Rehosting EBCDIC applications on ASCII environments for mainframe migrations

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Rehosting EBCDIC applications on ASCII environments for mainframe migrations

Table of Contents

Rehosting EBCDIC applications on ASCII environments for mainframe migrations

Peter West, Amazon Web Services (AWS)

October 2022 ([document history](#))

As part of your mainframe migration, it's critical to understand the challenges that you can face when moving an EBCDIC (Extended Binary Coded Decimal Interchange Code) application to an ASCII (American Standard Code for Information Interchange) or (ANSI) American National Standards Institute environment in the AWS Cloud. The challenges arise from the fact that mainframe applications use the EBCDIC character set, while Windows, Unix, and Linux environments use the ASCII character set. This guide analyzes these challenges and helps you weigh the pros and cons of keeping your mainframe applications in EBCDIC or migrating them to ASCII. The guide divides the challenges into the following categories:

- Data migration challenges
- Application migration challenges
- Physical environment challenges

Data migration challenges

In addition to a relational database, most mainframe environments have a combination of physical datasets, such as sequential files and Virtual Storage Access Method (VSAM) files. Database migration often relies on tooling to extract, transform, and inject the data into the target database. This section outlines the challenges for physical datasets.

Character set conversion

You must use a conversion table to convert a data file from EBCDIC to ASCII. A conversion table describes a byte-to-byte conversion that's applied to every byte in the file. You can use a conversion table to correctly convert COBOL files that contain alphanumeric and numeric data with explicit sign characters.

The following table shows what data types are affected by conversion issues when you use an EBCDIC-to-ASCII conversion table.

Data types with no character set conversion issues	Data types with character set conversion issues
PIC X(n)	PIC S9(n)
PIC A(n)	PIC S9 or 9 BINARY/COMP
PIC 9(n) (Unsigned)	PIC S9 or 9 COMP-3
PIC S9(n) SIGN SEPARATE	

Zoned-decimal format

Data in the zoned-decimal format can look like regular character numeric data. The difference is in the way the sign is stored. A zoned-decimal field looks like a regular numeric display field except for the sign indicator, as in PIC S9(n) for a signed numeric display field and PIC 9(n) for an unsigned numeric display field. In a zoned-decimal field, the sign is stored in the high nibble of the last byte.

The following table shows the values stored in the data to represent "+/-" 1–9 in both their EBCDIC and ASCII values.

Digit	EBCDIC Hex-Binary	EBCDIC Display	ASCII Hex-Binary	ASCII Display
+0	X"C0"–1100 0000	{	X"30"–0011 0000	0
+1	X"C1"–1100 0001	A	X"31"–0011 0001	1
+2	X"C2"–1100 0010	B	X"32"–0011 0010	2
+3	X"C3"–1100 0011	C	X"33"–0011 0011	3
+4	X"C4"–1100 0100	D	X"34"–0011 0100	4
+5	X"C5"–1100 0101	E	X"35"–0011 0101	5
+6	X"C6"–1100 0110	F	X"36"–0011 0110	6
+7	X"C7"–1100 0111	G	X"37"–0011 0111	7
+8	X"C8"–1100 1000	H	X"38"–0011 1000	8
+9	X"C9"–1100 1001	I	X"39"–0011 1001	9
-0	X"D0"–1101 0000	}	X"70"–0111 0000	p
-1	X"D1"–1101 0001	J	X"71"–0111 0001	q

-2	X"D2"–1101 0010	K	X"72"–0111 0010	r
-3	X"D3"–1101 0011	L	X"73"–0111 0011	s
-4	X"D4"–1101 0100	M	X"74"–0111 0100	t
-5	X"D5"–1101 0101	N	X"75"–0111 0101	u
-6	X"D6"–1101 0110	O	X"76"–0111 0110	v
-7	X"D7"–1101 0111	P	X"77"–0111 0111	w
-8	X"D8"–1101 0100	Q	X"78"–0111 1000	x
-9	X"D9"–1101 1001	R	X"79"–0111 1001	y

For example, if a PIC S9(4) contains the value "-1234", then its value is stored in EBCDIC as X"F1F2F3D4". Each byte is looked up in the EBCDIC-to-ASCII conversion table and translated to the corresponding ASCII value. In this example, the first three characters convert correctly from their EBCDIC numeric value to the ASCII numeric value X"313233". However, the last character, which contains the overpunched sign nibble X"D4" "M", is converted to the ASCII equivalent of "M", which is X"4D".

