



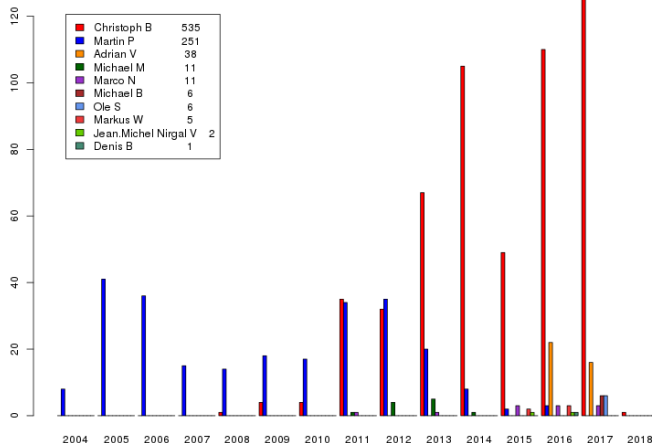
PostgreSQL and Debian under the Elephant-Shed

Integrated Administration and Monitoring

Michael Banck <michael.banck@creativ.de>

MiniDebConf Hamburg 2018

Uploaders of pkg-postgresql team





- ▶ Extensible, object-relational database system
- ▶ Created as a research project at Berkeley, community-based development since the mid-90s
- ▶ BSD/MIT-style licence
- ▶ No copyright assignments, no open-core, no dual licensing

- ▶ “Postgres Global Development Group”, Core team (5 members), Release team (3 members), around 20 Committers
- ▶ Vendor-neutral, commercial support available from multiple companies
- ▶ Yearly release-cycle
  - ▶ Time-based feature-freeze
  - ▶ Release when it's ready
- ▶ 5 years maintenance per release
  - ▶ Quarterly patch-releases, whitelisted by Debian SRM
- ▶ Many (also proprietary) forks

- ▶ Packaged in Debian
- ▶ Rock Solid
- ▶ Good and consistent coverage of the SQL-standard, useful modern SQL extensions
- ▶ Cost-based query-planner
- ▶ Transactional changes to database structure (DDL)
- ▶ Drivers for many programming languages
- ▶ Many different procedural languages
- ▶ Foreign-data-wrapper (FDW) for federated access to many other data sources/databases
- ▶ Large number of extensions and associated projects



- ▶ `amcheck` - Index consistency checking
- ▶ `bgw-replstatus` - Report whether node is master or standby
- ▶ `hypopg` - Hypothetical indexes extension
- ▶ `jsonquery` - JSON query language with indexing support
- ▶ `londiste` - Trigger-base logical replication
- ▶ `pgaudit` - Audit extension
- ▶ `pglogical` - Logical replication extension
- ▶ `pg-dirtyread` - Read dead but unvacuumed rows from a table
- ▶ `pg-cron` - Run periodic jobs in PostgreSQL
- ▶ `postgresql-rum` - RUM access method
- ▶ `pspg` - Interactive pager for PostgreSQL query output
- ▶ `wal2json` - Logical decoding JSON output plugin

Major New Features since stretch



- ▶ Logical replication
- ▶ Native, declarative partitionierung
- ▶ Extended parallel query
- ▶ Just-in-Time compilation of expressions
- ▶ Client-based connection failover
- ▶ Quorum-commit for synchronous standbys
- ▶ Improved and production-ready hash index
- ▶ Transaction-controlled SQL procedures (Stored Procedures)
- ▶ Multi-column statistics for the query planer

Debian packages one Postgres release per version

Problem:

- ▶ There is Debian sid, stretch, jessie, wheezy
- ▶ There is PostgreSQL 11, 10, 9.6, 9.5, 9.4, 9.3, ...
- ▶ PostgreSQL major releases have incompatible on-disk format
- ▶ Upgrading needs both versions installed in parallel (or lots of disk space and a complicated plan)

Solution: make packages for 9.6, 10 etc. co-installable

- ▶ `postgresql-common` takes care of creating database clusters in the correct locations
- ▶ Each cluster is listening on a different port
- ▶ Still, Debian stretch has 9.6 only (jessie had 9.4, wheezy had 9.1)
  - ▶ Users want to try new PostgreSQL versions
  - ▶ Users want to upgrade Debian without upgrading PostgreSQL

- ▶ Apt repository hosting packages of all upstream supported PostgreSQL versions built for all Debian (and Ubuntu) releases
- ▶ Somewhat a superset of what backports would do
- ▶ 6 PostgreSQL releases: 9.3 9.4 9.5 9.6 10 (11)
- ▶ 8 Debian and Ubuntu releases: wheezy jessie stretch buster sid trusty xenial bionic
- ▶ 3 architectures: amd64 i386 ppc64el
- ▶ **6x8x3 = 144 targets**
- ▶ Automatic builds via Jenkins and jenkins-debian-glue
- ▶ Hosted at <https://pgdgbuild.dus.dg-i.net/> (Thanks to DG-i)
- ▶ Running autopkgtests as well

