PITR (Point in Time Recovery) Support in Crunchy Container Suite

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Description

PITR (point-in-time-recovery) is a feature that lets a DBA recreate a database from backup and log files at a certain point in time. This is useful in that the log files record only the small changes made to a database over time. For large databases, doing a full database backup is not workable given the time and space it requires. Therefore, with PITR, you only need to save the log files plus a known full backup to recreate your database in the event of a database or system failure.

Support was added into the Crunchy Container Suite as of version 1.2.3.

This document will discuss the design and usage of the PITR features as implemented so far.

1.1 WAL

Write Ahead Logs (WAL) is created when you enable continuous archiving as described here.

By default in the crunchy-postgres container, WAL logging is **not** enabled. To enable WAL logging, you set the following environment variables when starting the crunchy-postgres container:

```
ARCHIVE_MODE=on
ARCHIVE_TIMEOUT=60
```

These variables set the same name settings within the postgresql.conf file that is used by the database.

When set, WAL files generated by the database will be written out to the /pgwal mount point.

1.2 Restoring from WAL

A full backup is required to do a PITR. crunchy-backup currently performs this role, running a pg_basebackup for you. This is required for PITR.

After a backup has been performed, code was added into crunchy-postgres which during a backup restore, will also check to see if you want to do a PITR.

When you run a crunchy-postgres container and specify a /recover volume mount, this will trigger logic for PITR during container startup.

/backup will still be used to find the base backup you want to use, just like a normal database backup restore.

/pgwal can still be used to write out new WAL files from the restored database container.

The RECOVERY_TARGET_NAME environment variable is used to tell the PITR logic what target name you want to use for the PITR. RECOVERY_TARGET_TIME is also an optional environment variable that lets you restore using a known time stamp. If you don't specify either of these environment variables, then the PITR logic will assume you want to restore using all the WAL files or essentially the last known recovery point.

The RECOVERY_TARGET_INCLUSIVE environment variable is also available to let you control the setting of the recovery.conf setting **recovery_target_inclusive**. If you do not set this environment variable the default is **true**.

1.3 Hint

Once you recover a database using PITR, it will be in read-only mode. To make the database resume as a writable database, run the following sql command:

```
select pg_xlog_replay_resume();
```

1.4 Examples

Examples for PITR have been created within the standalone, openshift, and kubernetes examples directories.

There is a blog on how the example works documented here: https://blog.openshift.com/crunchy-postgresql-containers-on-openshift-performing-point-in-time-recovery/

2 Legal Notices