AWS re:Invent

DAT317-R1

PostgreSQL options on AWS: Self-managed, managed, and serverless

Jim Mlodgenski

Principal Database Engineer

Amazon Web Services

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Agenda

Managed PostgreSQL on AWS

PostgreSQL deployment

Security

Monitoring and management

Building an application

Amazon Relational Database Service (Amazon RDS)

Managed relational database service with your choice of database engine

Amazon Aurora | MySQL | PostgreSQL | MariaDB | Microsoft SQL Server | Oracle

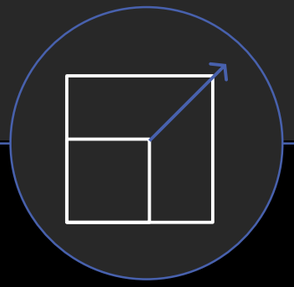
Easy to administer

Easily deploy and maintain hardware, OS and DB software; built-in monitoring

Available & durable

Automatic multi-AZ data replication; automated backup, snapshots, failover

Performant & scalable



Scale compute and storage with a few clicks; minimal downtime for your application

Secure & compliant

Data encryption at rest and in transit; industry compliance and assurance programs

Why PostgreSQL?

Open source

In active development for more than 30 years

Owned by a community, not a single company

Performance & scale

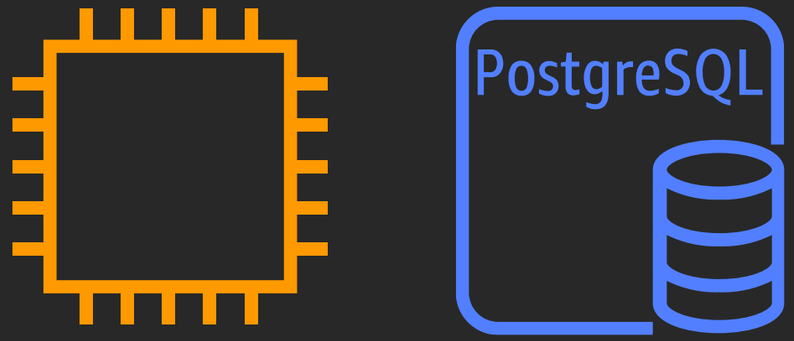
Extensive index support for a number of use cases

Parallel processing for complex queries

Native partitioning for large tables

Running PostgreSQL on AWS

Self-managed on Amazon Elastic Compute Cloud (Amazon EC2)

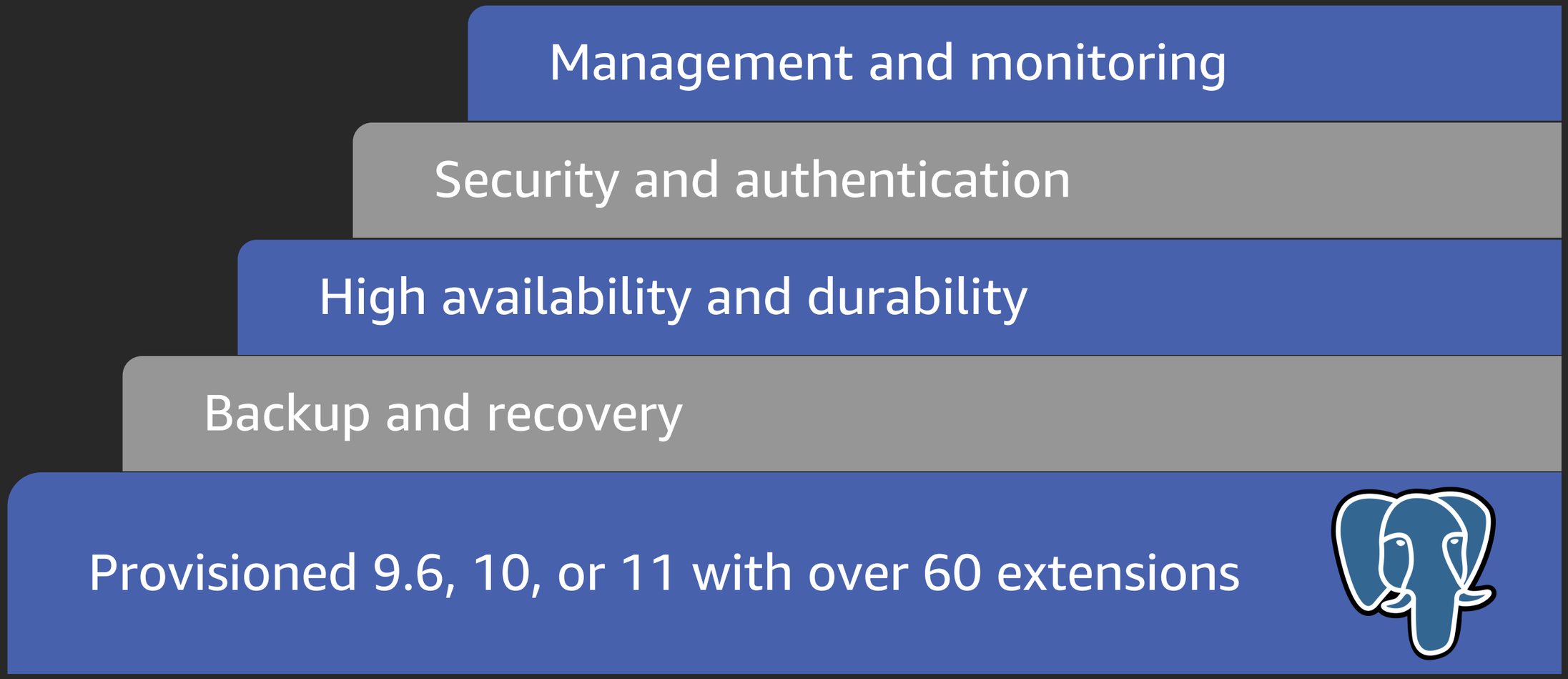


Fully-managed services

Amazon RDS for PostgreSQL

Aurora with PostgreSQL compatibility

Managed PostgreSQL on AWS



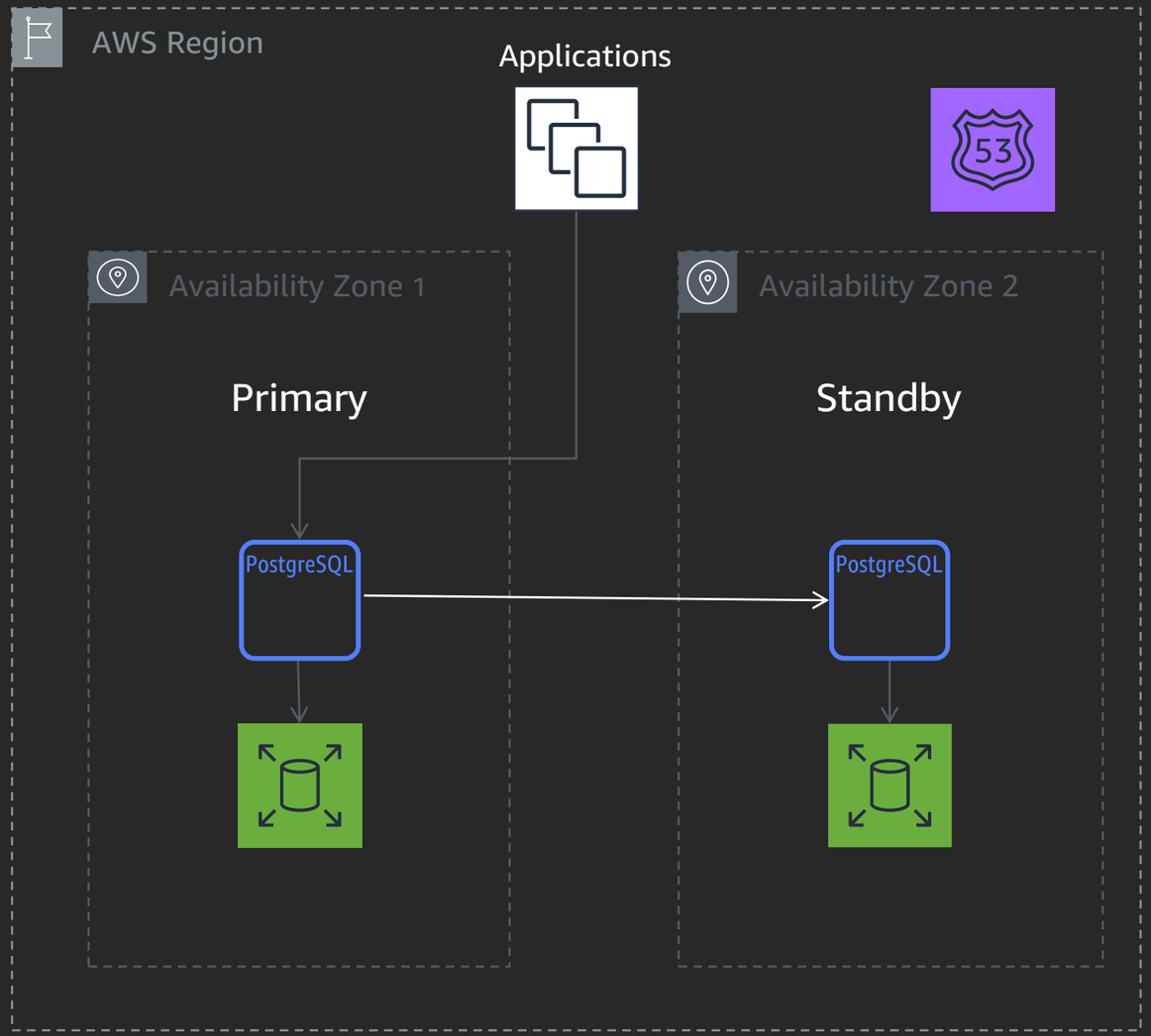
Amazon RDS for PostgreSQL

PostgreSQL community version with easy configuration and management

Supports 9.4, 9.5, 9.6, 10, and 11

PostgreSQL 12.0 can be tested on RDS Preview Region (New)