- ▶ Allows multiple major versions to be co-installed
- ▶ Instances ("Clusters") are identified by version and cluster name
- ▶ Every new instance gets an incremented port, starting from 5432
- ▶ pg_wrapper: Wrapper for PostgreSQL client binaries
  - ▶ `psql --cluster 9.4/main`
  - ▶ `PGCLUSTER=9.4/main psql`
- ▶ pg_conftool: Read and edit parameters in configuration files
- ▶ pg_*cluster: Debian-specific cluster administration commands
  - ▶ `pg_createcluster <version> <name>`
- ▶ pg_createcluster behaviour can be configured in `/etc/postgresql-common/createcluster.conf`
  - ▶ `create_main_cluster, start_conf, data_directory, xlogdir, initdb_options'`
  - ▶ Additional config parameters are copied into `postgresql.conf`

What's Missing?



- ▶ PostgreSQL in Production implies more than just the Database Server

PostgreSQL - In a Vacuum?

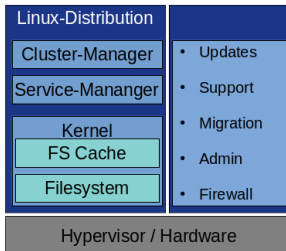


PostgreSQL

PostgreSQL - In a Vacuum?



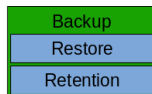
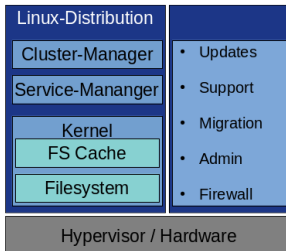
PostgreSQL



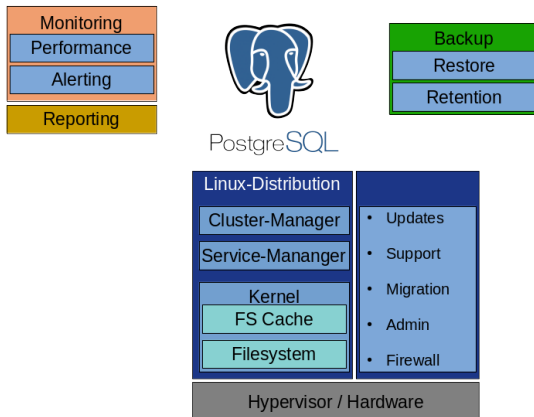
PostgreSQL - In a Vacuum?



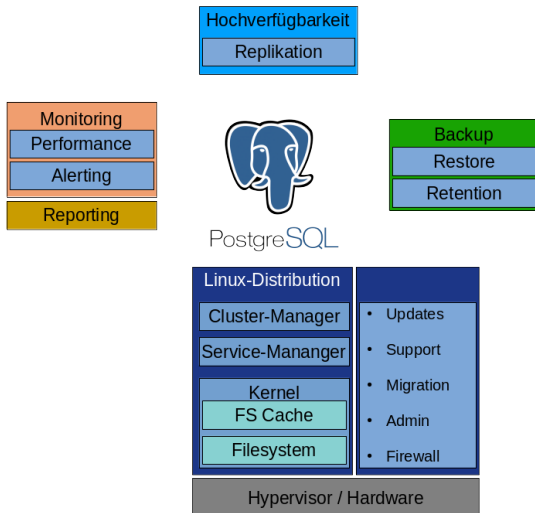
PostgreSQL



PostgreSQL - Ecosystem



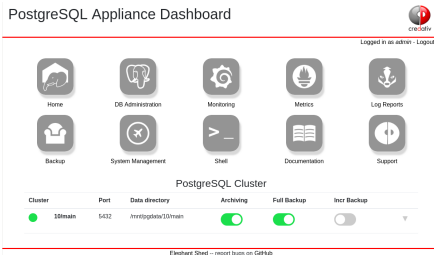
PostgreSQL - Ecosystem



- ▶ Make using PostgreSQL as easy as possible
- ▶ Complete Offering
  - ▶ All required components included
  - ▶ All components geared to each other
- ▶ Modular
  - ▶ Extensible
  - ▶ Components as building sets for individual setups
- ▶ 100% Open Source, no proprietary components
- ▶ Long-term maintenance and commercial support

- ▶ Monitoring
- ▶ Database administration
- ▶ Backups
- ▶ Logfile analysis
- ▶ System administration

PostgreSQL Appliance Dashboard



The dashboard features a grid of icons for various functions: Home, DB Administration, Monitoring, Metrics, Log Reports, Backup, System Management, Shell, Documentation, and Support. Below the icons is a table titled "PostgreSQL Cluster" showing the status of a cluster named "101main".

