



Optimisation des performances des requêtes PostgreSQL

## AWS Conseils prescriptifs



# AWS Conseils prescriptifs: Optimisation des performances des requêtes PostgreSQL

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

# Table of Contents

Introduction .....	1
Cas d'utilisation pour le réglage des performances des requêtes .....	1
EXPLIQUEZ le plan .....	2
La déclaration EXPLAIN .....	2
Utiliser EXPLAIN ANALYZE .....	2
Comment lire le plan de requête EXPLAIN .....	2
.....	4
Classements .....	13
Incompatibilité des types de données .....	16
Appel de fonction dans SELECT .....	18
DANS ou EXISTE .....	19
Sous-requêtes ou CTE .....	23
FAQ .....	26
Qu'est-ce que EXPLAIN ? .....	26
Qu'est-ce que EXPLAIN ANALYZE ? .....	26
Qu'est-ce que le classement dans PostgreSQL ? .....	27
Qu'est-ce qu'un CTE ? .....	27
Quelles sont les catégories de fonctions dans PostgreSQL ? .....	27
.....	29
Collaborateurs .....	30
Historique de la documentation .....	31
Glossaire .....	32
# .....	32
A .....	33
B .....	36
C .....	38
D .....	41
E .....	45
F .....	48
G .....	50
H .....	51
I .....	53
L .....	55
M .....	56

O .....	61
P .....	63
Q .....	66
R .....	67
S .....	70
T .....	74
U .....	75
V .....	76
W .....	77
Z .....	78
	lxxix

# Optimisation des performances des requêtes PostgreSQL

Amazon Web Services ([contributeurs](#))

Avril 2024 ([historique du document](#))

PostgreSQL est un système de base de données relationnelle objet open source puissant, flexible et fiable. Il existe de nombreuses manières d'optimiser les performances d'une requête PostgreSQL. Le processus d'optimisation de la requête dépend du cas d'utilisation. Connaître le plan de requête actuel peut vous aider à identifier et à comprendre les problèmes éventuels et à apporter les modifications nécessaires. Il se peut que vous deviez analyser les tables pour maintenir les statistiques de la base de données à jour. L'optimiseur PostgreSQL utilisera ces statistiques pour exécuter la requête plus rapidement. Ce guide se concentre sur les meilleures pratiques pour améliorer les performances des requêtes PostgreSQL.

Ce guide part du principe que vous disposez d'une instance de base de données Amazon Relational Database Service (Amazon RDS) existante pour PostgreSQL ou Amazon Aurora PostgreSQL compatible.

## Cas d'utilisation pour le réglage des performances des requêtes

Ce guide couvre cinq cas d'utilisation, avec des explications et des exemples :

- Classements
- Incompatibilité entre les types de données
- Appel de fonction dans l'指令 SELECT
- IN ou EXISTS
- Sous-requêtes ou expressions de table communes (CTE)

Chaque cas d'utilisation fournit des détails sur le plan d'exécution initial, explique comment analyser le plan pour identifier le problème et propose une solution. La mise en œuvre de ces cas d'utilisation se traduit généralement par des temps de réponse plus rapides pour les requêtes, une réduction de la charge sur le serveur et une amélioration globale de l'efficacité du système. Ces améliorations peuvent améliorer l'expérience utilisateur et améliorer la fiabilité du système.

# Le plan de requête EXPLAIN

PostgreSQL fournit les options EXPLAIN ANALYZE et pour renvoyer EXPLAIN des plans de requête avec des détails sur la façon dont la requête sera exécutée.

## La déclaration EXPLAIN

L'EXPLAIN instruction renvoie le plan de requête généré par le planificateur PostgreSQL pour une instruction donnée. Le plan de requête indique les éléments suivants :

- Comment les tables impliquées dans un relevé seront numérisées (par exemple, par analyse d'index ou analyse séquentielle)
- Comment plusieurs tables seront jointes (par exemple, jointure par hachage, jointure par fusion ou jointure par boucle imbriquée)

Il est essentiel de comprendre le plan pour améliorer les performances de la requête. Après avoir compris le plan, vous pouvez vous concentrer sur les points où la requête prend trop de temps et prendre des mesures pour réduire le temps.

## Utiliser EXPLAIN ANALYZE

Dans PostgreSQL EXPLAIN ne générera un plan que pour l'instruction donnée. Si vous ajoutez le ANALYZE mot clé, il EXPLAIN renverra le plan, exécutera la requête et affichera le temps d'exécution réel et le nombre de lignes pour chaque étape. Cela est indispensable pour analyser les performances des requêtes.

### ⚠️ Important

Lors de l'utilisation EXPLAIN ANALYZE, soyez prudent avec INSERT, UPDATE, et DELETE.

## Comment lire le plan de requête EXPLAIN

Un plan de requête PostgreSQL est une structure arborescente composée de plusieurs nœuds. Le plan de EXPLAIN requête indique les étapes utilisées par le moteur de base de données pour exécuter une requête. Le plan de requête fournit les informations suivantes :

- Type d'opérations effectuées, telles que les analyses séquentielles, les analyses d'index ou les jointures par boucles imbriquées.
- Une étiquette, telle que Seq Scan, ou Index ScanNested Loop, pour décrire l'opération en cours d'exécution.
- Nom de la table ou de l'index traité par la requête.
- Colonnes de coûts et de lignes contenant des informations sur le coût estimé dans une unité de calcul arbitraire et le nombre de lignes traitées.
- État du filtre appliqué à l'opération, tel que la where condition.
- Une représentation visuelle des étapes, chaque opération étant représentée par un nœud et des flèches reliant les opérations. L'ordre des opérations est affiché de gauche à droite, les opérations antérieures alimentant les opérations ultérieures.

La capture d'écran suivante montre le plan de requête pour un scan séquentiel.

The screenshot shows the pgAdmin interface with the following details:

```
4 EXPLAIN ANALYZE SELECT * FROM mig.test_ts
5
Data Output Messages Notifications
QUERY PLAN
text
1 Seq Scan on test_ts (cost=0.00..32.60 rows=2260 width=8) (actual time=0.120..0.121 rows=1 loops=1)
2 Planning Time: 0.029 ms
3 Execution Time: 0.132 ms
```

L'estimation des coûts ( cost=0.00 .. 32.60 rows=2260 width=8 ) signifie que PostgreSQL prévoit que la requête nécessitera 32,60 unités de calcul pour renvoyer des résultats.

La 0.00 valeur est le coût auquel ce nœud peut commencer à fonctionner (dans ce cas, l'heure de démarrage de la requête). La rows valeur est le nombre estimé de lignes que le scan séquentiel renverra. La width valeur est la taille estimée en octets des lignes renvoyées.

Comme l'exemple montre EXPLAIN l'ANALYZEoption, la requête a été exécutée et les informations temporelles ont été capturées. Le résultat (actual\_time=0.120 .. 0.121 rows=1 loops=1) signifie ce qui suit :

- Le scan séquentiel a été exécuté une fois (loopsvaleur).
- Le scan a renvoyé une ligne.
- Le temps réel était de 0,12 milliseconde.

# Cas d'utilisation pour le réglage des requêtes

Ce guide couvre les cas d'utilisation suivants pour optimiser les performances des requêtes :

- Classements
- Incompatibilité des types de données
- Appel de fonction dans l'SELECTinstruction
- IN ou EXISTS
- Sous-requêtes ou expressions de table communes (CTE)

Pour tester le réglage des performances dans ces cas d'utilisation liés aux performances des requêtes, utilisez votre base de données existante et les exemples de données fournis dans ce guide. L'exemple utilise les données d'une compagnie aérienne fictive XX. Pour préparer les données d'exemple, exécutez l'exemple de code suivant :

```
--Creating required tables along with data.

--Creating user and schema
create user perf_user;
create schema perf_user AUTHORIZATION perf_user;
set search_path to perf_user;

--Table1:

CREATE TABLE IF NOT EXISTS perf_user.rnr_expiry_date
(
    airline_iata_code character(2) COLLATE pg_catalog."default",
    pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL,
    calculated_pnr_expiry_date timestamp(0) without time zone,
    row_num bigint,
    arc_expiry_date timestamp(0) without time zone,
    status character varying(10) COLLATE pg_catalog."default"
);

insert into perf_user.rnr_expiry_date
select 'XX' , upper(substring(concat(md5(random()::text), md5(random()::text)), 0,
7)),'2023-01-01 00:00:00',generate_series(1,100000), '2023-02-02 00:00:00' ,null;
```

```
CREATE INDEX rnr_expiry_date_idx1 ON perf_user.rnr_expiry_date (row_num ASC NULLS LAST);

CREATE INDEX rnr_expiry_date_idx2 ON perf_user.rnr_expiry_date (airline_iata_code COLLATE pg_catalog."default" ASC NULLS LAST, pnr_number COLLATE pg_catalog."default" ASC NULLS LAST);

CREATE INDEX rnr_expiry_date_idx3 ON perf_user.rnr_expiry_date (pnr_number ASC NULLS LAST);

vacuum analyze perf_user.rnr_expiry_date;

-----
--Table2:

CREATE TABLE IF NOT EXISTS perf_user.rnr_segment_pax
(
    airline_iata_code character varying(6) COLLATE pg_catalog."default" NOT NULL,
    pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL,
    segment_pax_id numeric(25,0) NOT NULL,
    oandd_id numeric(25,0) NOT NULL,
    segment_id numeric(25,0) NOT NULL,
    cabin_class character varying(15) COLLATE pg_catalog."default",
    pax_id numeric(25,0) NOT NULL,
    ticket_number character varying(25) COLLATE pg_catalog."default",
    ticket_type character varying(10) COLLATE pg_catalog."default",
    archive_status smallint NOT NULL DEFAULT (0)::smallint,
    certificate_number character varying(100) COLLATE pg_catalog."default",
    loyalty_number character varying(25) COLLATE pg_catalog."default",
    arc_expiry_date timestamp(0) without time zone,
    CONSTRAINT rnr_segment_pax_pk PRIMARY KEY (airline_iata_code, pnr_number, segment_id, pax_id),
    CONSTRAINT rnr_segment_pax_ck1 CHECK (ticket_type::text = ANY (ARRAY['E'::character varying::text, 'A'::character varying::text, 'C'::character varying::text, 'M'::character varying::text, 'I'::character varying::text]))
);

insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id, oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date )
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)), generate_series(1,10000000),generate_series(1,10000000), generate_series(1,10000000),generate_series(1,10000000), 'A','2023-01-01 00:00:00';
```

```

insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id,
oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date )
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(10000001,20000000),generate_series(10000001,20000000),
generate_series(10000001,20000000),generate_series(10000001,20000000), 'I','2023-01-01
00:00:00';

insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id,
oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date)
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(20000001,30000000),generate_series(20000001,30000000),
generate_series(20000001,30000000),generate_series(20000001,30000000), 'E','2023-01-01
00:00:00';

insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id,
oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date)
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(30000001,40000000),generate_series(30000001,40000000),
generate_series(30000001,40000000),generate_series(30000001,40000000), 'M','2023-01-01
00:00:00';

CREATE INDEX rnr_segment_pax_idx1
  ON perf_user.rnr_segment_pax USING btree
  (loyalty_number COLLATE pg_catalog."default" ASC NULLS LAST, airline_iata_code
  COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS rnr_segment_pax_pn_idx1
  ON perf_user.rnr_segment_pax USING btree
  (pnr_number COLLATE pg_catalog."default" ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS rnr_segment_pax_seq_idx1
  ON perf_user.rnr_segment_pax USING btree
  (segment_id ASC NULLS LAST);

vacuum analyze perf_user.rnr_segment_pax;

-----
--Table3:

CREATE TABLE IF NOT EXISTS perf_user.rnr_segment
(
    airline_iata_code character varying(6) COLLATE pg_catalog."default" NOT NULL,

```

```
pnr_number character varying(15) COLLATE pg_catalog."C" NOT NULL,
segment_id numeric(25,0) NOT NULL,
oandd_id numeric(25,0),
price_id numeric(25,0),
flight_carrier character varying(6) COLLATE pg_catalog."default" ,
flight_number integer ,
flight_suffix character varying(1) COLLATE pg_catalog."default" ,
flight_date_ltc timestamp(0) without time zone ,
airline_company_code character varying(6) COLLATE pg_catalog."default",
bd_airport_code character varying(5) COLLATE pg_catalog."default" ,
off_airport_code character varying(5) COLLATE pg_catalog."default" ,
segment_status character varying(50) COLLATE pg_catalog."default" ,
flight_status character varying(30) COLLATE pg_catalog."default",
flight_type character varying(15) COLLATE pg_catalog."default",
cabin_class character varying(15) COLLATE pg_catalog."default",
arc_expiry_date timestamp(0) without time zone,
oandd_dep_date_ltc timestamp(0) without time zone,
added_time timestamp(6) without time zone,
dep_date_ltc timestamp(0) without time zone ,
arr_date_utc timestamp(0) without time zone,
dep_date_utc timestamp(0) without time zone,
origin character varying(5) COLLATE pg_catalog."default",
destination character varying(5) COLLATE pg_catalog."default",
CONSTRAINT rnr_segment_pk PRIMARY KEY (pnr_number, segment_id, airline_iata_code)
);

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
FLIGHT_CARRIER,FLIGHT_NUMBER,FLIGHT_SUFFIX,FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(1,10000000), 'XX',110,'*', '2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
FLIGHT_CARRIER, FLIGHT_NUMBER, FLIGHT_SUFFIX, FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(10000001,20000000), 'XX',120,'*', '2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
FLIGHT_CARRIER, FLIGHT_NUMBER,FLIGHT_SUFFIX,FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(20000001,30000000), 'XX',130,'*', '2023-01-01 00:00:00';
```

```
insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
FLIGHT_CARRIER, FLIGHT_NUMBER, FLIGHT_SUFFIX, FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(30000001,40000000), 'XX', 140, '*', '2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
FLIGHT_CARRIER, FLIGHT_NUMBER, FLIGHT_SUFFIX, FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(40000001,50000000), 'XX', 150, '*', '2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
FLIGHT_CARRIER, FLIGHT_NUMBER, FLIGHT_SUFFIX, FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(50000001,60000000), 'XX', 160, '*', '2023-01-01 00:00:00';

CREATE INDEX rnr_segment_idx1 ON perf_user.rnr_segment USING btree
(flight_date_ltc ASC NULLS LAST, bd_airport_code COLLATE pg_catalog."default"
ASC NULLS LAST, off_airport_code COLLATE pg_catalog."default" ASC NULLS LAST,
flight_number ASC NULLS LAST, flight_carrier COLLATE pg_catalog."default" ASC NULLS
LAST, flight_suffix COLLATE pg_catalog."default" ASC NULLS LAST, airline_iata_code
COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);

CREATE INDEX rnr_segment_idx2
ON perf_user.rnr_segment USING btree
(dep_date_ltc ASC NULLS LAST, flight_number ASC NULLS LAST, bd_airport_code COLLATE
pg_catalog."default" ASC NULLS LAST, off_airport_code COLLATE pg_catalog."default" ASC
NULLS LAST, flight_carrier COLLATE pg_catalog."default" ASC NULLS LAST, flight_suffix
COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);

CREATE INDEX rnr_segment_idx3
ON perf_user.rnr_segment USING btree
(pnr_number COLLATE pg_catalog."default" ASC NULLS LAST, arr_date_utc ASC NULLS
LAST, airline_iata_code COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date
ASC NULLS LAST);

CREATE INDEX rnr_segment_idx4
ON perf_user.rnr_segment USING btree
(dep_date_utc ASC NULLS LAST, added_time ASC NULLS LAST, airline_iata_code COLLATE
pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);
```

```
CREATE INDEX rnr_segment_idx5
  ON perf_user.rnr_segment USING btree
  (origin COLLATE pg_catalog."default" ASC NULLS LAST, destination COLLATE
  pg_catalog."default" ASC NULLS LAST, oandd_dep_date_ltc ASC NULLS LAST,
  airline_iata_code COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC
  NULLS LAST);

CREATE INDEX rnr_segment_idx6
  ON perf_user.rnr_segment USING btree
  (pnr_number COLLATE pg_catalog."default" ASC NULLS LAST, oandd_id ASC NULLS LAST,
  segment_id ASC NULLS LAST, airline_iata_code COLLATE pg_catalog."default" ASC NULLS
  LAST, arc_expiry_date ASC NULLS LAST);

vacuum analyze perf_user.rnr_segment;
```

