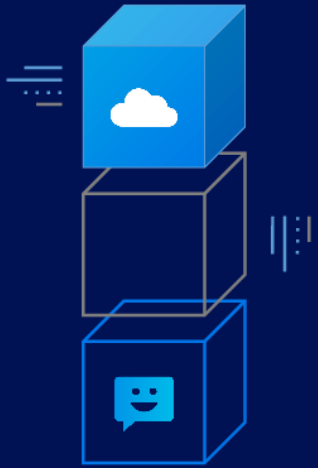
instaclustr

Postgres für nicht-Datenbank AdministratorInnen



Michael Banck
michael.banck@instaclustr.com

FrOSCon 2022

Postgres für nicht-Datenbank AdministratorInnen

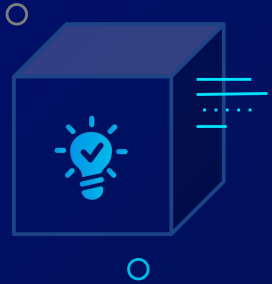
- Überblick und Architektur
- Best-Practises Installation, Konfiguration und Monitoring
- Release- und Upgrade-Zyklus
- Operations Best-Practises und Praktikable Limits
- Pitfalls

https://share.credativ.com/~mba/talks/postgresql-administration.pdf

Prior Art

- **Christophe Pettus, PgDay SCALE 10x 2012: “PostgreSQL Performance… when it’s not your job”**

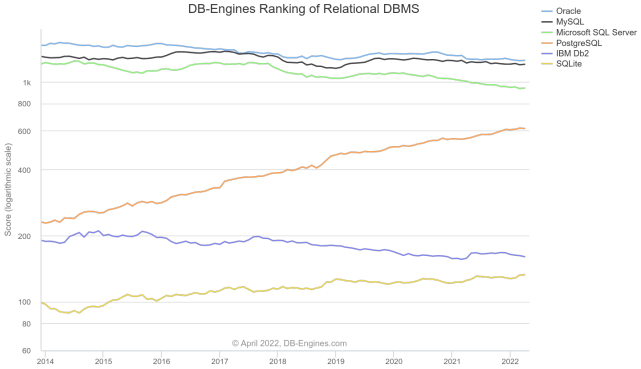
`https://thebuild.com/presentations/not-my-job.pdf`



Überblick und Architektur

instaclustr

Popularity Contest



https://db-engines.com/en/ranking_trend/relational+dbms

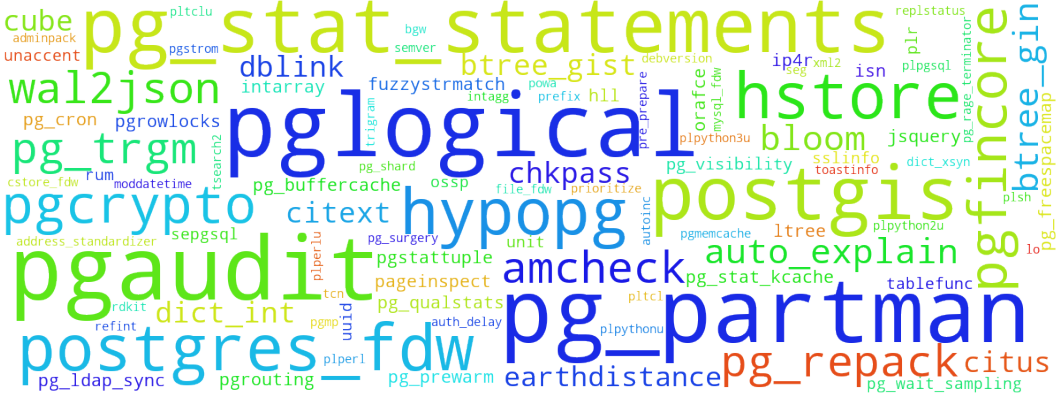
PostgreSQL - Überblick

- “The World’s Most Advanced Open Source Relational Database”
- Erweiterbares, Objekt-Relationales Datenbanksystem
- Entstanden als Forschungsprojekt in Berkeley, Community-basierte Entwicklung seit Mitte der 90er
- Hersteller-Unabhängig, kommerzieller Support von mehreren Firmen erhältlich
- ‘’Postgres Global Development Group”, 7 köpfiges Core Team, ca. 30 Committer
- Keine Copyright-Assignments, Open-Core oder Dual-Lizenzierung
- BSD/MIT-artige Lizenz
- Viele (auch proprietäre) Forks