The correct ASCII value for this zoned-decimal character is X"74" "t". If you use the EBCDIC-to-ASCII conversion table to convert this field, the result is X"3132334Dv" "123M" and not the correct X"31323374" "123t". If you don't handle this data correctly, you can get data corruption and invalid numeric data errors in the application.

BINARY (COMP or COMP-4)

BINARY (COMP or COMP-4) data is stored in a binary format. COMP fields are typically found in multiples of two bytes and are used to store numeric data values that exceed the numeric range of an equivalent numeric display item. For example, PIC 9(4) is a 2-byte numeric display item that can contain the values "0000"–"9999". The value "1234" is stored as X"F1F2F3F4" in EBCDIC and X"31323334" when translated to ASCII.

PIC 9(4) COMP is a 2-byte binary field (the same as a short int in C) and can contain the values "0"–"65535". Storing the maximum value in a display numeric field requires a PIC 9(6), which consumes two additional bytes. The following example shows the value "1234" represented in binary.

Byte 1								Byte 2							
0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0

The preceding example, represented as hexadecimal (hex) data, contains the value X"04D2". If you use the EBCDIC-to-ASCII conversion table to convert this data, the data converts to X"1A4B". When interpreted as a numeric, the value is "6731" and not the expected "1234".

COMP data doesn't require conversion because it's stored the same in both EBCDIC and ASCII. If you don't handle COMP data correctly, you can get data corruption. Because every value in a COMP data item is a valid numeric value there would be no numeric handling errors reported by the application. The value would simply be incorrect. As the preceding example shows, the "1234" value becomes "6731".

Packed-decimal format (COMP-3)

A COMP-3 field stores two digits per byte by storing only the low nibble of each byte. For example, the value "1" is represented by X"F1" in EBCDIC and X"31" in ASCII. The value "2" is represented by X"F2" in EBCDIC and X"32" in ASCII. In a COMP-3 field, the nibble of each byte is stored in a single byte. So, the value "12" X"F1F2" or X"3132" in ASCII is stored in COMP-3 as X"012C".

A COMP-3 field contains a "C" character for unsigned and positive signed values in the low nibble of the last byte and a "D" for negative signed values. COMP-3 data doesn't require conversion

because it's stored the same in EBCDIC and ASCII. If you use the EBCDIC-to-ASCII conversion table to convert COMP-3 data, then the data becomes corrupted and you get invalid numeric data errors in the application.

Complex record layouts

A file containing a zoned-decimal, COMP, or COMP-3 field isn't the only issue to consider when converting a data file from EBCDIC to ASCII. The file itself can contain different record layouts or even areas that use the REDEFINES clause to introduce a combination of data areas that need special handling. In such cases, use the Micro Focus Data File Editor to create a record layout that defines the structure of the file so that it can be converted from EBCDIC to ASCII correctly. Before you start creating record layouts, you must first identify the type of file and structure that you're working with. Here are the possible types:

- Single record structure with non-text data

- Single record structure with non-text data within areas of REDEFINES

- Multiple 01 record structures with non-text data

Single record structure with non-text data

A single record structure with non-text data is any file that contains a single 01 record area with no REDEFINES, where the data areas contain ZD, COMP, or COMP-3 data types. A single record structure with non-text data requires a simple "Default Record Layout" that's built by using the Micro Focus Data File Editor. The following example shows a single record structure with non-text data.

```
01 S-PARTS-RECORD.
   05 S-PART-ID        PIC 9(9) COMP.
   05 S-PART-TYPE      PIC X(2).
   05 S-PART-NAME      PIC X(40).
   05 S-SUB-PART-DATA.
     10 S-SUB-DESC     PIC X(40).
     10 S-SUB-COST     PIC S9(4)V99 COMP.
     10 S-SUB-WEIGHT   PIC 9(4)V99 COMP-3.
     10 FILLER         PIC X(34).
```

