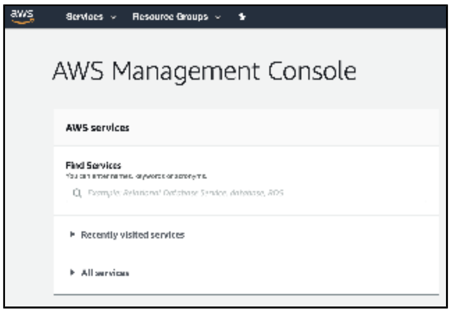
Amazon RDS as an Identity Management Repository

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. With this value proposition in mind, customers want to leverage cloud databases such as RDS to become the ForgeRock Identity Management (IDM) repository. This guide describes how to configure the Postgres flavor of RDS as an IDM repository.

Configuration guide

1. Setup AWS RDS Postgres instance.
2. Configure network and security.
3. Modify ForgeRock Postgres script.
4. Complete remaining steps for Postgres as an IDM repository.

Setup Amazon RDS Postgres


aws
Services
Resource Groups
AWS Management Console
AWS services
Find Services
You can enter names, keywords or acronyms.
Example: Relational Database Service, database, RDS
Recently visited services
All services

Before starting, some key configuration details are needed:

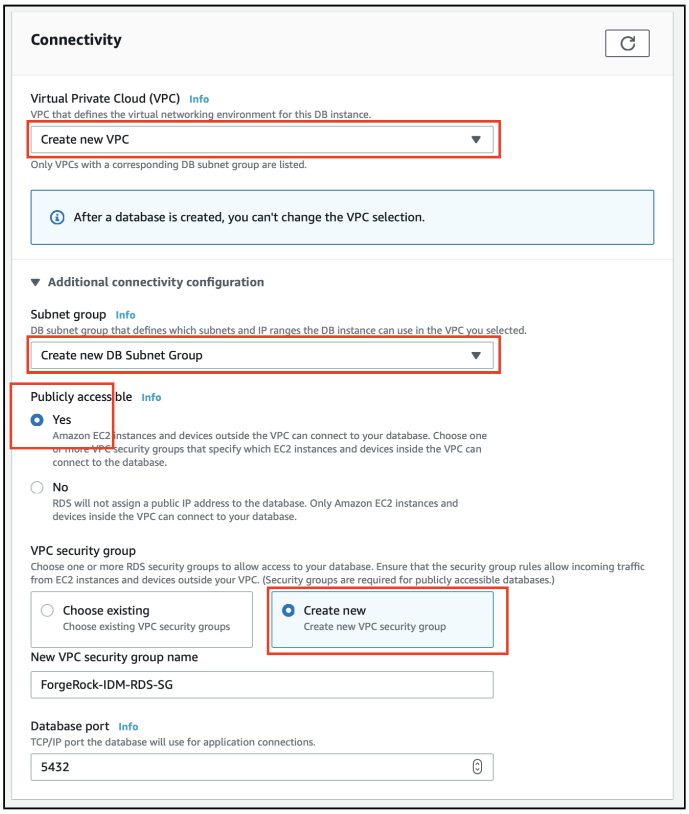
Postgres

- Port 5432
- PubliclyAccessible (true or false)

Security Groups

- Inbound add rules to accommodate the Postgres port (ie 5432) limited by IP addresses (optional)
- Outbound rules "CidrIp": "0.0.0.0/0"

1. Login to AWS admin console and chose to manage RDS
2. Create a PostgreSQL DB Instance that matches a version ForgeRock Identity Management supports per our release notes: https://backstage.forgerock.com/docs/idm
3. For Postgres, configure based upon https://aws.amazon.com/getting-started/tutorials/create-connect-postgresql-db/



Connectivity

Virtual Private Cloud (VPC) Info
VPC that defines the virtual networking environment for this DB instance.

Create new VPC ▼

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change the VPC selection.

▼ Additional connectivity configuration

Subnet group Info
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

Create new DB Subnet Group ▼

Publicly accessible Info

● Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

○ No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose one or more RDS security groups to allow access to your database. Ensure that the security group rules allow incoming traffic from EC2 instances and devices outside your VPC. (Security groups are required for publicly accessible databases.)

○ Choose existing
Choose existing VPC security groups

● Create new
Create new VPC security group

New VPC security group name

ForgeRock-IDM-RDS-SG

Database port Info
TCP/IP port the database will use for application connections.

5432

The above highlights changes from default settings. These changes open network settings and create new security groups, VPC, subnets and to make the instance publicly available.