PostgreSQL - Forks

PostgreSQL - Forks

PostgreSQL - Erweiterungen



PostgreSQL Wire Protokoll - Die universelle Datenbank API

- **Google Cloud Spanner Ankündigung bei der Google Cloud Next ’21: * ”PostgreSQL has emerged as the “API” for operational databases”**

https://dbdb.io/browse?compatible=postgresql

PostgreSQL - Cloud Provider

Top Feature-Requests 2009

- Einfache, eingebaute Replikation
- In-Place Upgrades
- Administration/Monitoring
- Treiber-Qualität/Wartung
- Erweiterungs-Management
- Locales/Collationen pro Spalte
- Materialisierte und Aktualisierbare Sichten
- Autonome Transaktionen
- Parallelisierte Queries
- Index-Only Scans
- Merge/Upsert Anweisung
- Automatische Partitionierung
- Hot Standby
- Rekursive Abfragen und Window Funktionen

Top Feature-Requests 2009 - Stand 2022

- ~~Einfache, eingebaute Replikation~~
- ~~In-Place Upgrades~~
- ~~Administration/Monitoring~~
- ~~Treiber-Qualität/Wartung~~
- ~~Erweiterungs-Management~~
- ~~Locales/Collationen pro Spalte~~
- ~~Materialisierte und Aktualisierbare Sichten~~
- Autonome Transaktionen
- ~~Parallelisierte Queries~~
- ~~Index-Only Scans~~
- ~~Merge/Upsert Anweisung~~
- ~~Automatische Partitionierung~~
- ~~Hot Standby~~
- ~~Rekursive Abfragen und Window Funktionen~~

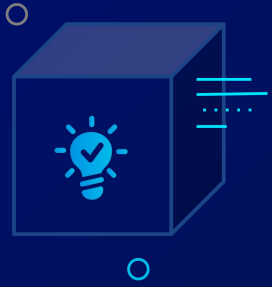
PostgreSQL ist Unix-basiert

Viele Aufgaben ans Betriebssystem delegiert

- **Lokale Verbindungen verwenden Unix-Sockets**
- **Remote Verbindungen verwenden TCP/IP (per Default Port 5432)**
- **Datenbanken sind Unterverzeichnisse in `data/base`**
- **Caching wird hauptsächlich an den Dateisystem Page Cache delegiert**
- **Tabellen sind normale Dateien (pro GB segmentiert)**
- **Tablespaces sind symbolische Links in `pg_tblspc`**
- **Verbindungen sind dedizierte, geforkte Prozesse (genannt Backends)**
- **Prozesse setzen nützliche Namen für `ps`/`top`**
- **Unix-Signale werden verwendet**

Logische Architektur

- **Instanz (“Cluster”)**
  - Server Prozess mit Datenverzeichnis und `postgresql.conf`
  - Dedizierte IP-Adresse/Port Kombination
- **Rollen, Gruppen und Tablespaces sind globale/geteilte Objekte per Instanz**
- **Datenbanken (Default: `postgres`)**
  - Client verbindet sich zu dedizierter Datenbank
  - Kein Datenaustausch zwischen Datenbanken einer Instanz
  - Erweiterungen sind Datenbank-spezifisch
- **Schemas (Default: `public`)**
  - Namespace innerhalb einer Datenbank, `SELECT * FROM foo.t1;`
  - `public`-Schema erlaubt jedem Anlegen von Objekten
  - Schema-Suchpfad `search_path` erlaubt implizite Verwendung von Schemas