-----  
--Table4:

```
CREATE TABLE IF NOT EXISTS perf_user.rnr_seat_numbers
(
    airline_iata_code character varying(6) COLLATE pg_catalog."default" NOT NULL,
    pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL,
    segment_id numeric(25,0) NOT NULL,
    pax_id numeric(25,0) NOT NULL,
    seat_id numeric(25,0) NOT NULL,
    bd_airport_code character varying(5) COLLATE pg_catalog."default",
    off_airport_code character varying(5) COLLATE pg_catalog."default",
    seat_number character varying(5) COLLATE pg_catalog."default",
    seat_status character varying(20) COLLATE pg_catalog."default",
    ssr_id character varying(100) COLLATE pg_catalog."default",
    archive_status smallint DEFAULT (0)::smallint,
    seat_alloc_id numeric(25,0),
    archive_date timestamp(0) without time zone,
    seat_attribute_code character varying(201) COLLATE pg_catalog."default",
    channel_code character varying(20) COLLATE pg_catalog."default",
    arc_expiry_date timestamp(0) without time zone,
    CONSTRAINT rnr_seat_numbers_pk PRIMARY KEY (pnr_number, segment_id, pax_id,
    seat_id, airline_iata_code)
);
```

```

insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(1,10000000),generate_series(1,10000000),generate_series(1,10000000),'XX';

insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(10000001,20000000),generate_series(10000001,20000000),generate_series(10000001,20000000);

insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(20000001,30000000),generate_series(20000001,30000000),generate_series(20000001,30000000);

insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(30000001,40000000),generate_series(30000001,40000000),generate_series(30000001,40000000);

vacuum Analyze perf_user.rnr_seat_numbers;

--Table5:
CREATE TABLE IF NOT EXISTS perf_user.test_veh
(
    test_veh_id bigint NOT NULL,
    oiltype_id bigint,
    vehicle_id character varying(50) COLLATE pg_catalog."default",
    serviceprogram_id character varying(100) COLLATE pg_catalog."default",
    startdate timestamp without time zone,
    enddate timestamp without time zone,
    last_update_dt timestamp without time zone,
    CONSTRAINT test_veh_pkey PRIMARY KEY (test_veh_id),
    CONSTRAINT test_veh_oiltype_id_fkey FOREIGN KEY (oiltype_id)
        REFERENCES perf_user.oiltype (oiltype_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT test_veh_oiltype_id_fkey1 FOREIGN KEY (oiltype_id)
        REFERENCES perf_user.oiltype (oiltype_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
);
CREATE INDEX IF NOT EXISTS test_veh_enddate_ind

```

```
ON perf_user.test_veh USING btree
(enndate ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS test_veh_oiltype_id_ind
ON perf_user.test_veh USING btree
(oiltype_id ASC NULLS LAST);

--Table6:
CREATE TABLE IF NOT EXISTS perf_user.oiltype
(
    oiltype_id bigint NOT NULL,
    descr character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT oiltype_pkey PRIMARY KEY (oiltype_id)
);

CREATE INDEX IF NOT EXISTS oiltype_oiltyp_in
ON perf_user.oiltype USING btree
(oiltype_id ASC NULLS LAST);

--Table7:
CREATE TABLE IF NOT EXISTS perf_user.serviceprogram
(
    serial bigint NOT NULL,
    serviceprogram_id character varying(50) COLLATE pg_catalog."default",
    progname character varying(150) COLLATE pg_catalog."default",
    CONSTRAINT serviceprogram_pkey PRIMARY KEY (serial)
);

CREATE INDEX IF NOT EXISTS progname_id_ind
ON perf_user.serviceprogram USING btree
(progname COLLATE pg_catalog."default" ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS serviceprogram_id_ind
ON perf_user.serviceprogram USING btree
(serviceprogram_id COLLATE pg_catalog."default" ASC NULLS LAST);

--Table8:
CREATE TABLE IF NOT EXISTS perf_user.vehicleservicehistory
(
    v_id bigint NOT NULL,
    test_veh_id bigint,
```

```
desc_1 character varying(50) COLLATE pg_catalog."default",
start_date timestamp without time zone,
end_date timestamp without time zone,
CONSTRAINT vehicleservicehistory_pkey PRIMARY KEY (v_id)
);

CREATE INDEX IF NOT EXISTS veh_end_date_id_ind
    ON perf_user.vehicleservicehistory USING btree
    (end_date ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS veh_ser_ind
    ON perf_user.vehicleservicehistory USING btree
    (test_veh_id ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS vehicleservicehistory_v_id_ind
    ON perf_user.vehicleservicehistory USING btree
    (test_veh_id ASC NULLS LAST);

--Function creation
CREATE OR REPLACE FUNCTION perf_user.return_data()
RETURNS character varying
LANGUAGE 'plpgsql'
COST 100
VOLATILE PARALLEL UNSAFE
AS $BODY$
BEGIN
return 'EE9F41' ;
END;
$BODY$;

-----  
CREATE TABLE IF NOT EXISTS ITEM_DETAILS
(
    ITEMID INTEGER,
    ORDID INTEGER,
    ITEMNAME CHARACTER VARYING(200)
);

CREATE TABLE IF NOT EXISTS ORDER_DETAILS
(
    ORDID INTEGER,
    ORDNAME CHARACTER VARYING(200),
    ORDEREDPLACE CHARACTER VARYING(55)
);
```

```
CREATE TABLE IF NOT EXISTS PAYMENT_DETAILS
(
    PAYID INTEGER,
    ORDID INTEGER,
    PAYPLACE CHARACTER VARYING(55)
);
```

## Cas d'utilisation 1 — Collations

Dans une base de données, un classement est un ensemble de règles permettant de déterminer comment les données sont triées et comparées. Un classement est généralement appliqué à la manière dont les données de texte sont triées dans différentes langues à des fins d'indexation et de comparaison entre les valeurs de texte. Les langues ont des jeux de caractères et un ordre différents. Avec un classement, vous pouvez trier les données de caractères pour une langue donnée en utilisant des règles qui définissent la séquence de caractères correcte. Vous pouvez aussi spécifier les éléments suivants :

- Sensibilité majuscules
- Marques d'accent
- Types de personnages Kana
- Utilisation de symboles ou de signes de ponctuation
- Largeur des caractères
- Tri des mots

Cela peut avoir un impact sur les performances si la colonne de jointure utilise un classement différent. L'exemple de requête suivant utilise trois tables, avec un classement différent pour la colonne de jointure.

Nom de la table

rnr\_segment

Nom de colonne

pnr\_number character varying(15)  
COLLATE pg\_catalog."C" NOT NULL

rnr_segment_pax	pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL
rnr_seat_numbers	pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL

```

EXPLAIN ANALYZE SELECT
A.PNR_NUMBER,
A.PAX_ID,
A.SEGMENT_ID,
B.OANDD_ID,
C.SEAT_ID,
C.BD_AIRPORT_CODE,
C.OFF_AIRPORT_CODE,
C.SEAT_NUMBER ,
B.CABIN_CLASS ,
A.SEGMENT_PAX_ID,
C.SEAT_ALLOC_ID,
C.SSR_ID,
C.SEAT_ATTRIBUTE_CODE
from
RNR_SEGMENT_PAX A,
RNR_SEGMENT B,
RNR_SEAT_NUMBERS C
where
B.AIRLINE_IATA_CODE = 'XX'
and B.FLIGHT_CARRIER = 'XX'
and B.FLIGHT_NUMBER = 140
and B.FLIGHT_SUFFIX ='*'
and B.FLIGHT_DATE_LTC = TO_DATE('01-JAN-2023', 'DD-MON-YYYY')
and A.AIRLINE_IATA_CODE = B.AIRLINE_IATA_CODE
and A.PNR_NUMBER = B.PNR_NUMBER
and A.SEGMENT_ID = B.SEGMENT_ID
and C.AIRLINE_IATA_CODE = B.AIRLINE_IATA_CODE
and C.PNR_NUMBER = B.PNR_NUMBER
and C.SEGMENT_ID = B.SEGMENT_ID
and A.PAX_ID = C.PAX_ID
and B.PNR_NUMBER in ('9F1588','E37DE0','04E82B','813D11','BFF10F');

```

Le plan de requête de la requête précédente utilise un scan de séquence sur la `rnr_seat_numbers` table, même si cette table possède un index approprié sur les colonnes jointes. Le planificateur n'utilise pas d'analyse d'index car ces colonnes jointes utilisent des classements différents :

```

Nested Loop  (cost=1112.14..927363.51 rows=1 width=833) (actual time=5395.367..5397.253
rows=0 loops=1)
  Join Filter: (((b.pnr_number)::text = (a.pnr_number)::text) AND (b.segment_id =
a.segment_id))
    -> Gather  (cost=1111.58..670766.48 rows=1 width=843) (actual
time=5395.367..5397.251 rows=0 loops=1)
      Workers Planned: 2
      Workers Launched: 2
        -> Hash Join  (cost=111.58..669766.38 rows=1 width=843) (actual
time=5388.992..5388.993 rows=0 loops=3)
          Hash Cond: (((c.pnr_number)::text = (b.pnr_number)::text) AND
(c.segment_id = b.segment_id))
            -> Parallel Seq Scan on rnr_seat_numbers c  (cost=0.00..582154.96
rows=16666637 width=760) (actual time=0.008..2963.019 rows=13333333 loops=3)
              Filter: ((airline_iata_code)::text = 'XX'::text)
            -> Hash  (cost=111.52..111.52 rows=4 width=86) (actual time=0.121..0.121
rows=2 loops=3)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              -> Index Scan using rnr_segment_pk on rnr_segment b
(cost=0.56..111.52 rows=4 width=86) (actual time=0.082..0.116 rows=2 loops=3)
                Index Cond: (((pnr_number)::text = ANY
('9F1588,E37DE0,04E82B,813D11,BFF10F')::text[]) AND ((airline_iata_code)::text =
'XX'::text))
                Filter: (((flight_carrier)::text = 'XX'::text) AND
(flight_number = 140) AND ((flight_suffix)::text = '*'::text) AND (flight_date_ltc =
to_date('01-JAN-2023)::text, 'DD-MON-YYYY)::text)))
                Rows Removed by Filter: 20
      -> Index Scan using rnr_segment_pax_pk on rnr_segment_pax a  (cost=0.56..256597.02
rows=1 width=28) (never executed)
        Index Cond: (((airline_iata_code)::text = 'XX'::text) AND (segment_id =
c.segment_id) AND (pax_id = c.pax_id))
        Filter: ((c.pnr_number)::text = (pnr_number)::text)
Planning Time: 0.982 ms
Execution Time: 5397.314 ms

```

Pour remplacer le classement des colonnes de la table par le "C" langage par le classement par défaut fourni par PostgreSQL, exécutez l'instruction alter suivante, puis analysez le tableau :

```
alter table rnr_segment alter column pnr_number type character varying(15) COLLATE pg_catalog."default";
Analyze rnr_segment;
```

Le plan de requête utilise désormais un scan d'index, ce qui réduit le temps d'exécution.

```
Nested Loop  (cost=1.69..146.63 rows=1 width=833) (actual time=0.155..0.155 rows=0 loops=1)
  -> Nested Loop  (cost=1.13..145.89 rows=1 width=111) (actual time=0.154..0.155
      rows=0 loops=1)
        -> Index Scan using rnr_segment_pk on rnr_segment b  (cost=0.56..111.51 rows=4
            width=86) (actual time=0.048..0.097 rows=2 loops=1)
          Index Cond: (((pnr_number)::text = ANY
            ('{9F1588,E37DE0,04E82B,813D11,BFF10F}'::text[])) AND ((airline_iata_code)::text =
            'XX'::text))
          Filter: (((flight_carrier)::text = 'XX'::text) AND (flight_number =
            140) AND ((flight_suffix)::text = '*'::text) AND (flight_date_ltc = to_date('01-
            JAN-2023'::text, 'DD-MON-YYYY'::text)))
          Rows Removed by Filter: 20
        -> Index Scan using rnr_segment_pax_pk on rnr_segment_pax a  (cost=0.56..8.58
            rows=1 width=28) (actual time=0.027..0.027 rows=0 loops=2)
          Index Cond: (((airline_iata_code)::text = 'XX'::text) AND
            ((pnr_number)::text = (b.pnr_number)::text) AND (segment_id = b.segment_id))
        -> Index Scan using rnr_seat_numbers_pk on rnr_seat_numbers c  (cost=0.56..0.72
            rows=1 width=760) (never executed)
          Index Cond: (((pnr_number)::text = (a.pnr_number)::text) AND (segment_id =
            a.segment_id) AND (pax_id = a.pax_id) AND ((airline_iata_code)::text = 'XX'::text))
Planning Time: 1.432 ms
Execution Time: 0.207 ms
```

## Cas d'utilisation 2 — Incompatibilité des types de données

Le choix du type de données approprié en fonction des données permet de trouver l'équilibre optimal entre la taille du stockage et les performances.

L'exemple de requête suivant utilise la `pnr_number` colonne pour joindre deux tables. La `pnr_number` colonne contient différents types de données dans différentes tables.

Nom de la table

Nom de colonne et type de données

```

perf_user.rnr_segment_pax          pnr_number character varying(6)
perf_user.rnr_expiry_date         pnr_number character(2)

```

```

EXPLAIN ANALYZE UPDATE perf_user.RNR_SEGMENT_PAX x SET ARC_EXPIRY_DATE =
y.ARC_EXPIRY_DATE
  FROM (SELECT AIRLINE_IATA_CODE, PNR_NUMBER, ARC_EXPIRY_DATE, 0+row_num ROW_NUM
        FROM perf_user.RNR_EXPIRY_DATE
       WHERE airline_iata_code = 'XX'
         AND row_num BETWEEN (1*5000)+0 AND (1+1)*5000) y
      WHERE x.airline_iata_code = y.airline_iata_code
        AND x.PNR_NUMBER =y.PNR_NUMBER;
-----
```

```

Update on rnr_segment_pax x  (cost=290.97..1104986.32 rows=15515 width=460) (actual
time=14574.118..14574.120 rows=0 loops=1)
 -> Hash Join  (cost=290.97..1104986.32 rows=15515 width=460) (actual
time=16.967..14101.983 rows=11953 loops=1)
     Hash Cond: ((x.pnr_number)::text = (rnr_expiry_date.pnr_number)::text)
     -> Seq Scan on rnr_segment_pax x  (cost=0.00..954539.00 rows=40000320
width=446) (actual time=0.011..9702.989 rows=40000000 loops=1)
         Filter: ((airline_iata_code)::bpchar = 'XX'::bpchar)
     -> Hash  (cost=225.37..225.37 rows=5248 width=24) (actual time=16.540..16.541
rows=5001 loops=1)
         Buckets: 8192  Batches: 1  Memory Usage: 338kB
         -> Index Scan using rnr_expiry_date_idx1 on rnr_expiry_date
(cost=0.29..225.37 rows=5248 width=24) (actual time=3.102..15.331 rows=5001 loops=1)
             Index Cond: ((row_num >= 5000) AND (row_num <= 10000))
             Filter: (airline_iata_code = 'XX'::bpchar)
Planning Time: 4.445 ms
Execution Time: 14574.322 ms

```

Lorsque vous l'exéutez EXPLAIN ANALYZE, le planificateur utilise un scan de séquence au rnr\_segment\_pax lieu d'un scan d'index, même si les colonnes utilisées dans la jointure contiennent des index. Le planificateur n'utilise pas de scan d'index car les colonnes utilisées dans la jointure ont des longueurs différentes.

Modifiez les colonnes de la table pour conserver le même type de données pour les deux tables impliquées dans la condition de jointure, puis analysez la table :

```
alter table perf_user.rnr_expiry_date alter column airline_iata_code type character varying(6) ;

analyze perf_user.rnr_expiry_date;
```

Les tables ont désormais la même longueur sur les deux colonnes utilisées dans la condition de jointure.

Exécutez à nouveau EXPLAIN ANALYZE. Le planificateur effectue une analyse d'index, ce qui améliore considérablement les performances des requêtes.

```
Update on rnr_segment_pax x  (cost=0.86..59733.09 rows=14637 width=460) (actual
time=416.653..416.654 rows=0 loops=1)
 -> Nested Loop  (cost=0.86..59733.09 rows=14637 width=460) (actual
time=0.103..91.106 rows=11953 loops=1)
     -> Index Scan using rnr_expiry_date_idx1 on rnr_expiry_date
(cost=0.29..212.69 rows=4951 width=24) (actual time=0.025..3.023 rows=5001 loops=1)
         Index Cond: ((row_num >= 5000) AND (row_num <= 10000))
         Filter: ((airline_iata_code)::text = 'XX)::text)
     -> Index Scan using rnr_segment_pax_pk on rnr_segment_pax x  (cost=0.56..11.99
rows=3 width=446) (actual time=0.014..0.016 rows=2 loops=5001)
         Index Cond: (((airline_iata_code)::text = 'XX)::text) AND
((pnr_number)::text = (rnr_expiry_date.pnr_number)::text))
Planning Time: 0.310 ms
Execution Time: 416.696 ms
```

## Cas d'utilisation 3 — Appel de fonction dans l'instruction SELECT

L'appel d'une fonction dans une where clause peut réduire les performances des requêtes lorsque la fonction existe VOLATILE et que vous n'utilisez pas le select mot-clé lors de l'appel de la fonction :

```
Select * from tab_name where FieldName = FunctionName(parameters);
```

Une analyse d'index est exécutée si l'selectinstruction est utilisée lors de l'appel de la fonction :

```
Select * from tab_name where FieldName = ( select FunctionName(parameters) );
```

Le pnr\_number champ possède un index dans la rnr\_expiry\_date table. L'index est utilisé lors de la comparaison de la valeur de la where clause.

```
explain analyze select * from perf_user.rnr_expiry_date where pnr_number= 'EE9F41';

"Index Scan using rnr_expiry_date_idx3 on rnr_expiry_date  (cost=0.29..8.31 rows=1
width=72) (actual time=0.020..0.021 rows=1 loops=1)"
"  Index Cond: ((pnr_number)::text = 'EE9F41'::text)"
"Planning Time: 0.063 ms"
"Execution Time: 0.038 ms"
```

Un scan séquentiel est effectué lorsqu'une fonction est appelée sans le `select` mot-clé, même lorsqu'un index est disponible sur le champ.

```
explain analyze select * from perf_user.rnr_expiry_date where pnr_number=
perf_user.return_data();

"Seq Scan on rnr_expiry_date  (cost=0.00..27084.00 rows=1 width=72) (actual
time=0.112..135.917 rows=1 loops=1)"
"  Filter: ((pnr_number)::text = (perf_user.return_data())::text)"
"  Rows Removed by Filter: 99999"
"Planning Time: 0.053 ms"
"Execution Time: 136.803 ms"
```

Un scan d'index est effectué lorsque la fonction est appelée avec le `select` mot clé.

```
explain analyze select * from perf_user.rnr_expiry_date where pnr_number= (select
perf_user.return_data() );

"Index Scan using rnr_expiry_date_idx3 on rnr_expiry_date  (cost=0.55..8.57 rows=1
width=72) (actual time=0.058..0.061 rows=1 loops=1)"
"  Index Cond: ((pnr_number)::text = ($0)::text)"
"  InitPlan 1 (returns $0)"
"    -> Result  (cost=0.00..0.26 rows=1 width=32) (actual time=0.021..0.022 rows=1
loops=1)"
"Planning Time: 0.147 ms"
"Execution Time: 0.111 ms"
```

## Cas d'utilisation 4 — IN ou EXISTS

Si la requête comporte des NOT IN opérateurs IN ou, nous vous recommandons de vérifier le plan de requête pour vérifier que l'index approprié est utilisé. Si l'index approprié n'est pas utilisé et que

les performances des requêtes prennent plus de temps que prévu, essayez de réécrire la requête en utilisant les NOT EXISTS conditions EXISTS or.

