A study on point in time recovery on relational databases

Author
Ștefan Stan

Scientific Coordinator
Lect. dr. Cristian Frăsinaru
Company Supervisor
DevOps Engineer Gabi Ichim, TSS-Yonder

Faculty of Computer Science
University Alexandru Ioan Cuza from Iași

6 July 2017

Table of contents

Introduction
Keywords
Objectives
Studies
Point in time recovery
Logical Volumes Management
Implementation
Logical volumes architecture
System architecture
Use Cases
Future dev
Conclusions
Bibliography

Keywords

checkpoint, recovery, point in time, database, logical volume, UNIX scripts, RESTful, API, cloud, Amazon Web Service

Objectives

- *study* the way relational databases (ex. PostgreSQL) manages transactions;
- *study* and find a way to use PostgreSQL logging to create a management mechanism of the database changes;
- *create* a tool that enables a user to restore a database instance state to one in a specific point in time;
- *investigate* the possibilities and find a solution to *facilitate* usage of previously created tool to persons without UNIX/ PostgreSQL specific knowledge.

Studies - Point in time recovery

Basebackup

- checkpoint from which it can be started the point in time recovery

Write Ahead Logs (WALs)

- binary sequences that PostgreSQL generates to keep its logging (information about transactions)

Point in time recovery (PITR)

- the process of changing a database current state to one at a specific point in time, given as input.

Studies - Logical Volumes Management (LVM)

Advantages:

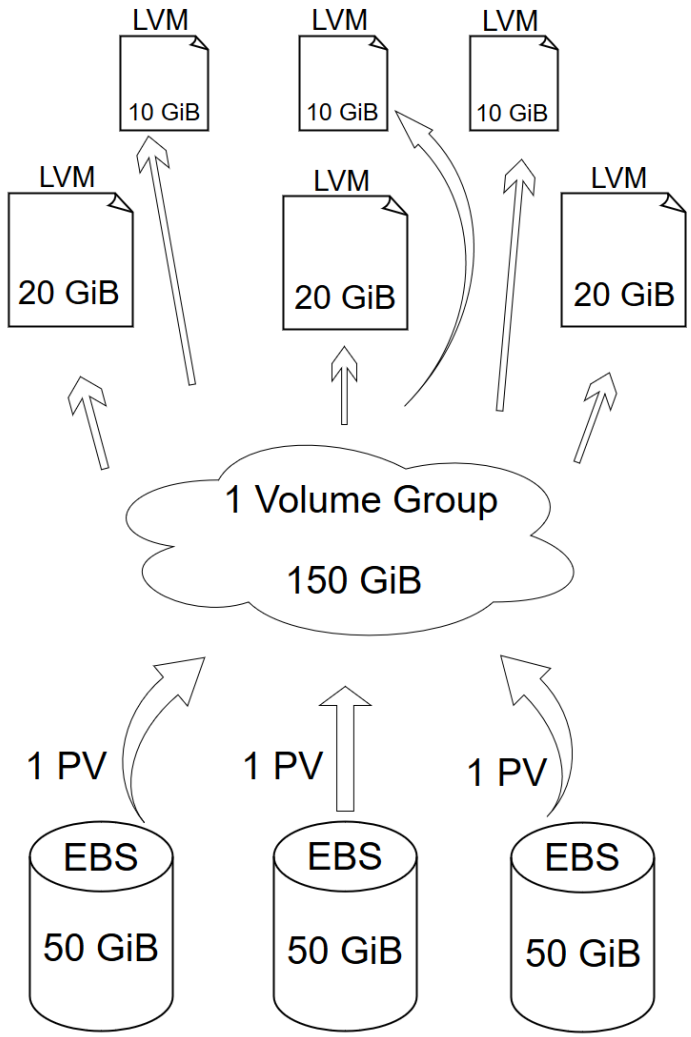
- flexible "disks"called logical volumes (no longer depend on physical space of harddisks)
- abstraction layer above storage units

Physical volume (PV) - a partition with LVM metadata asociated.