Best Practises Installation, Konfiguration und Monitoring

instaclustr

Installation

- **Community-Pakete verwenden**
  - `apt.postgresql.org` (identisch zu Upstream)
  - `yum.postgresql.org` (unterschiedlich zu Upstream)
  - `zypp.postgresql.org` (gleich wie `yum.postgresql.org`)
- **Unter Debian/Ubuntu `postgreql-common` Framework verwenden**
  - Instanz-Management/Wrapper
  - Konfiguration unter `/etc`
- **Dediziertes Dateisystem verwenden**
  - Datenverzeichnis mindestens ein Level unterhalb des Dateisystem-Roots
  - Erlaubt In-Place Upgrades (Link-Modus) zu neuer Instanz im gleichen Dateisystem
- **Overcommit abschalten**
  - Postgres kann schlecht mit OOM-Killer umgehen
  - Alle laufenden Abfragen werden abgebrochen und Verbindungen neugestartet

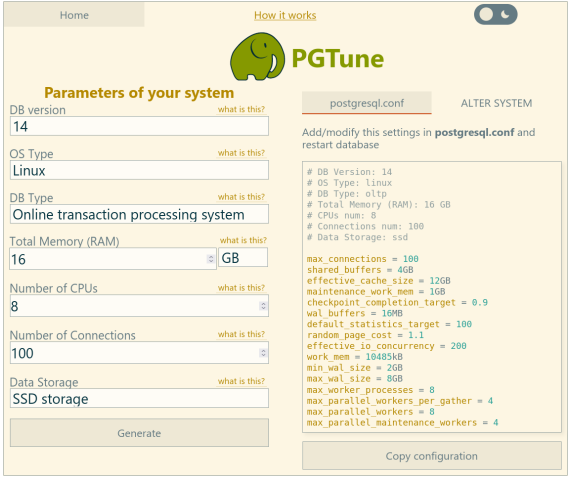
Installation

- **Dedizierten Server/VM für Postgres falls Performance wichtig ist**
- **Transaktions-Log Verzeichnis (`data/pg_wal`) kann auf eigenem Dateisystem sein mit Symlink**
- **Lokale NVME/SSD viel besser als iSCSI/SAN, aber letztere einfacher zu Verwenden bzw. erstere evtl. nicht verfügbar**
- **Managed Postgres Services in der Cloud einfach zu Verwenden**
  - Automatische Backups, Patching, Monitoring
  - Oft schlechtere I/O-Performance
  - Ermöglichen weniger Flexibilität (Erweiterungen usw.)

Konfiguration

- **Postgres ist nicht für exklusive System-Verwendung ausgerichtet**
- **Neuere Versionen haben deutlich bessere Defaults**
- **Einige Einstellungen benötigen einen Server-Neustart (Downtime)**
- **Wichtige vom lokalen System abhängige Konfigurations-Einstellungen**
  - `max_connections` → schwierig zu tunen
  - `shared_buffers` → 25% RAM
  - `effective_cache_size` 75% RAM
  - `work_mem` → schwierig zu tunen
  - `maintenance_work_mem` → 1 GB
  - `max_wal_size` → erhöhen bis Checkpoints zeitbasiert sind
  - `random_page_cost` → 1.0 für SSD/NVME
- **Mehrere (auch web-basierte) Konfigurations-Tuning tools/Services vorhanden**
  - `https://pgtune.leopard.in.ua/`, `pg_cloudconfig`

Konfiguration



Konfiguration

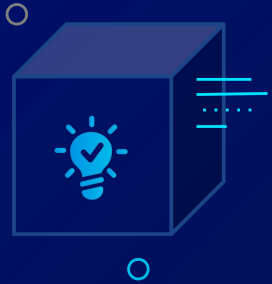
- **Nützliche initiale Einstellungen die später einen Neustart erfordern**
  - `shared_preload_libraries=pg_stat_statements`
  - `pg_stat_statements.max=10000`
  - `archive_mode=on`, `archive_command=/bin/true`
- **Nützliche Logging-Einstellungen**
  - `log_min_duration_statement=5s`
  - `log_lock_waits=on`
  - `log_temp_files=10MB`
- **Andere nützliche Einstellungen**
  - `idle_in_transaction_session_timeout=1h`