High availability across two availability zones



Amazon Aurora with PostgreSQL compatibility

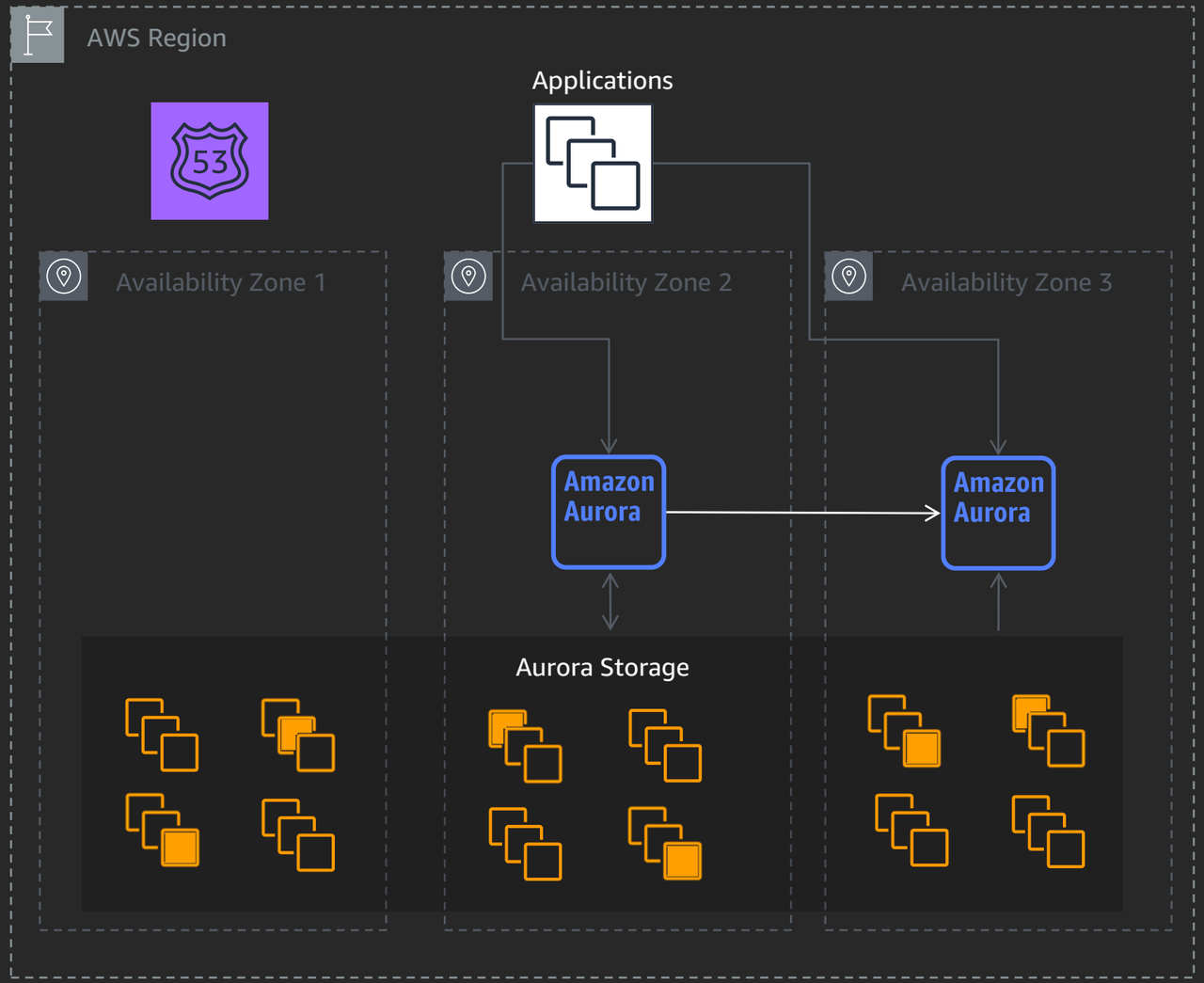
Built from the ground up to leverage AWS services

Supports 9.6, 10, and 11

Up to 2-3x better throughput on the same instance sizes

Highly available, durable, and fault-tolerant storage layer with 6-way persistence across 3 Availability Zones

Scalable up to 64 TiB



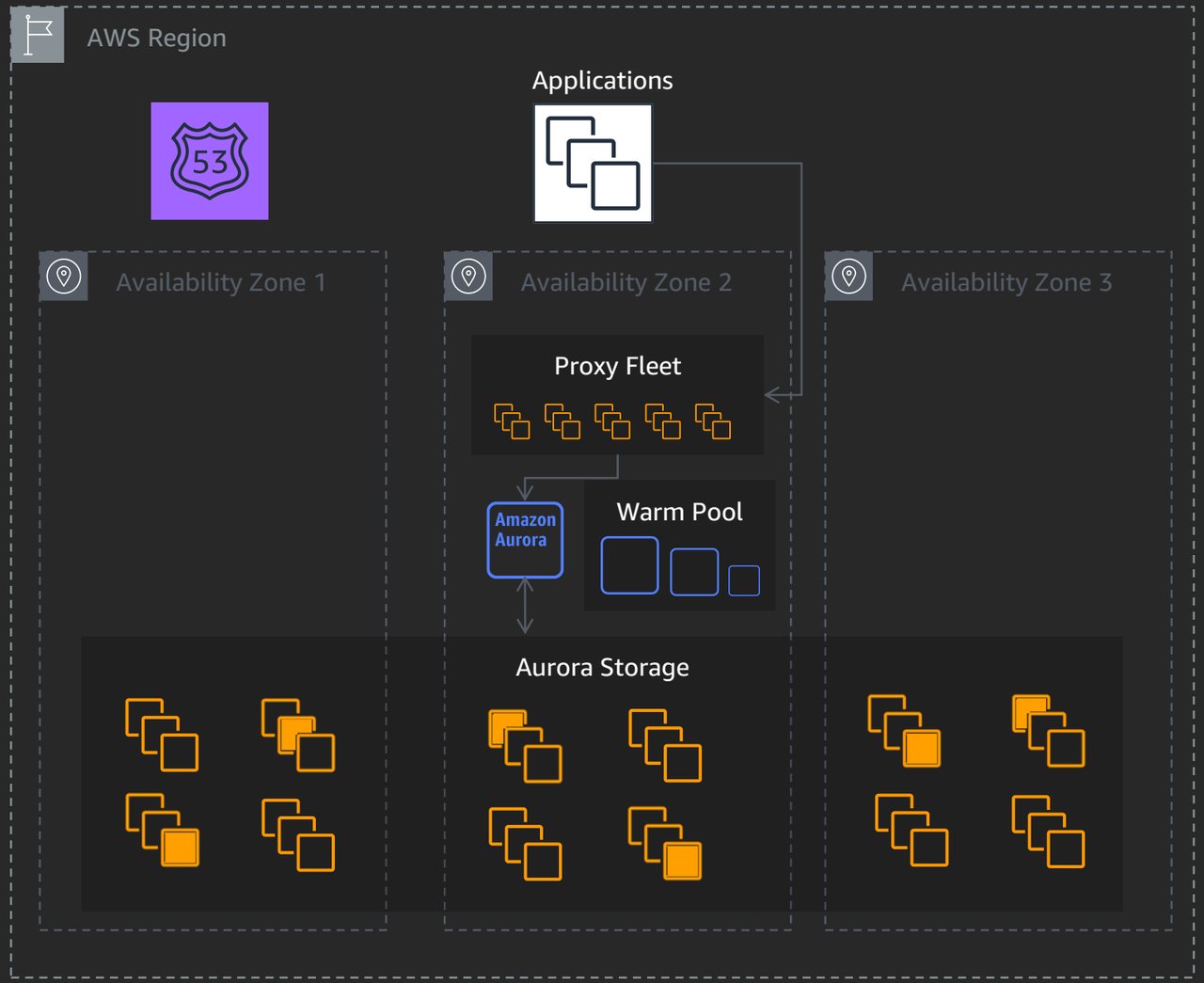
Amazon Aurora serverless

Supports 10.7

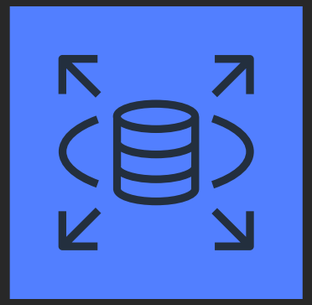
Scales up/down automatically

Highly available, durable, and fault-tolerant storage layer with 6-way persistence across 3 Availability Zones

Secure HTTP endpoint using the data API for running queries without a persistent database connection



Managed PostgreSQL on AWS



RDS PostgreSQL

Keeps in close lockstep with community releases

Available in small instance sizes

Aurora PostgreSQL

Up to 3x better throughput

Durability across 3 availability zones

Aurora Serverless

Auto scale from a paused instance to 768GB of RAM

Data access over HTTPS through the Data API

RDS PostgreSQL deployment

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon RDS for PostgreSQL: Instance types

T family

- **Burstable instances**
- 1 vCPU/1 GB RAM > 8 vCPU 32 GB RAM
- Moderate networking performance
- Good for smaller or variable workloads
- T2.micro is eligible for the AWS Free Tier
- T3 will enable unlimited mode—can burst above baseline for extra charge

M family

- **General purpose instances**
- 2 vCPU/8 GiB RAM > 64 vCPU 256 GiB RAM
- High-performance networking
- Good for running CPU-intensive workloads
- M5 offers up to 96 vCPU / 384 GiB RAM

R family

- **Memory-optimized instances**
- 2 vCPU/16 GiB RAM > 64 vCPU 488 GiB RAM
- High-performance networking
- Good for query-intensive workloads or high connection counts
- R5 offers up to 96 vCPU 768 GiB RAM

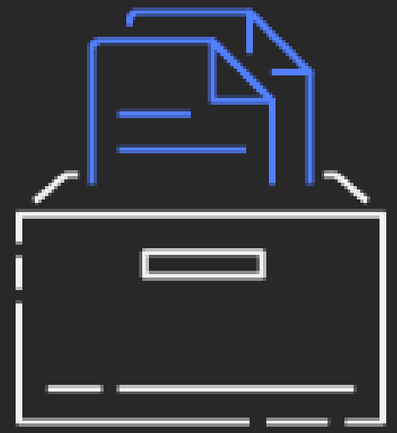
High-performance database storage

General purpose (GP2)

- SSD storage
- Auto scale up to 64 TiB
- Latency in milliseconds
- IOPS determined by volume size
- Affordable performance

Provisioned IOPS (IO1)

- SSD storage
- Auto scale up to 64 TiB
- Single digit millisecond latencies
- Maximum of 80 K IOPS
- Delivers within 10% of the IOPS performance, 99.9% of the time
- High performance and consistency



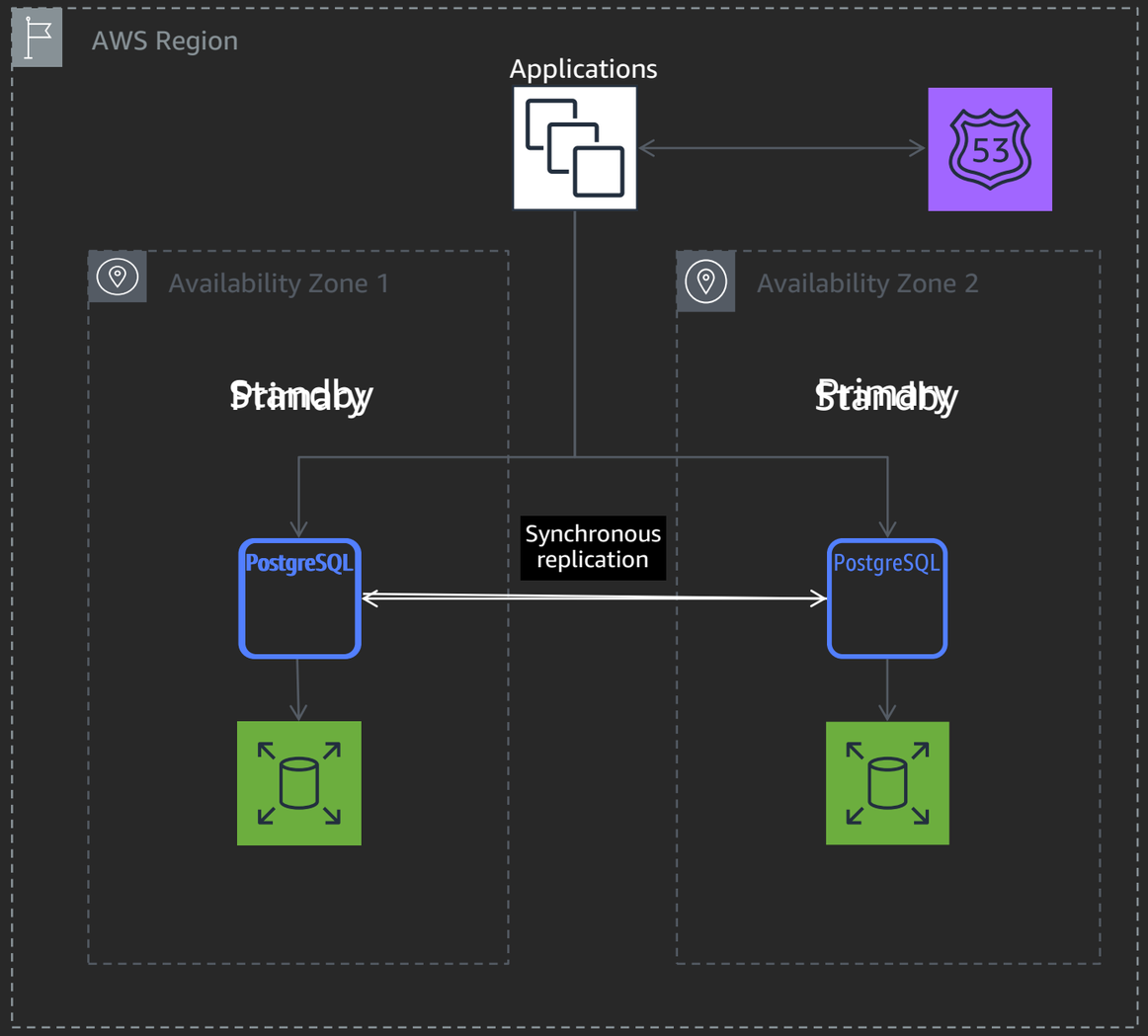
High availability for RDS PostgreSQL

Multi-AZ configurations provide a fault-tolerant solution for PostgreSQL

Each host manages a set of volumes with a full copy of the data

Redirection to the new primary instance is provided through DNS (Route 53)

