service. You can use the endpoint to connect programmatically to the target service. For more information, see AWS service endpoints in *AWS General Reference*.

service-level agreement (SLA)

An agreement that clarifies what an IT team promises to deliver to their customers, such as service uptime and performance.

service-level indicator (SLI)

A measurement of a performance aspect of a service, such as its error rate, availability, or throughput.

service-level objective (SLO)

A target metric that represents the health of a service, as measured by a service-level indicator.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

SIEM

See security information and event management system.

single point of failure (SPOF)

A failure in a single, critical component of an application that can disrupt the system.

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization’s capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

A database organizational structure that uses one large fact table to store transactional or measured data and uses one or more smaller dimensional tables to store data attributes. This structure is designed for use in a data warehouse or for business intelligence purposes.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

subnet

A range of IP addresses in your VPC. A subnet must reside in a single Availability Zone.

supervisory control and data acquisition (SCADA)

In manufacturing, a system that uses hardware and software to monitor physical assets and production operations.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

synthetic testing

Testing a system in a way that simulates user interactions to detect potential issues or to monitor performance. You can use Amazon CloudWatch Synthetics to create these tests.

T

tags

Key-value pairs that act as metadata for organizing your AWS resources. Tags can help you manage, identify, organize, search for, and filter resources. For more information, see Tagging your AWS resources.

target variable

The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.

task list

A tool that is used to track progress through a runbook. A task list contains an overview of the runbook and a list of general tasks to be completed. For each general task, it includes the estimated amount of time required, the owner, and the progress.

test environment

See environment.

training

To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don’t know the target.

transit gateway

A network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see What is a transit gateway in the AWS Transit Gateway documentation.

trunk-based workflow

An approach in which developers build and test features locally in a feature branch and then merge those changes into the main branch. The main branch is then built to the development, preproduction, and production environments, sequentially.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

tuning

To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development.

U

uncertainty

A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data. For more information, see the Quantifying uncertainty in deep learning systems guide.

undifferentiated tasks

Also known as *heavy lifting*, work that is necessary to create and operate an application but that doesn’t provide direct value to the end user or provide competitive advantage. Examples of undifferentiated tasks include procurement, maintenance, and capacity planning.

upper environments

See environment.

V

vacuuming

A database maintenance operation that involves cleaning up after incremental updates to reclaim storage and improve performance.

version control

Processes and tools that track changes, such as changes to source code in a repository.

VPC peering

A connection between two VPCs that allows you to route traffic by using private IP addresses. For more information, see What is VPC peering in the Amazon VPC documentation.

vulnerability

A software or hardware flaw that compromises the security of the system.

W

warm cache

A buffer cache that contains current, relevant data that is frequently accessed. The database instance can read from the buffer cache, which is faster than reading from the main memory or disk.

warm data

Data that is infrequently accessed. When querying this kind of data, moderately slow queries are typically acceptable.

window function

A SQL function that performs a calculation on a group of rows that relate in some way to the current record. Window functions are useful for processing tasks, such as calculating a moving average or accessing the value of rows based on the relative position of the current row.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.

workstream

Functional groups in a migration project that are responsible for a specific set of tasks. Each workstream is independent but supports the other workstreams in the project. For example, the portfolio workstream is responsible for prioritizing applications, wave planning, and collecting migration metadata. The portfolio workstream delivers these assets to the migration workstream, which then migrates the servers and applications.

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

A storage model that writes data a single time and prevents the data from being deleted or modified. Authorized users can read the data as many times as needed, but they cannot change it. This data storage infrastructure is considered immutable.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a zero-day vulnerability.

zero-day vulnerability

An unmitigated flaw or vulnerability in a production system. Threat actors can use this type of vulnerability to attack the system. Developers frequently become aware of the vulnerability as a result of the attack.

zombie application

An application that has an average CPU and memory usage below 5 percent. In a migration project, it is common to retire these applications.