Prenons l'exemple suivant, qui utilise NOT IN :

```
EXPLAIN ANALYZE SELECT
    TEST_VEH.TEST_VEH_ID,
    TEST_VEH.VEHICLE_ID,
    TEST_VEH.SERVICEPROGRAM_ID,
    TEST_VEH.STARTDATE,
    TEST_VEH.ENDDATE,
    TEST_VEH.OILTYPE_ID
FROM PERF_USER.TEST_VEH TEST_VEH
JOIN PERF_USER.OILTYPE OT ON OT.OILTYPE_ID =TEST_VEH.OILTYPE_ID
JOIN PERF_USER.SERVICEPROGRAM SP ON SP.SERVICEPROGRAM_ID = TEST_VEH.SERVICEPROGRAM_ID
WHERE SP.PROGNAME = '18FCE8FDCAF365BB'
AND OT.OILTYPE_ID =3
AND TEST_VEH.ENDDATE IS NOT NULL
AND TEST_VEH.TEST_VEH_ID NOT IN
    (SELECT TEST_VEH_ID
        FROM PERF_USER.VEHICLESERVICEHISTORY
        WHERE TEST_VEH_ID > 1
    );
-----
"Nested Loop (cost=1009.16..1188860356305.01 rows=1 width=76) (actual
 time=37299.891..37347.853 rows=0 loops=1)"
" -> Gather (cost=1009.16..1188860356303.88 rows=1 width=76) (actual
 time=37299.890..37347.849 rows=0 loops=1)"
"     Workers Planned: 2"
"     Workers Launched: 2"
"         -> Hash Join (cost=9.16..1188860355303.78 rows=1 width=76) (actual
 time=37286.742..37286.751 rows=0 loops=3)"
"             Hash Cond: ((test_veh.serviceprogram_id)::text =
(sp.serviceprogram_id)::text)"
"                 -> Parallel Index Scan using test_veh_oiltype_id_ind on test_veh
(cost=0.56..1188860351273.04 rows=1072570 width=76) (actual time=37276.290..37276.292
rows=1 loops=3)"
"                     Index Cond: (oiltype_id = 3)"
"                     Filter: ((enddate IS NOT NULL) AND (NOT (SubPlan 1)))"
"                     Rows Removed by Filter: 0"
"                     SubPlan 1"
"                         -> Materialize (cost=0.00..1025071.31 rows=33333332 width=8)
(actual time=0.418..23201.432 rows=25001498 loops=4)"
```

```

"
          -> Seq Scan on vehicleservicehistory
(cost=0.00..728195.65 rows=33333332 width=8) (actual time=0.416..13249.975
rows=25001498 loops=4)"
"
          Filter: (test_veh_id > 1)"
"
          -> Hash  (cost=8.58..8.58 rows=1 width=11) (actual time=9.045..9.046
rows=0 loops=3)"
"
          Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"
          -> Index Scan using progname_id_ind on serviceprogram sp
(cost=0.56..8.58 rows=1 width=11) (actual time=9.043..9.044 rows=0 loops=3)"
"
          Index Cond: ((progname)::text = '18FCE8FDCAF365BB'::text)"
"
          -> Seq Scan on oiltype ot  (cost=0.00..1.12 rows=1 width=8) (never executed)"
"
          Filter: (oiltype_id = 3)"
"Planning Time: 37.696 ms"
"Execution Time: 37366.335 ms"

```

La requête prend plus de 37 secondes (366 millisecondes) pour récupérer 4 millions d'enregistrements.

Le plan de requête indique qu'un scan de séquence est effectué sur la table utilisée dans la sous-requête `vehicleservicehistory`. Le scan des séquences produit un grand nombre d'enregistrements. Pour chacun de ces enregistrements de la sous-requête, la requête effectue une analyse complète de la table, ce qui est à l'origine du problème de performances.

Pour éviter le scan séquentiel de la sous-requête, réécrivez la sous-requête pour utiliser une sous-requête corrélée avec `NOT EXISTS`. La sous-requête corrélée utilisera une analyse d'index et un nombre réduit d'analyses de tables :

```

EXPLAIN ANALYZE SELECT
TEST_VEH.TEST_VEH_ID,
TEST_VEH.VEHICLE_ID,
TEST_VEH.SERVICEPROGRAM_ID,
TEST_VEH.STARTDATE,
TEST_VEH.ENDDATE,
TEST_VEH.OILTYPE_ID
FROM PERF_USER.TEST_VEH TEST_VEH
JOIN PERF_USER.OILTYPE OT ON OT.OILTYPE_ID =TEST_VEH.OILTYPE_ID
JOIN PERF_USER.SERVICEPROGRAM SP ON SP.SERVICEPROGRAM_ID = TEST_VEH.SERVICEPROGRAM_ID
WHERE SP.PROGNAME  = '18FCE8FDCAF365BB'
AND OT.OILTYPE_ID  =3
AND TEST_VEH.ENDDATE  IS NOT NULL
AND NOT EXISTS
(SELECT TEST_VEH_ID

```

```

        FROM PERF_USER.VEHICLESERVICEHISTORY
        WHERE
TEST_VEH.TEST_VEH_ID=VEHICLESERVICEHISTORY.TEST_VEH_ID
            AND TEST_VEH_ID > 1
        );

-----
"Nested Loop Anti Join  (cost=1009.03..936146.10 rows=1 width=76) (actual
 time=12.693..12.810 rows=0 loops=1)"
"  -> Nested Loop  (cost=1008.59..936141.78 rows=1 width=76) (actual
 time=12.692..12.809 rows=0 loops=1)"
"      -> Gather  (cost=1008.59..936140.64 rows=1 width=76) (actual
 time=12.691..12.807 rows=0 loops=1)"
"          Workers Planned: 2"
"          Workers Launched: 2"
"          -> Hash Join  (cost=8.59..935140.54 rows=1 width=76) (actual
 time=0.773..0.774 rows=0 loops=3)"
"              Hash Cond: ((test_veh.serviceprogram_id)::text =
(sp.serviceprogram_id)::text)"
"              -> Parallel Seq Scan on test_veh  (cost=0.00..927087.67
rows=2145139 width=76) (actual time=0.672..0.672 rows=1 loops=3)"
"                  Filter: ((enddate IS NOT NULL) AND (oiltype_id = 3))"
"                  Rows Removed by Filter: 7"
"                  -> Hash  (cost=8.58..8.58 rows=1 width=11) (actual
 time=0.040..0.040 rows=0 loops=3)"
"                      Buckets: 1024 Batches: 1 Memory Usage: 8kB"
"                      -> Index Scan using progname_id_ind on serviceprogram sp
(cost=0.56..8.58 rows=1 width=11) (actual time=0.039..0.040 rows=0 loops=3)"
"                          Index Cond: ((progname)::text =
'18FCE8FDAB365BB)::text)"
"                          -> Seq Scan on oiltype ot  (cost=0.00..1.12 rows=1 width=8) (never executed)"
"                              Filter: (oiltype_id = 3)"
" -> Index Only Scan using veh_ser_ind on vehicleservicehistory  (cost=0.44..4.32
rows=1 width=8) (never executed)"
"     Index Cond: ((test_veh_id = test_veh.test_veh_id) AND (test_veh_id > 1))"
"     Heap Fetches: 0"
"Planning Time: 11.115 ms"
"Execution Time: 12.871 ms"

```

Après la modification, la requête prend moins de 13 ms pour traiter 4 millions d'enregistrements

Selon le plan de requête de la requête modifiée, la table `vehicleservicehistory` peut être analysée par index. L'utilisation d'un scan d'index permet de réduire le coût et le nombre de

lignes concernées. Vous pouvez ainsi réduire le temps d'exécution d'une requête et améliorer ses performances.

## Cas d'utilisation 5 — Sous-requêtes ou CTE

Les expressions de table communes (CTE) permettent de décomposer les requêtes volumineuses en requêtes plus petites. Cela facilite la maintenance de l'ensemble de la requête.

Les jointures de sous-requêtes sont remplacées par des jointures CTE, qui sont plus lisibles car la requête est nommée et séparée dans la section CTE. Cela est particulièrement utile lorsque la taille de la requête augmente et qu'elle devient plus difficile à gérer. De plus, les résultats du CTE dans PostgreSQL sont matérialisés. Si vous appelez le CTE à plusieurs endroits, la définition de requête réelle ne sera exécutée qu'une seule fois. Le résultat sera enregistré en mémoire. Vous pouvez l'utiliser pour toute logique complexe qui doit être utilisée à plusieurs endroits dans la même requête. Mettez cette logique dans un CTE etappelez le CTE autant de fois que vous le souhaitez.

Par exemple, un client utilisait des requêtes d'application en ligne comportant de nombreuses sous-requêtes au sein des requêtes. Les sous-requêtes ont été filtrées en fonction des valeurs des paramètres d'entrée envoyées par les applications.

```
EXPLAIN ANALYZE
SELECT * FROM
ORDER_DETAILS A
WHERE A.ORDID IN (SELECT ORDID FROM PAYMENT_DETAILS)
AND A.ORDID IN (SELECT ORDID FROM ITEM_DETAILS )
AND A.ORDID = 1000000;
```

```
"Nested Loop Semi Join  (cost=3000.00..194258.21 rows=5 width=74) (actual
 time=201.605..747.945 rows=5 loops=1)"
"  -> Nested Loop Semi Join  (cost=2000.00..135040.47 rows=5 width=74) (actual
 time=146.016..666.779 rows=5 loops=1)"
"    -> Gather  (cost=1000.00..78580.31 rows=5 width=74) (actual
 time=58.893..463.570 rows=5 loops=1)"
"      Workers Planned: 2"
"      Workers Launched: 2"
"      -> Parallel Seq Scan on order_details a  (cost=0.00..77579.81 rows=2
 width=74) (actual time=165.627..549.702 rows=2 loops=3)"
"          Filter: (ordid = 1000000)"
"          Rows Removed by Filter: 1666665"
"      -> Materialize  (cost=1000.00..56460.07 rows=3 width=4) (actual
 time=17.424..40.638 rows=1 loops=5)"
```

```

"           -> Gather  (cost=1000.00..56460.06 rows=3 width=4) (actual
    time=87.113..203.178 rows=1 loops=1)"
"               Workers Planned: 2"
"               Workers Launched: 2"
"           -> Parallel Seq Scan on payment_details  (cost=0.00..55459.76
    rows=1 width=4) (actual time=174.431..423.792 rows=1 loops=3)"
"                   Filter: (ordid = 1000000)"
"                   Rows Removed by Filter: 1333002"
"   -> Materialize  (cost=1000.00..59217.64 rows=4 width=4) (actual time=11.117..16.231
    rows=1 loops=5)"
"       -> Gather  (cost=1000.00..59217.62 rows=4 width=4) (actual
    time=55.581..81.148 rows=1 loops=1)"
"           Workers Planned: 2"
"           Workers Launched: 2"
"           -> Parallel Seq Scan on item_details  (cost=0.00..58217.22 rows=2
    width=4) (actual time=287.030..411.004 rows=1 loops=3)"
"               Filter: (ordid = 1000000)"
"               Rows Removed by Filter: 1333080"
"Planning Time: 0.266 ms"
"Execution Time: 747.986 ms"

```

Après avoir modifié les sous-requêtes à l'aide d'un CTE et ajouté des filtres afin que seuls les ensembles de lignes requis soient récupérés, les performances des requêtes s'améliorent.

```

EXPLAIN ANALYZE
WITH PAYMENT AS
(
  SELECT * FROM PAYMENT_DETAILS WHERE ORDID = 1000000
),
ITEM AS
(SELECT * FROM ITEM_DETAILS WHERE ORDID = 1000000)
SELECT * FROM
ORDER_DETAILS A JOIN PAYMENT B
ON A.ORDID=B.ORDID
JOIN ITEM C ON B.ORDID=C.ORDID

```

```

"Nested Loop  (cost=3000.00..194258.91 rows=60 width=166) (actual time=586.410..732.918
rows=80 loops=1)"
"   -> Nested Loop  (cost=2000.00..115677.83 rows=12 width=92) (actual
    time=456.760..457.083 rows=16 loops=1)"
"       -> Gather  (cost=1000.00..59217.62 rows=4 width=48) (actual
    time=153.802..154.060 rows=4 loops=1)"

```

```
"          Workers Planned: 2"
"          Workers Launched: 2"
"              -> Parallel Seq Scan on item_details  (cost=0.00..58217.22 rows=2
width=48) (actual time=85.417..249.045 rows=1 loops=3)"
"                  Filter: (ordid = 1000000)"
"                  Rows Removed by Filter: 1333332"
"          -> Materialize  (cost=1000.00..56460.07 rows=3 width=44) (actual
time=75.738..75.753 rows=4 loops=4)"
"              -> Gather  (cost=1000.00..56460.06 rows=3 width=44) (actual
time=302.947..303.005 rows=4 loops=1)"
"                  Workers Planned: 2"
"                  Workers Launched: 2"
"                      -> Parallel Seq Scan on payment_details  (cost=0.00..55459.76
rows=1 width=44) (actual time=184.609..294.784 rows=1 loops=3)"
"                          Filter: (ordid = 1000000)"
"                          Rows Removed by Filter: 1333332"
"      -> Materialize  (cost=1000.00..78580.34 rows=5 width=74) (actual time=8.103..17.238
rows=5 loops=16)"
"          -> Gather  (cost=1000.00..78580.31 rows=5 width=74) (actual
time=129.641..275.795 rows=5 loops=1)"
"              Workers Planned: 2"
"              Workers Launched: 2"
"                  -> Parallel Seq Scan on order_details a  (cost=0.00..77579.81 rows=2
width=74) (actual time=78.556..268.994 rows=2 loops=3)"
"                      Filter: (ordid = 1000000)"
"                      Rows Removed by Filter: 1666665"
"Planning Time: 0.108 ms"
"Execution Time: 732.953 ms"
```

Voici les observations issues des exemples de données. Lorsque vous exécutez la requête sur un énorme ensemble de données, la différence de performances est très importante.

# FAQ

Trouvez les réponses aux questions fréquemment posées sur le réglage des performances des requêtes.

## Qu'est-ce que EXPLAIN ?

EXPLAIN est un mot clé que vous ajoutez à une requête PostgreSQL SELECT (UPDATE,, INSERTDELETE,) pour générer un plan de requête. Le plan de requête PostgreSQL détaille la manière dont la base de données entend exécuter la requête. Ce plan inclut des informations sur l'ordre d'analyse d'une table, l'utilisation de l'index et les jointures.

Utilisez le plan de requêtes pour identifier les goulets d'étranglement potentiels, optimiser les requêtes et améliorer les performances globales. Lorsque vous examinez le plan de requête, tenez compte des facteurs suivants :

- Approches d'accès aux tables
- Joignez les approches
- Conditions du filtre
- Opérations de tri
- Utilisation de l'index
- Parallelism
- Statistiques
- Estimations des coûts
- Lignes extraites de chaque étape
- Distribution de données

Pour plus d'informations sur [EXPLAIN](#), consultez la documentation de PostgreSQL.