Detects infrastructure issues, not database engine problems



RDS PostgreSQL backups

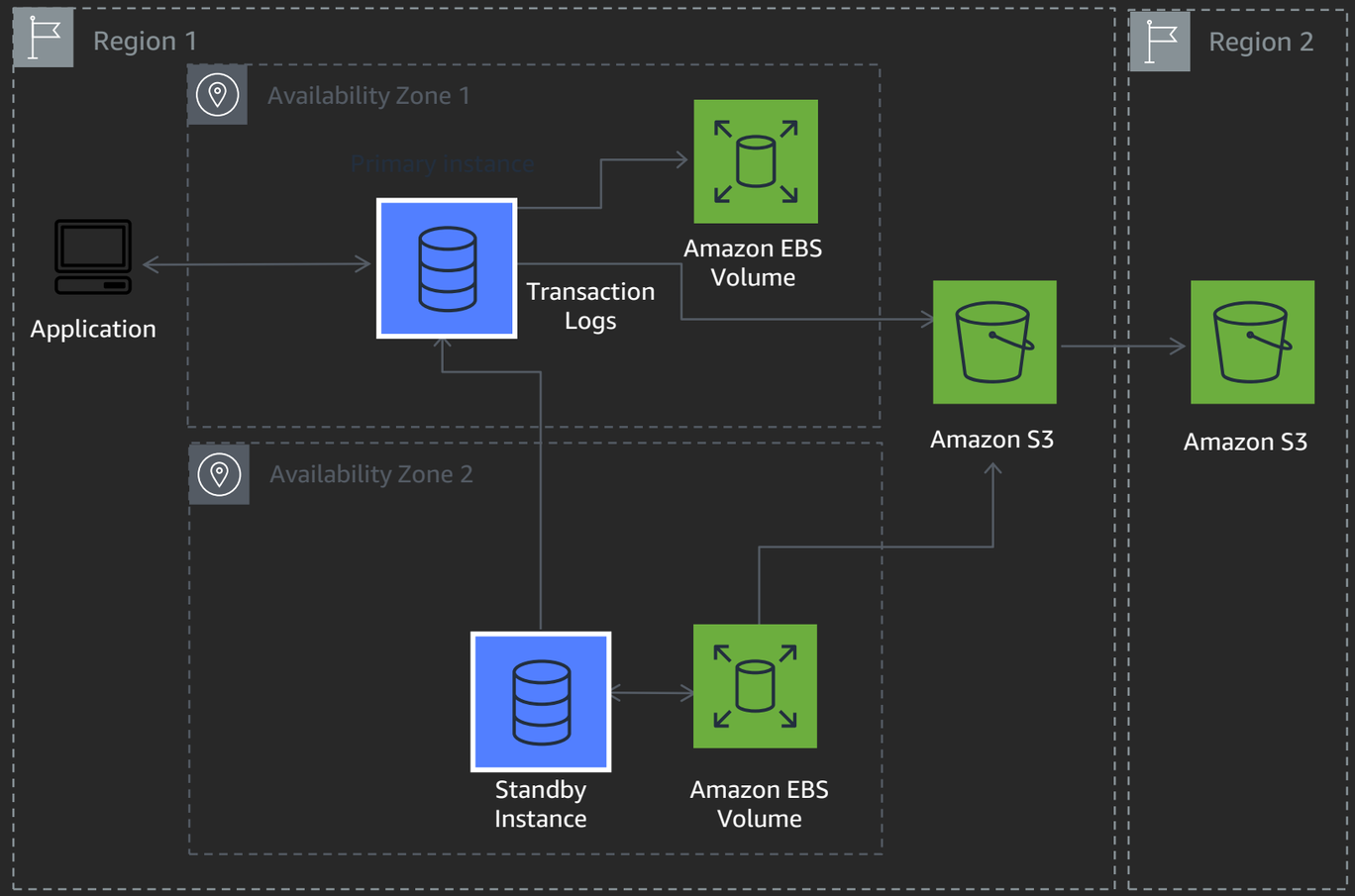
Two options: automated backups and manual snapshots

Amazon RDS leverages Amazon Elastic Block Store (Amazon EBS) snapshots

Snapshots are stored in Amazon Simple Storage Service (Amazon S3)

Transaction logs are stored every five minutes in Amazon S3 to support PITR

Snapshots can be copied across regions or with other accounts



RDS PostgreSQL security

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

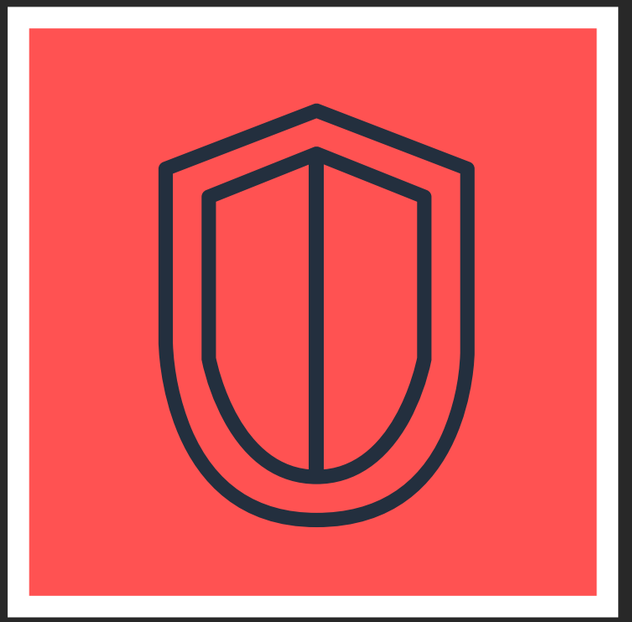
Security

Encryption to ensure unauthorized users cannot see your data

- In transit using SSL
- At rest using AWS Key Management Service (AWS KMS)

Authentication to ensure a user is known and trusted

Auditing to verify the trusted user is only accessing permitted content



Security, Identity, and Compliance

Network encryption

An SSL certificate is available on RDS instances

- Used to encrypt network traffic
- Also used to verify the endpoint to guard against spoofing attacks

By default, SSL is optional

- Set rds.force_ssl to 1 to force SSL

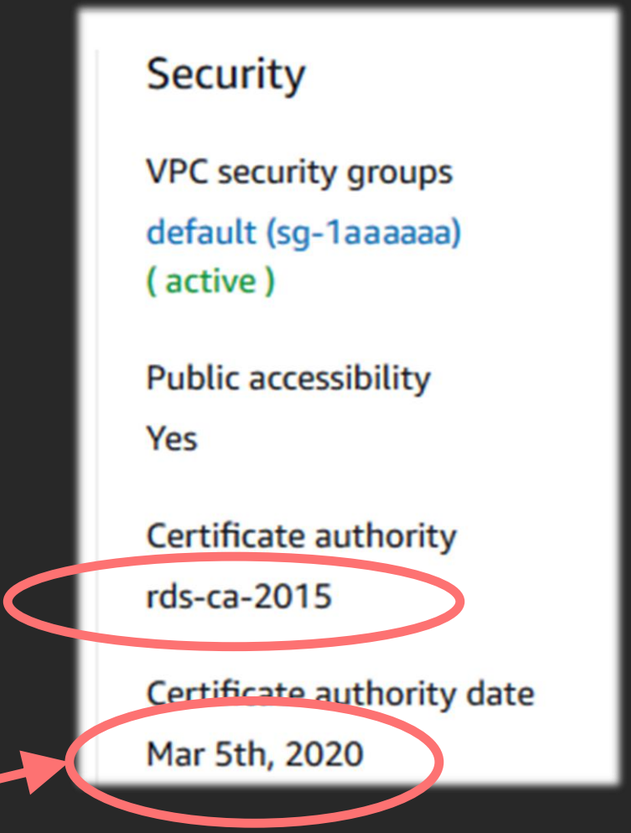
The client requests the type of SSL connection

```
$ psql -h $RDSHOST \
 "user=jim dbname=postgres sslmode=require"

postgres=>  SELECT ssl,version, bits
              FROM pg_stat_ssl
             WHERE pid = pg_backend_pid();
 ssl | version | bits
-----+---------+------
 t   | TLSv1.2 |  256
(1 row)
```

Network encryption

The SSL connection can be verified by using a stricter SSL mode

```
psql -h $RDSHOST \
"user=postgres dbname=postgres \
 sslmode=verify-full \
 sslrootcert=rds-ca-2015-root.pem"
```



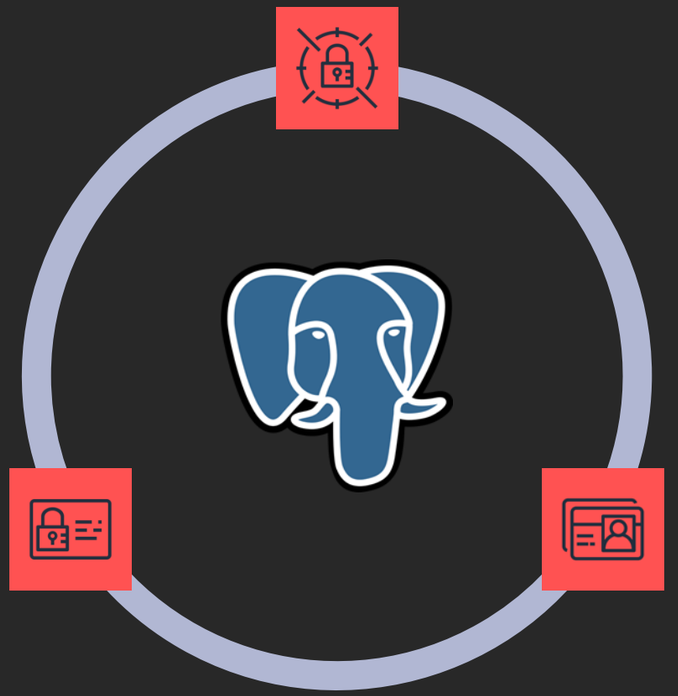
Certificates need to be updated

Authentication

Ensures only authorized users connect to the database

Several authentication mechanisms available based on need and environment

- AWS Identity and Access Management (IAM)
- Kerberos (New)
- PostgreSQL local accounts