Single record structure with non-text data within areas of REDEFINES

A single record structure with non-text data within areas of REDEFINES is any file that contains a single 01 record area with REDEFINES, where the data areas contain ZD, COMP, or COMP-3 data types. The following example shows two REDEFINES that define a common PIC X(80) area with a combination of text and non-text data.

```
01  PARTS-RECORD.
    05  PART-ID   PIC 9(9) COMP.
    05  PART-TYPE  PIC X(2).
    05  PART-NAME  PIC X(40).
    05  PART-DATA  PIC X(80).
    05  MAIN-PART REDEFINES PART-DATA.
      10  MAIN-DESC       PIC X(40).
      10  MAIN-SUB-COUNT   PIC 9(2) COMP.
      10  MAIN-ASSEMBLIES  OCCURS 10.
        15  MAIN-SUB-ID    PIC 9(9) COMP.
    05  SUB-PART  REDEFINES PART-DATA.
      10  SUB-DESC    PIC X(40).
      10  SUB-COST    PIC S9(4)V99 COMP.
      10  SUB-WEIGHT  PIC 9(4)V99 COMP-3.
      10  FILLER      PIC X(34).
```

This type of record structure must first be decomposed to remove the REDEFINES statements, which create multiple views of the record structure. If you don't decompose the record structure, then the structure file that's created ignores the areas under the REDEFINES and treats the entire area as text data. The preceding example describes two different non-text structures under a REDEFINES clause. The structure file that's created must describe these areas as part of a unique structure that the data converter can target. The following example shows two unique layouts after the REDEFINES are removed.

```
01  M-PARTS-RECORD.
    05  M-PART-ID          PIC 9(9) COMP.
    05  M-PART-TYPE        PIC X(2).
    05  M-PART-NAME        PIC X(40).
    05  M-PART-DATA.
      10  M-MAIN-DESC        PIC X(40).
      10  M-MAIN-SUB-COUNT   PIC 9(2) COMP.
      10  M-MAIN-ASSEMBLIES  OCCURS 10.
        15  M-MAIN-SUB-ID    PIC 9(9) COMP.
01  S-PARTS-RECORD.
    05  S-PART-ID          PIC 9(9) COMP.
    05  S-PART-TYPE        PIC X(2).
```

```
   05   S-PART-NAME          PIC X(40).
   05   S-SUB-PART-DATA.
     10   S-SUB-DESC         PIC X(40).
     10   S-SUB-COST         PIC S9(4)V99 COMP.
     10   S-SUB-WEIGHT       PIC 9(4)V99 COMP-3.
     10   FILLER             PIC X(34).
```

The next step is to determine the conditional statement that you can use to isolate one record layout from the other. To determine the conditional statement, we recommend that you consult with a subject matter expert or inspect the source code. The following example shows the source code.

```
MOVE "M" TO PART-TYPE
  MOVE "MAIN ASSEMBLY" TO PART-NAME
MOVE "S" TO PART-TYPE
  MOVE "SUB ASSEMBLY 1" TO PART-NAME
```

In the source code, you can identify that the "PART-TYPE" field is used to determine the record type. The value "M" is used for the "M-PART-RECORD" and the value "S" is used for the "S-PART-RECORD". Now, you can create a structure file containing either two conditional records: one for each using the identified condition on the "M-PART-ID" and "S-PART-ID" fields. Or, you can create a single default layout and a single conditional layout.

Multiple 01 record structures with non-text data

A multiple 01 record structure with non-text data is any file that contains a multiple 01 record area, which contains ZD, COMP, or COMP-3 data types, as shown in the following example.

```
01   M-PARTS-RECORD.
   05   M-PART-ID             PIC 9(9) COMP.
   05   M-PART-TYPE           PIC X(2).
   05   M-PART-NAME           PIC X(40).
   05   M-PART-DATA.
     10   M-MAIN-DESC         PIC X(40).
     10   M-MAIN-SUB-COUNT    PIC 9(2) COMP.
     10   M-MAIN-ASSEMBLIES   OCCURS 10.
       15   M-MAIN-SUB-ID     PIC 9(9) COMP.

01   S-PARTS-RECORD.
   05   S-PART-ID             PIC 9(9) COMP.
   05   S-PART-TYPE           PIC X(2).
```

```
   05  S-PART-NAME          PIC X(40).
   05  S-SUB-PART-DATA.
     10  S-SUB-DESC          PIC X(40).
     10  S-SUB-COST          PIC S9(4)V99 COMP.
     10  S-SUB-WEIGHT        PIC 9(4)V99 COMP-3.
     10  FILLER              PIC X(34).
```

