Implementing infrastructure as a product (IaP) on AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Implementing infrastructure as a product (IaP) on AWS

Table of Contents

Implementing IaP on AWS

Kirsten Kissmeyer, Amazon Web Services (AWS)

January 2023 (document history)

This guide explores approaches for managing your AWS infrastructure as a product (IaP). IaP provides a higher level of abstraction and control than infrastructure as code (IaC) but uses IaC methods to achieve its goals. The guide also explores AWS services and tools for managing IaP and highlights how each tool can support your objectives for managing your infrastructure. The information in this guide is based on learnings from an AWS Service Catalog enablement initiative for a very large financial sector company.

This guide is intended for users who want to develop functional AWS Cloud infrastructure services that can easily be allocated and authorized as needed for different organizational users, business units, and third parties.

Why manage infrastructure as products?

The advantage of managing your infrastructure resources as products is that you can package consumer capabilities as a set of resources that have standardized definitions and configurations. Products provide a convenient way for an organization to manage and control how AWS capabilities are allocated and consumed. A product might be restricted to only designated organizational units (OUs) or to individuals who need those functional capabilities. A product can be restricted to specific AWS Regions as well.

A product provisioning model also lets you encapsulate and update the definition of a product from a central location. You can then distribute product updates on a one-time or scheduled basis, as its implementation changes over time.

Targeted business outcomes

Organizations always look for better ways to manage and provision their AWS infrastructure. Your objectives might include:

- Achieving a high degree of agility, reliability, fault tolerance, and centralized control, where single points of configuration satisfy compliance with evolving internal and external standards.

- A low-touch or push-button mechanism to distribute infrastructure in a centralized way, while allowing self-serve access when needed for specific teams or individuals.
- The ability to provision AWS infrastructure and services to internal staff, client accounts, and partner OU accounts. You might also want to control which OUs or organizations have access to specific infrastructure components in specific Regions.
- If you use third-party tools (such as ServiceNow) or custom tools to manage requests to access and provision your enterprise assets and infrastructure, easy integration between your AWS infrastructure and these tools.
- The ability to provision AWS infrastructure to dozens or even hundreds of target accounts at the same time.
- Support for provisioning multiple AWS resources to provide a single capability.
- The ability to create new accounts with required infrastructure within a tight schedule.
- Access to an inventory of the infrastructure that you have provisioned, and the ability to update or remove infrastructure components.
- Approaches and technologies that make the provisioning and maintenance process easier, faster, and more secure and reliable.

Using AWS Service Catalog to manage IaP

AWS provides a service called AWS Service Catalog that supports managing and provisioning AWS infrastructure as a product. You can use Service Catalog to rapidly define the infrastructure you need to provision as a set of products, grant permission for those products to desired parties, and implement the provisioning and update patterns required for individual products.

Service Catalog is backed by AWS CloudFormation. Service Catalog portfolios, products, and their provisioning templates are managed as CloudFormation stacks. You can define these stacks in four ways:

- By using standard CloudFormation templates.
- By using the AWS Cloud Development Kit (AWS CDK) and the Service Catalog Construct Library with a supported programming language that you prefer.
- By using a framework provided by a third-party tool to generate the CloudFormation stack definitions from declarative metadata that describes the stacks.
- By using the Service Catalog API. This API provides methods for everything except for building the product. You can add products to portfolios, remove products from portfolios, tag products and portfolios, define administrative and operational product service actions, and browse and search for portfolio and product definitions.

At its core, a Service Catalog product is a set of one or more AWS resources that are configured to provide a collective, customizable (through parameterization) capability. For example, you can define a Service Catalog product to provision a private Amazon Simple Storage Service (Amazon S3) bucket in a target account. The S3 bucket is a product that might have input parameters such as the bucket name, an internet address range to allow access from, a set of users who can access the bucket, a lifecycle tiering policy, or a bucket versioning specification. You can also define an AWS Identity and Access Management (IAM) role to provide access to the bucket as part of the product.

You can add a Service Catalog product to one or more *portfolios*. A Service Catalog portfolio is a collection of products that are grouped together, generally because they serve a similar purpose (for example, analysis, development, client access services, partner access services, and so on).