## Qu'est-ce que EXPLAIN ANALYZE ?

Lorsque vous préparez EXPLAIN ANALYZE une requête et que vous l'exécutez, PostgreSQL exécute la requête et renvoie à la fois le plan de requête et les statistiques d'exécution. Le temps d'exécution réel, les lignes traitées à partir de chaque étape et les autres informations pertinentes

sont affichés avec le plan de requête. L'utilisation EXPLAIN ANALYZE sur une base de données de production doit être effectuée avec prudence, car l'exécution de la requête peut avoir un impact sur les performances de la base de données pendant l'analyse.

Pour plus d'informations sur [EXPLAIN ANALYZE](#), consultez la documentation de PostgreSQL.

## Qu'est-ce que le classement dans PostgreSQL ?

Dans PostgreSQL, un classement est un ensemble de règles permettant de déterminer comment les chaînes sont comparées et triées. Le classement définit l'ordre dans lequel les caractères sont pris en compte dans les comparaisons, en tenant compte des règles et des conversions spécifiques à la langue.

Pour plus d'informations sur le [classement](#), consultez la documentation de PostgreSQL.

## Qu'est-ce qu'un CTE ?

Dans une base de données PostgreSQL, une expression de table commune (CTE) est un jeu de résultats temporaire nommé auquel vous pouvez faire référence. Les CTE permettent de créer des requêtes SQL plus lisibles et plus modulaires en décomposant la logique complexe en unités nommées plus petites.

Pour plus d'informations sur les [CTE](#), consultez la documentation de PostgreSQL.

## Quelles sont les catégories de fonctions dans PostgreSQL ?

Chaque fonction PostgreSQL possède une classification de volatilité, les possibilités VOLATILE étant STABLE les suivantes : IMMUTABLE

- VOLATILE — Une VOLATILE fonction peut tout faire, y compris modifier la base de données. Il peut renvoyer des résultats différents lors d'appels successifs avec les mêmes arguments. L'optimiseur ne fait aucune hypothèse quant au comportement de telles fonctions. Une requête utilisant une fonction volatile réévaluera la fonction à chaque ligne où sa valeur est requise.
- STABLE — Une STABLE fonction ne peut pas modifier la base de données. Il est garanti de renvoyer les mêmes résultats avec les mêmes arguments pour toutes les lignes d'une seule instruction. Lorsque vous utilisez cette classification, l'optimiseur peut optimiser plusieurs appels de la fonction en un seul appel. En particulier, il est prudent d'utiliser une expression contenant une telle fonction dans une condition d'analyse d'index. (Étant donné qu'une analyse d'index n'évaluera

- la valeur de comparaison qu'une seule fois, et non une fois par ligne, il n'est pas valide d'utiliser une VOLATILE fonction dans une condition d'analyse d'index.)
- **IMMUBLE** — Une IMMUTABLE fonction ne peut pas modifier la base de données et il est garanti qu'elle renverra les mêmes résultats pour toujours avec les mêmes arguments. Lorsque vous utilisez cette classification, l'optimiseur peut préévaluer la fonction lorsqu'une requête l'appelle avec des arguments constants. Par exemple, une requête telle que celle-ci `SELECT ... WHERE x = 2 + 2` peut être simplifiée à `SELECT ... WHERE x = 4`, car la fonction sous-jacente à l'opérateur d'addition d'entiers est marquée IMMUTABLE.

VOLATILE est la valeur par défaut si la CREATE FUNCTION commande ne spécifie aucune catégorie. Pour plus d'informations sur les [types de fonctions](#), consultez la documentation de PostgreSQL.

# Ressources

## Références

- [EXPLIQUEZ](#)
- [Utilisation d'EXPLAIN](#)
- [Support de collation](#)
- [Requêtes WITH \(expressions de table communes\)](#)

## Guides

- [Activités de maintenance des bases de données PostgreSQL dans Amazon RDS et Amazon Aurora afin d'éviter les problèmes de performances](#)
- [Réglage des paramètres PostgreSQL dans Amazon RDS et Amazon Aurora](#)

# Collaborateurs

Les personnes qui ont contribué à ce document incluent :

- Tirumala Dasari, consultante principale en bases de données, AWS
- Veeranjaneyulu Grandhi, consultante principale — Bases de données, AWS
- Vamsikrishna Jammula, consultante en bases de données, AWS
- Srinivas Potlachervoo, consultant principal principal en bases de données, AWS
- Naga Srinivas Reddy Ravulapati, consultante en bases de données, AWS

# Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un [fil RSS](#).

Modification	Description	Date
<a href="#"><u>Publication initiale</u></a>	—	23 avril 2024

# AWS Glossaire des directives prescriptives

Les termes suivants sont couramment utilisés dans les stratégies, les guides et les modèles fournis par les directives AWS prescriptives. Pour suggérer des entrées, veuillez utiliser le lien Faire un commentaire à la fin du glossaire.

## Nombres

### 7 R

Sept politiques de migration courantes pour transférer des applications vers le cloud. Ces politiques s'appuient sur les 5 R identifiés par Gartner en 2011 et sont les suivantes :

- Refactorisation/réarchitecture : transférez une application et modifiez son architecture en tirant pleinement parti des fonctionnalités natives cloud pour améliorer l'agilité, les performances et la capacité de mise à l'échelle. Cela implique généralement le transfert du système d'exploitation et de la base de données. Exemple : migrez votre base de données Oracle sur site vers l'édition compatible avec Amazon Aurora PostgreSQL.
- Replateformer (déplacer et remodeler) : transférez une application vers le cloud et introduisez un certain niveau d'optimisation pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Amazon Relational Database Service (Amazon RDS) pour Oracle dans le AWS Cloud
- Racheter (rachat) : optez pour un autre produit, généralement en passant d'une licence traditionnelle à un modèle SaaS. Exemple : migrez votre système de gestion de la relation client (CRM) vers Salesforce.com.
- Réhéberger (lift and shift) : transférez une application vers le cloud sans apporter de modifications pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle locale vers Oracle sur une EC2 instance du AWS Cloud.
- Relocaliser (lift and shift au niveau de l'hyperviseur) : transférez l'infrastructure vers le cloud sans acheter de nouveau matériel, réécrire des applications ou modifier vos opérations existantes. Vous migrez des serveurs d'une plateforme sur site vers un service cloud pour la même plateforme. Exemple : migrer une Microsoft Hyper-V application vers AWS.
- Retenir : conservez les applications dans votre environnement source. Il peut s'agir d'applications nécessitant une refactorisation majeure, que vous souhaitez retarder, et d'applications existantes que vous souhaitez retenir, car rien ne justifie leur migration sur le plan commercial.

- Retirer : mettez hors service ou supprimez les applications dont vous n'avez plus besoin dans votre environnement source.

# A

## ABAC

Voir contrôle [d'accès basé sur les attributs](#).

## services abstraits

Consultez la section [Services gérés](#).

## ACIDE

Voir [atomicité, consistance, isolation, durabilité](#).

## migration active-active

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue (à l'aide d'un outil de réPLICATION bidirectionnelle ou d'opérations d'écriture double), tandis que les deux bases de données gèrent les transactions provenant de la connexion d'applications pendant la migration. Cette méthode prend en charge la migration par petits lots contrôlés au lieu d'exiger un basculement ponctuel. Elle est plus flexible mais demande plus de travail qu'une migration [active-passive](#).

## migration active-passive

Méthode de migration de base de données dans laquelle les bases de données source et cible sont synchronisées, mais seule la base de données source gère les transactions liées à la connexion des applications pendant que les données sont répliquées vers la base de données cible. La base de données cible n'accepte aucune transaction pendant la migration.

## fonction d'agrégation

Fonction SQL qui agit sur un groupe de lignes et calcule une valeur de retour unique pour le groupe. Des exemples de fonctions d'agrégation incluent SUM et MAX.

## AI

Voir [intelligence artificielle](#).

## AIOps

Voir les [opérations d'intelligence artificielle](#).

## anonymisation

Processus de suppression définitive d'informations personnelles dans un ensemble de données. L'anonymisation peut contribuer à protéger la vie privée. Les données anonymisées ne sont plus considérées comme des données personnelles.

## anti-motif

Solution fréquemment utilisée pour un problème récurrent lorsque la solution est contre-productive, inefficace ou moins efficace qu'une alternative.

## contrôle des applications

Une approche de sécurité qui permet d'utiliser uniquement des applications approuvées afin de protéger un système contre les logiciels malveillants.

## portefeuille d'applications

Ensemble d'informations détaillées sur chaque application utilisée par une organisation, y compris le coût de génération et de maintenance de l'application, ainsi que sa valeur métier. Ces informations sont essentielles pour [le processus de découverte et d'analyse du portefeuille](#) et permettent d'identifier et de prioriser les applications à migrer, à moderniser et à optimiser.

## intelligence artificielle (IA)

Domaine de l'informatique consacré à l'utilisation des technologies de calcul pour exécuter des fonctions cognitives généralement associées aux humains, telles que l'apprentissage, la résolution de problèmes et la reconnaissance de modèles. Pour plus d'informations, veuillez consulter [Qu'est-ce que l'intelligence artificielle ?](#)

## opérations d'intelligence artificielle (AIOps)

Processus consistant à utiliser des techniques de machine learning pour résoudre les problèmes opérationnels, réduire les incidents opérationnels et les interventions humaines, mais aussi améliorer la qualité du service. Pour plus d'informations sur son AIOps utilisation dans la stratégie de AWS migration, consultez le [guide d'intégration des opérations](#).

## chiffrement asymétrique

Algorithme de chiffrement qui utilise une paire de clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement. Vous pouvez partager la clé publique, car elle n'est pas utilisée pour le déchiffrement, mais l'accès à la clé privée doit être très restreint.

## atomicité, cohérence, isolement, durabilité (ACID)

Ensemble de propriétés logicielles garantissant la validité des données et la fiabilité opérationnelle d'une base de données, même en cas d'erreur, de panne de courant ou d'autres problèmes.

## contrôle d'accès par attributs (ABAC)

Pratique qui consiste à créer des autorisations détaillées en fonction des attributs de l'utilisateur, tels que le service, le poste et le nom de l'équipe. Pour plus d'informations, consultez [ABAC pour AWS](#) dans la documentation AWS Identity and Access Management (IAM).

## source de données faisant autorité

Emplacement où vous stockez la version principale des données, considérée comme la source d'information la plus fiable. Vous pouvez copier les données de la source de données officielle vers d'autres emplacements à des fins de traitement ou de modification des données, par exemple en les anonymisant, en les expurgant ou en les pseudonymisant.

## Zone de disponibilité

Un emplacement distinct au sein d'un Région AWS réseau isolé des défaillances dans d'autres zones de disponibilité et fournissant une connectivité réseau peu coûteuse et à faible latence aux autres zones de disponibilité de la même région.

## AWS Cadre d'adoption du cloud (AWS CAF)

Un cadre de directives et de meilleures pratiques visant AWS à aider les entreprises à élaborer un plan efficace pour réussir leur migration vers le cloud. AWS La CAF organise ses conseils en six domaines prioritaires appelés perspectives : les affaires, les personnes, la gouvernance, les plateformes, la sécurité et les opérations. Les perspectives d'entreprise, de personnes et de gouvernance mettent l'accent sur les compétences et les processus métier, tandis que les perspectives relatives à la plateforme, à la sécurité et aux opérations se concentrent sur les compétences et les processus techniques. Par exemple, la perspective liée aux personnes cible les parties prenantes qui s'occupent des ressources humaines (RH), des fonctions de dotation en personnel et de la gestion des personnes. Dans cette perspective, la AWS CAF fournit des conseils pour le développement du personnel, la formation et les communications afin de préparer l'organisation à une adoption réussie du cloud. Pour plus d'informations, veuillez consulter le [site Web AWS CAF](#) et le [livre blanc AWS CAF](#).

## AWS Cadre de qualification de la charge de travail (AWS WQF)

Outil qui évalue les charges de travail liées à la migration des bases de données, recommande des stratégies de migration et fournit des estimations de travail. AWS Le WQF est inclus avec

AWS Schema Conversion Tool (AWS SCT). Il analyse les schémas de base de données et les objets de code, le code d'application, les dépendances et les caractéristiques de performance, et fournit des rapports d'évaluation.

## B

mauvais bot

Un [bot](#) destiné à perturber ou à nuire à des individus ou à des organisations.

BCP

Consultez la section [Planification de la continuité des activités](#).

graphique de comportement

Vue unifiée et interactive des comportements des ressources et des interactions au fil du temps. Vous pouvez utiliser un graphique de comportement avec Amazon Detective pour examiner les tentatives de connexion infructueuses, les appels d'API suspects et les actions similaires. Pour plus d'informations, veuillez consulter [Data in a behavior graph](#) dans la documentation Detective.

système de poids fort

Système qui stocke d'abord l'octet le plus significatif. Voir aussi [endianité](#).

classification binaire

Processus qui prédit un résultat binaire (l'une des deux classes possibles). Par exemple, votre modèle de machine learning peut avoir besoin de prévoir des problèmes tels que « Cet e-mail est-il du spam ou non ? » ou « Ce produit est-il un livre ou une voiture ? ».

filtre de Bloom

Structure de données probabiliste et efficace en termes de mémoire qui est utilisée pour tester si un élément fait partie d'un ensemble.

déploiement bleu/vert

Stratégie de déploiement dans laquelle vous créez deux environnements distincts mais identiques. Vous exécutez la version actuelle de l'application dans un environnement (bleu) et la nouvelle version de l'application dans l'autre environnement (vert). Cette stratégie vous permet de revenir rapidement en arrière avec un impact minimal.

## bot

Application logicielle qui exécute des tâches automatisées sur Internet et simule l'activité ou l'interaction humaine. Certains robots sont utiles ou bénéfiques, comme les robots d'exploration Web qui indexent des informations sur Internet. D'autres robots, appelés « bots malveillants », sont destinés à perturber ou à nuire à des individus ou à des organisations.

## botnet

Réseaux de [robots](#) infectés par des [logiciels malveillants](#) et contrôlés par une seule entité, connue sous le nom d'herder ou d'opérateur de bots. Les botnets sont le mécanisme le plus connu pour faire évoluer les bots et leur impact.

## branche

Zone contenue d'un référentiel de code. La première branche créée dans un référentiel est la branche principale. Vous pouvez créer une branche à partir d'une branche existante, puis développer des fonctionnalités ou corriger des bogues dans la nouvelle branche. Une branche que vous créez pour générer une fonctionnalité est communément appelée branche de fonctionnalités. Lorsque la fonctionnalité est prête à être publiée, vous fusionnez à nouveau la branche de fonctionnalités dans la branche principale. Pour plus d'informations, consultez [À propos des branches](#) (GitHub documentation).

## accès par brise-vitre

Dans des circonstances exceptionnelles et par le biais d'un processus approuvé, il s'agit d'un moyen rapide pour un utilisateur d'accéder à un accès auquel Compte AWS il n'est généralement pas autorisé. Pour plus d'informations, consultez l'indicateur [Implementation break-glass procedures](#) dans le guide Well-Architected AWS .

## stratégie existante (brownfield)

L'infrastructure existante de votre environnement. Lorsque vous adoptez une stratégie existante pour une architecture système, vous concevez l'architecture en fonction des contraintes des systèmes et de l'infrastructure actuels. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et [greenfield](#) (inédites).

## cache de tampon

Zone de mémoire dans laquelle sont stockées les données les plus fréquemment consultées.

## capacité métier

Ce que fait une entreprise pour générer de la valeur (par exemple, les ventes, le service client ou le marketing). Les architectures de microservices et les décisions de développement

peuvent être dictées par les capacités métier. Pour plus d'informations, veuillez consulter la section [Organisation en fonction des capacités métier](#) du livre blanc [Exécution de microservices conteneurisés sur AWS](#).

planification de la continuité des activités (BCP)

Plan qui tient compte de l'impact potentiel d'un événement perturbateur, tel qu'une migration à grande échelle, sur les opérations, et qui permet à une entreprise de reprendre ses activités rapidement.

C

CAF

Voir le [cadre d'adoption du AWS cloud](#).

déploiement de Canary

Diffusion lente et progressive d'une version pour les utilisateurs finaux. Lorsque vous êtes sûr, vous déployez la nouvelle version et remplacez la version actuelle dans son intégralité.

CCo E

Voir [le Centre d'excellence du cloud](#).

CDC

Consultez la section [Capture des données de modification](#).

capture des données de modification (CDC)

Processus de suivi des modifications apportées à une source de données, telle qu'une table de base de données, et d'enregistrement des métadonnées relatives à ces modifications. Vous pouvez utiliser la CDC à diverses fins, telles que l'audit ou la réPLICATION des modifications dans un système cible afin de maintenir la synchronisation.

ingénierie du chaos

Introduire intentionnellement des défaillances ou des événements perturbateurs pour tester la résilience d'un système. Vous pouvez utiliser [AWS Fault Injection Service \(AWS FIS\)](#) pour effectuer des expériences qui stressent vos AWS charges de travail et évaluer leur réponse.

CI/CD

Découvrez [l'intégration continue et la livraison continue](#).

## classification

Processus de catégorisation qui permet de générer des prédictions. Les modèles de ML pour les problèmes de classification prédisent une valeur discrète. Les valeurs discrètes se distinguent toujours les unes des autres. Par exemple, un modèle peut avoir besoin d'évaluer la présence ou non d'une voiture sur une image.

## chiffrement côté client

Chiffrement des données localement, avant que la cible ne les Service AWS reçoive.