The first step is to determine the conditional statement that you can use to isolate one record layout from the other. To determine the conditional statement, we recommend that you consult with a subject matter expert or inspect the source code. The following example shows the source code.

```
MOVE "M" TO PART-TYPE
 MOVE "MAIN ASSEMBLY" TO PART-NAME
MOVE "S" TO PART-TYPE
 MOVE "SUB ASSEMBLY 1" TO PART-NAME
```

In the source code, you can identify that the "PART-TYPE" field is used to determine the record type. The value "M" is used for the "M-PART-RECORD" and the value "S" is used for the "S-PART-RECORD". Now, you can create a structure file containing either two conditional records: one for each using the identified condition on the "M-PART-ID" and "S-PART-ID" fields. Or, you can create a single default layout and a single conditional layout.

Programs using partial record descriptions

While it's good programming style to use a copy file to describe the entire format of a file, programmers sometimes hand code only the structure required by the program when targeting specific records or data structures. This might include sections of the file that are masked out by a large FILLER item or involve multiple programs that each describe part of the file. In these cases, you must create a single program that contains a complete description of the file, as shown in the following PROGRAM1 and PROGRAM2 examples.

PROGRAM1 example:

```
01  M-PARTS-RECORD.
   05  M-PART-ID            PIC 9(9) COMP.
   05  M-PART-TYPE          PIC X(2).
   05  M-PART-NAME          PIC X(40).
   05  M-PART-DATA.
```

```
      10  M-MAIN-DESC          PIC X(40).
      10  M-MAIN-SUB-COUNT     PIC 9(2) COMP.
      10  M-MAIN-ASSEMBLIES    OCCURS 10.
        15  M-MAIN-SUB-ID      PIC 9(9) COMP.
```

PROGRAM2 example:

```
01  S-PARTS-RECORD.
    05  S-PART-ID            PIC 9(9) COMP.
    05  S-PART-TYPE          PIC X(2).
    05  S-PART-NAME          PIC X(40).
    05  S-SUB-PART-DATA.
      10  S-SUB-DESC         PIC X(40).
      10  S-SUB-COST         PIC S9(4)V99 COMP.
      10  S-SUB-WEIGHT       PIC 9(4)V99 COMP-3.
      10  FILLER             PIC X(34).
```

In the preceding examples, you can load the data dictionary of each program one at a time, and
then add the different structures to the record layout. However, more complex situations can occur,
as the following PROGRAM1 and PROGRAM2 examples show.

PROGRAM1 example:

```
01  M-PARTS-RECORD.
    05  M-PART-ID            PIC 9(9) COMP.
    05  M-PART-TYPE          PIC X(2).
    05  M-PART-NAME          PIC X(40).
    05  FILLER               PIC X(82).
```

PROGRAM2 example:

```
01  S-PARTS-RECORD.
    05  S-PART-ID            PIC 9(9) COMP.
    05  S-PART-TYPE          PIC X(2).
    05  S-PART-NAME          PIC X(40).
    05  S-SUB-PART-DATA.
      10  S-SUB-DESC         PIC X(40).
      10  S-SUB-COST         PIC S9(4)V99 COMP.
      10  S-SUB-WEIGHT       PIC 9(4)V99 COMP-3.
      10  FILLER             PIC X(34).
```

In this case, part of the record layout in PROGRAM1 is masked out using a FILLER statement. If you build a record layout using this information, then the data within the FILLER block of "M-PARTS-RECORD" is treated as text and converted incorrectly. The developer will need to perform their due diligence to identify the absolute structure of the file before constructing a record layout.

Relational database

There are several considerations involved in converting a mainframe relational database to a distributed relational database. For example, moving to an ASCII or ANSI character set can result in collating sequence issues with the data. For more information on these issues, see *Collating sequence* in the *Application migration challenges* section of this guide.

For instance, key columns could return data in a different order from how they are returned on the mainframe. CURSORS and WHERE clauses respect the collating sequence of the database. If the data has dependencies on the EBCDIC collating sequence, then make your database use an EBCDIC collating sequence, if possible. Or, change the application logic to use the EBCDIC collating sequence when needed. Consider that specifying an alternate collating sequence in a SQL statement introduces additional latency when the statement runs.