You provide permissions for a user, group, or role to have access to provision a product at the portfolio level. For provisioning, products are associated with either a launch IAM role (for launching the product in a self-serve manner to anyone who can assume the role), or with a stack set that defines one or more accounts that the product can be provisioned to. To use a stack set, you must define a Service Catalog administrator role in the Service Catalog hub account and a Service Catalog product provisioning execution role in each target account of the stack set.

The following sections discuss Service Catalog IaP functionality in more detail.

Topics

Support for modularity and code reuse

You can assemble a product from many different AWS resources or even from other products. Ideally, you define resources in a modular way so you can reuse them in multiple products. Resource-level reuse enables you to make any future changes in one place rather than across every product that uses that resource type.

Service Catalog provides a feature called *chaining* to support reusability at the product level. You can chain a product to one or more other products. For example, you might want to chain an S3 logging bucket product to a higher-level monitoring product. Although chaining supports modularity, it imposes some operational complexities because you have to manage dependencies. Service Catalog doesn’t automatically maintain versioning between chained products, so it can’t ensure that changes to one product don’t break other products that depend on it. Use chaining with care, and develop your own mechanisms for ensuring versioning and maintaining dependencies.

Service Catalog uses CloudFormation natively to deploy a product provisioning template as a CloudFormation stack. However, Service Catalog imposes some limitations on CloudFormation deployment of the product stack. In particular, Service Catalog provisioning doesn’t support the

CloudFormation include macro for inserting reusable script segments or referencing nested CloudFormation scripts (or stacks) to more than one level. These Service Catalog restrictions limit the ability to define products from reusable CloudFormation templates or components, which is a standard best practice when you define stacks natively in CloudFormation.

Note

Service Catalog allows you to successfully define products with provisioning templates that use these CloudFormation constructs. However, you will encounter provision-time errors if you use the include macro or nest multiple levels of scripts in a Service Catalog CloudFormation template.

These restrictions might make it difficult to implement modular and reusable products in Service Catalog. If modularity is a requirement, you might explore using the AWS CDK to implement your products and their provisioning templates, or use the provisioning workflows and engine in the AWS Labs Service Catalog Tools project. Both alternatives are described later in this guide.

Programming options for defining products in Service Catalog

Two programming options for using Service Catalog to provision AWS infrastructure are CloudFormation templates or the AWS CDK. Currently there are no declarative or no-code mechanisms for defining a Service Catalog product.

CloudFormation scripting

AWS CloudFormation is a tried and true IaC native scripting language for provisioning AWS infrastructure. You can develop a CloudFormation script in the AWS Management Console or by using a development tool such as Visual Studio Code (or a simple text editor) and the AWS Command Line Interface (AWS CLI).

For more information, see the CloudFormation documentation. For more information about using a CloudFormation template to specify a Service Catalog product, see the AWS::ServiceCatalog::CloudFormationProduct resource in the CloudFormation documentation.

Programmatic approach with the AWS CDK

The AWS CDK provides an elegant and powerful object-oriented programming framework for defining and maintaining AWS infrastructure by using a selection of programming languages. You can use the AWS CDK to develop object-oriented, fine-grained customizations and extensions to the AWS class framework. The AWS CDK is for users who want to customize AWS services for more sophisticated infrastructure needs, and who have the requisite programming skills and experience.

To implement Service Catalog solutions by using the AWS CDK, you use the built-in Service Catalog classes to define your products and portfolios. These classes are provided by the AWS CDK aws-cdk-lib.aws_servicecatalog module.

You can use the AWS CDK to implement products in many ways. To avoid having to write the provisioning template for a product in CloudFormation and to maintain reusability, we recommend that you use the AWS CDK ProductStack class to represent the provisioning template. A `ProductStack` instance is an AWS CDK stack that you programmatically add resources to. For example, you can add an S3 bucket, IAM roles, or an Amazon CloudWatch log. When you add the `ProductStack` instance to a defined `servicecatalog.CloudFormationProduct` instance as its provisioning template by calling `servicecatalog.CloudFormationTemplate.fromProductStack (<ProductStack instance>)`, the AWS CDK automatically generates the CloudFormation template.