AWS Identity and Access Management (IAM)

AWS Identity and Access Management

PostgreSQL authentication is managed externally using IAM

Authentication tokens are used to validate the user

- Tokens have a lifetime of 15 minutes

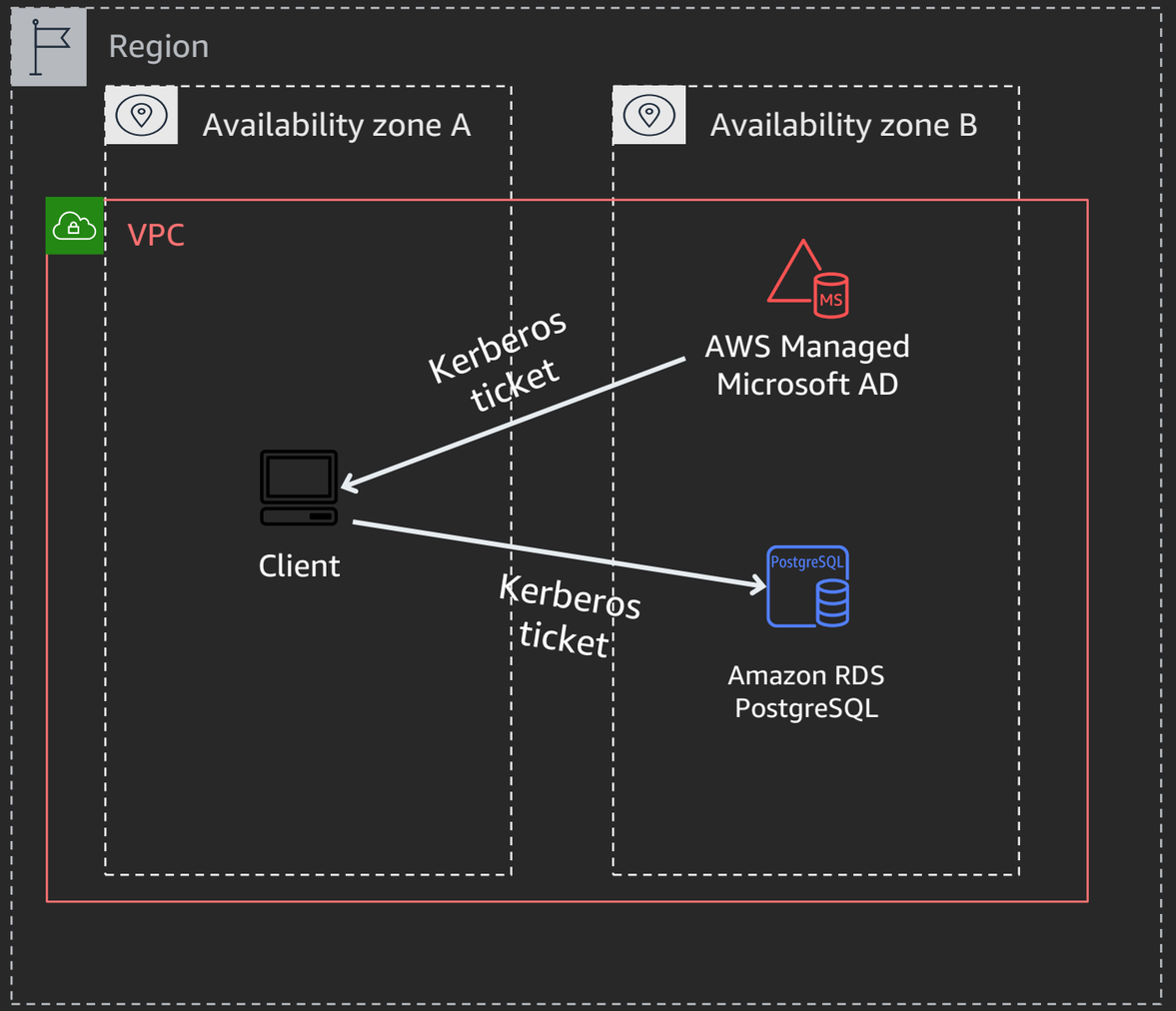
Connections per second are limited by the instance type of the database

Kerberos authentication

Provides a fast and secure authentication mechanism

Uses AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) as the authentication source

The client must be part of the AWS Managed Microsoft AD domain



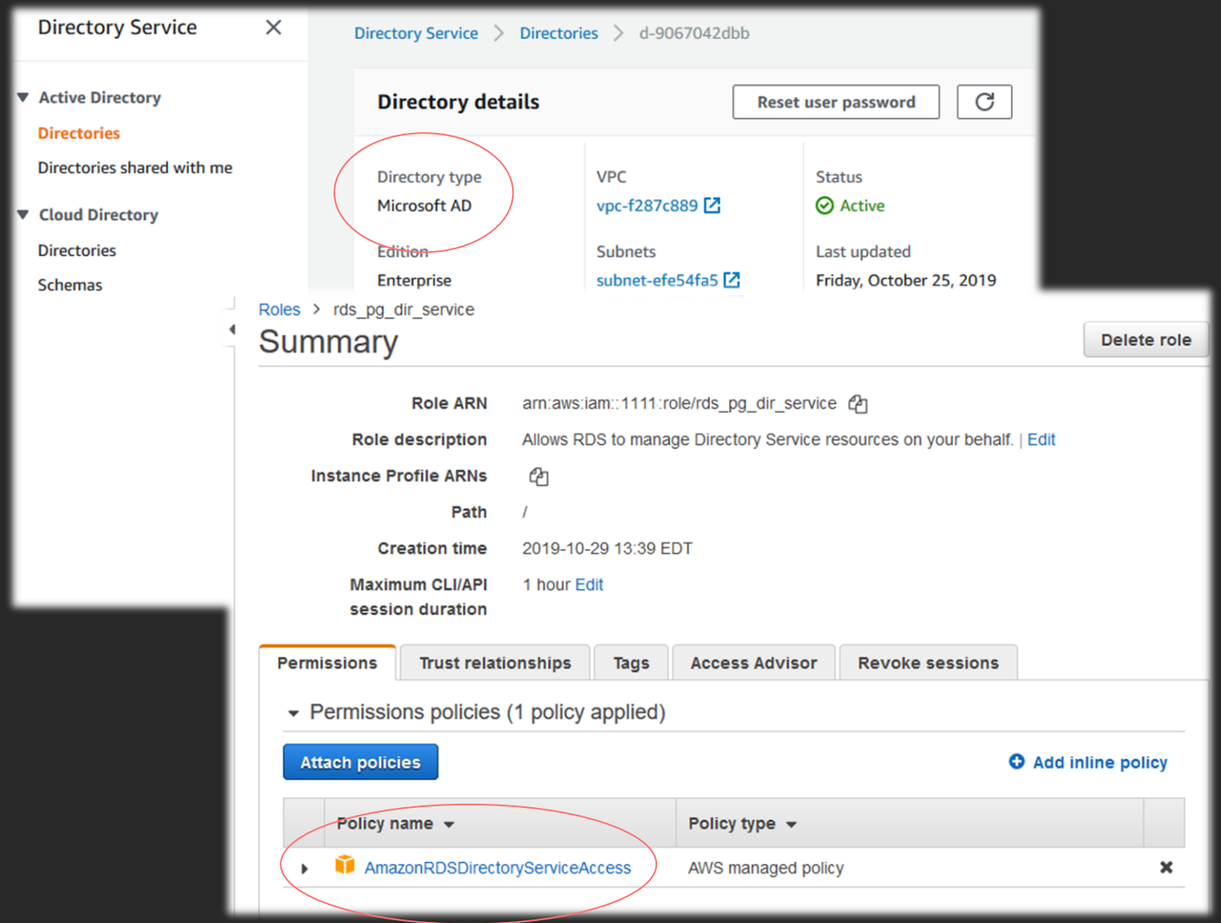
Kerberos authentication

Create a directory

It needs to be an AWS Managed Microsoft AD

Create an IAM role so RDS can access the directory

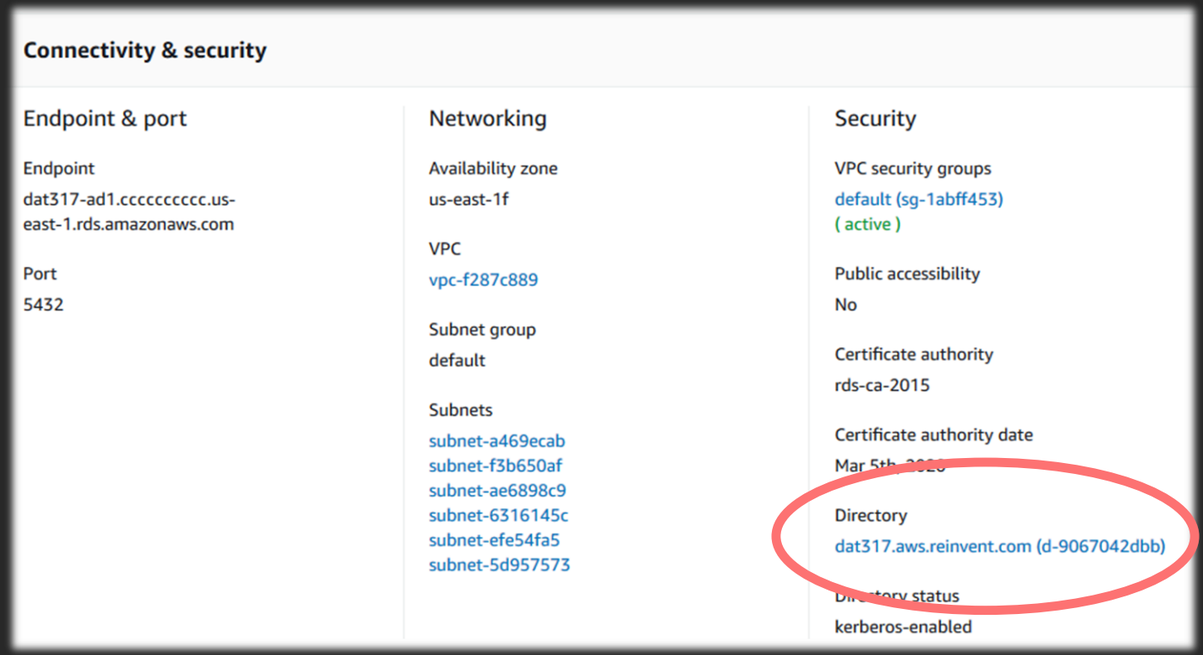
The role needs AmazonRDSDirectoryServiceAccess privileges



Kerberos authentication

Connect the directory and role to the database instance

```
aws rds modify-db-instance \
  --db-instance-identifier dat317-ad \
  --domain d-9067042dbb  \
  --domain-iam-role-name rds_pg_ad_service
```



Kerberos authentication

Create a role with login privileges that matches the domain user

ALL CAPS

```
CREATE ROLE
"jim@DAT317.AWS.REINVENT.COM"
WITH LOGIN;
```

Case matters

```
CREATE ROLE
“Kevin@DAT317.AWS.REINVENT.COM"
WITH LOGIN;
```

Add the login role to the rds_ad role

```
GRANT rds_ad TO
"jim@DAT317.AWS.REINVENT.COM";

GRANT rds_ad TO
“Kevin@DAT317.AWS.REINVENT.COM";
```

Object security

Granting privileges on objects directly to the Kerberos role requires the full name in quotes

```
GRANT SELECT ON events TO
"Kevin@DAT317.AWS.REINVENT.COM";
```