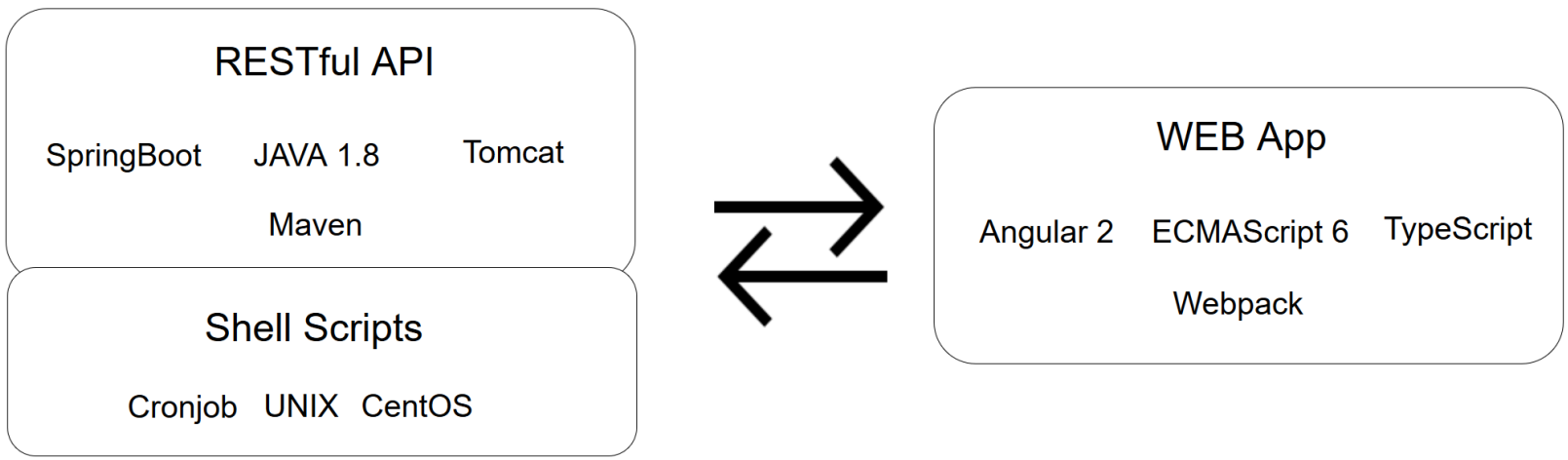
Volume Group (VG) - a way of organizing physical volumes to enable subdivisions and their management.

Logical Volume (LV) - a subdivision of a volume group, which may has its space allocated from different physical volumes.

Implementation - Logical volumes architecture



Implementation - System architecture



Deployment - Amazon Web Services (AWS)

Advantages:

- simulate a production environment;

- physical volumes used in LVM architecture are provided by EBS (Elastic Block Storage) service which guarantees data guarantees availability in case of disaster by using replication in 3 different data centers;

- IaaS (Infrastructure as a service) access, which is needed to setup LVM architecture.

Disadvantages:

- AWS costs.

Use Cases

- The worst has happened, databased crushed/stopped and after restarting it has inconsistent data. The tool build can be used from its web interface to restore the database to a consistent state at a specified time;
- A tester wants to run a suite of test cases which will alter the data in some way. It realises that what he did was wrong and he wants to restore the database in the exact state it was at a specified time.
  He has no knowledge of SQL or Unix.

Future development directions

- study different databases also (right now works only with PostgreSQL);
- offer an abstraction layer over different databases;
- improve web design, refactor Angular components and also HTML and CSS

Conclusions

- studied and understood the way PostgreSQL manages database changes;
- studied and understood how to use LVM to combine database point in time recovery with "flexible" UNIX partitions to minimize downtime of the database;
- build an app which facilitates restoring of PostgreSQL databases at a given time, and management (start/ stop/ status queries);
- deployed and tested the created solution on a similar production cloud Amazon Web Services (AWS) environment.

Bibliography

Sander van Vugt,
Red Hat® RHCSA™/RHCE® 7 Cert Guide: Red Hat Enterprise Linux 7 (EX200 and EX300)

Postgres Documentation
https://www.postgresql.org/docs/9.5/static/continuous-archiving.html

E. Gamma, R. Helm, R. Johnson, and J. Vlissides.
Design Patterns: Elements of Reusable Object-Oriented Software.
Addison-Wesley, 2005.

Craig Walls,
Spring in Action, Third Edition