Data volume

As with all migrations, you must migrate the data from the mainframe to the distributed environment. Before considering what migration tools to use, it's important to first consider the volume of data to be migrated. Migrating data can be split into four parts:

- Integration test data

- System test data

- Cut-over or go-live data

- Historical and archival data

The migration of both integration and system test data doesn't generally pose a significant challenge because you can move this data over an extended period of time prior to its use. Instead, it's better to spend your time and effort on cut-over or go-live data and historical or archival data.

It's a best practice to determine the volume of data for the go-live event early in the migration, including both physical files and database tables. By using data gathered from exporting files,

converting this data from EBCDIC to ASCII, and then importing the data to the target system, you can estimate how long it will take to cut over the data from the mainframe. The data migration window may not be acceptable, depending on the amount of time the business can be down and the time it would take to catch up after the go-live deployment. Then, you must decide if you want to remain in EBCDIC, which might reduce the data cut over by up to 50 percent, or make other accommodations, such as planning for an extended business outage and recovery.

Historical data is often conducted as a parallel project and typically isn't completed prior to the go-live deployment. Converting historical data can be challenging, especially if record or table layouts have changed over time. We recommend that you develop a plan to migrate the required data so that your organization can meet its compliance and regulatory requirements.

Application migration challenges

When you migrate an application from EBCDIC to ASCII, we recommend that you analyze and mitigate any EBCDIC dependencies in the application. This involves analyzing source code dependencies and collating sequences.

Source code dependencies

It's critical to check the characters used to represent numeric values, because these characters will get changed when your source code is converted from EBCDIC to ASCII. Some mainframe programmers use the following technique to represent numeric data in a COMP field in a copybook or in working storage, as the following example shows.

```
01 CHAR-74        PIC 99 COMP VALUE "¢".
```

In EBCDIC, the value of the preceding example is X"41" ("74" as a numeric). If the source data is converted to ASCII, then this character is changed to X"5B" ("91" as a numeric). This change won't cause a compiler issue but can result in programming issues that are challenging to debug. The following example shows an effective solution that changes the source to contain the actual numeric value.

```
01 CHAR-74        PIC 99 COMP VALUE 74.
```

We also recommend that you check for the use of embedded hex characters, as in the following example.

```
IF WS-DATA(1:1) = X"5B" …
```

In EBCDIC, the preceding example is testing the first character of the field to see if it's "$". The ASCII representation of "$" is X"24". The following example shows an effective solution that uses the intended character.

```
IF WS-DATA(1:1) = "$" …
```

Not all X"nn" values cause issues. For instance, the constant LOW-VALUE is X"00" and HIGH-VALUE is X"FF". LOW-VALUES (X"00") are typically used to initialize unsigned binary fields. For example, the following statement doesn't require any changes.

```
MOVE ALL X"00" TO WS-DATA
```

HIGH-VALUES (X"FF") are sometimes used as data markers to ensure that records are positioned
at the end of a SORT or to mark the end of a file. For example, the following statement doesn't
require any changes.

```
MOVE ALL X"FF" TO WS-DATA
```

Both of the preceding statements work the same in an EBCDIC or ASCII environment.

Collating sequence

In EBCDIC, alphabetic characters precede numeric values. In ASCII, numeric values precede
alphabetic characters. Consequently, an alphanumeric key field in VSAM or a database that
contains a mixture of alphabetic and numeric values returns data in different orders after being
migrated. We recommend that you analyze the impact of the collating sequences at the application
level to determine the correct remediation strategy.

The following table shows a simple example of alphanumeric data and how this data is sorted in
EBCDIC and ASCII.

EBCDIC	ASCII
A1	1A
A2	2B
1A	A1
2B	A2

The collating sequence affects the return of the following:

- VSAM keyed access (**Note**: VSAM files return data in the order of the collating sequence the file
  is created with. Even if the COBOL program is compiled to use an EBCDIC collating sequence,
  a VSAM file built with an ASCII collating sequence returns data by using an ASCII collating
  sequence. The ASCII collating sequence could cause application issues.)

- Internal or external SORT functions

- Conditional statements

- SQL cursors (**Note**: A SQL cursor returns data in the order of the collating sequence the database
  is created with, unless the collating sequence is defined in the DECLARE statement. Even if the
  COBOL program is compiled to use an EBCDIC collating sequence, a database built with an
  ASCII collating sequence returns data by using an ASCII collating sequence. The ASCII collating
  sequence could cause application issues.)