Konfiguration

- **Client-Verbindungen sollten SSL-verschlüsselt sein**
- **Passwort-basierte Verbindungen sollten SCRAM und nicht MD5 verwenden**
- **Alternativ Client-basierte SSL-Zertifikate oder LDAP/AD-basierte Authentifizierung**
- **Dedizierte Anwendungs-Nutzer, niemals `postgres`/`SUPERUSER` rausgeben**
  - Nutzer mit `CREATEROLE`-Privileg sind auch gefährlich
- **Wenn möglich ein eigenes Schema, ansonsten `REVOKE ALL ON SCHEMA public FROM PUBLIC`**

Monitoring

- **Das verwenden was bereits vorhanden ist**
- **Icinga für Alerting**
  - `check_postgres`, `check_pgactivity`
  - Nicht toll für Performance-Metriken
- **Prometheus/Grafana für Performance-Metriken**
  - Mehrere verfügbare Exporter, `postgres_exporter`, `sql_exporter`, …
- **pgBadger für Logdatei-Analyse**
  - HTML-Reports, auch hilfreich für Entwickler
- **POWA für Workload-Analyse**



Release- und Upgrade-Zyklus

instaclustr

Major Release Zyklus

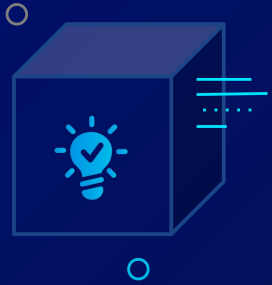
- **Eine Major-Version pro Jahr, Release üblicherweise in September/Oktober**

Version	Release Datum	Version	Release Datum
14	30.09.2021	9.5	07.01.2016
13	24.09.2020	9.4	18.12.2014
12	03.10.2019	9.3	09.09.2013
11	18.10.2018	9.2	10.09.2012
10	04.10.2017	9.1	12.09.2011
9.6	29.09.2016	9.0	20.09.2010

- **Release Management Team seit 2016**

Verlässliche und Lange Supportete Versionen

- **Zeitbasierter Code-Freeze (Q1), anschließende Beta-Phase**
  - Release erfolgt wenn keine schweren Bugs mehr vorhanden sind
- **Major-Releases werden 5 Jahre lang supportet (sog. Backbranches)**
  - Vierteljährliche, planbare Point-Releases für kritische und sicherheitsrelevante Bugs
  - Jeweils am zweiten Donnerstag des zweiten Monats eines Quartals
  - Zeitplan: `https://www.postgresql.org/developer/roadmap/`
  - Security Team behandelt Sicherheitsvorfälle
  - Im Notfall außerordentliche Point-Releases
    - 14.4 Release am 16. Juni 2022
    - 10.3, 9.6.8, … Release am 1. März 2018



Operations Best-Practises und Praktikable Limits

instaclustr

Daily Operations

- **Neustarts**
  - Beenden alle laufenden Verbindungen, keine neuen Verbindungen möglich
  - `smart`-Shutdown nicht verwenden (war Default)
  - Ausführen eines `CHECKPOINT` beschleunigt Runterfahren
- **Langlaufende, bösartige Abfragen oder Verbindungen**
  - Können folgendermaßen beendet werden
    - Via `kill` (nicht `kill -9`) auf die PID des Backends
    - Via `SELECT pg_cancel_backend('$PID')` von `psql` für Abfragen
    - Via `SELECT pg_terminate_backend('$PID')` von `psql` für wartende Transaktionen
  - PID herausfinden z.B. via `SELECT * FROM pg_stat_activity` oder aus dem Monitoring

Upgrades

- **Minor/Patch-Releases**
  - Neue Community-Pakete installieren
  - PostgreSQL neustarten
  - Bei Replikations-Setups zunächst den Standby patchen, dann Switchover, dann den früheren Primary
- **Major Upgrades**
  - Komplizierter, erfordern Downtime
  - In-Place (`pg_upgrade` im Link-Modus)
  - Dump/Restore (langsame)
  - Logische Replikation (kompliziert)