## Centre d'excellence du cloud (CCoE)

Une équipe multidisciplinaire qui dirige les efforts d'adoption du cloud au sein d'une organisation, notamment en développant les bonnes pratiques en matière de cloud, en mobilisant des ressources, en établissant des délais de migration et en guidant l'organisation dans le cadre de transformations à grande échelle. Pour plus d'informations, consultez les [CCoarticles électroniques](#) du blog sur la stratégie AWS Cloud d'entreprise.

## cloud computing

Technologie cloud généralement utilisée pour le stockage de données à distance et la gestion des appareils IoT. Le cloud computing est généralement associé à la technologie [informatique de pointe](#).

## modèle d'exploitation du cloud

Dans une organisation informatique, modèle d'exploitation utilisé pour créer, faire évoluer et optimiser un ou plusieurs environnements cloud. Pour plus d'informations, consultez la section [Création de votre modèle d'exploitation cloud](#).

## étapes d'adoption du cloud

Les quatre phases que les entreprises traversent généralement lorsqu'elles migrent vers AWS Cloud :

- Projet : exécution de quelques projets liés au cloud à des fins de preuve de concept et d'apprentissage
- Base : réaliser des investissements fondamentaux pour accélérer votre adoption du cloud (par exemple, créer une zone de landing zone, définir un CCo E, établir un modèle opérationnel)
- Migration : migration d'applications individuelles
- Réinvention : optimisation des produits et services et innovation dans le cloud

Ces étapes ont été définies par Stephen Orban dans le billet de blog [The Journey Toward Cloud-First & the Stages of Adoption](#) publié sur le blog AWS Cloud Enterprise Strategy. Pour plus d'informations sur leur lien avec la stratégie de AWS migration, consultez le [guide de préparation à la migration](#).

## CMDB

Consultez la base de [données de gestion des configurations](#).

## référentiel de code

Emplacement où le code source et d'autres ressources, comme la documentation, les exemples et les scripts, sont stockés et mis à jour par le biais de processus de contrôle de version. Les référentiels cloud courants incluent GitHub ou Bitbucket Cloud. Chaque version du code est appelée branche. Dans une structure de microservice, chaque référentiel est consacré à une seule fonctionnalité. Un seul pipeline CI/CD peut utiliser plusieurs référentiels.

## cache passif

Cache tampon vide, mal rempli ou contenant des données obsolètes ou non pertinentes. Cela affecte les performances, car l'instance de base de données doit lire à partir de la mémoire principale ou du disque, ce qui est plus lent que la lecture à partir du cache tampon.

## données gelées

Données rarement consultées et généralement historiques. Lorsque vous interrogez ce type de données, les requêtes lentes sont généralement acceptables. Le transfert de ces données vers des niveaux ou classes de stockage moins performants et moins coûteux peut réduire les coûts.

## vision par ordinateur (CV)

Domaine de l'[IA](#) qui utilise l'apprentissage automatique pour analyser et extraire des informations à partir de formats visuels tels que des images numériques et des vidéos. Par exemple, Amazon SageMaker AI fournit des algorithmes de traitement d'image pour les CV.

## dérive de configuration

Pour une charge de travail, une modification de configuration par rapport à l'état attendu. Cela peut entraîner une non-conformité de la charge de travail, et cela est généralement progressif et involontaire.

## base de données de gestion des configurations (CMDB)

Référentiel qui stocke et gère les informations relatives à une base de données et à son environnement informatique, y compris les composants matériels et logiciels ainsi que leurs

configurations. Vous utilisez généralement les données d'une CMDB lors de la phase de découverte et d'analyse du portefeuille de la migration.

## pack de conformité

Ensemble de AWS Config règles et d'actions correctives que vous pouvez assembler pour personnaliser vos contrôles de conformité et de sécurité. Vous pouvez déployer un pack de conformité en tant qu'entité unique dans une région Compte AWS et, ou au sein d'une organisation, à l'aide d'un modèle YAML. Pour plus d'informations, consultez la section [Packs de conformité](#) dans la AWS Config documentation.

## intégration continue et livraison continue (CI/CD)

Processus d'automatisation des étapes de source, de construction, de test, de préparation et de production du processus de publication du logiciel. CI/CD est communément décrit comme un pipeline. CI/CD peut vous aider à automatiser les processus, à améliorer la productivité, à améliorer la qualité du code et à accélérer les livraisons. Pour plus d'informations, veuillez consulter [Avantages de la livraison continue](#). CD peut également signifier déploiement continu. Pour plus d'informations, veuillez consulter [Livraison continue et déploiement continu](#).

## CV

Voir [vision par ordinateur](#).

## D

### données au repos

Données stationnaires dans votre réseau, telles que les données stockées.

### classification des données

Processus permettant d'identifier et de catégoriser les données de votre réseau en fonction de leur sévérité et de leur sensibilité. Il s'agit d'un élément essentiel de toute stratégie de gestion des risques de cybersécurité, car il vous aide à déterminer les contrôles de protection et de conservation appropriés pour les données. La classification des données est une composante du pilier de sécurité du AWS Well-Architected Framework. Pour plus d'informations, veuillez consulter [Classification des données](#).

### dérive des données

Une variation significative entre les données de production et les données utilisées pour entraîner un modèle ML, ou une modification significative des données d'entrée au fil du temps. La dérive

des données peut réduire la qualité, la précision et l'équité globales des prédictions des modèles ML.

#### données en transit

Données qui circulent activement sur votre réseau, par exemple entre les ressources du réseau.

#### maillage de données

Un cadre architectural qui fournit une propriété des données distribuée et décentralisée avec une gestion et une gouvernance centralisées.

#### minimisation des données

Le principe de collecte et de traitement des seules données strictement nécessaires. La pratique de la minimisation des données AWS Cloud peut réduire les risques liés à la confidentialité, les coûts et l'empreinte carbone de vos analyses.

#### périmètre de données

Ensemble de garde-fous préventifs dans votre AWS environnement qui permettent de garantir que seules les identités fiables accèdent aux ressources fiables des réseaux attendus. Pour plus d'informations, voir [Création d'un périmètre de données sur AWS](#).

#### prétraitement des données

Pour transformer les données brutes en un format facile à analyser par votre modèle de ML. Le prétraitement des données peut impliquer la suppression de certaines colonnes ou lignes et le traitement des valeurs manquantes, incohérentes ou en double.

#### provenance des données

Le processus de suivi de l'origine et de l'historique des données tout au long de leur cycle de vie, par exemple la manière dont les données ont été générées, transmises et stockées.

#### sujet des données

Personne dont les données sont collectées et traitées.

#### entrepôt des données

Un système de gestion des données qui prend en charge les informations commerciales, telles que les analyses. Les entrepôts de données contiennent généralement de grandes quantités de données historiques et sont généralement utilisés pour les requêtes et les analyses.

## langage de définition de base de données (DDL)

Instructions ou commandes permettant de créer ou de modifier la structure des tables et des objets dans une base de données.

## langage de manipulation de base de données (DML)

Instructions ou commandes permettant de modifier (insérer, mettre à jour et supprimer) des informations dans une base de données.

### DDL

Voir [langage de définition de base](#) de données.

### ensemble profond

Sert à combiner plusieurs modèles de deep learning à des fins de prédiction. Vous pouvez utiliser des ensembles profonds pour obtenir une prévision plus précise ou pour estimer l'incertitude des prédictions.

### deep learning

Un sous-champ de ML qui utilise plusieurs couches de réseaux neuronaux artificiels pour identifier le mappage entre les données d'entrée et les variables cibles d'intérêt.

### defense-in-depth

Approche de la sécurité de l'information dans laquelle une série de mécanismes et de contrôles de sécurité sont judicieusement répartis sur l'ensemble d'un réseau informatique afin de protéger la confidentialité, l'intégrité et la disponibilité du réseau et des données qu'il contient. Lorsque vous adoptez cette stratégie AWS, vous ajoutez plusieurs contrôles à différentes couches de la AWS Organizations structure afin de sécuriser les ressources. Par exemple, une defense-in-depth approche peut combiner l'authentification multifactorielle, la segmentation du réseau et le chiffrement.

### administrateur délégué

Dans AWS Organizations, un service compatible peut enregistrer un compte AWS membre pour administrer les comptes de l'organisation et gérer les autorisations pour ce service. Ce compte est appelé administrateur délégué pour ce service. Pour plus d'informations et une liste des services compatibles, veuillez consulter la rubrique [Services qui fonctionnent avec AWS Organizations](#) dans la documentation AWS Organizations .

## déploiement

Processus de mise à disposition d'une application, de nouvelles fonctionnalités ou de corrections de code dans l'environnement cible. Le déploiement implique la mise en œuvre de modifications dans une base de code, puis la génération et l'exécution de cette base de code dans les environnements de l'application.

## environnement de développement

Voir [environnement](#).

## contrôle de détection

Contrôle de sécurité conçu pour détecter, journaliser et alerter après la survenue d'un événement. Ces contrôles constituent une deuxième ligne de défense et vous alertent en cas d'événements de sécurité qui ont contourné les contrôles préventifs en place. Pour plus d'informations, veuillez consulter la rubrique [Contrôles de détection](#) dans Implementing security controls on AWS.

## cartographie de la chaîne de valeur du développement (DVSM)

Processus utilisé pour identifier et hiérarchiser les contraintes qui nuisent à la rapidité et à la qualité du cycle de vie du développement logiciel. DVSM étend le processus de cartographie de la chaîne de valeur initialement conçu pour les pratiques de production allégée. Il met l'accent sur les étapes et les équipes nécessaires pour créer et transférer de la valeur tout au long du processus de développement logiciel.

## jumeau numérique

Représentation virtuelle d'un système réel, tel qu'un bâtiment, une usine, un équipement industriel ou une ligne de production. Les jumeaux numériques prennent en charge la maintenance prédictive, la surveillance à distance et l'optimisation de la production.

## tableau des dimensions

Dans un [schéma en étoile](#), table plus petite contenant les attributs de données relatifs aux données quantitatives d'une table de faits. Les attributs des tables de dimensions sont généralement des champs de texte ou des nombres discrets qui se comportent comme du texte. Ces attributs sont couramment utilisés pour la contrainte des requêtes, le filtrage et l'étiquetage des ensembles de résultats.

## catastrophe

Un événement qui empêche une charge de travail ou un système d'atteindre ses objectifs commerciaux sur son site de déploiement principal. Ces événements peuvent être des

catastrophes naturelles, des défaillances techniques ou le résultat d'actions humaines, telles qu'une mauvaise configuration involontaire ou une attaque de logiciel malveillant.

## reprise après sinistre (DR)

La stratégie et le processus que vous utilisez pour minimiser les temps d'arrêt et les pertes de données causés par un [sinistre](#). Pour plus d'informations, consultez [Disaster Recovery of Workloads on AWS : Recovery in the Cloud in the AWS Well-Architected Framework](#).

## DML

Voir [langage de manipulation de base](#) de données.

## conception axée sur le domaine

Approche visant à développer un système logiciel complexe en connectant ses composants à des domaines évolutifs, ou objectifs métier essentiels, que sert chaque composant. Ce concept a été introduit par Eric Evans dans son ouvrage Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston : Addison-Wesley Professional, 2003). Pour plus d'informations sur l'utilisation du design piloté par domaine avec le modèle de figuier étrangleur, veuillez consulter [Modernizing legacy Microsoft ASP.NET \(ASMX\) web services incrementally by using containers and Amazon API Gateway](#).

## DR

Voir [reprise après sinistre](#).

## détection de dérive

Suivi des écarts par rapport à une configuration de référence. Par exemple, vous pouvez l'utiliser AWS CloudFormation pour [détecter la dérive des ressources du système](#) ou AWS Control Tower pour [détecter les modifications de votre zone d'atterrissement](#) susceptibles d'affecter le respect des exigences de gouvernance.

## DVSM

Voir la [cartographie de la chaîne de valeur du développement](#).

## E

## EDA

Voir [analyse exploratoire des données](#).

## EDI

Voir échange [de données informatisé](#).

## informatique de périphérie

Technologie qui augmente la puissance de calcul des appareils intelligents en périphérie d'un réseau IoT. Comparé au [cloud computing, l'informatique](#) de pointe peut réduire la latence des communications et améliorer le temps de réponse.

## échange de données informatisé (EDI)

L'échange automatique de documents commerciaux entre les organisations. Pour plus d'informations, voir [Qu'est-ce que l'échange de données informatisé ?](#)

## chiffrement

Processus informatique qui transforme des données en texte clair, lisibles par l'homme, en texte chiffré.

## clé de chiffrement

Chaîne cryptographique de bits aléatoires générée par un algorithme cryptographique. La longueur des clés peut varier, et chaque clé est conçue pour être imprévisible et unique.

## endianisme

Ordre selon lequel les octets sont stockés dans la mémoire de l'ordinateur. Les systèmes de poids fort stockent d'abord l'octet le plus significatif. Les systèmes de poids faible stockent d'abord l'octet le moins significatif.

## point de terminaison

Voir [point de terminaison de service](#).

## service de point de terminaison

Service que vous pouvez héberger sur un cloud privé virtuel (VPC) pour le partager avec d'autres utilisateurs. Vous pouvez créer un service de point de terminaison avec AWS PrivateLink et accorder des autorisations à d'autres Comptes AWS ou à AWS Identity and Access Management (IAM) principaux. Ces comptes ou principaux peuvent se connecter à votre service de point de terminaison de manière privée en créant des points de terminaison d'un VPC d'interface. Pour plus d'informations, veuillez consulter [Création d'un service de point de terminaison](#) dans la documentation Amazon Virtual Private Cloud (Amazon VPC).

## planification des ressources d'entreprise (ERP)

Système qui automatise et gère les principaux processus métier (tels que la comptabilité, le [MES](#) et la gestion de projet) pour une entreprise.

## chiffrement d'enveloppe

Processus de chiffrement d'une clé de chiffrement à l'aide d'une autre clé de chiffrement. Pour plus d'informations, consultez la section [Chiffrement des enveloppes](#) dans la documentation AWS Key Management Service (AWS KMS).

## environment

Instance d'une application en cours d'exécution. Les types d'environnement les plus courants dans le cloud computing sont les suivants :

- Environnement de développement : instance d'une application en cours d'exécution à laquelle seule l'équipe principale chargée de la maintenance de l'application peut accéder. Les environnements de développement sont utilisés pour tester les modifications avant de les promouvoir dans les environnements supérieurs. Ce type d'environnement est parfois appelé environnement de test.
- Environnements inférieurs : tous les environnements de développement d'une application, tels que ceux utilisés pour les générations et les tests initiaux.
- Environnement de production : instance d'une application en cours d'exécution à laquelle les utilisateurs finaux peuvent accéder. Dans un CI/CD pipeline, l'environnement de production est le dernier environnement de déploiement.
- Environnements supérieurs : tous les environnements accessibles aux utilisateurs autres que l'équipe de développement principale. Ils peuvent inclure un environnement de production, des environnements de préproduction et des environnements pour les tests d'acceptation par les utilisateurs.

## épopée

Dans les méthodologies agiles, catégories fonctionnelles qui aident à organiser et à prioriser votre travail. Les épopées fournissent une description détaillée des exigences et des tâches d'implémentation. Par exemple, les points forts de la AWS CAF en matière de sécurité incluent la gestion des identités et des accès, les contrôles de détection, la sécurité des infrastructures, la protection des données et la réponse aux incidents. Pour plus d'informations sur les épopées dans la stratégie de migration AWS , veuillez consulter le [guide d'implémentation du programme](#).

## ERP

Voir [Planification des ressources d'entreprise.](#)

### analyse exploratoire des données (EDA)

Processus d'analyse d'un jeu de données pour comprendre ses principales caractéristiques. Vous collectez ou agrégez des données, puis vous effectuez des enquêtes initiales pour trouver des modèles, détecter des anomalies et vérifier les hypothèses. L'EDA est réalisée en calculant des statistiques récapitulatives et en créant des visualisations de données.

## F

### tableau des faits

La table centrale dans un [schéma en étoile](#). Il stocke des données quantitatives sur les opérations commerciales. Généralement, une table de faits contient deux types de colonnes : celles qui contiennent des mesures et celles qui contiennent une clé étrangère pour une table de dimensions.

### échouer rapidement

Une philosophie qui utilise des tests fréquents et progressifs pour réduire le cycle de vie du développement. C'est un élément essentiel d'une approche agile.

### limite d'isolation des défauts

Dans le AWS Cloud, une limite telle qu'une zone de disponibilité Région AWS, un plan de contrôle ou un plan de données qui limite l'effet d'une panne et contribue à améliorer la résilience des charges de travail. Pour plus d'informations, consultez la section [Limites d'isolation des AWS pannes](#).

### branche de fonctionnalités

Voir [succursale](#).

### fonctionnalités

Les données d'entrée que vous utilisez pour faire une prédition. Par exemple, dans un contexte de fabrication, les fonctionnalités peuvent être des images capturées périodiquement à partir de la ligne de fabrication.

## importance des fonctionnalités

Le niveau d'importance d'une fonctionnalité pour les prédictions d'un modèle. Il s'exprime généralement sous la forme d'un score numérique qui peut être calculé à l'aide de différentes techniques, telles que la méthode Shapley Additive Explanations (SHAP) et les gradients intégrés. Pour plus d'informations, voir [Interprétabilité du modèle d'apprentissage automatique avec AWS](#).