Collating sequence dependencies can also extend beyond the application and into the physical
world. For example, the barcodes on storage bins in a warehouse could be ordered using an EBCDIC
value. Changing the application to ASCII could require physically changing the barcodes on the
storage bins.

Physical environment challenges

The physical environment is often one of the most overlooked factors in a mainframe migration. Analyzing the physical environment can help you determine if your organization should remain in EBCDIC or migrate to ASCII. We recommend that you consider the following issues concerning your physical environment:

- **On-premises issues** – It's possible to have a physical dependency on EBCDIC if, for example, your organization manages warehouses and the warehouse bins are labeled using an EBCDIC alphanumeric key. Because alphanumeric keys sort differently between EBCDIC and ASCII, you would need to relabel physical bins so that the ordering on the shop floor matches an ASCII collating sequence.

- **External issues** – External interfaces with other customers could require receiving and sending files in EBCDIC to external mainframes. Changes made to the migrated application to support ASCII could inadvertently affect processing of this data by external clients. At a minimum, you must put in place extract, transform, and load (ETL) processes to convert the inbound data from EBCDIC to ASCII and the outbound data from ASCII back to EBCDIC.

Pros and cons

During the design phase of your migration, use this guide to inform your decision of remaining with EBCDIC or migrating to ASCII. The following section provides an overview of the pros and cons of remaining in EBCDIC compared to migrating to ASCII.

Remaining in EBCDIC

Pros

- Risk reduction by avoiding the conversion of data from EBCDIC to ASCII
- Risk reduction by avoiding application issues where EBCDIC dependencies are missed
- Risk reduction for external clients sending and receiving EBCDIC data (**Note**: There is likely to be some ETL for file type conversion but no data conversion.)
- Risk reduction by avoiding physical EBCDIC dependencies
- Risk reduction by reducing time required for data migration, especially for go-live deployments
- Cost reduction by reducing time required to migrate data
- Cost reduction by reducing time required to analyze code for EBCDIC dependencies
- Cost reduction by reducing time required to migrate and test code

Cons

- Increased complexity for non–COBOL downstream processing, especially applications written to use an ASCII collating sequence
- Increased complexity for upstream and downstream interfaces that require ASCII data

Migrating to ASCII

Pros

- Ability to use a native character set and collating sequence
- Simplified integration with non-COBOL environments

Cons

- Increased complexity for data exchange with mainframe environments

Issues with hybrid solutions

Programming flexibility allows for a hybrid solution. For example, you can convert the data to ASCII and run the application in ASCII, while also using an EBCDIC collating sequence. This approach can solve many problems, but can also create the following challenges:

- Not all programming languages can selectively change the working character set and collating sequence independently. This could create a situation where a non-COBOL application can't process the data correctly because it can't use an EBCDIC collating sequence. For instance, data from a SQL table is returned by using an EBCDIC collating sequence, but the conditional statements in the program rely on an ASCII collating sequence.
- Not all third-party tools and applications work in a hybrid way.
- Programmers might not be aware of the hybrid requirements for new applications.

Ideally, your first option is to either remain in EBCDIC or migrate fully to ASCII. We recommend that you consider a hybrid solution only when the challenges of either staying in EBCDIC or migrating to ASCII cannot be overcome.

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	—	October 4, 2022

AWS Prescriptive Guidance glossary

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

Numbers

7 Rs

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

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.

- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.

- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.

- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.

- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. You migrate servers from an on-premises platform to a cloud service for the same platform. Example: Migrate a Microsoft Hyper-V application to AWS.

- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there's no business justification for migrating them.

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

A

ABAC

> See [attribute-based access control](#).

abstracted services

> See [managed services](#).

ACID

> See [atomicity, consistency, isolation, durability](#).

active-active migration

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

active-passive migration

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

aggregate function

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

AI

> See [artificial intelligence](#).

AIOps

> See [artificial intelligence operations](#).

anonymization

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

anti-pattern

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

application control

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

application portfolio

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

artificial intelligence (AI)

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

artificial intelligence operations (AIOps)

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

asymmetric encryption

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

atomicity, consistency, isolation, durability (ACID)

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

attribute-based access control (ABAC)

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

authoritative data source

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

Availability Zone

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

AWS Cloud Adoption Framework (AWS CAF)

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

AWS Workload Qualification Framework (AWS WQF)

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

B

bad bot

A [bot](#) that is intended to disrupt or cause harm to individuals or organizations.