Here’s an example of the Java `ProductStack` implementation for an Amazon S3 product.

```
import * as s3 from 'aws-cdk-lib/aws-s3';
import * as cdk from 'aws-cdk-lib';

class S3BucketProduct extends servicecatalog.ProductStack {
  constructor(scope: Construct, id: string) {
    super(scope, id);

    new s3.Bucket(this, 'BucketProduct');
  }
}

const product = new servicecatalog.CloudFormationProduct(this, 'Product', {
  productName: "My Product",
  owner: "Product Owner",
  productVersions: [
    {
      productVersionName: "v1",
      cloudFormationTemplate:
 servicecatalog.CloudFormationTemplate.fromProductStack(new S3BucketProduct(this,
 'S3BucketProduct')),
    },
  ],
});
```

The AWS CDK provides built-in continuous integration and continuous deployment (CI/CD) pipelines. You can customize these built-in pipelines and software development lifecycle (SDLC) processes to meet your own process standards and objectives.

Custom AWS CDK classes can inherit from other classes to provide specialized functions, and a class may be composed from instances of other classes. If you use shared AWS CDK class frameworks to implement multiple Service Catalog products, consider any versioning or compatibility implications, especially across multiple development teams. You will have to ensure that changes are backward compatible, or that you have a versioning scheme that is being followed so that class changes you make for one product's doesn't break another product.

For more information, see the AWS CDK documentation.

Integration with external provisioning processes and workflows

You can interact with Service Catalog components by using AWS SDK APIs or the AWS CLI. You can use the AWS SDK Service Catalog API to manage Service Catalog products from any tool that can integrate Service Catalog API calls. The API covers all aspects of Service Catalog creation and management. For example, Terraform supports the launching (provisioning) of Service Catalog products by calling the AWS SDK Service Catalog API in its Launch Wizard. For more information, see Launch AWS Service Catalog products with Terraform in the AWS documentation.

You can also call the AWS CLI Service Catalog commands to perform actions on Service Catalog. For more information about supported commands, see servicecatalog in the AWS CLI Command Reference.

Product provisioning specifications

Service Catalog initiates the provisioning process as a CloudFormation stack set deployment of the resources that are specified in the CloudFormation provisioning template. (The template can be created directly in AWS CloudFormation or generated by the AWS CDK `ProductStack` construct.) Service Catalog product provisioning is a closed process—you cannot customize it to add preliminary or post-process steps, or tune it. However, you can modify the provisioning template to add steps in the form of CloudFormation resource specifications. These could be AWS Lambda or AWS Step Functions, or Lambda-backed custom resources that perform preliminary steps (such as custom bootstrapping to set up a bastion host that is used during provisioning) and post-steps (such as tearing down the bastion host). This method of implementing pre-provisioning and post-provisioning steps is subject to the same `include` and nested stack restrictions as the provisioning template.

You can specify target accounts as individual accounts, not as organizational units (OUs). You can write a custom resource or function to work around this limitation. Most organizations provision portfolios of products to OUs and not to individual accounts, because they automate the generation of accounts and don't want to maintain account lists manually.

DevSecOps lifecycle support

Currently, products that are provisioned with Service Catalog CloudFormation scripts don't have built-in support for CI/CD processes. We recommend that you create a CI/CD process in AWS CodePipeline or other DevOps tools to develop, test, and release a product through lifecycle environments such as development, test, stage, and production.

The AWS CDK does provide built-in CI/CD support for products, as discussed earlier in this guide.

Customized reuse and account-specific provisioning

Products should be made reusable for as many different customized purposes as possible. Service Catalog supports reusability through product parameters. You can provide these parameters as input to a product at provisioning time.

You can also specify these parameters as AWS Systems Manager Parameter Store values at the CloudFormation template level, to apply account-specific and OU-specific values. This is a best practice for CloudFormation provisioning template design. The value of the named parameter within the target account is applied when the product is provisioned. For example, you can specify a subnet parameter as a Parameter Store value and apply that subnet at product provisioning time for a specific OU account. For more information about Parameter Store values as CloudFormation template parameters, see Using dynamic references to specify template values in the AWS CloudFormation documentation.

Defining and managing Service Catalog product resources as applications

AWS Service Catalog AppRegistry provides centralized application search, reporting, and management capabilities. An AppRegistry application can include one or more provisioned product stacks as well as CloudFormation stacks that are independent of Service Catalog. You can group and view all your application resource collections across the AWS accounts that you define as deployment targets. These accounts could be your development, test, and production lifecycle accounts.