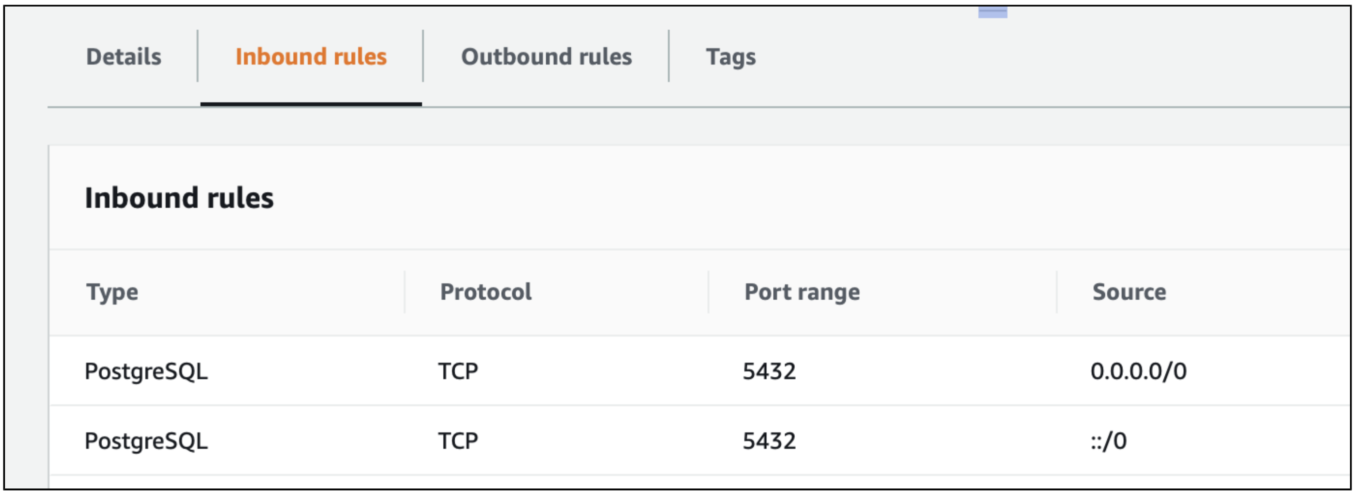
In production tighter control over the network may be desired, but the scope of this paper is not to illustrate a hardened configuration, but rather provide an easy configuration example for understanding.

> ⓘ Ensure network connectivity is accessible from the Identity Management instance. This includes settings in RDS setup that configure the specific, Amazon VPC, Availability Zones and Security Groups.

Configure network and security

In the ForgeRock Installation Guide regarding the topic of Postgres repository configuration, there is a step that details how to setup security related to client connections. This step details how to edit the Postgres client authentication configuration file, *pg_hba.conf*. This step can be ignored, as it cannot be performed against the AWS RDS service. The functional equivalent is to setup an AWS Security Group configuration that allows remote clients (Identity Management process) to connect.

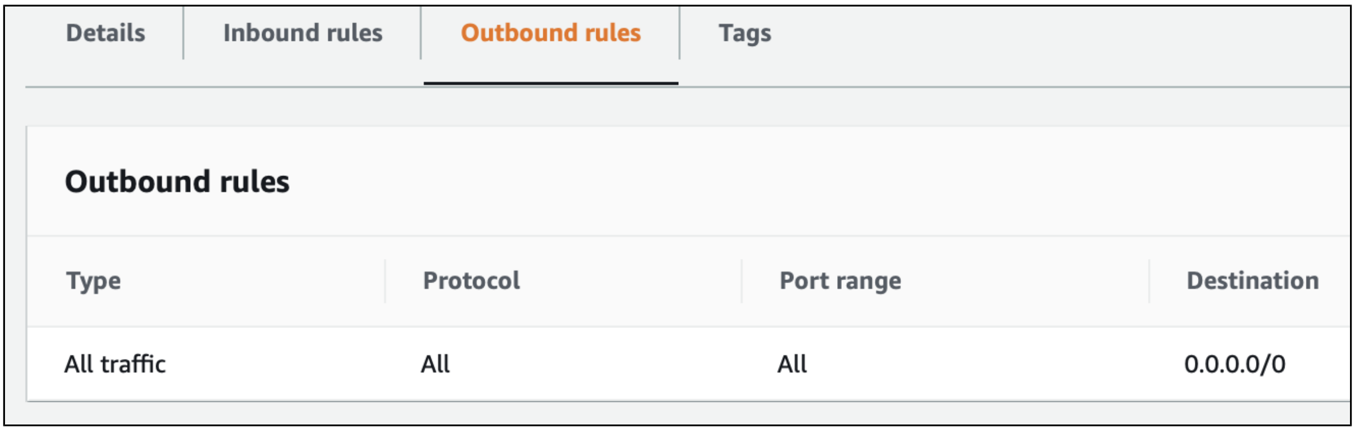
Inbound rules should look like this:



