Postgres Service

Contents

Copyright GE Digital

Postgres Service

PostgresService

Postgres service allows you to provision PostgreSQL databases instances and store your data in them. PostgreSQL is an object-oriented relational database management system to store data securely for retrieval at the request of other software applications.

The PostgreSQL offers features that emphasize extensibility and standards compliance. PostgreSQL supports multiple data types such as, XML, JSON, arrays, geometric, and many others in different domains.

The database can handle workloads ranging from single-machine applications to internet-facing applications with many concurrent users.

For more information and details about using PostgreSQL Database, see the PostgreSQL documentation.

Cloud Foundry Provisioning

After choosing a plan, you can provision and de-provision using the Cloud Foundry CLI.

```
cf create-service postgres-2.0 dedicated-5.1 my-db  -c
'{"postgres_version":"12", "encryption_at_rest":true}'
cf service my-db
cf delete-service my-db
```

High Availability

The database instances are configured for high availability and automatic failover. Every database instance has a standby synchronous replica in a separate datacenter. In case of an instance failure or a datacenter disruption, the standby synchronous replica takes over.

Superuser Access

A uniquely named database superuser role is created within every new instance. The credentials for the superuser role are provided to the Cloud Foundry apps via the `VCAP_SERVICES` environment variable. This user is a member of the system-defined `postgres_dba` role. The `postgres_dba` role should not be dropped or altered in any way. This initial superuser role can be used to create additional databases and additional roles in the instance. Any of those roles can also be granted access to the `postgres_dba` role. Membership in the role grants users broadly destructive operations. Use caution when assigning the `postgres_dba` role.

System Objects in the Database Instance

Every database instance has a database superuser named `postgres_operator` and a database named `postgres_operator_db`. Do not modify them in any way, as the service depends on them.

Automatic Backups and Retention

Database instances are backed-up automatically every day, at a system-chosen time. These automatic backups are retained for seven (7) days.

Maintenance Window

Database instances are automatically maintained during a weekly 30-minute maintenance window assigned by the system upon instance creation. While this window is not used frequently, you should be aware that it exists and that the maintenance may cause down time, however brief. If this window is inconvenient to you, you may request that its start time be changed. The maintenance window of a database instance is included in the database instance's `credentials` property, as discussed in the Credentials on page 2 section.

Postgres Major-version Upgrades

Major version upgrades must be initiated explicitly, via the Cloud Foundry CLI. A DB Instance's major version is never upgraded automatically because the newer major versions of PostgreSQL may introduce backwards-incompatible changes. Please see PostgreSQL community policy for more details.

Postgres Minor-version Upgrades

Minor version upgrades are strictly bug fixes which are deemed by the PostgreSQL community to be safer to apply than to defer. They occur automatically after roll-out during the next maintenance window.

Credentials

Use `credentials` to identify the database service object.

There are a number of ways to identify the database service `credentials`. For example the `credentials` object is available in `VCAP_SERVICES` of an application. The following steps provide another way of inspecting the credentials.

1. Find an existing service key.

```
cf service-keys my-db
```

2. Create a service key, if you cannot find one that is already usable by you.

```
cf create-service-key my-db my-db-key
```

3. Get the credentials.

```
cf service-key my-db my-db-key
 ...
 {
  "hostname": "db-hostname",
  "port": 5432,
  "database": "postgres",
  "username": "username",
  "password": "password",
  "uri": "postgres://username:password@db-hostname:5432/dbname",
  "uuid": "4ac6cb37-f486-4d71-a339-eb46bce4e399",
  "allocated_storage": 10,
  "maintenance_window": "fri:03:00-fri:03:30"
  "encryption_at_rest":true
 }
```

Using the Postgres Service

Postgres service uses Postgres and allows you to choose the major version.