Adding an additional role makes managing privileges simpler

```
CREATE ROLE ad_users;

GRANT SELECT ON events TO ad_users;

GRANT ad_users TO
"Kevin@DAT317.AWS.REINVENT.COM";
```

Repeatedly creating users

```
CREATE OR REPLACE PROCEDURE create_ad_user(
      p_user text,
      p_domain text DEFAULT 'DAT317.AWS.REINVENT.COM')
AS $$
DECLARE
  v_user text;
  v_role text;
BEGIN
  v_user := p_user || '@' || upper(p_domain);
  IF v_user IS NULL THEN
    RAISE EXCEPTION 'The user or domain can
                     not be null';
  END IF;

  IF length(v_user) > 63 THEN
    RAISE EXCEPTION 'The domain username (%)
        is too long. Usernames are limited to
        63 characters', v_user;
  END IF;

  EXECUTE format('CREATE ROLE %I WITH LOGIN',
         v_user);
  EXECUTE format('GRANT rds_ad TO %I', v_user);
  EXECUTE format('GRANT ad_users TO %I', v_user);
  RAISE NOTICE 'Created AD user %', v_user;

END $$
LANGUAGE plpgsql
SECURITY DEFINER;
```

Local authentication

Stores the login information in the local system catalogs for the instance

Default method uses MD5 with a salt for authentication

Fast authentication times but no control over password complexity

```
CREATE USER rptuser
  PASSWORD ‘5trOngP@sswOrd!'
  VALID UNTIL '2019-12-31 12:15';

ALTER USER rptuser
   PASSWORD ‘weakpassword’;
```

Restricted password management

Limits the users capable of changing passwords and expiration times

By default, it is disabled

- Set rds. restrict_password_commands to 1 to enable the feature

Only members of the rds_password role can make changes

```
ALTER USER rptuser
   PASSWORD ‘weakpassword’;

ERROR:  must be a member of rds_password
        to alter passwords
```

AWS Secrets Manager

Lifecycle management for secrets such as database passwords

Safely rotates passwords

Built-in integration with RDS and Aurora PostgreSQL

With restricted password management, all passwords can be controlled by Secrets Manager

AWS Secrets Manager

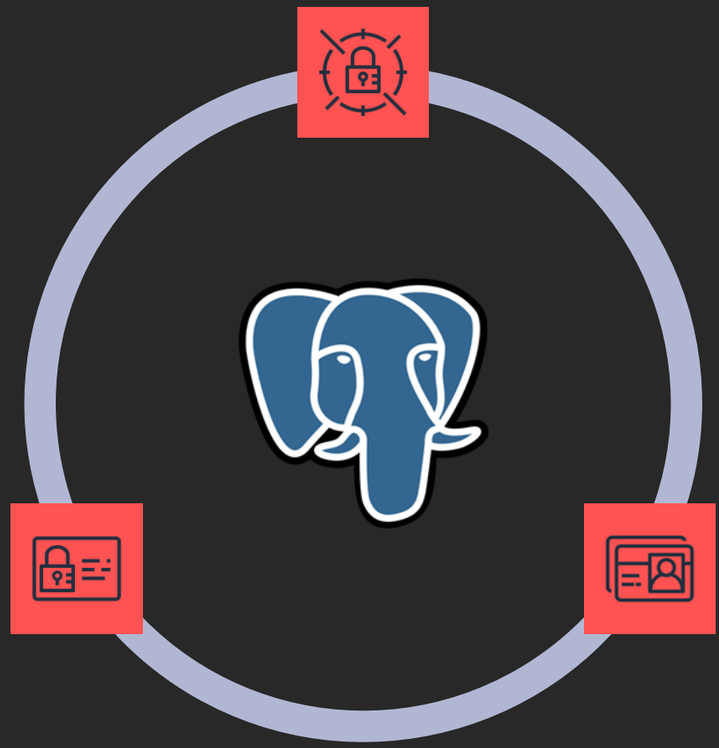
Choosing an authentication method

Multiple methods can be used

Use Kerberos in active directory environments

Use IAM for high-security environments with low to moderate connection requirements

Use local authentication with Secrets Manager for high connection requirements



Monitoring and management

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Monitoring

Enhanced monitoring for Amazon RDS

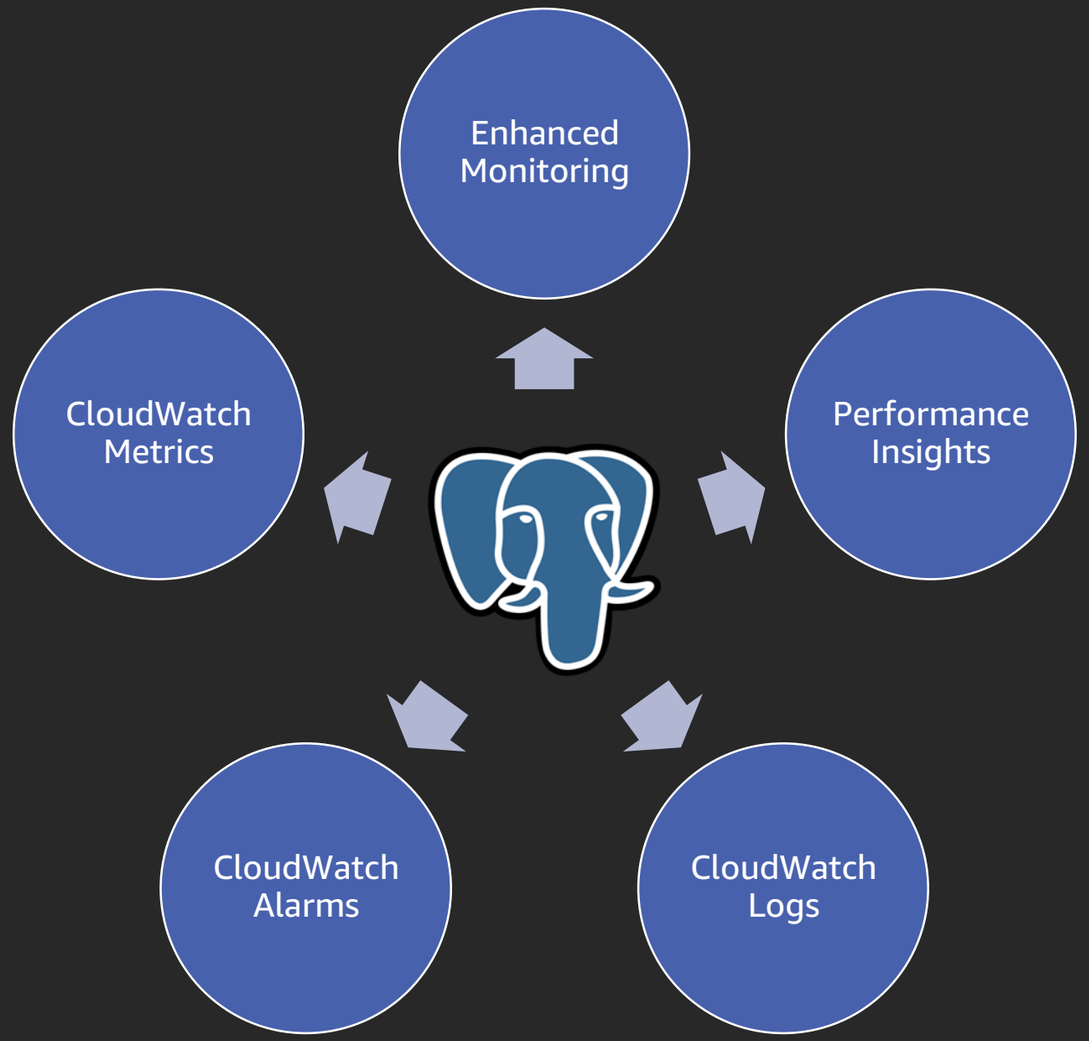
Access to over 50 CPU, memory, file system, and disk I/O metrics

Amazon CloudWatch Metrics

Displayed in the RDS console or in personalized CloudWatch dashboards

Amazon CloudWatch alarms

Alarms triggered based on metric values crossing configurable thresholds



Performance insights

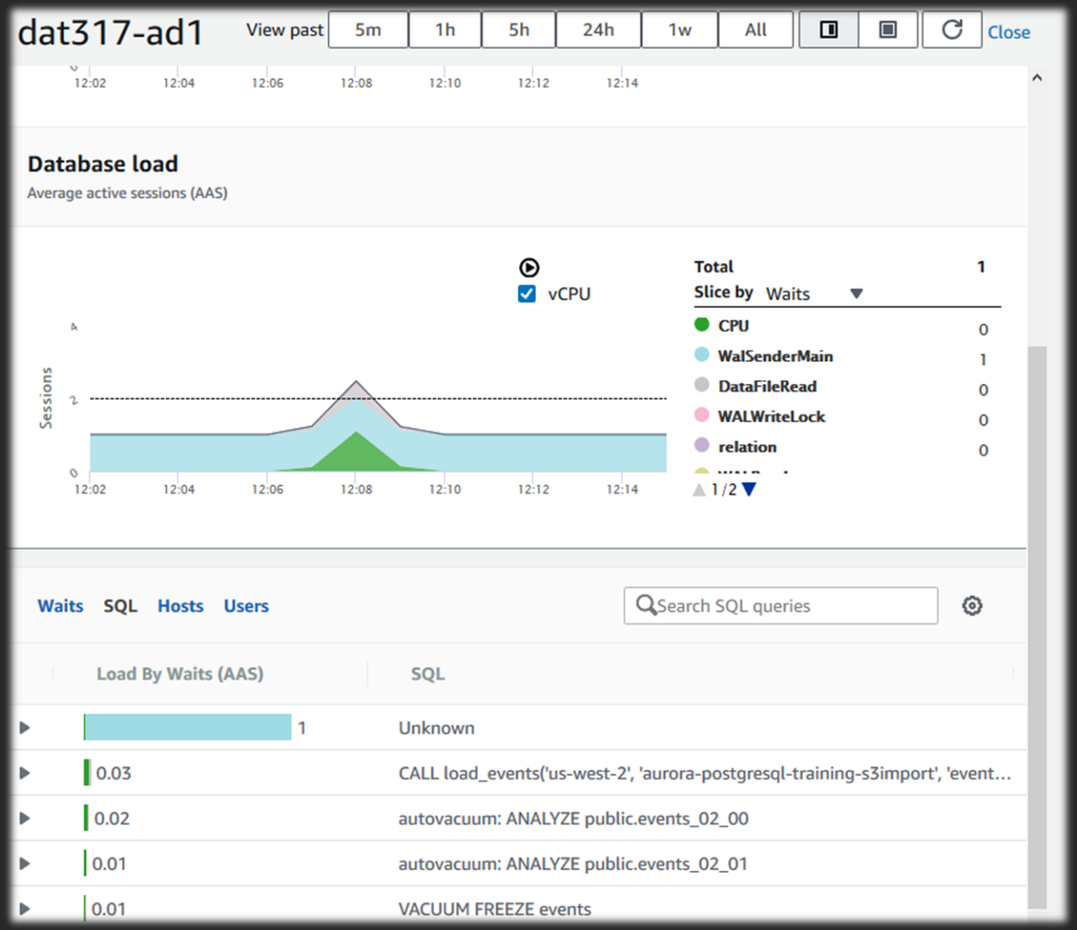
Database performance tuning and monitoring feature