You can also use AppRegistry to associate metadata attributes with an application. You can assign reusable attribute groups that contain sets of attributes. You can then search and act on application resources that have the given attributes by using AppRegistry or integrated services. These integrated services include:

- Application Manager, a capability of AWS Systems Manager, to investigate and remediate issues with AWS resources in the context of your applications and clusters
- AWS Resource Access Manager, to share applications and attribute groups with AWS organization principals
- AWS services that work with AWS Resource Groups
- AWS Resilience Hub for product structure discovery and resilience assessment
- AWS Service Management Connector to declare and set up connections to ServiceNow, JIRA, and other popular tools

For more information about AppRegistry, see the following:

- AppRegistry Administrator Guide.
- Increase application visibility and governance using AWS Service Catalog AppRegistry blog post. This article provides an overview of how to use AppRegistry in infrastructure governance, with command-line examples of registering your infrastructure as applications in AppRegistry.
- Govern your applications centrally using AppRegistry and Application Manager blog post. This article provides an overview with a tutorial of how to apply AppRegistry to register a LAMP web application on the AWS Management Console and manage it by using Application Manager.

Inventory management

Service Catalog has its own internal inventory management capability that registers products when they are provisioned through product sharing and self-service. However, we recommend that you use AWS Config or AppRegistry and related services to manage your product-provisioned resources. These tools provide a more comprehensive and integrated approach to managing your provisioned Service Catalog products with the rest of your AWS infrastructure. AWS Config lets you inventory and perform actions on provisioned products on the console or by using the AWS SDK API. AppRegistry, which is integrated with Application Manager, also provides inventory management for Service Catalog provisioned products.

Using AWS Service Catalog Tools

If you want to provision your IaC products with customized provisioning workflows in a more declarative manner, you might want to augment portions of the Service Catalog functionality. AWS provides several tools to support these requirements. Two popular tools are provided in the AWS Labs project: Service Catalog Puppet and Service Catalog Factory.

Topics

Service Catalog Puppet

Service Catalog Puppet is implemented in Python by using the AWS Boto3 API. This tool offers several powerful features for configuring and provisioning Service Catalog products. Developers can configure Service Catalog product and portfolio provisioning information by using YAML templates that serve as manifests. Service Catalog Puppet provisioning workflows support products that require more complex deployment processes than Service Catalog. They also support performance optimizations to provision products at scale within aggressive time windows.

Service Catalog Puppet accesses the Service Catalog CloudFormation templates for product provisioning at deployment time. It calls CloudFormation directly to deploy the provisioning template stack for a product and bypasses the restrictions imposed by Service Catalog's own stack set provisioning process. If the provisioning template uses macros to include other CloudFormation scripts or uses nested CloudFormation scripts, you must provide access to those scripts in the target account in the bootstrapping portion of the provisioning workflow.

For more information:

- See the Service Catalog Puppet documentation and GitHub repository.
- If you want to use the Service Catalog Puppet SDK to interact with the tool programmatically to initiate product and portfolio provisioning, see the SDK documentation.
- GitOps is the default mechanism for managing the Service Catalog Puppet environment.

Service Catalog Puppet is fairly easy for developers to learn. It requires familiarity with CloudFormation to implement product provisioning templates and YAML templates to implement manifests. There are good workshops available to bring new developers up to speed, such as self-paced workshops.

Support for provisioning workflows

Service Catalog Puppet employs the Python Luigi task orchestration engine to implement bootstrapping and provisioning workflows. All steps in these workflows are implemented as Luigi workflow tasks. For an overview of Luigi and how it compares to other popular workflow tools, see Airflow vs. Luigi vs. Argo vs. MLFlow vs. KubeFlow on the Data Revenue blog.

Luigi allows Service Catalog Puppet to control the number of workers associated with workflow tasks, and to control other aspects of the workflows, for better scaling and performance. Service Catalog Puppet also provides a depends_on mechanism for managing product and step dependencies, and for orchestrating product provisioning. This feature helps you implement and operationally manage fine-grained product definitions and complex dependencies.