Inbound rules

Type	Protocol	Port range	Source
PostgreSQL	TCP	5432	0.0.0.0/0
PostgreSQL	TCP	5432	::/0

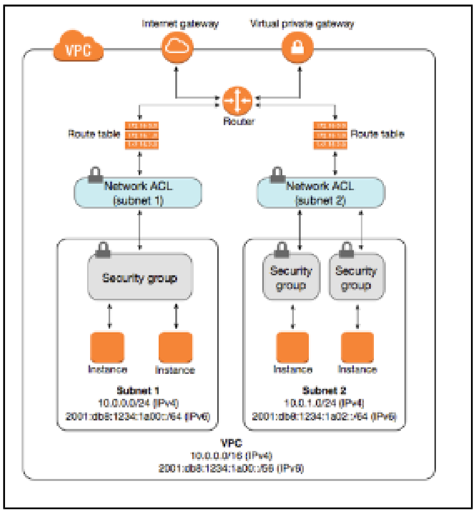
Outbound rules should look like this:



Outbound rules

Type	Protocol	Port range	Destination
All traffic	All	All	0.0.0.0/0

For reference of the security model that involves Security Groups and VPCs:



Security Group / VPC Architecture

Note there is a relationship with the Amazon concept of Virtual Private Cloud (VPC) settings and the associated Amazon concept of a Security Group. Both are key to connectivity to services, including RDS.

To test network connectivity: From the environment that ForgeRock IDM runs:

NetCat test network

```
nc -zv my-rds-instance.us-east-1.rds.amazonaws.com 5432
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 172.30.0.250:5432.
Ncat: 0 bytes sent, 0 bytes received in 0.04 seconds.
```

The above command will prove connectivity into the AWS RDS instance of Postgres from the IDM environment. As shown in the response there will be a status of time taken to test the connection, or instead of output there will be a timeout. A timeout means something in network needs to be debugged.

Modify ForgeRock Postgres script

There exist some RDS nuances, that require modification to the c*reateuser.psql* script from the standard Postgres configuration.

Create a new file to use in place of the *createuser.pgsql.* Call the new file *create-user-aws-rds.pgsql* and edit as below.

`/path/to/openidm/db/postgresql/scripts/create-user-aws-rds.pgsql`

create-user-aws-rds.psql

```
create USER openidm with password 'openidm';
grant openidm TO postgres;
create database openidm encoding 'utf8' owner openidm;
grant all privileges on database openidm to openidm;
```

execute the new create-user-aws.pgsql script

Execute script

```
psql -U postgres < /path/to/openidm/db/postgresql/scripts/createuser.pgsql
```

After this runs a new user called openidm will exist and in Postgres and can now be used the execute the remaining scripts.

Complete steps for Postgres as an IDM repository

From this point the steps in the ForgeRock Installation Guide regarding the topic of Postgres as a repository can be completed as prescribed.

In summary the two details that change from the guide are:

1. Instead of configuring a Postgres client authentication file, perform functional equivalent in AWS Security Groups
2. The createuser.psql script for Postgres needs to be altered.

The remaining steps detailed here: https://backstage.forgerock.com/docs/idm/6.5/install-guide/#repository-postgresql, in brief are:

Execute remaining scripts:

Execute scripts

```
psql -h my-rds-instance.us-east-1.rds.amazonaws.com -p 5432 -U opening < openidm.pgsql
psql -h my-rds-instance.us-east-1.rds.amazonaws.com -p 5432 -U openidm < audit.pgsql
psql -h my-rds-instance.us-east-1.rds.amazonaws.com -p 5432 -d openidm -U openidm < activiti.postgres.create.
engine.sql
psql -h my-rds-instance.us-east-1.rds.amazonaws.com -p 5432 -d openidm -U openidm < activiti.postgres.create.
history.sql
psql -h my-rds-instance.us-east-1.rds.amazonaws.com -p 5432 -d openidm -U openidm < activiti.postgres.create.
identity.sql
```

Docker Tips

The steps in the guide related to Postgres configuration after these script execution that related to file copy and edits need to occur prior to building from the Dockerfile.

Related articles

- Amazon RDS as an Identity Management Repository
- Developing Against DB2
- Change mail example for IDM 6.5
- Selfservices in IDM6
- Bulk Import of users with rolemembership