Dashboard shows database load over time

Identifies source of bottlenecks

- Sort by top SQL
- Slice by host, user, wait events

Store up to 2 years of metrics



Amazon CloudWatch logs

PostgreSQL logs pushed into CloudWatch in real time

Create alarms on the content of the logs

- Long lock times
- Failed connections
- Audit violations

Perform analysis on logs across database instances

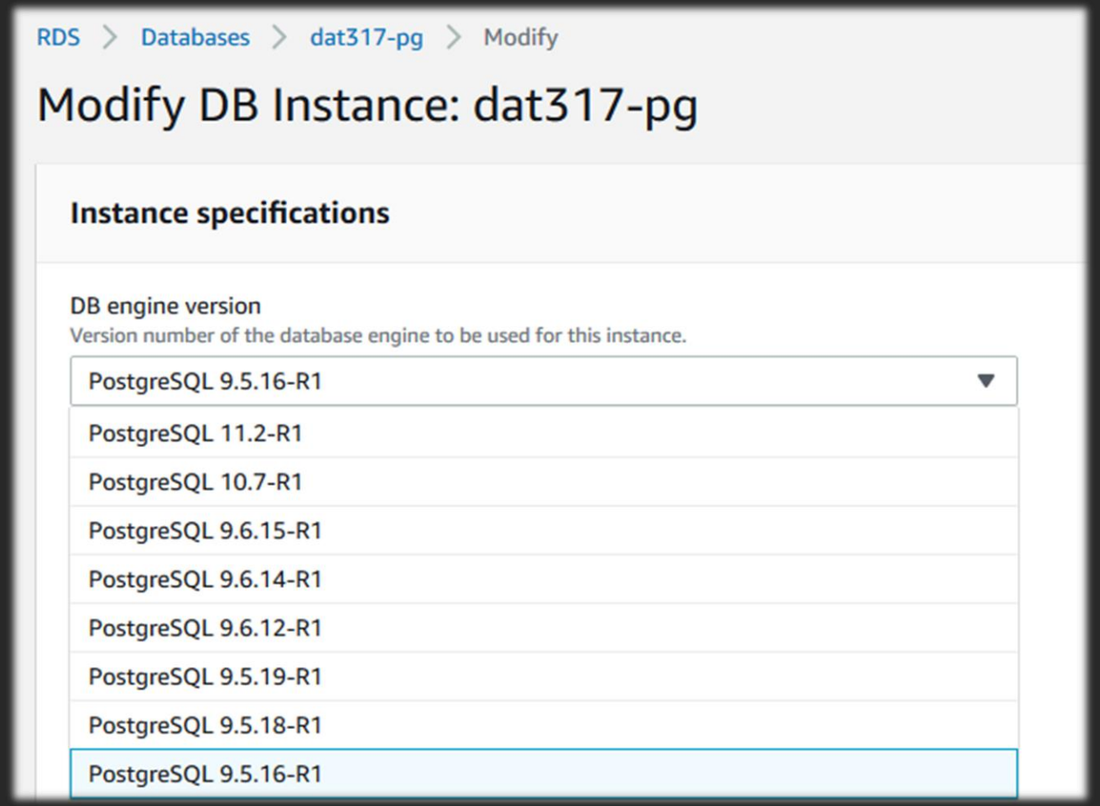
Upgrades

Minor version upgrades

- Patches to the binaries
- No new functionality
- May contain important security fixes

Major version upgrades

- Tracks the community yearly release cycle
- Introduces new functionality
- May change system catalogs and page formats

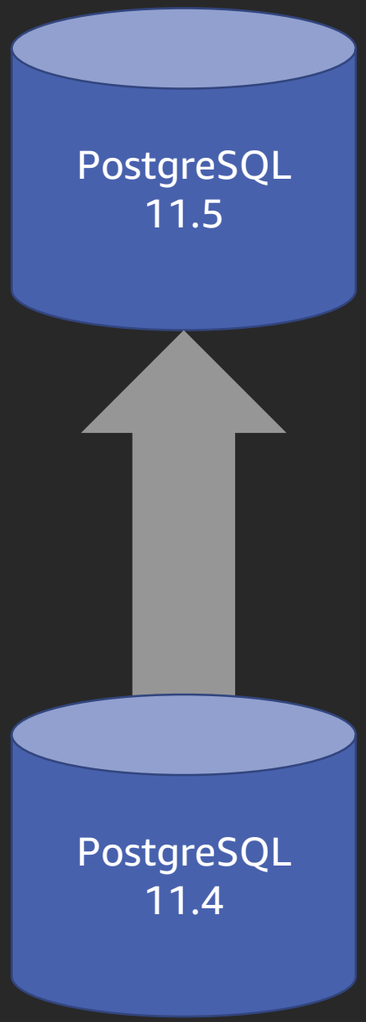


Minor version upgrades

The minor version upgrade process typically takes less than 5 to 10 minutes

- Normal shutdown of the instance
  - The time can be extended with long in-flight transactions
- Replace version binaries
- Start the instance

Can be performed manually or automatically

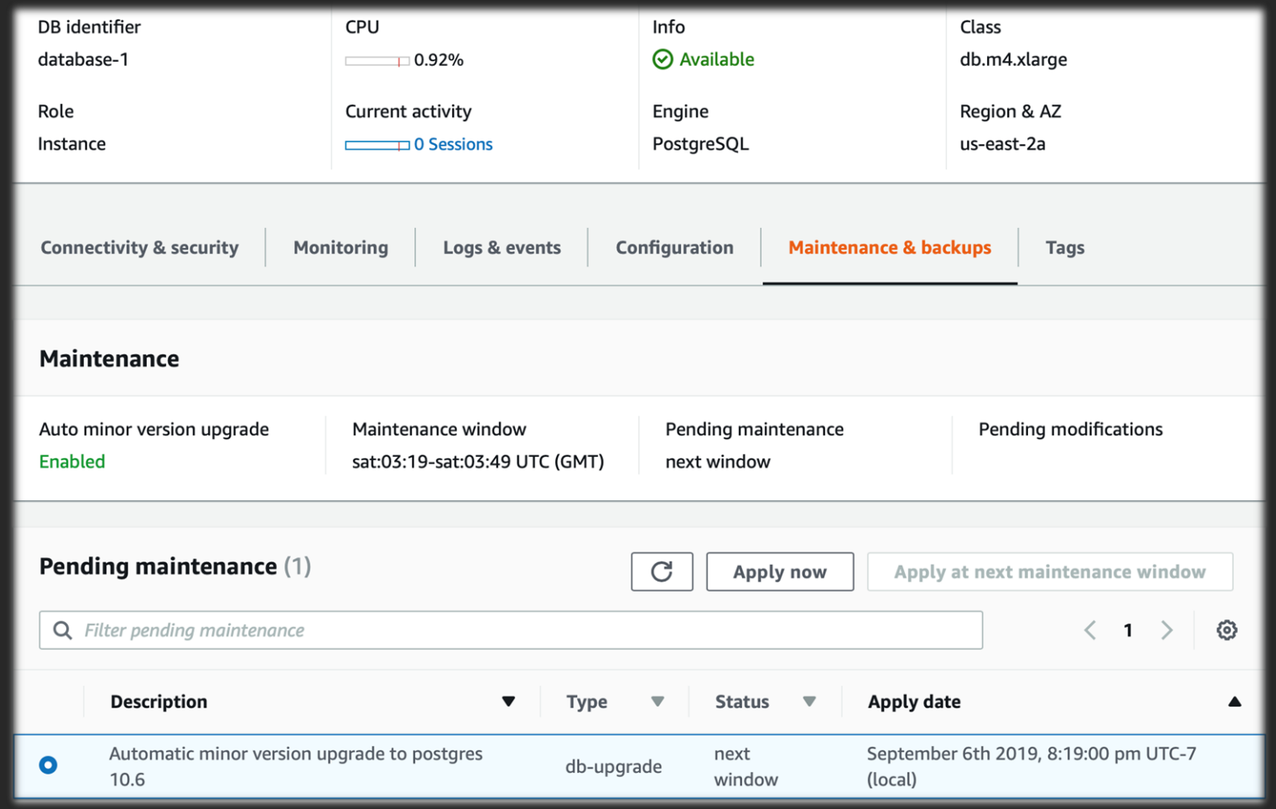


Auto minor version upgrade (AmVU)

Scheduled when the newer minor version is marked as preferred

By default, AmVU is ON

If a database is on a lower version, the instance is upgraded to the preferred version in the next maintenance window



Major version upgrades

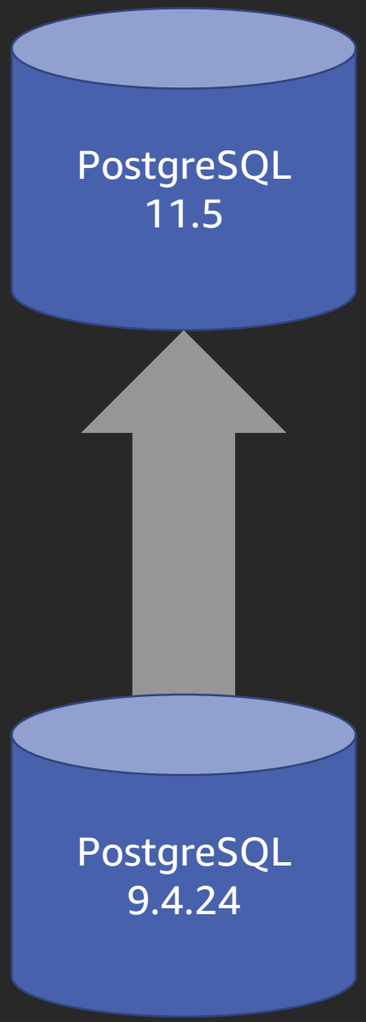
Upgrades can skip major versions with multi major version upgrade (New)

The time needed to upgrade depends more on the number of database objects than the database size