BCP

See [business continuity planning](#).

behavior graph

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

big-endian system

A system that stores the most significant byte first. See also [endianness](#).

binary classification

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

bloom filter

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

blue/green deployment

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

bot

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

botnet

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

branch

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

break-glass access

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

brownfield strategy

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

buffer cache

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

business capability

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

business continuity planning (BCP)

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

C

CAF

See [AWS Cloud Adoption Framework](#).

canary deployment

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

CCoE

See [Cloud Center of Excellence](#).

CDC

See [change data capture](#).

change data capture (CDC)

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

chaos engineering

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

CI/CD

See [continuous integration and continuous delivery](#).

classification

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

client-side encryption

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

Cloud Center of Excellence (CCoE)

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

cloud computing

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

cloud operating model

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

cloud stages of adoption

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

- Project – Running a few cloud-related projects for proof of concept and learning purposes

- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a
  landing zone, defining a CCoE, establishing an operations model)

- Migration – Migrating individual applications

- Re-invention – Optimizing products and services, and innovating in the cloud

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

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts,
are stored and updated through version control processes. Common cloud repositories include
GitHub or Bitbucket Cloud. 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, Amazon SageMaker AI 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.

EDI

See electronic data interchange.

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.

electronic data interchange (EDI)

The automated exchange of business documents between organizations. For more information, see What is Electronic Data Interchange.

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.

few-shot prompting

Providing an LLM with a small number of examples that demonstrate the task and desired output before asking it to perform a similar task. This technique is an application of in-context learning, where models learn from examples (*shots*) that are embedded in prompts. Few-shot prompting can be effective for tasks that require specific formatting, reasoning, or domain knowledge. See also zero-shot prompting.

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.

FM

See foundation model.

foundation model (FM)

A large deep-learning neural network that has been training on massive datasets of generalized and unlabeled data. FMs are capable of performing a wide variety of general tasks, such as understanding language, generating text and images, and conversing in natural language. For more information, see What are Foundation Models.

G

generative AI

A subset of AI models that have been trained on large amounts of data and that can use a simple text prompt to create new content and artifacts, such as images, videos, text, and audio. For more information, see What is Generative AI.

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.

golden image

A snapshot of a system or software that is used as a template to deploy new instances of that system or software. For example, in manufacturing, a golden image can be used to provision software on multiple devices and helps improve speed, scalability, and productivity in device manufacturing operations.

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.

holdout data

A portion of historical, labeled data that is withheld from a dataset that is used to train a [machine learning](#) model. You can use holdout data to evaluate the model performance by comparing the model predictions against the holdout data.

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 language model (LLM)

A deep learning [AI](#) model that is pretrained on a vast amount of data. An LLM can perform multiple tasks, such as answering questions, summarizing documents, translating text into other languages, and completing sentences. For more information, see [What are LLMs](#).

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.

LLM

> See large language model.

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

A system engineering approach that takes privacy into account through the whole development 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.

prompt chaining

Using the output of one [LLM](#) prompt as the input for the next prompt to generate better responses. This technique is used to break down a complex task into subtasks, or to iteratively refine or expand a preliminary response. It helps improve the accuracy and relevance of a model's responses and allows for more granular, personalized results.

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).

RAG

See Retrieval Augmented Generation.

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](#).

Retrieval Augmented Generation (RAG)

A [generative AI](#) technology in which an [LLM](#) references an authoritative data source that is outside of its training data sources before generating a response. For example, a RAG model might perform a semantic search of an organization's knowledge base or custom data. For more information, see [What is RAG](#).

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 by design

A system engineering approach that takes security into account through the whole development process.

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.

system prompt

A technique for providing context, instructions, or guidelines to an LLM to direct its behavior. System prompts help set context and establish rules for interactions with users.

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.

zero-shot prompting

> Providing an LLM with instructions for performing a task but no examples (*shots*) that can help guide it. The LLM must use its pre-trained knowledge to handle the task. The effectiveness of zero-shot prompting depends on the complexity of the task and the quality of the prompt. See also few-shot prompting.

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.