Choosing a Postgres Version

When creating a new database instance, you must specify a Postgres major version using a JSON object which consists of the `postgres_version` key and the corresponding value, as follows:

```
cf create-service postgres-2.0 dedicated-5.1 my-db -c
'{"postgres_version":"12", "encryption_at_rest":true}'
```

Point-in-time Recovery

Point-in-time Recovery (PITR) creates a new instance whose state begins at your chosen point in the past, up to seven (7) days, for an existing instance. From that moment forward, the new instance is independent of its parent. Storage for the new instance is general-purpose SSD, just as it is when creating ordinary instances. The new instance must be created with the same plan as the existing instance. PITR can be used for recovery from human errors like unqualified deletes, and for creating development databases which need to contain a data set which reflects production. The unique ID (uuid) of a database instance is included in the `credentials` property, as shown in the section.

Use unique ID (`uuid`) to create a new service instance:

```
cf create-service postgres-2.0 dedicated-5.1 my-db-clone -c '{
    "restore_from" : {
        "source_db_instance_id": "4ac6cb37-f486-4d71-a339-
eb46bce4e399",
        "restore_to_time": "2016-08-01T23:52:25Z"
    }
}'

Creating service instance my-app-clone in org my-org / space test
as me@example.com...
OK
```

You must specify the value of the `restore_to_time` parameter in UTC. An instance created using PITR inherits the data and many configuration settings from the instance from which it is created, so specifying `encryption_at_rest` or `postgres_version` in a PITR request is an error. To check on the progress of service creation started by the above command, use the `cf service` command.

```
cf service my-db-clone
Service instance: my-db-clone
...

Last Operation
Status: create succeeded
...
```

After successful creation of the new instance, you can `cf unbind` the application from the previous instance and `cf bind` it to the new instance. Alternatively, you can use `cf rename-service` to achieve desired configuration.

Upgrading the DB Instance Postgres major-version

You can upgrade the Postgres major-version of your DB instance using the Cloud Foundry CLI as shown below:

```
cf update-service my-db -c '{"postgres_version":"12"}'
```

During the upgrade process, the DB Instance is unavailable and applications cannot connect to it. The duration of the DB instance downtime depends on the size of the database.

Note that you can only specify the Postgres major-version you want to upgrade to. The Postgres service automatically chooses the latest minor-version for the specified major-version.

Changing the Plan of a Service Instance

You can change the plan of the Cloud Foundry Postgres Service Instance in Cloud Foundry using the CLI, for example:

```
cf update-service my-db -p dedicated-10.1
```

During the update process, the DB Instance is unavailable and applications cannot connect to it.

Increasing DB Instance Storage

You can increase the storage allocation of your DB instance by using Cloud Foundry CLI, for example:

```
cf update-service my-db -c '{"allocated_storage":100}'
```

This is an asynchronous operation, and can take some time to finish. The database remains available during execution, although you might experience a minor performance impact.

Set the `"allocated_storage"` parameter value to be the database's allocated storage in GB. You cannot use this command to decrease the allocated storage for a DB instance.

Encryption at Rest

Postgres service instances support encryption at rest. All storage instances are encrypted with a unique encryption key except when they are created using Point-in-Time-Recovery (PITR). Database instances created using PITR inherit the encryption key from the instance they are created from. When creating a new database instance, you must enable storage encryption using a JSON object which consists of one key, encryption_at_rest and the corresponding boolean value in command line format:

```
cf create-service postgres-2.0 dedicated-5.1 my-db -c
'{"postgres_version":"12", "encryption_at_rest":true}'
```

Connecting to the Postgres Service

Only applications in the Cloud Foundry environment are allowed to connect to the DB instances provisioned by the Postgres Service.

You may connect to your PostgreSQL DB Instance using the pgStudio application. Directions for installing and using pgStudio are provided in KB0010829.

Shared-NR PostgreSQL

About Shared NR - PostgreSQL Database

The Shared-NR uses PostgreSQL to provide as a Postgres service-relational database management system to store data securely for retrieval at the request of other software applications.