Cluster	Port	Data directory	Archiving	Full Backup	Incr Backup
101main	5432	/mnt/pgsql/data/101main	☒	☒	☐

Elephant Shed -- report bugs on GitHub

- Web based interface
- SSO via PAM authentication

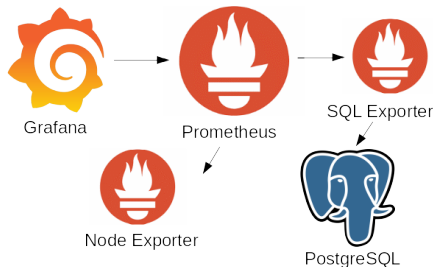


PostgreSQL

Adjustments to the standard configuration:

- ▶ Listen parameter
- ▶ Logging
- ▶ Backup
- ▶ Replication
- ▶ Extensions

- ▶ Grafana - Metrics rendering and display
- ▶ Prometheus - Metrics storage
- ▶ SQL Exporter - Export of PostgreSQL metrics
- ▶ Node Exporter - Export of system metrics



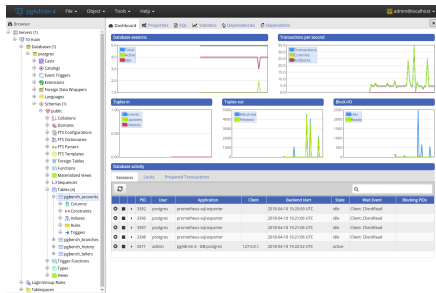
Monitoring - Grafana Dashboards



- ▶ Rundown of current performance data
- ▶ System metrics
- ▶ Instance metrics
- ▶ Database metrics
- ▶ Replication

Via pgAdmin4

- ▶ Database Administration
  - ▶ Schema management
  - ▶ User management
    - ▶ Dump / Restore
- ▶ Ad-hoc SQL queries

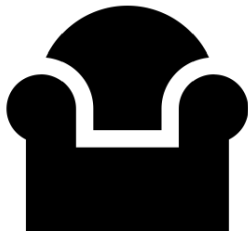


Via pgBackRest

- ▶ Physical backups / base backups
- ▶ WAL Archiving / Point-in-Time-Recovery

Features:

- ▶ Compression (On Disk / Network)
- ▶ Checksumming
- ▶ Encryption
- ▶ Full, Differential and Incremental Backups
- ▶ Backup to disk and S3



Backups - Base Configuration

- ▶ Prepared postgresql.conf
- ▶ Preconfigured retention policy
- ▶ Prepared systemd-timer
- ▶ Ad-hoc backups possible
  - ▶ Activated through webinterface

PostgreSQL Cluster

Cluster	Port	Data directory	Archiving	Full Backup	Incr Backup	
 10/main	5432	/mnt/pgdata/10/main				

systemd

Report

Backup

Service

Log

Run

Show

Full

Incremental

Info

Generation of HTML reports via pgBadger

- ▶ Processing of logfiles, e.g.:
  - ▶ Error messages
  - ▶ Statement duration
(`log_min_duration_statement`)
  - ▶ Connections over a time period
- ▶ Automatic generation:
  - ▶ Daily via systemd-timer
  - ▶ Ad-Hoc via webinterface
  - ▶ For every Instance



Via Cockpit

- ▶ Service management (start, stop, enable, disable)
- ▶ Review and installation of updates
- ▶ Journal liveview
- ▶ User administration



- ▶ Platforms

- ▶ Appliance and Packages (Database + Management)

- ▶ Debian 9 (stable)
    - ▶ Ubuntu 18.04 (since 1.2.0)

- ▶ Database-Server (PostgreSQL + Prometheus exporter)

- ▶ Debian / Ubuntu
    - ▶ RHEL / CentOS
    - ▶ SLES

- ▶ Architectures

- ▶ x86_64
  - ▶ ppc64el

1.2.0 Minor Release (2018-05-18)

- ▶ Version updates
  - ▶ Grafana 5
  - ▶ Prometheus 2.0
- ▶ Overhauled Monitoring-Dashboard
- ▶ Ubuntu 18.04 support

Future Major Updates

- ▶ RHEL / CentOS 7 support
- ▶ postgresql-common REST API
  - ▶ Configuration Changes via Webinterface
  - ▶ Instance Management via Webinterface
  - ▶ <https://github.com/credativ/pgapi>
- ▶ Multi-Host-Administration

Virtual Machine via Vagrant + VirtualBox

```
vagrant init credativ/elephant-shed  
vagrant up
```

- ▶ Default user: admin / admin
- ▶ Web-GUI: <https://localhost:4433>
- ▶ PG-Port: 55432 (psql -h localhost -p 55432 -U admin)
- ▶ pgAdmin4: admin@localhost / admin

```
https://elephant-shed.io/project/vagrant/  
https://elephant-shed.io/#download  
https://app.vagrantup.com/credativ/boxes/elephant-shed
```


Team

- ▶ Christoph Berg <christoph.berg@credativ.de>
- ▶ Alexander Sosna <alexander.sosna@credativ.de>
- ▶ Adrian Vondendriesch <adrian.vondendriesch@credativ.de>

Further Informationen

- ▶ <https://elephant-shed.io>
- ▶ <https://github.com/credativ/elephant-shed>
- ▶ <https://credativ.de/postgresql-competence-center>