Snapshots taken before and after the upgrade

Statistics are not upgraded so ANALYZE is needed on the new version

A new parameter group is needed for the new version



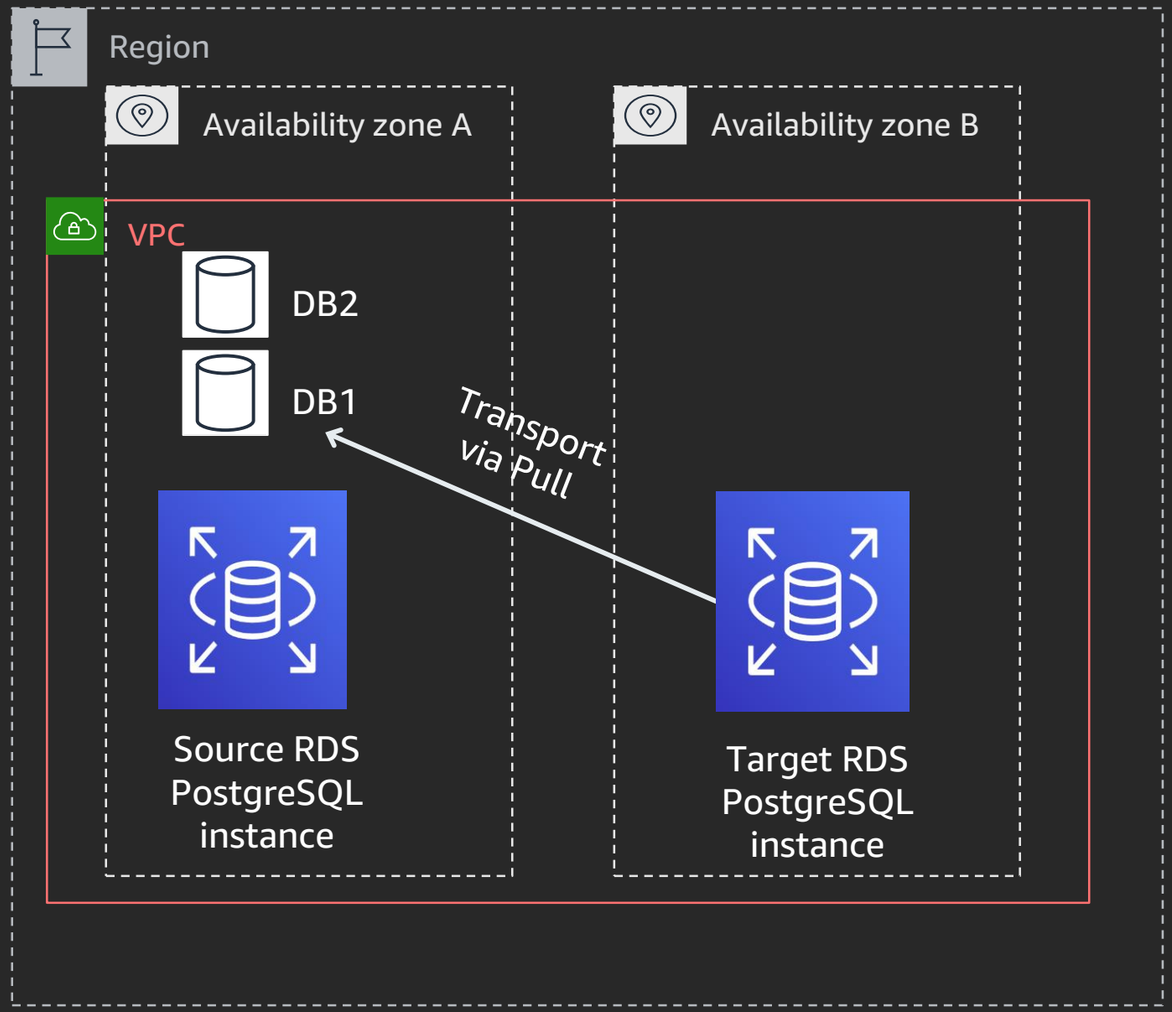
Transportable databases

Fast method of data export and import between RDS instances with minimal downtime

Supported on RDS PostgreSQL versions 11.5 and 10.10 onwards

Available through the pg_transport extension

Source database is made read-only during transport



Building a PostgreSQL application

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

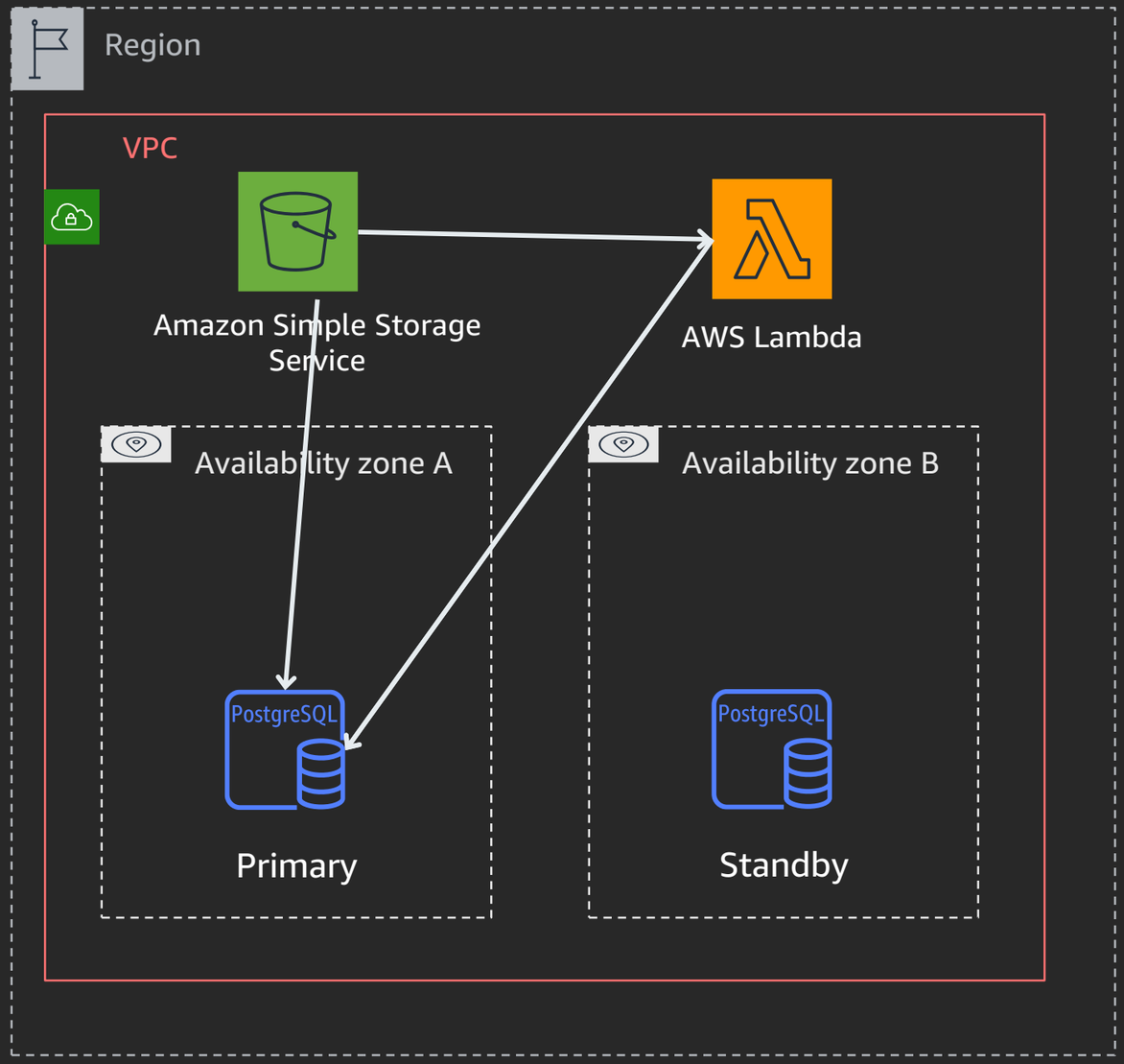
aws

Application requirements

Files containing JSON documents need to be regularly loaded from an Amazon Simple Storage Service (S3) bucket

Users need to run queries on the documents, along with other transactional data

User access needs to be secure and highly available



Amazon S3 import

Loads a file from Amazon S3 directly into a PostgreSQL table using the COPY syntax

Available using the aws_s3 extension

Two methods of security access

- AWS Identity and Access Management (IAM)
- Security credentials

Amazon Simple Storage Service

Amazon S3 import: IAM access

Create an IAM policy to allow access to the S3 bucket

Create an IAM role for RDS

Attach the role to the policy

Add the role to the RDS instance

```
{ {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::your-s3-bucket-arn",
        "arn:aws:s3:::your-s3-bucket-arn/*"
      ]
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
aws iam attach-role-policy \
  --policy-arn your-policy-arn \
  --role-name rds-s3-integration-role
```

```
aws rds add-role-to-db-instance \
  --db-instance-identifier mydbinstance \
  --feature-name s3Import \
  --role-arn your-role-arn  \
  --region us-west-2
```

Amazon S3 import: Using security credentials

Instead of IAM policies, security credentials can be used

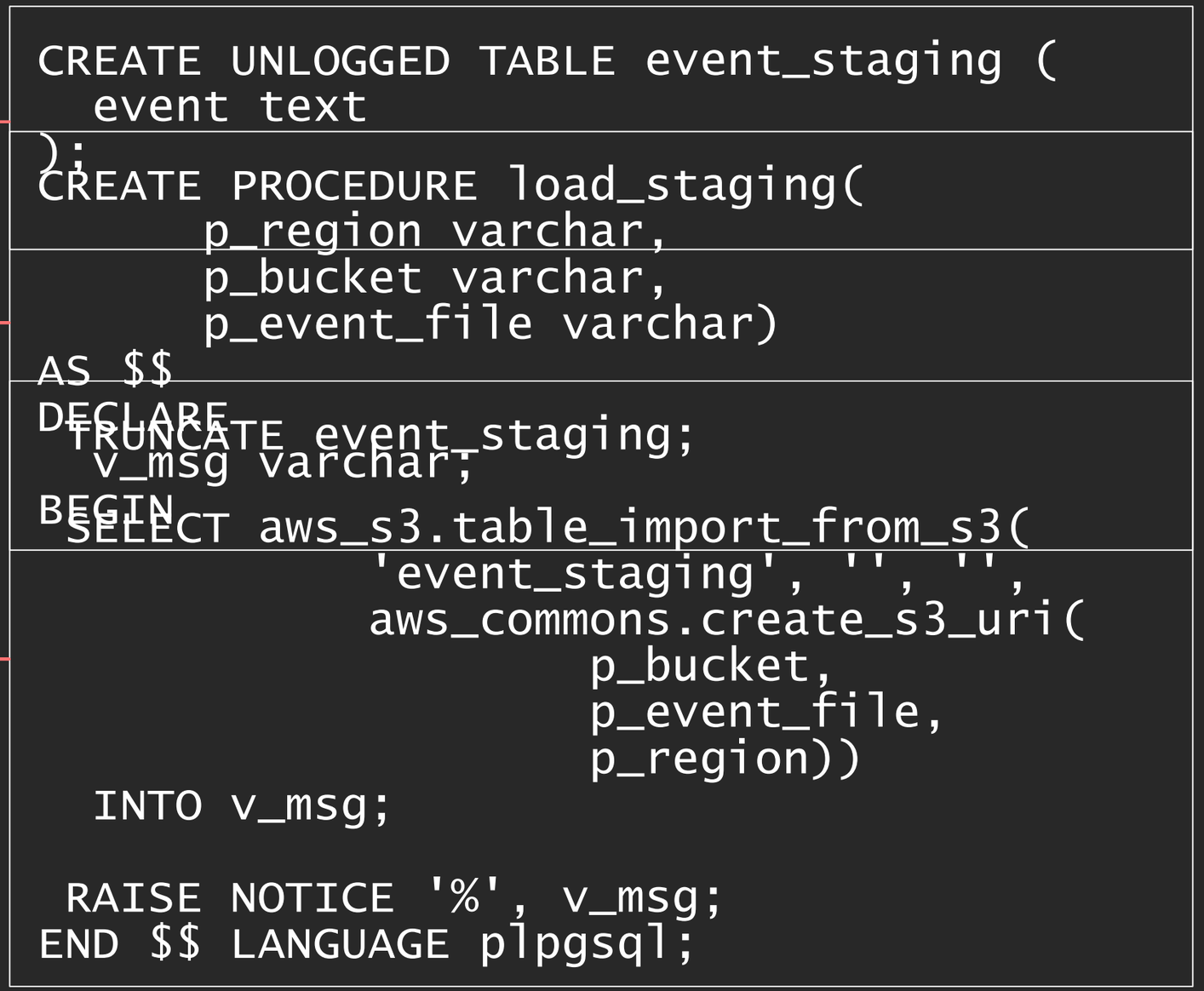
The S3 import function takes an optional parameter for the AWS credentials

Secure key management is the responsibility of the application

```
SELECT aws_commons.create_aws_credentials(
'AKIAIOSFODNN7EXAMPLE',
'walrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY‘,
'');
```

Loading the data

```
CREATE UNLOGGED TABLE event_staging (
  event text
);


CREATE PROCEDURE load_staging(
      p_region varchar,
      p_bucket varchar,
      p_event_file varchar)
AS $$
DECLARE
  v_msg varchar;
BEGIN
  TRUNCATE event_staging;

  SELECT aws_s3.table_import_from_s3(
           'event_staging', '', '',
           aws_commons.create_s3_uri(
                  p_bucket,
                  p_event_file,
                  p_region))
  INTO v_msg;

  RAISE NOTICE '%', v_msg;
END $$ LANGUAGE plpgsql;
```