The PostgreSQL Database relational database management system (RDBMS) as a service on Predix, offers features that emphasize extensibility and standards compliance. PostgreSQL supports multiple data types such as, XML, JSON, arrays, geometric and many others in different domains.

The database can handle workloads ranging from single-machine apps to internet-facing applications with many concurrent users.

For more information and details about using PostgreSQL Database, see the PostgreSQL documentation.

Note: The Shared-NR plan is deprecated, no longer available, and will be decommissioned soon. If you are a current Shared-NR customer, you may continue using the service. (See status here). Please move to Postures 2.0 as soon as possible. Consider the following limitations as you continue to use your current Shared-NR plan:

- New instances cannot be created
- Current instances cannot be modified or enlarged.

PostgreSQL Features

High Availability and Scalability

The PostgreSQL Database architecture ensures high availability and automatic scaling. Postgres supports the Predix goal of 99.9 percent up time and automatically scales to add nodes as usage levels increase.

Each subscription plan has an associated storage restriction, concurrent connection limit, and other specific features such as data-at-rest encryption.

Daily Backups

All Postgres databases are backed up daily and uploaded to archive storage for 14 days as long as a service is active. All archived backups older than 14 days are automatically deleted.

Note: Custom business rules for individual databases cannot be defined for backups.

On-Demand Backup

An embedded Cloud Foundry application enables users to schedule a backup as needed for events such as, preparing for data migration, rolling out a system upgrade, or performing a bulk data upload. One on-demand backup is allowed per database, per day.

Replication and Failover

All subscription plans (with the exception of the shared-nr plan) replicate the database to a standby instance. If a failover is activated, all connections are automatically redirected to the standby instance. An outage of up to 30 minutes, Recovery Time Objective, (RTO) might occur until the transfer to the standby is complete and the standby is established as the new active database. The length of the outage during the failover depends on the number of write transactions that were in process at the time of the failure. In the rare case that the master database is completely down and the standby is unable to replicate the latest data, there will be a data loss; the new master will start without the latest replicated data within the RTO time.

Point-In-Time Recovery

Point-In-Time Recovery (PITR) backs up the system continuously so that data can be restored or recovered to a particular time in the past within the 7-day archival period.

Database recovery is a manual process. To fall back to a previous version, submit your request to support.

Related Information

#unique_11
#unique_12

Connecting to your Shared-NR PostgreSQL Database

Learn how to connect to your database and schedule backups.

Connecting to the PostgreSQL Database

About This Task

Use one of these ways to connect to the PostgreSQL Database:

Connect using PHPpgAdmin

1. Follow the steps listed in this article: KB0010829
2. Refer to the Admin description for database management options: https://github.com/cloudfoundry-community/phppgadmin-cf

Connect using PGStudio

1. Download the Cloud Foundry version of PGStudio and push the application to your org/space.
2. Refer to this PG Studio link for usage instructions: https://github.com/cloudfoundry-community/phppgadmin-cfhttps://github.com/john-k-ge/pg_studio_1.2_cf

Connect using a Predix HTTP Data Tunnel

https://digitalexchange.ge.com/solution/predix-http-data-tunnel

Deleting a Shared-NR PostgreSQL Database Service Instance

About This Task

You can remove the Shared NR PostgreSQL Database application and service instance.

Procedure

1. Log into your Cloud Foundry account.

```
cf login
```

2. Stop the application by entering the following command:

```
cf stop <application_name>
```

3. Unbind your SQL Database service instance from the application.

```
cf us <application_name> <my-postgres_service_instance>
```