## transformation de fonctionnalité

Optimiser les données pour le processus de ML, notamment en enrichissant les données avec des sources supplémentaires, en mettant à l'échelle les valeurs ou en extrayant plusieurs ensembles d'informations à partir d'un seul champ de données. Cela permet au modèle de ML de tirer parti des données. Par exemple, si vous décomposez la date « 2021-05-27 00:15:37 » en « 2021 », « mai », « jeudi » et « 15 », vous pouvez aider l'algorithme d'apprentissage à apprendre des modèles nuancés associés à différents composants de données.

## invitation en quelques coups

Fournir à un [LLM](#) un petit nombre d'exemples illustrant la tâche et le résultat souhaité avant de lui demander d'effectuer une tâche similaire. Cette technique est une application de l'apprentissage contextuel, dans le cadre de laquelle les modèles apprennent à partir d'exemples (prises de vue) intégrés dans des instructions. Les instructions en quelques clics peuvent être efficaces pour les tâches qui nécessitent un formatage, un raisonnement ou des connaissances de domaine spécifiques. Voir également l'[invite Zero-Shot](#).

## FGAC

Découvrez le [contrôle d'accès détaillé](#).

## contrôle d'accès détaillé (FGAC)

Utilisation de plusieurs conditions pour autoriser ou refuser une demande d'accès.

## migration instantanée (flash-cut)

Méthode de migration de base de données qui utilise la réplication continue des données par [le biais de la capture des données de modification](#) afin de migrer les données dans les plus brefs délais, au lieu d'utiliser une approche progressive. L'objectif est de réduire au maximum les temps d'arrêt.

## FM

Voir le [modèle de fondation](#).

## modèle de fondation (FM)

Un vaste réseau neuronal d'apprentissage profond qui s'est entraîné sur d'énormes ensembles de données généralisées et non étiquetées. FMs sont capables d'effectuer une grande variété de tâches générales, telles que comprendre le langage, générer du texte et des images et converser en langage naturel. Pour plus d'informations, voir [Que sont les modèles de base ?](#)

# G

## IA générative

Sous-ensemble de modèles d'[IA](#) qui ont été entraînés sur de grandes quantités de données et qui peuvent utiliser une simple invite textuelle pour créer de nouveaux contenus et artefacts, tels que des images, des vidéos, du texte et du son. Pour plus d'informations, consultez [Qu'est-ce que l'IA générative.](#)

## blocage géographique

Voir les [restrictions géographiques.](#)

## restrictions géographiques (blocage géographique)

Sur Amazon CloudFront, option permettant d'empêcher les utilisateurs de certains pays d'accéder aux distributions de contenu. Vous pouvez utiliser une liste d'autorisation ou une liste de blocage pour spécifier les pays approuvés et interdits. Pour plus d'informations, consultez [la section Restreindre la distribution géographique de votre contenu](#) dans la CloudFront documentation.

## Flux de travail Gitflow

Approche dans laquelle les environnements inférieurs et supérieurs utilisent différentes branches dans un référentiel de code source. Le flux de travail Gitflow est considéré comme existant, et le [flux de travail basé sur les troncs](#) est l'approche moderne préférée.

## image dorée

Un instantané d'un système ou d'un logiciel utilisé comme modèle pour déployer de nouvelles instances de ce système ou logiciel. Par exemple, dans le secteur de la fabrication, une image dorée peut être utilisée pour fournir des logiciels sur plusieurs appareils et contribue à améliorer la vitesse, l'évolutivité et la productivité des opérations de fabrication des appareils.

## stratégie inédite

L'absence d'infrastructures existantes dans un nouvel environnement. Lorsque vous adoptez une stratégie inédite pour une architecture système, vous pouvez sélectionner toutes les nouvelles technologies sans restriction de compatibilité avec l'infrastructure existante, également appelée [brownfield](#). Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

## barrière de protection

Règle de haut niveau qui permet de régir les ressources, les politiques et la conformité au sein des unités organisationnelles (OUs). Les barrières de protection préventives appliquent des politiques pour garantir l'alignement sur les normes de conformité. Elles sont mises en œuvre à l'aide de politiques de contrôle des services et de limites des autorisations IAM. Les barrières de protection de détection détectent les violations des politiques et les problèmes de conformité, et génèrent des alertes pour y remédier. Ils sont implémentés à l'aide d'Amazon AWS Config AWS Security Hub CSPM GuardDuty AWS Trusted Advisor, d'Amazon Inspector et de AWS Lambda contrôles personnalisés.

# H

## HA

Découvrez [la haute disponibilité](#).

## migration de base de données hétérogène

Migration de votre base de données source vers une base de données cible qui utilise un moteur de base de données différent (par exemple, Oracle vers Amazon Aurora). La migration hétérogène fait généralement partie d'un effort de réarchitecture, et la conversion du schéma peut s'avérer une tâche complexe. [AWS propose AWS SCT](#) qui facilite les conversions de schémas.

## haute disponibilité (HA)

Capacité d'une charge de travail à fonctionner en continu, sans intervention, en cas de difficultés ou de catastrophes. Les systèmes HA sont conçus pour basculer automatiquement, fournir constamment des performances de haute qualité et gérer différentes charges et défaillances avec un impact minimal sur les performances.

## modernisation des historiens

Approche utilisée pour moderniser et mettre à niveau les systèmes de technologie opérationnelle (OT) afin de mieux répondre aux besoins de l'industrie manufacturière. Un historien est un type de base de données utilisé pour collecter et stocker des données provenant de diverses sources dans une usine.

## données de rétention

Partie de données historiques étiquetées qui n'est pas divulguée dans un ensemble de données utilisé pour entraîner un modèle d'[apprentissage automatique](#). Vous pouvez utiliser les données de blocage pour évaluer les performances du modèle en comparant les prévisions du modèle aux données de blocage.

## migration de base de données homogène

Migration de votre base de données source vers une base de données cible qui partage le même moteur de base de données (par exemple, Microsoft SQL Server vers Amazon RDS for SQL Server). La migration homogène s'inscrit généralement dans le cadre d'un effort de réhébergement ou de replateforme. Vous pouvez utiliser les utilitaires de base de données natifs pour migrer le schéma.

## données chaudes

Données fréquemment consultées, telles que les données en temps réel ou les données translationnelles récentes. Ces données nécessitent généralement un niveau ou une classe de stockage à hautes performances pour fournir des réponses rapides aux requêtes.

## correctif

Solution d'urgence à un problème critique dans un environnement de production. En raison de son urgence, un correctif est généralement créé en dehors du flux de travail de DevOps publication habituel.

## période de soins intensifs

Immédiatement après le basculement, période pendant laquelle une équipe de migration gère et surveille les applications migrées dans le cloud afin de résoudre les problèmes éventuels. En règle générale, cette période dure de 1 à 4 jours. À la fin de la période de soins intensifs, l'équipe de migration transfère généralement la responsabilité des applications à l'équipe des opérations cloud.

I

laC

Considérez [l'infrastructure comme un code.](#)

politique basée sur l'identité

Politique attachée à un ou plusieurs principaux IAM qui définit leurs autorisations au sein de l' AWS Cloud environnement.

application inactive

Application dont l'utilisation moyenne du processeur et de la mémoire se situe entre 5 et 20 % sur une période de 90 jours. Dans un projet de migration, il est courant de retirer ces applications ou de les retenir sur site.

Illo T

Voir [Internet industriel des objets.](#)

infrastructure immuable

Modèle qui déploie une nouvelle infrastructure pour les charges de travail de production au lieu de mettre à jour, d'appliquer des correctifs ou de modifier l'infrastructure existante. Les infrastructures immuables sont intrinsèquement plus cohérentes, fiables et prévisibles que les infrastructures [mutables](#). Pour plus d'informations, consultez les meilleures pratiques de [déploiement à l'aide d'une infrastructure immuable](#) dans le AWS Well-Architected Framework.

VPC entrant (d'entrée)

Dans une architecture AWS multi-comptes, un VPC qui accepte, inspecte et achemine les connexions réseau depuis l'extérieur d'une application. L'[architecture AWS de référence de sécurité](#) recommande de configurer votre compte réseau avec les fonctions entrantes, sortantes et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

migration incrémentielle

Stratégie de basculement dans le cadre de laquelle vous migrez votre application par petites parties au lieu d'effectuer un basculement complet unique. Par exemple, il se peut que vous ne transfériez que quelques microservices ou utilisateurs vers le nouveau système dans un premier temps. Après avoir vérifié que tout fonctionne correctement, vous pouvez transférer

progressivement des microservices ou des utilisateurs supplémentaires jusqu'à ce que vous puissiez mettre hors service votre système hérité. Cette stratégie réduit les risques associés aux migrations de grande ampleur.

## Industry 4.0

Un terme introduit par [Klaus Schwab](#) en 2016 pour désigner la modernisation des processus de fabrication grâce aux avancées en matière de connectivité, de données en temps réel, d'automatisation, d'analyse et d'IA/ML.

## infrastructure

Ensemble des ressources et des actifs contenus dans l'environnement d'une application.

## infrastructure en tant que code (IaC)

Processus de mise en service et de gestion de l'infrastructure d'une application via un ensemble de fichiers de configuration. IaC est conçue pour vous aider à centraliser la gestion de l'infrastructure, à normaliser les ressources et à mettre à l'échelle rapidement afin que les nouveaux environnements soient reproductibles, fiables et cohérents.

## Internet industriel des objets (IIoT)

L'utilisation de capteurs et d'appareils connectés à Internet dans les secteurs industriels tels que la fabrication, l'énergie, l'automobile, les soins de santé, les sciences de la vie et l'agriculture.

Pour plus d'informations, voir [Élaboration d'une stratégie de transformation numérique de l'Internet des objets \(IIoT\) industriel](#).

## VPC d'inspection

Dans une architecture AWS multi-comptes, un VPC centralisé qui gère les inspections du trafic réseau VPCs entre (identique ou Régions AWS différent), Internet et les réseaux locaux.

L'[architecture AWS de référence de sécurité](#) recommande de configurer votre compte réseau avec les fonctions entrantes, sortantes et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

## Internet des objets (IoT)

Réseau d'objets physiques connectés dotés de capteurs ou de processeurs intégrés qui communiquent avec d'autres appareils et systèmes via Internet ou via un réseau de communication local. Pour plus d'informations, veuillez consulter la section [Qu'est-ce que l'IoT ?](#)

## interprétabilité

Caractéristique d'un modèle de machine learning qui décrit dans quelle mesure un être humain peut comprendre comment les prédictions du modèle dépendent de ses entrées. Pour plus d'informations, voir [Interprétabilité du modèle d'apprentissage automatique avec AWS](#).

## IoT

Voir [Internet des objets](#).

## Bibliothèque d'informations informatiques (ITIL)

Ensemble de bonnes pratiques pour proposer des services informatiques et les aligner sur les exigences métier. L'ITIL constitue la base de l'ITSM.

## gestion des services informatiques (ITSM)

Activités associées à la conception, à la mise en œuvre, à la gestion et à la prise en charge de services informatiques d'une organisation. Pour plus d'informations sur l'intégration des opérations cloud aux outils ITSM, veuillez consulter le [guide d'intégration des opérations](#).

## ITIL

Consultez la [bibliothèque d'informations informatiques](#).

## ITSM

Voir [Gestion des services informatiques](#).

## L

## contrôle d'accès basé sur des étiquettes (LBAC)

Une implémentation du contrôle d'accès obligatoire (MAC) dans laquelle une valeur d'étiquette de sécurité est explicitement attribuée aux utilisateurs et aux données elles-mêmes. L'intersection entre l'étiquette de sécurité utilisateur et l'étiquette de sécurité des données détermine les lignes et les colonnes visibles par l'utilisateur.

## zone de destination

Une zone d'atterrissement est un AWS environnement multi-comptes bien conçu, évolutif et sécurisé. Il s'agit d'un point de départ à partir duquel vos entreprises peuvent rapidement lancer et déployer des charges de travail et des applications en toute confiance dans leur environnement de sécurité et d'infrastructure. Pour plus d'informations sur les zones de destination, veuillez consulter [Setting up a secure and scalable multi-account AWS environment](#).

## grand modèle de langage (LLM)

Un modèle d'[intelligence artificielle basé](#) sur le deep learning qui est préentraîné sur une grande quantité de données. Un LLM peut effectuer plusieurs tâches, telles que répondre à des questions, résumer des documents, traduire du texte dans d'autres langues et compléter des phrases. Pour plus d'informations, voir [Que sont LLMs.](#)

## migration de grande envergure

Migration de 300 serveurs ou plus.

## LBAC

Voir contrôle d'[accès basé sur des étiquettes.](#)

## principe de moindre privilège

Bonne pratique de sécurité qui consiste à accorder les autorisations minimales nécessaires à l'exécution d'une tâche. Pour plus d'informations, veuillez consulter la rubrique [Accorder les autorisations de moindre privilège](#) dans la documentation IAM.

## lift and shift

Voir [7 Rs.](#)

## système de poids faible

Système qui stocke d'abord l'octet le moins significatif. Voir aussi [endianité.](#)

## LLM

Voir le [grand modèle de langage.](#)

## environnements inférieurs

Voir [environnement.](#)

# M

## machine learning (ML)

Type d'intelligence artificielle qui utilise des algorithmes et des techniques pour la reconnaissance et l'apprentissage de modèles. Le ML analyse et apprend à partir de données enregistrées, telles que les données de l'Internet des objets (IoT), pour générer un modèle statistique basé sur des modèles. Pour plus d'informations, veuillez consulter [Machine Learning.](#)

## branche principale

Voir [succursale](#).

## malware

Logiciel conçu pour compromettre la sécurité ou la confidentialité de l'ordinateur. Les logiciels malveillants peuvent perturber les systèmes informatiques, divulguer des informations sensibles ou obtenir un accès non autorisé. Parmi les malwares, on peut citer les virus, les vers, les rançongiciels, les chevaux de Troie, les logiciels espions et les enregistreurs de frappe.

## services gérés

Services AWS pour lequel AWS fonctionnent la couche d'infrastructure, le système d'exploitation et les plateformes, et vous accédez aux points de terminaison pour stocker et récupérer des données. Amazon Simple Storage Service (Amazon S3) et Amazon DynamoDB sont des exemples de services gérés. Ils sont également appelés services abstraits.

## système d'exécution de la fabrication (MES)

Un système logiciel pour le suivi, la surveillance, la documentation et le contrôle des processus de production qui convertissent les matières premières en produits finis dans l'atelier.

## MAP

Voir [Migration Acceleration Program](#).

## mécanisme

Processus complet au cours duquel vous créez un outil, favorisez son adoption, puis inspectez les résultats afin de procéder aux ajustements nécessaires. Un mécanisme est un cycle qui se renforce et s'améliore lorsqu'il fonctionne. Pour plus d'informations, voir [Création de mécanismes](#) dans le cadre AWS Well-Architected.

## compte membre

Tous, à l' exception du compte de gestion, qui font partie d'une organisation dans AWS Organizations. Un compte ne peut être membre que d'une seule organisation à la fois.

## MAILLES

Voir le [système d'exécution de la fabrication](#).

## Transport télémétrique en file d'attente de messages (MQTT)

[Protocole de communication léger machine-to-machine \(M2M\), basé sur le modèle de publication/d'abonnement, pour les appareils IoT aux ressources limitées.](#)

## microservice

Un petit service indépendant qui communique via un réseau bien défini APIs et qui est généralement détenu par de petites équipes autonomes. Par exemple, un système d'assurance peut inclure des microservices qui mappent à des capacités métier, telles que les ventes ou le marketing, ou à des sous-domaines, tels que les achats, les réclamations ou l'analytique. Les avantages des microservices incluent l'agilité, la flexibilité de la mise à l'échelle, la facilité de déploiement, la réutilisation du code et la résilience. Pour plus d'informations, consultez la section [Intégration de microservices à l'aide de services AWS sans serveur](#).

## architecture de microservices

Approche de création d'une application avec des composants indépendants qui exécutent chaque processus d'application en tant que microservice. Ces microservices communiquent via une interface bien définie en utilisant Lightweight APIs. Chaque microservice de cette architecture peut être mis à jour, déployé et mis à l'échelle pour répondre à la demande de fonctions spécifiques d'une application. Pour plus d'informations, consultez la section [Implémentation de microservices sur AWS](#).

## Programme d'accélération des migrations (MAP)

Un AWS programme qui fournit un support de conseil, des formations et des services pour aider les entreprises à établir une base opérationnelle solide pour passer au cloud, et pour aider à compenser le coût initial des migrations. MAP inclut une méthodologie de migration pour exécuter les migrations héritées de manière méthodique, ainsi qu'un ensemble d'outils pour automatiser et accélérer les scénarios de migration courants.

## migration à grande échelle

Processus consistant à transférer la majeure partie du portefeuille d'applications vers le cloud par vagues, un plus grand nombre d'applications étant déplacées plus rapidement à chaque vague. Cette phase utilise les bonnes pratiques et les enseignements tirés des phases précédentes pour implémenter une usine de migration d'équipes, d'outils et de processus en vue de rationaliser la migration des charges de travail grâce à l'automatisation et à la livraison agile. Il s'agit de la troisième phase de la [stratégie de migration AWS](#).