Creating the partitioned table

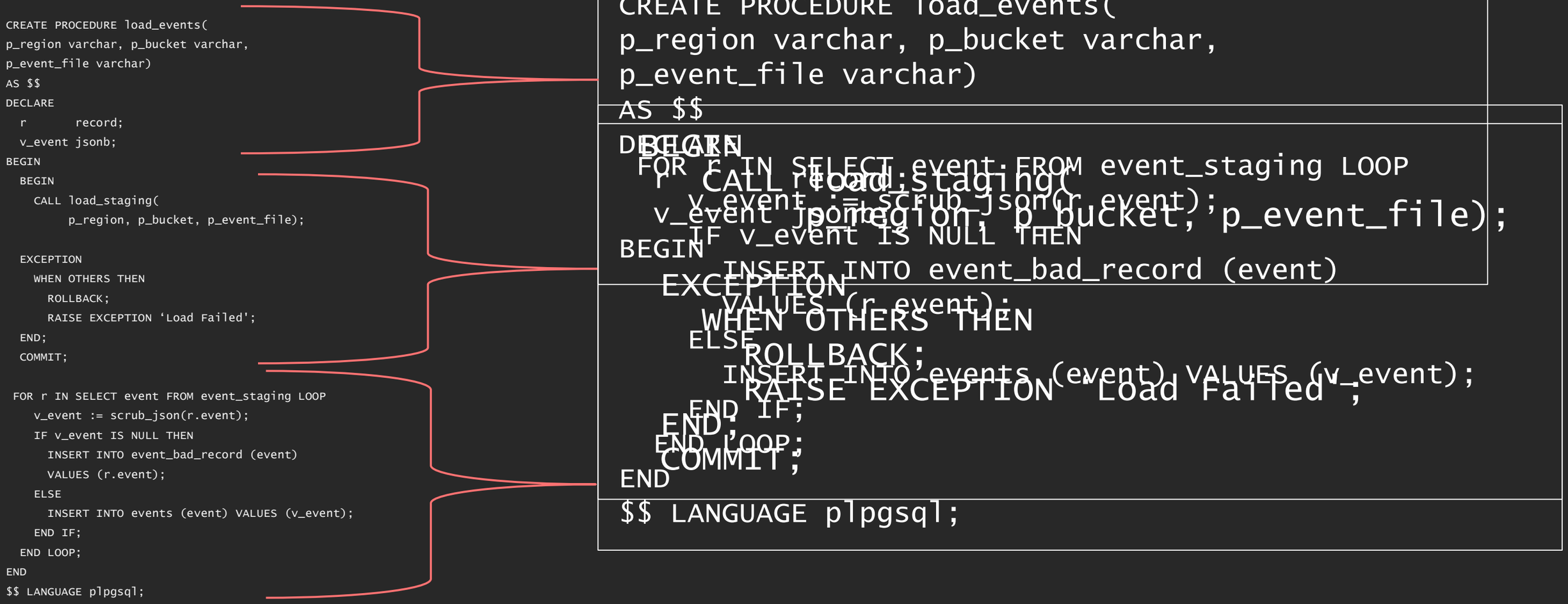
Split a large table into smaller, more manageable pieces

Performance can increase by pruning away large amounts of data

Simplifies data management by allowing operations to be performed on a subset of the table

```
CREATE TABLE events (
  id    uuid DEFAULT uuid_generate_v4(),
  event jsonb
) PARTITION BY HASH (id);

CREATE TABLE events_02_00
  PARTITION OF events
  FOR VALUES
  WITH (MODULUS 2, REMAINDER 0);

CREATE TABLE events_02_01
  PARTITION OF events
  FOR VALUES
  WITH (MODULUS 2, REMAINDER 1);

CREATE UNIQUE INDEX events_id_idx
       ON events (id);
CREATE INDEX events_event_gin_idx
       ON EVENTS USING gin(event);
```

Loading the table



Scrubbing the data

```
CREATE FUNCTION scrub_json(str text)
  RETURNS json AS
$$
  try {
    // Change all of the single quotes
    // to double quotes to be valid JSON

    str = str.replace(/'/g, '"');

    var obj = JSON.parse(str);
  } catch (e) {
    return null;
  }
  return obj;
$$
LANGUAGE plv8 IMMUTABLE;
```

Available procedural languages

- PL/pgSQL
- SQL
- Perl
- TCL
- JavaScript (v8)
- LiveScript
- CoffeeScript

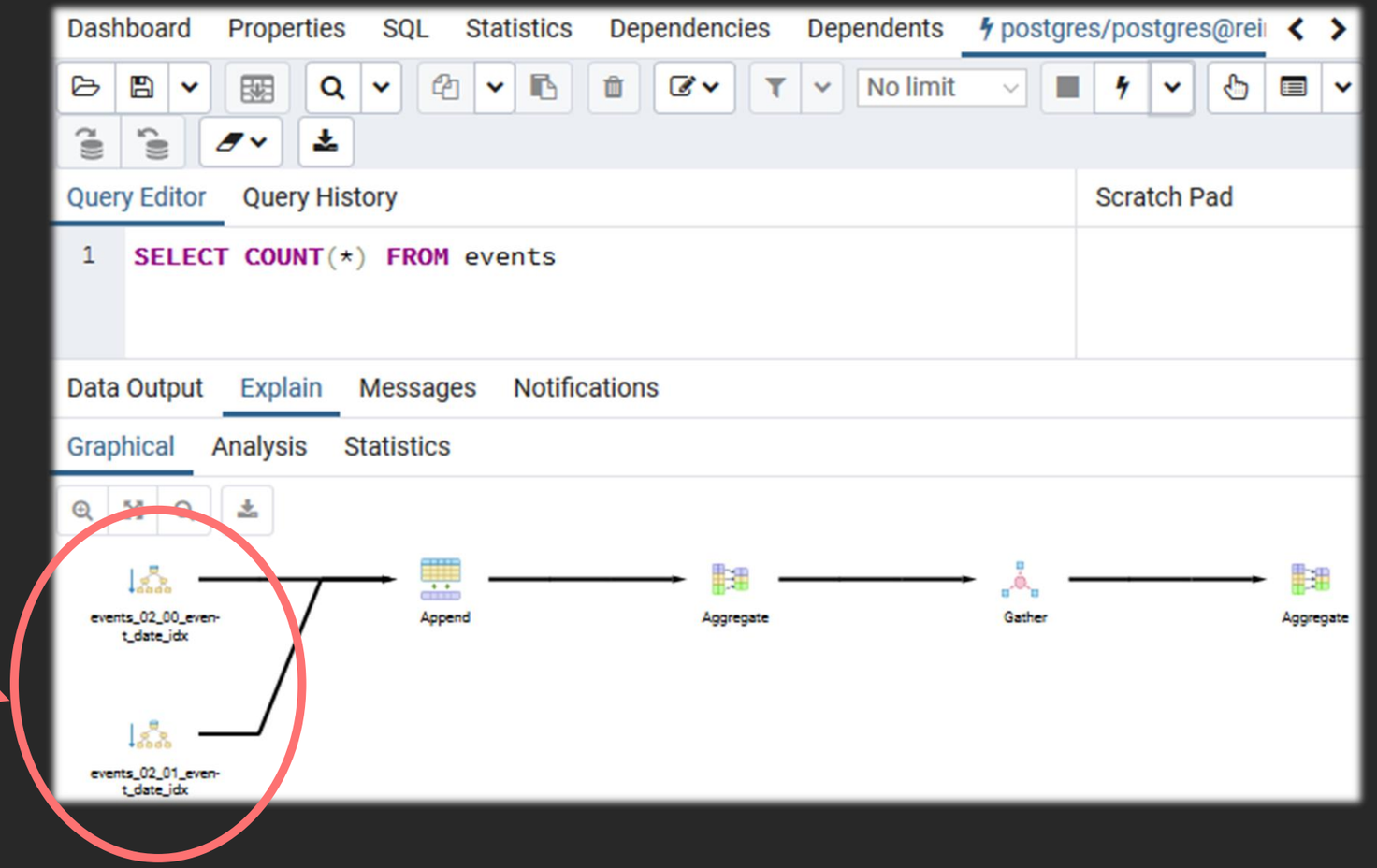
Querying the data

PostgreSQL supports query parallelism for many operations

Supports parallel

- Scans
- Joins
- Aggregations
- Appends

Parallel index-only scans



SQL/JSON path

The legacy PostgreSQL syntax requires comparisons to JSON snippets

```
SELECT *
  FROM events
 WHERE event @>
    '{"date": ["1969/07/16"]}'::jsonb;
```

Implements the SQL:2016 path language for JSON

```
SELECT *
  FROM events
 WHERE event @@
    '$.date[0] == "1969/07/16"';
```

Available in PostgreSQL 12.0 and higher

Generated columns

A special column computed from other columns of the table

Can reference immutable functions

Useful for frequently computed values

Less overhead than using triggers

Available in PostgreSQL 12.0

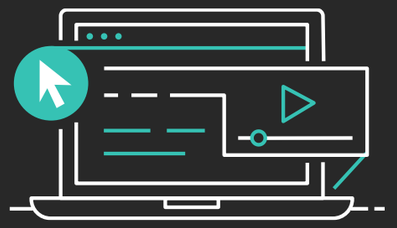
```
ALTER TABLE events
  ADD COLUMN event_date date
  GENERATED ALWAYS
    AS (get_date(event)) STORED;

CREATE FUNCTION get_date(p_event jsonb)
  RETURNS date AS
$$
  SELECT to_date(
           jsonb_path_query(
               p_event,
               '$.date[0]')::text,
           '\"YYYY/MM/DD\"');
$$ LANGUAGE sql
IMMUTABLE;
```

Things to remember

- Use multi-AZ for high availability
- Enable automated backups with retention longer than the default of 7 days
- Force SSL connections
- Implement a strong password policy with password rotation
- Enable performance insights and keep the history
- Set up enhanced monitoring: at least 10 second collection
- Push logs to CloudWatch
- Apply minor version updates
- Use native PostgreSQL functionality for best performance

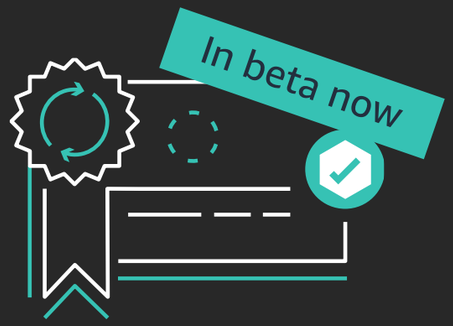
Learn databases with AWS Training and Certification

Resources created by the experts at AWS to help you build and validate database skills



25+ free digital training courses cover topics and services related to databases, including:

- Amazon Aurora
- Amazon Neptune
- Amazon DocumentDB
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon Redshift
- Amazon RDS



Validate expertise with the new **AWS Certified Database - Specialty** beta exam

Visit aws.training

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Related breakouts

DAT202	What's new in Amazon Aurora
DAT309	Amazon Aurora storage demystified: How it all works
DAT328	Deep dive on Amazon Aurora with PostgreSQL compatibility
DAT350	Migrating open-source databases from Amazon EC2 to Amazon Aurora PostgreSQL
GPSTEC345	Operational excellence with Amazon Aurora PostgreSQL
WIN312	Active Directory on AWS to support Windows workloads

Thank you!

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.