4. Remove the SQL Database application.

   ```
   cf d <application_name>
   ```

5. Remove the SQL Database service instance.

   ```
   cf ds <my-postres_service_instance>
   ```

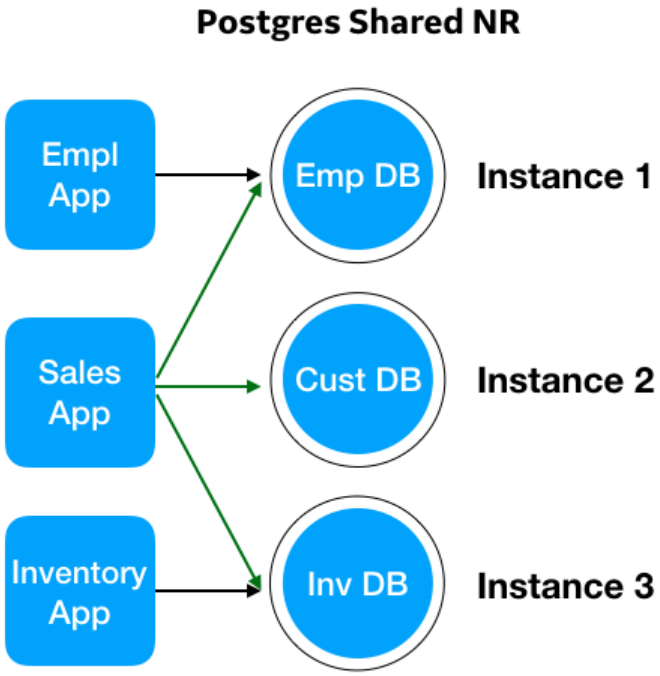
   The database is deleted.

Moving to Postgres

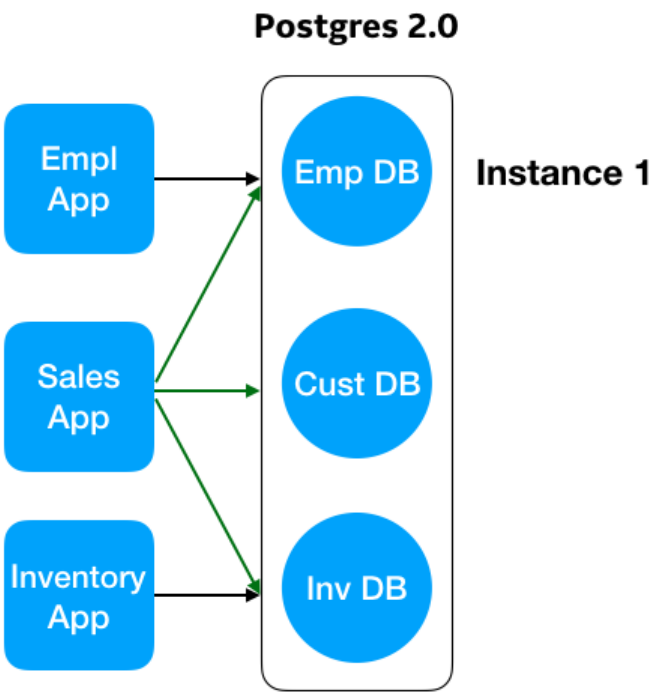
Moving from Shared-NR to Postgres 2.0

This topic provides some points worth considering as you plan your migration from Postgres Shared-NR to Postgres-2.0.

The Postgres Shared-NR plan is a free cloud-based service allowing you to create as many service instances (as created with cf create service) as you need for your application. Each service instance contains a single database on a shared database instance. When you need another database, you must create another service instance. Further, there was only one user per service instance, and it did not have Super User privileges. You could not create new users or new databases on demand.



Migrating to Postgres-2.0 service introduces you to a robust computing infrastructure where a service instance corresponds to a single database instance that can hold all of your databases (as created with the CREATE DATABASE SQL command). As Predix charges by the service instance, this represents a considerable savings for implementations that were using tens or hundreds of service instances with Shared-NR plan. In this environment, as the Super User, you can create additional databases and users as needed, and manage their privileges.



To create additional databases or users you can use the `CREATE DATABASE` or `CREATE USER` SQL commands. Additional information about ad-hoc SQL commands and pgStudio in available at KB0010829.