Hochverfügbarkeit

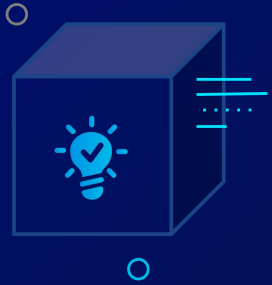
- **PostgreSQL physische Replikation ermöglicht Read-Replicas**
  - Aber lang-laufenden Abfragen auf dem/den Standby(s) können Replikation verzögern
- **Patroni viel-benutzte HA-Lösung, die auch die Administration vereinfacht**
  - Benötigt einen DCS (etcd, consul, …) der keine Ressourcen-Engpässe haben sollte
  - REST-Interface kann für Konfigurations-Änderungen oder zur Planung von Switchovers/Nestarts verwendet werden
- **Pacemaker gibt es weiterhin, ist aber kompliziert aufzusetzen und zu benutzen**
  - PAF Ressourcen-Agent macht es besser aber nicht perfekt
  - Timeouts sind ohne richtige Last-Tests schwierig einzustellen

Backups

- `pg_dump`/`pg_basebackup` **Programme ermöglichen einfache Backup-Lösungen**
  - Dumps sind für langfristige Archivierung, Schema-Versionierung und gezieltes Daten-Restore sinnvoll
- **pgBackRest/Barman sind nützliche dedizierte Backup-Lösungen mit Point-in-Time-Recovery**
  - pgBackRest ausgereifter, aber auch etwas schrullig
  - Mehrere mögliche Topologien (lokale Backups, zentraler Backup-Server)

Praktikable Limits

- **Bis 100 GB sind Postgres-Instanzen normalerweise unproblematisch**
  - Sofern genug Speicher vorhanden ist und es nicht zu viel Traffic gibt
  - Tägliche `pg_basebackup` Backups mit Xlog-Archivierung möglich
  - Oder tägliche (oder stündliche) Dumps
  - Gesamt-Performance üblicherweise OK, einzelne Queries können länger dauern
- **Bis 1 TB sind Postgres-Instanzen immer noch handhabbar**
  - Dedizierte Backup-Lösung (z.B. pgBackRest)
  - Tägliche Dumps könnten zu lange dauern, einmal am Wochenende machbar
  - Performance-Probleme kann bei Ressourcen-Knappheit oder schlechten Queries
- **Ab 1 TB werden üblicherweise DBAs oder externer Support benötigt**
  - Tabellen sollten ab ein paar Hundert GB Größer partitioniert werden
  - Autovacuum sollte für große Tabellen spezifisch eingestellt werden
  - Working Set passt vermutlich nicht in Hauptspeicher, verlangt Schema/Query-Tuning



Pitfalls

instaclustr

Pitfalls

- **Multi-Master Replikation als Lösung für alles ansehen**

Pitfalls

- **Multi-Master Replikation als Lösung für alles ansehen**
- **Nicht mehr unterstützte Major-Versionen verwenden**

Pitfalls

- **Multi-Master Replikation als Lösung für alles ansehen**
- **Nicht mehr unterstützte Major-Versionen verwenden**
- **Patch-Releases nicht einspielen**

Pitfalls

- Multi-Master Replikation als Lösung für alles ansehen
- Nicht mehr unterstützte Major-Versionen verwenden
- Patch-Releases nicht einspielen
- Datenverlust durch Änderungen an glibc-Sortierreihenfolgen bei OS Updates

Pitfalls

- Multi-Master Replikation als Lösung für alles ansehen
- Nicht mehr unterstützte Major-Versionen verwenden
- Patch-Releases nicht einspielen
- Datenverlust durch Änderungen an glibc-Sortierreihenfolgen bei OS Updates
- Unbedingt Tablespaces verwenden wollen

Pitfalls

- **Multi-Master Replikation als Lösung für alles ansehen**
- **Nicht mehr unterstützte Major-Versionen verwenden**
- **Patch-Releases nicht einspielen**
- **Datenverlust durch Änderungen an glibc-Sortierreihenfolgen bei OS Updates**
- **Unbedingt Tablespaces verwenden wollen**
- **Autovacuum abschalten**

instaclustr

www.instaclustr.com

info@instaclustr.com

@instaclustr