## usine de migration

Équipes interfonctionnelles qui rationalisent la migration des charges de travail grâce à des approches automatisées et agiles. Les équipes de Migration Factory comprennent généralement des responsables des opérations, des analystes commerciaux et des propriétaires, des ingénieurs de migration, des développeurs et DevOps des professionnels travaillant dans le cadre de sprints.

Entre 20 et 50 % du portefeuille d'applications d'entreprise est constitué de modèles répétés qui peuvent être optimisés par une approche d'usine. Pour plus d'informations, veuillez consulter la rubrique [discussion of migration factories](#) et le [guide Cloud Migration Factory](#) dans cet ensemble de contenus.

## métadonnées de migration

Informations relatives à l'application et au serveur nécessaires pour finaliser la migration. Chaque modèle de migration nécessite un ensemble de métadonnées de migration différent. Les exemples de métadonnées de migration incluent le sous-réseau cible, le groupe de sécurité et le AWS compte.

## modèle de migration

Tâche de migration reproductible qui détaille la stratégie de migration, la destination de la migration et l'application ou le service de migration utilisé. Exemple : réorganisez la migration vers Amazon EC2 avec le service de migration AWS d'applications.

## Évaluation du portefeuille de migration (MPA)

Outil en ligne qui fournit des informations pour valider l'analyse de rentabilisation en faveur de la migration vers le AWS Cloud La MPA propose une évaluation détaillée du portefeuille (dimensionnement approprié des serveurs, tarification, comparaison du coût total de possession, analyse des coûts de migration), ainsi que la planification de la migration (analyse et collecte des données d'applications, regroupement des applications, priorisation des migrations et planification des vagues). L'[outil MPA](#) (connexion requise) est disponible gratuitement pour tous les AWS consultants et consultants APN Partner.

## Évaluation de la préparation à la migration (MRA)

Processus qui consiste à obtenir des informations sur l'état de préparation d'une organisation au cloud, à identifier les forces et les faiblesses et à élaborer un plan d'action pour combler les lacunes identifiées, à l'aide du AWS CAF. Pour plus d'informations, veuillez consulter le [guide de préparation à la migration](#). La MRA est la première phase de la [stratégie de migration AWS](#).

## stratégie de migration

L'approche utilisée pour migrer une charge de travail vers le AWS Cloud. Pour plus d'informations, reportez-vous aux [7 R](#) de ce glossaire et à [Mobiliser votre organisation pour accélérer les migrations à grande échelle](#).

## ML

Voir [apprentissage automatique](#).

## modernisation

Transformation d'une application obsolète (héritée ou monolithique) et de son infrastructure en un système agile, élastique et hautement disponible dans le cloud afin de réduire les coûts, de gagner en efficacité et de tirer parti des innovations. Pour plus d'informations, consultez [la section Stratégie de modernisation des applications dans le AWS Cloud.](#)

## évaluation de la préparation à la modernisation

Évaluation qui permet de déterminer si les applications d'une organisation sont prêtes à être modernisées, d'identifier les avantages, les risques et les dépendances, et qui détermine dans quelle mesure l'organisation peut prendre en charge l'état futur de ces applications. Le résultat de l'évaluation est un plan de l'architecture cible, une feuille de route détaillant les phases de développement et les étapes du processus de modernisation, ainsi qu'un plan d'action pour combler les lacunes identifiées. Pour plus d'informations, consultez la section [Évaluation de l'état de préparation à la modernisation des applications dans le AWS Cloud.](#)

## applications monolithiques (monolithes)

Applications qui s'exécutent en tant que service unique avec des processus étroitement couplés. Les applications monolithiques ont plusieurs inconvénients. Si une fonctionnalité de l'application connaît un pic de demande, l'architecture entière doit être mise à l'échelle. L'ajout ou l'amélioration des fonctionnalités d'une application monolithique devient également plus complexe lorsque la base de code s'élargit. Pour résoudre ces problèmes, vous pouvez utiliser une architecture de microservices. Pour plus d'informations, veuillez consulter [Decomposing monoliths into microservices.](#)

## MPA

Voir [Évaluation du portefeuille de migration.](#)

## MQTT

Voir [Message Queuing Telemetry Transport.](#)

## classification multi-classes

Processus qui permet de générer des prédictions pour plusieurs classes (prédiction d'un résultat parmi plus de deux). Par exemple, un modèle de ML peut demander « Ce produit est-il un livre, une voiture ou un téléphone ? » ou « Quelle catégorie de produits intéresse le plus ce client ? ».

## infrastructure mutable

Modèle qui met à jour et modifie l'infrastructure existante pour les charges de travail de production. Pour améliorer la cohérence, la fiabilité et la prévisibilité, le AWS Well-Architected Framework recommande l'utilisation [d'une infrastructure immuable comme](#) meilleure pratique.

# O

## OAC

Voir [Contrôle d'accès à l'origine.](#)

## OAI

Voir [l'identité d'accès à l'origine.](#)

## OCM

Voir [gestion du changement organisationnel.](#)

## migration hors ligne

Méthode de migration dans laquelle la charge de travail source est supprimée au cours du processus de migration. Cette méthode implique un temps d'arrêt prolongé et est généralement utilisée pour de petites charges de travail non critiques.

## OI

Consultez la section [Intégration des opérations.](#)

## OLA

Voir l'accord [au niveau opérationnel.](#)

## migration en ligne

Méthode de migration dans laquelle la charge de travail source est copiée sur le système cible sans être mise hors ligne. Les applications connectées à la charge de travail peuvent continuer à fonctionner pendant la migration. Cette méthode implique un temps d'arrêt nul ou minimal et est généralement utilisée pour les charges de travail de production critiques.

## OPC-UA

Voir [Open Process Communications - Architecture unifiée.](#)

## Communications par processus ouvert - Architecture unifiée (OPC-UA)

Un protocole de communication machine-to-machine (M2M) pour l'automatisation industrielle.

L'OPC-UA fournit une norme d'interopérabilité avec des schémas de cryptage, d'authentification et d'autorisation des données.

### accord au niveau opérationnel (OLA)

Accord qui précise ce que les groupes informatiques fonctionnels s'engagent à fournir les uns aux autres, afin de prendre en charge un contrat de niveau de service (SLA).

### examen de l'état de préparation opérationnelle (ORR)

Une liste de questions et de bonnes pratiques associées qui vous aident à comprendre, à évaluer, à prévenir ou à réduire l'ampleur des incidents et des défaillances possibles. Pour plus d'informations, voir [Operational Readiness Reviews \(ORR\)](#) dans le AWS Well-Architected Framework.

### technologie opérationnelle (OT)

Systèmes matériels et logiciels qui fonctionnent avec l'environnement physique pour contrôler les opérations, les équipements et les infrastructures industriels. Dans le secteur manufacturier, l'intégration des systèmes OT et des technologies de l'information (IT) est au cœur des transformations de [l'industrie 4.0](#).

### intégration des opérations (OI)

Processus de modernisation des opérations dans le cloud, qui implique la planification de la préparation, l'automatisation et l'intégration. Pour en savoir plus, veuillez consulter le [guide d'intégration des opérations](#).

### journal de suivi d'organisation

Un parcours créé par AWS CloudTrail qui enregistre tous les événements pour tous les membres Comptes AWS d'une organisation dans AWS Organizations. Ce journal de suivi est créé dans chaque Compte AWS qui fait partie de l'organisation et suit l'activité de chaque compte. Pour plus d'informations, consultez [la section Crédit d'un suivi pour une organisation dans la CloudTrail documentation](#).

### gestion du changement organisationnel (OCM)

Cadre pour gérer les transformations métier majeures et perturbatrices du point de vue des personnes, de la culture et du leadership. L'OCM aide les organisations à se préparer et à effectuer la transition vers de nouveaux systèmes et de nouvelles politiques en accélérant

l'adoption des changements, en abordant les problèmes de transition et en favorisant des changements culturels et organisationnels. Dans la stratégie de AWS migration, ce cadre est appelé accélération du personnel, en raison de la rapidité du changement requise dans les projets d'adoption du cloud. Pour plus d'informations, veuillez consulter le [guide OCM](#).

## contrôle d'accès d'origine (OAC)

Dans CloudFront, une option améliorée pour restreindre l'accès afin de sécuriser votre contenu Amazon Simple Storage Service (Amazon S3). L'OAC prend en charge tous les compartiments S3 dans leur ensemble Régions AWS, le chiffrement côté serveur avec AWS KMS (SSE-KMS) et les requêtes dynamiques PUT adressées au compartiment S3. DELETE

## identité d'accès d'origine (OAI)

Dans CloudFront, une option permettant de restreindre l'accès afin de sécuriser votre contenu Amazon S3. Lorsque vous utilisez OAI, il CloudFront crée un principal auprès duquel Amazon S3 peut s'authentifier. Les principaux authentifiés ne peuvent accéder au contenu d'un compartiment S3 que par le biais d'une distribution spécifique CloudFront . Voir également [OAC](#), qui fournit un contrôle d'accès plus précis et amélioré.

## ORR

Voir [l'examen de l'état de préparation opérationnelle](#).

## DE

Voir [technologie opérationnelle](#).

## VPC sortant (de sortie)

Dans une architecture AWS multi-comptes, un VPC qui gère les connexions réseau initiées depuis une application. L'[architecture AWS de référence de sécurité](#) recommande de configurer votre compte réseau avec les fonctions entrantes, sortantes et d'inspection VPCs afin de protéger l'interface bidirectionnelle entre votre application et l'Internet en général.

## P

### limite des autorisations

Politique de gestion IAM attachée aux principaux IAM pour définir les autorisations maximales que peut avoir l'utilisateur ou le rôle. Pour plus d'informations, veuillez consulter la rubrique [Limites des autorisations](#) dans la documentation IAM.

## informations personnelles identifiables (PII)

Informations qui, lorsqu'elles sont consultées directement ou associées à d'autres données connexes, peuvent être utilisées pour déduire raisonnablement l'identité d'une personne. Les exemples d'informations personnelles incluent les noms, les adresses et les coordonnées.

### PII

Voir les [informations personnelles identifiables](#).

## manuel stratégique

Ensemble d'étapes prédéfinies qui capturent le travail associé aux migrations, comme la fourniture de fonctions d'opérations de base dans le cloud. Un manuel stratégique peut revêtir la forme de scripts, de runbooks automatisés ou d'un résumé des processus ou des étapes nécessaires au fonctionnement de votre environnement modernisé.

### PLC

Voir [contrôleur logique programmable](#).

### PLM

Consultez la section [Gestion du cycle de vie des](#) produits.

## politique

Objet capable de définir les autorisations (voir la [politique basée sur l'identité](#)), de spécifier les conditions d'accès (voir la [politique basée sur les ressources](#)) ou de définir les autorisations maximales pour tous les comptes d'une organisation dans AWS Organizations (voir la politique de contrôle des [services](#)).

## persistance polyglotte

Choix indépendant de la technologie de stockage de données d'un microservice en fonction des modèles d'accès aux données et d'autres exigences. Si vos microservices utilisent la même technologie de stockage de données, ils peuvent rencontrer des difficultés d'implémentation ou présenter des performances médiocres. Les microservices sont plus faciles à mettre en œuvre, atteignent de meilleures performances, ainsi qu'une meilleure capacité de mise à l'échelle s'ils utilisent l'entrepôt de données le mieux adapté à leurs besoins. Pour plus d'informations, veuillez consulter [Enabling data persistence in microservices](#).

## évaluation du portefeuille

Processus de découverte, d'analyse et de priorisation du portefeuille d'applications afin de planifier la migration. Pour plus d'informations, veuillez consulter [Evaluating migration readiness](#).

## predicate

Une condition de requête qui renvoie true ou false, généralement située dans une WHERE clause.

## prédictat pushdown

Technique d'optimisation des requêtes de base de données qui filtre les données de la requête avant le transfert. Cela réduit la quantité de données qui doivent être extraites et traitées à partir de la base de données relationnelle et améliore les performances des requêtes.

## contrôle préventif

Contrôle de sécurité conçu pour empêcher qu'un événement ne se produise. Ces contrôles constituent une première ligne de défense pour empêcher tout accès non autorisé ou toute modification indésirable de votre réseau. Pour plus d'informations, veuillez consulter [Preventative controls](#) dans [Implementing security controls on AWS](#).

## principal

Entité AWS capable d'effectuer des actions et d'accéder aux ressources. Cette entité est généralement un utilisateur root pour un Compte AWS rôle IAM ou un utilisateur. Pour plus d'informations, veuillez consulter la rubrique Principal dans [Termes et concepts relatifs aux rôles](#), dans la documentation IAM.

## confidentialité dès la conception

Une approche d'ingénierie système qui prend en compte la confidentialité tout au long du processus de développement.

## zones hébergées privées

Conteneur contenant des informations sur la manière dont vous souhaitez qu'Amazon Route 53 réponde aux requêtes DNS pour un domaine et ses sous-domaines au sein d'un ou de plusieurs VPCs domaines. Pour plus d'informations, veuillez consulter [Working with private hosted zones](#) dans la documentation Route 53.

## contrôle proactif

[Contrôle de sécurité](#) conçu pour empêcher le déploiement de ressources non conformes. Ces contrôles analysent les ressources avant qu'elles ne soient provisionnées. Si la ressource n'est pas conforme au contrôle, elle n'est pas provisionnée. Pour plus d'informations, consultez le [guide de référence sur les contrôles](#) dans la AWS Control Tower documentation et consultez la section [Contrôles proactifs dans Implémentation](#) des contrôles de sécurité sur AWS.

## gestion du cycle de vie des produits (PLM)

Gestion des données et des processus d'un produit tout au long de son cycle de vie, depuis la conception, le développement et le lancement, en passant par la croissance et la maturité, jusqu'au déclin et au retrait.

## environnement de production

Voir [environnement](#).

## contrôleur logique programmable (PLC)

Dans le secteur manufacturier, un ordinateur hautement fiable et adaptable qui surveille les machines et automatise les processus de fabrication.

## chaînage rapide

Utiliser le résultat d'une invite [LLM](#) comme entrée pour l'invite suivante afin de générer de meilleures réponses. Cette technique est utilisée pour décomposer une tâche complexe en sous-tâches ou pour affiner ou développer de manière itérative une réponse préliminaire. Cela permet d'améliorer la précision et la pertinence des réponses d'un modèle et permet d'obtenir des résultats plus précis et personnalisés.

## pseudonymisation

Processus de remplacement des identifiants personnels dans un ensemble de données par des valeurs fictives. La pseudonymisation peut contribuer à protéger la vie privée. Les données pseudonymisées sont toujours considérées comme des données personnelles.

## publish/subscribe (pub/sub)

Modèle qui permet des communications asynchrones entre les microservices afin d'améliorer l'évolutivité et la réactivité. Par exemple, dans un [MES](#) basé sur des microservices, un microservice peut publier des messages d'événements sur un canal auquel d'autres microservices peuvent s'abonner. Le système peut ajouter de nouveaux microservices sans modifier le service de publication.

# Q

## plan de requête

Série d'étapes, telles que des instructions, utilisées pour accéder aux données d'un système de base de données relationnelle SQL.

## régression du plan de requêtes

Le cas où un optimiseur de service de base de données choisit un plan moins optimal qu'avant une modification donnée de l'environnement de base de données. Cela peut être dû à des changements en termes de statistiques, de contraintes, de paramètres d'environnement, de liaisons de paramètres de requêtes et de mises à jour du moteur de base de données.

# R

## Matrice RACI

Voir [responsable, responsable, consulté, informé \(RACI\)](#).

## CHIFFON

Voir [Retrieval Augmented Generation](#).

## rançongiciel

Logiciel malveillant conçu pour bloquer l'accès à un système informatique ou à des données jusqu'à ce qu'un paiement soit effectué.

## Matrice RASCI

Voir [responsable, responsable, consulté, informé \(RACI\)](#).

## RCAC

Voir [contrôle d'accès aux lignes et aux colonnes](#).

## réplica en lecture

Copie d'une base de données utilisée en lecture seule. Vous pouvez acheminer les requêtes vers le réplica de lecture pour réduire la charge sur votre base de données principale.

## réarchitecte

Voir [7 Rs.](#)

## objectif de point de récupération (RPO)

Durée maximale acceptable depuis le dernier point de récupération des données. Il détermine ce qui est considéré comme étant une perte de données acceptable entre le dernier point de reprise et l'interruption du service.

objectif de temps de récupération (RTO)

Le délai maximum acceptable entre l'interruption du service et le rétablissement du service.

refactoriser

Voir [7 Rs.](#)

Région

Ensemble de AWS ressources dans une zone géographique. Chacun Région AWS est isolé et indépendant des autres pour garantir tolérance aux pannes, stabilité et résilience. Pour plus d'informations, voir [Spécifier ce que Régions AWS votre compte peut utiliser](#).

régression

Technique de ML qui prédit une valeur numérique. Par exemple, pour résoudre le problème « Quel sera le prix de vente de cette maison ? », un modèle de ML pourrait utiliser un modèle de régression linéaire pour prédire le prix de vente d'une maison sur la base de faits connus à son sujet (par exemple, la superficie en mètres carrés).

réhéberger

Voir [7 Rs.](#)

version

Dans un processus de déploiement, action visant à promouvoir les modifications apportées à un environnement de production.

déplacer

Voir [7 Rs.](#)

replateforme

Voir [7 Rs.](#)

rachat

Voir [7 Rs.](#)

résilience

La capacité d'une application à résister aux perturbations ou à s'en remettre. [La haute disponibilité et la reprise après sinistre](#) sont des considérations courantes lors de la planification de la résilience dans le AWS Cloud. Pour plus d'informations, consultez [AWS Cloud Résilience](#).