Provisioning modes

Service Catalog Puppet supports three execution modes: hub, spoke, and async. All three modes provision products within portfolios that are already defined in Service Catalog. They rely on Service Catalog product sharing to the target accounts and use Service Catalog administrator and launch roles to realize provisioning in those targets. Service Catalog Puppet performs the bootstrapping steps within the same organization based on the role configurations provided in the YAML configuration files. The tool also supports provisioning to multiple organizations from a single hub account. In this scenario, bootstrapping must be performed manually in the external organizations to allow Service Catalog Puppet to perform required provisioning actions in the external organization’s accounts.

In all provisioning modes, Service Catalog Puppet implements product provisioning directly without calling Service Catalog’s provisioning process. You can configure a provisioning manifest to use the role and target account specifications in an existing Service Catalog stack set constraint. Service Catalog Puppet uses this information to do its own provisioning with Luigi workflows.

You can define targets for product portfolio provisioning based on an account tagging approach, in addition to specifying OUs or accounts directly. In account tag-based provisioning, a portfolio product is provisioned to all accounts that have all the tags in the specified manifest provisioning tag set. For example, if you want to issue a portfolio product to all institutional production accounts in US East Regions, you could specify the tags `type:prod`, `partition:us-east`, and `scope:institutional-client`. You can also declare account and OU exclusions to prohibit provisioning to OUs or accounts that have the tags that you specify, or to accounts that are members of the OU-specified targets. For more information about account tagging, see the Service Catalog Tools documentation.

Hub mode

In hub provisioning mode, all Luigi workflows for the spoke accounts are managed from the designated central hub account. The hub account assumes an IAM role that allows it to perform actions in a spoke account, but the management of tasks occurs from within the hub account. The hub account waits synchronously until all spoke account provisioning tasks complete, either successfully or unsuccessfully. It then reports final status. The hub account mode is the oldest and most mature provisioning mode. However, many users have moved to the spoke provisioning mode to achieve greater provisioning concurrency and speed.

Spoke mode

In spoke mode, the Service Catalog hub account initiates the Luigi workflows to run in the designated bootstrapped spoke accounts. The hub account is notified when the spoke workflows complete. Failures in a spoke account rise up to the hub account. The hub account polls the spoke account to see if it’s finished and to determine status.

Spoke mode is least likely to require AWS service quota increases because almost everything runs in the separate spoke accounts. Spoke mode also provides far greater concurrency than hub mode while retaining central control. It can improve provisioning speed by 800 percent over hub mode. Spoke mode supports product chaining through `DependsOn` relationships between products, which ensures that a product that is depended on has already been provisioned. A product that comprises chained products can also provision a component chained product. You can also use specialized AWS Lambda function calls to perform required steps. Faults in one spoke are isolated from other spokes.

Spoke mode is used by enterprises that have over 980 accounts in up to 7 Regions. These enterprises are generally able to provision a product to all Regions and accounts in their infrastructure within an hour.

> **Note**
>
> These results might vary based on factors such as the networking infrastructure, the workload, and the quotas in place for the AWS organization hub and spoke accounts. They

> also depend on the product resources that are being provisioned, their inherent creation times, and their dependencies on other resources.

Aysnc mode

Async mode initiates provisioning workflows in spoke accounts, but it doesn't wait for or receive completion responses from the spokes.

Caching

Another mechanism that Service Catalog Puppet uses to optimize the speed of workflows is to cache common tasks that represent steps in the workflow. When a cached task is complete, it writes its output to Amazon Simple Storage Service (Amazon S3). The next time the task is invoked in the same session with the same parameters, Service Catalog Puppet uses the cached values instead of rerunning the task. For more information, see Caching in the Service Catalog Puppet documentation.

DevSecOps lifecycle support

Service Catalog Puppet includes support for managing the DevSecOps pipeline. You can use Service Catalog Tools actions (as illustrated in the Service Catalog Puppet overview) to automate testing and to promote products across your AWS lifecycle accounts, including the recommended canary account. For more information, see Managing your environments in the Service Catalog Puppet documentation.

To ensure that any issues related to a product change are detected before widespread production use, Service Catalog Puppet requires at least one canary account for initial deployment. After you test and gain confidence in the new release, you can promote it to non-canary production accounts. If you identify any issues, you can roll the release back and reintroduce it when the issues are resolved. When you use this approach, production issues might occur if you release a canary version that has an issue to production accounts. As an alternative approach, you can run full regression tests for each product change before releasing the change to production. This introduces additional overhead in the CI/CD process but helps avoid production issues. It is the DevSecOps administrator's responsibility to determine the best feature release scenarios and approaches for their development teams.