## politique basée sur les ressources

Politique attachée à une ressource, comme un compartiment Amazon S3, un point de terminaison ou une clé de chiffrement. Ce type de politique précise les principaux auxquels l'accès est autorisé, les actions prises en charge et toutes les autres conditions qui doivent être remplies.

matrice responsable, redevable, consulté et informé (RACI)

Une matrice qui définit les rôles et les responsabilités de toutes les parties impliquées dans les activités de migration et les opérations cloud. Le nom de la matrice est dérivé des types de responsabilité définis dans la matrice : responsable (R), responsable (A), consulté (C) et informé (I). Le type de support (S) est facultatif. Si vous incluez le support, la matrice est appelée matrice RASCI, et si vous l'excluez, elle est appelée matrice RACI.

## contrôle réactif

Contrôle de sécurité conçu pour permettre de remédier aux événements indésirables ou aux écarts par rapport à votre référence de sécurité. Pour plus d'informations, veuillez consulter la rubrique [Responsive controls](#) dans [Implementing security controls on AWS](#).

## retain

Voir [7 Rs.](#)

## se retirer

Voir [7 Rs.](#)

## Génération augmentée de récupération (RAG)

Technologie d'[IA générative](#) dans laquelle un [LLM](#) fait référence à une source de données faisant autorité qui se trouve en dehors de ses sources de données de formation avant de générer une réponse. Par exemple, un modèle RAG peut effectuer une recherche sémantique dans la base de connaissances ou dans les données personnalisées d'une organisation. Pour plus d'informations, voir [Qu'est-ce que RAG ?](#)

## rotation

Processus de mise à jour périodique d'un [secret](#) pour empêcher un attaquant d'accéder aux informations d'identification.

## contrôle d'accès aux lignes et aux colonnes (RCAC)

Utilisation d'expressions SQL simples et flexibles dotées de règles d'accès définies. Le RCAC comprend des autorisations de ligne et des masques de colonnes.

## RPO

Voir l'[objectif du point de récupération](#).

## RTO

Voir l'[objectif en matière de temps de rétablissement](#).

## runbook

Ensemble de procédures manuelles ou automatisées nécessaires à l'exécution d'une tâche spécifique. Elles visent généralement à rationaliser les opérations ou les procédures répétitives présentant des taux d'erreur élevés.

# S

## SAML 2.0

Un standard ouvert utilisé par de nombreux fournisseurs d'identité (IdPs). Cette fonctionnalité permet l'authentification unique fédérée (SSO), afin que les utilisateurs puissent se connecter AWS Management Console ou appeler les opérations de l' AWS API sans que vous ayez à créer un utilisateur dans IAM pour tous les membres de votre organisation. Pour plus d'informations sur la fédération SAML 2.0, veuillez consulter [À propos de la fédération SAML 2.0](#) dans la documentation IAM.

## SCADA

Voir [Contrôle de supervision et acquisition de données](#).

## SCP

Voir la [politique de contrôle des services](#).

## secret

Dans AWS Secrets Manager des informations confidentielles ou restreintes, telles qu'un mot de passe ou des informations d'identification utilisateur, que vous stockez sous forme cryptée. Il comprend la valeur secrète et ses métadonnées. La valeur secrète peut être binaire, une chaîne unique ou plusieurs chaînes. Pour plus d'informations, voir [Que contient le secret d'un Secrets Manager ?](#) dans la documentation de Secrets Manager.

## sécurité dès la conception

Une approche d'ingénierie système qui prend en compte la sécurité tout au long du processus de développement.

## contrôle de sécurité

Barrière de protection technique ou administrative qui empêche, détecte ou réduit la capacité d'un assaillant d'exploiter une vulnérabilité de sécurité. Il existe quatre principaux types de contrôles de sécurité : [préventifs](#), [détectifs](#), [réactifs](#) et [proactifs](#).

## renforcement de la sécurité

Processus qui consiste à réduire la surface d'attaque pour la rendre plus résistante aux attaques. Cela peut inclure des actions telles que la suppression de ressources qui ne sont plus requises, la mise en œuvre des bonnes pratiques de sécurité consistant à accorder le moindre privilège ou la désactivation de fonctionnalités inutiles dans les fichiers de configuration.

## système de gestion des informations et des événements de sécurité (SIEM)

Outils et services qui associent les systèmes de gestion des informations de sécurité (SIM) et de gestion des événements de sécurité (SEM). Un système SIEM collecte, surveille et analyse les données provenant de serveurs, de réseaux, d'appareils et d'autres sources afin de détecter les menaces et les failles de sécurité, mais aussi de générer des alertes.

## automatisation des réponses de sécurité

Action prédéfinie et programmée conçue pour répondre automatiquement à un événement de sécurité ou y remédier. Ces automatisations servent de contrôles de sécurité [détectifs ou réactifs](#) qui vous aident à mettre en œuvre les meilleures pratiques AWS de sécurité. Parmi les actions de réponse automatique, citons la modification d'un groupe de sécurité VPC, l'application de correctifs à une EC2 instance Amazon ou la rotation des informations d'identification.

## chiffrement côté serveur

Chiffrement des données à destination, par celui Service AWS qui les reçoit.

## Politique de contrôle des services (SCP)

Politique qui fournit un contrôle centralisé des autorisations pour tous les comptes d'une organisation dans AWS Organizations. SCPs définissez des garde-fous ou des limites aux actions qu'un administrateur peut déléguer à des utilisateurs ou à des rôles. Vous pouvez les utiliser SCPs comme listes d'autorisation ou de refus pour spécifier les services ou les actions autorisés ou interdits. Pour plus d'informations, consultez la section [Politiques de contrôle des services](#) dans la AWS Organizations documentation.

## point de terminaison du service

URL du point d'entrée pour un Service AWS. Pour vous connecter par programmation au service cible, vous pouvez utiliser un point de terminaison. Pour plus d'informations, veuillez consulter la rubrique [Service AWS endpoints](#) dans Références générales AWS.

## contrat de niveau de service (SLA)

Accord qui précise ce qu'une équipe informatique promet de fournir à ses clients, comme le temps de disponibilité et les performances des services.

## indicateur de niveau de service (SLI)

Mesure d'un aspect des performances d'un service, tel que son taux d'erreur, sa disponibilité ou son débit.

## objectif de niveau de service (SLO)

Mesure cible qui représente l'état d'un service, tel que mesuré par un indicateur de [niveau de service](#).

## modèle de responsabilité partagée

Un modèle décrivant la responsabilité que vous partagez en matière AWS de sécurité et de conformité dans le cloud. AWS est responsable de la sécurité du cloud, alors que vous êtes responsable de la sécurité dans le cloud. Pour de plus amples informations, veuillez consulter [Modèle de responsabilité partagée](#).

## SIEM

Consultez les [informations de sécurité et le système de gestion des événements](#).

## point de défaillance unique (SPOF)

Défaillance d'un seul composant critique d'une application susceptible de perturber le système.

## SLA

Voir le contrat [de niveau de service](#).

## SLI

Voir l'indicateur de [niveau de service](#).

## SLO

Voir l'objectif de [niveau de service](#).

## split-and-seed modèle

Modèle permettant de mettre à l'échelle et d'accélérer les projets de modernisation. Au fur et à mesure que les nouvelles fonctionnalités et les nouvelles versions de produits sont définies, l'équipe principale se divise pour créer des équipes de produit. Cela permet de mettre à l'échelle les capacités et les services de votre organisation, d'améliorer la productivité des développeurs et de favoriser une innovation rapide. Pour plus d'informations, consultez la section [Approche progressive de la modernisation des applications dans le AWS Cloud](#)

## SPOF

Voir [point de défaillance unique](#).

## schéma en étoile

Structure organisationnelle de base de données qui utilise une grande table de faits pour stocker les données transactionnelles ou mesurées et utilise une ou plusieurs tables dimensionnelles plus petites pour stocker les attributs des données. Cette structure est conçue pour être utilisée dans un [entrepôt de données](#) ou à des fins de business intelligence.

## modèle de figuier étrangleur

Approche de modernisation des systèmes monolithiques en réécrivant et en remplaçant progressivement les fonctionnalités du système jusqu'à ce que le système hérité puisse être mis hors service. Ce modèle utilise l'analogie d'un figuier de vigne qui se développe dans un arbre existant et qui finit par supplanter son hôte. Le schéma a été [présenté par Martin Fowler](#) comme un moyen de gérer les risques lors de la réécriture de systèmes monolithiques. Pour obtenir un exemple d'application de ce modèle, veuillez consulter [Modernizing legacy Microsoft ASP.NET \(ASMX\) web services incrementally by using containers and Amazon API Gateway](#).

## sous-réseau

Plage d'adresses IP dans votre VPC. Un sous-réseau doit se trouver dans une seule zone de disponibilité.

## contrôle de supervision et acquisition de données (SCADA)

Dans le secteur manufacturier, un système qui utilise du matériel et des logiciels pour surveiller les actifs physiques et les opérations de production.

## chiffrement symétrique

Algorithme de chiffrement qui utilise la même clé pour chiffrer et déchiffrer les données.

## tests synthétiques

Tester un système de manière à simuler les interactions des utilisateurs afin de détecter les problèmes potentiels ou de surveiller les performances. Vous pouvez utiliser [Amazon CloudWatch Synthetics](#) pour créer ces tests.

## invite du système

Technique permettant de fournir un contexte, des instructions ou des directives à un [LLM](#) afin d'orienter son comportement. Les instructions du système aident à définir le contexte et à établir des règles pour les interactions avec les utilisateurs.

# T

## balises

Des paires clé-valeur qui agissent comme des métadonnées pour organiser vos AWS ressources. Les balises peuvent vous aider à gérer, identifier, organiser, rechercher et filtrer des ressources. Pour plus d'informations, veuillez consulter la rubrique [Balisage de vos AWS ressources](#).

## variable cible

La valeur que vous essayez de prédire dans le cadre du ML supervisé. Elle est également qualifiée de variable de résultat. Par exemple, dans un environnement de fabrication, la variable cible peut être un défaut du produit.

## liste de tâches

Outil utilisé pour suivre les progrès dans un runbook. Liste de tâches qui contient une vue d'ensemble du runbook et une liste des tâches générales à effectuer. Pour chaque tâche générale, elle inclut le temps estimé nécessaire, le propriétaire et l'avancement.

## environnement de test

Voir [environnement](#).

## entraînement

Pour fournir des données à partir desquelles votre modèle de ML peut apprendre. Les données d'entraînement doivent contenir la bonne réponse. L'algorithme d'apprentissage identifie des modèles dans les données d'entraînement, qui mettent en correspondance les attributs des données d'entrée avec la cible (la réponse que vous souhaitez prédire). Il fournit un modèle de ML

qui capture ces modèles. Vous pouvez alors utiliser le modèle de ML pour obtenir des prédictions sur de nouvelles données pour lesquelles vous ne connaissez pas la cible.

### passerelle de transit

Un hub de transit réseau que vous pouvez utiliser pour interconnecter vos réseaux VPCs et ceux sur site. Pour plus d'informations, voir [Qu'est-ce qu'une passerelle de transit](#) dans la AWS Transit Gateway documentation.

### flux de travail basé sur jonction

Approche selon laquelle les développeurs génèrent et testent des fonctionnalités localement dans une branche de fonctionnalités, puis fusionnent ces modifications dans la branche principale. La branche principale est ensuite intégrée aux environnements de développement, de préproduction et de production, de manière séquentielle.

### accès sécurisé

Accorder des autorisations à un service que vous spécifiez pour effectuer des tâches au sein de votre organisation AWS Organizations et dans ses comptes en votre nom. Le service de confiance crée un rôle lié au service dans chaque compte, lorsque ce rôle est nécessaire, pour effectuer des tâches de gestion à votre place. Pour plus d'informations, consultez la section [Utilisation AWS Organizations avec d'autres AWS services](#) dans la AWS Organizations documentation.

### réglage

Pour modifier certains aspects de votre processus d'entraînement afin d'améliorer la précision du modèle de ML. Par exemple, vous pouvez entraîner le modèle de ML en générant un ensemble d'étiquetage, en ajoutant des étiquettes, puis en répétant ces étapes plusieurs fois avec différents paramètres pour optimiser le modèle.

### équipe de deux pizzas

Une petite DevOps équipe que vous pouvez nourrir avec deux pizzas. Une équipe de deux pizzas garantit les meilleures opportunités de collaboration possible dans le développement de logiciels.

## U

### incertitude

Un concept qui fait référence à des informations imprécises, incomplètes ou inconnues susceptibles de compromettre la fiabilité des modèles de ML prédictifs. Il existe deux types

d'incertitude : l'incertitude épistémique est causée par des données limitées et incomplètes, alors que l'incertitude aléatoire est causée par le bruit et le caractère aléatoire inhérents aux données. Pour plus d'informations, veuillez consulter le guide [Quantifying uncertainty in deep learning systems.](#)

## tâches indifférencierées

Également connu sous le nom de « levage de charges lourdes », ce travail est nécessaire pour créer et exploiter une application, mais qui n'apporte pas de valeur directe à l'utilisateur final ni d'avantage concurrentiel. Les exemples de tâches indifférencierées incluent l'approvisionnement, la maintenance et la planification des capacités.

## environnements supérieurs

Voir [environnement](#).

# V

## mise à vide

Opération de maintenance de base de données qui implique un nettoyage après des mises à jour incrémentielles afin de récupérer de l'espace de stockage et d'améliorer les performances.

## contrôle de version

Processus et outils permettant de suivre les modifications, telles que les modifications apportées au code source dans un référentiel.

## Appairage de VPC

Une connexion entre deux VPCs qui vous permet d'acheminer le trafic en utilisant des adresses IP privées. Pour plus d'informations, veuillez consulter la rubrique [Qu'est-ce que l'appairage de VPC ?](#) dans la documentation Amazon VPC.

## vulnérabilités

Défaut logiciel ou matériel qui compromet la sécurité du système.

# W

## cache actif

Cache tampon qui contient les données actuelles et pertinentes fréquemment consultées.

L'instance de base de données peut lire à partir du cache tampon, ce qui est plus rapide que la lecture à partir de la mémoire principale ou du disque.

## données chaudes

Données rarement consultées. Lorsque vous interrogez ce type de données, des requêtes modérément lentes sont généralement acceptables.

## fonction de fenêtre

Fonction SQL qui effectue un calcul sur un groupe de lignes liées d'une manière ou d'une autre à l'enregistrement en cours. Les fonctions de fenêtre sont utiles pour traiter des tâches, telles que le calcul d'une moyenne mobile ou l'accès à la valeur des lignes en fonction de la position relative de la ligne en cours.

## charge de travail

Ensemble de ressources et de code qui fournit une valeur métier, par exemple une application destinée au client ou un processus de backend.

## flux de travail

Groupes fonctionnels d'un projet de migration chargés d'un ensemble de tâches spécifique.

Chaque flux de travail est indépendant, mais prend en charge les autres flux de travail du projet. Par exemple, le flux de travail du portefeuille est chargé de prioriser les applications, de planifier les vagues et de collecter les métadonnées de migration. Le flux de travail du portefeuille fournit ces actifs au flux de travail de migration, qui migre ensuite les serveurs et les applications.

## VER

Voir [écrire une fois, lire plusieurs](#).

## WQF

Voir le [cadre AWS de qualification de la charge](#) de travail.

## écrire une fois, lire plusieurs (WORM)

Modèle de stockage qui écrit les données une seule fois et empêche leur suppression ou leur modification. Les utilisateurs autorisés peuvent lire les données autant de fois que nécessaire,

mais ils ne peuvent pas les modifier. Cette infrastructure de stockage de données est considérée comme [immuable](#).

## Z

exploit Zero-Day

Une attaque, généralement un logiciel malveillant, qui tire parti d'une [vulnérabilité de type « jour zéro »](#).

vulnérabilité « jour zéro »

Une faille ou une vulnérabilité non atténuée dans un système de production. Les acteurs malveillants peuvent utiliser ce type de vulnérabilité pour attaquer le système. Les développeurs prennent souvent conscience de la vulnérabilité à la suite de l'attaque.

invite Zero-Shot

Fournir à un [LLM](#) des instructions pour effectuer une tâche, mais aucun exemple (plans) pouvant aider à la guider. Le LLM doit utiliser ses connaissances pré-entraînées pour gérer la tâche. L'efficacité de l'invite zéro dépend de la complexité de la tâche et de la qualité de l'invite. Voir également les instructions [en quelques clics](#).

application zombie

Application dont l'utilisation moyenne du processeur et de la mémoire est inférieure à 5 %. Dans un projet de migration, il est courant de retirer ces applications.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.