Service Catalog Puppet allows multiple teams to develop and test the provisioning of Service Catalog product solutions simultaneously. As a best practice, a product shouldn't be changed by multiple developers at the same time. Instead, you can break out products into finer-grained components for separate, simultaneous modifications.

Service Catalog Puppet also helps automate testing through an assertion statement that provides static and unit test capabilities. You can test service control policies (SCPs) and IAM policies by using policy simulators. These are technically end-to-end tests but can be used in system integration test (SIT) environments. For more information, see Using policy simulations and Applying service control policies in the Service Catalog Puppet documentation.

Maturity, completeness, and support

Although Service Catalog Puppet isn't an officially supported AWS service, it has been widely adopted. This tool has been used by large organizations over the last few years to successfully and centrally provision products to hundreds of OU accounts within their desired provisioning time windows. It has proven to provide fault-tolerant product provisioning at scale. Users who encounter any issues with Service Catalog Puppet can log them in the GitHub repository for resolution by the contributors to this AWS Labs solution.

Service Catalog Factory

Service Catalog Factory is another tool provided by AWS Labs. It is similar to AWS Control Tower—it generates accounts and calls Service Catalog (potentially through Puppet) to provision IaP within those accounts. It uses many of the same mechanisms as Service Catalog Puppet to implement its capabilities. Service Catalog Factory can call Service Catalog or Service Catalog Puppet to provision the infrastructure for products in an account. This tool also supports account generation in multiple AWS Regions and organizations. For more information, see the Service Catalog Factory documentation and GitHub repository.

Summary and next steps

Service Catalog helps you quickly and reliably provision your infrastructure as a product. You can self-serve infrastructure from a defined catalog of products or push products to designated target accounts in a hub-and-spoke model. You can define Service Catalog products and their provisioning templates by using CloudFormation scripting or by using the AWS CDK. In both approaches, Service Catalog provisions a product by calling CloudFormation to deploy a stack that represents the product's provisioning template. The stack is deployed to all designated target accounts within a CloudFormation stack set.

The AWS CDK approach for Service Catalog development supports greater modularization and reuse than CloudFormation, because you can define products and their resources by using predefined Service Catalog product and portfolio classes as well as predefined resource types. An AWS CDK implementation requires more advanced programming skills. This might be justified if your organization wants to establish its own reusable product framework with standardized resource configurations and behaviors as a foundation for your AWS infrastructure development.

You can use Service Catalog Puppet and Service Catalog Factory to augment Service Catalog functionality, primarily for provisioning. Service Catalog Puppet features declarative and tag-based product provisioning specifications; built-in, customizable, high-performance, and purpose-built provisioning workflows; and built-in, customizable, action-based CI/CD and SDLC pipelines. By using the workflow dependency management and built-in test automation features, you can chain Service Catalog products with less operational risk. Service Catalog Puppet helps you provision products to hundreds of accounts within aggressive time windows reliably. Service Catalog Factory is similar to AWS Control Tower. It generates accounts and calls Service Catalog to provision IaP within those accounts.

Service Catalog and Service Catalog Tools provide extensive functionality to help you manage IaP on AWS. Service Catalog and these tools are undergoing constant improvements. For the latest features, see AWS Service Catalog features and the AWS Service Catalog Products repository.

Resources

References

- Service Catalog documentation
- Service Catalog API
- AppRegistry
- AWS CloudFormation documentation
- AWS CloudFormation stack sets
- AWS::ServiceCatalog::CloudFormationProduct resource
- Launch AWS Service Catalog products with Terraform
- AWS Cloud Development Kit (AWS CDK)
- Service Catalog Construct Library
- AWS CDK ProductStack class
- AWS Organizations documentation

Tools

- Service Catalog Puppet documentation
- Service Catalog Puppet GitHub repository
- Service Catalog Factory documentation
- Service Catalog Factory GitHub repository

AWS Prescriptive Guidance patterns

- Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions
- Copy AWS Service Catalog products across different AWS accounts and AWS Regions
- Automate AWS Service Catalog portfolio and product deployment by using AWS CDK

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	—	January 30, 2023

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.