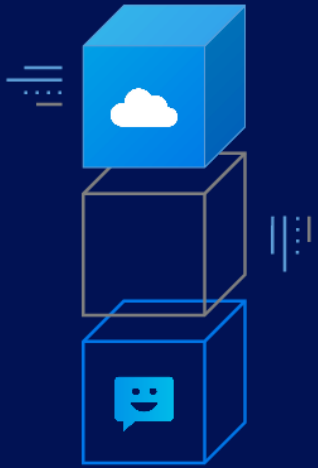
instaclustr

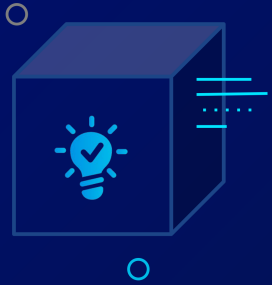
PostgreSQL Fallstricke



Michael Banck

13.5.2022

PGConf.DE, Leipzig



Allgemeine Fallstricke

instaclustr

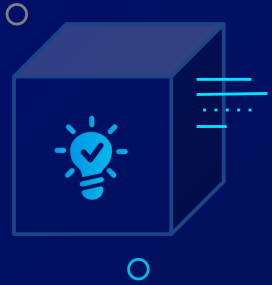
Allgemeine Fallstricke

- Feature X oder Verhalten Y erwarten, weil es in RDBMS Z auch so ist
- Multi-Master Replikation als Allheilmittel sehen
- Nicht mehr unterstützte Major-Releases verwenden
- Minor Releases nicht einspielen

Allgemeine Fallstricke

‘’While promptly updating PostgreSQL is the best remediation for most users, a user unable to do that can work around the vulnerability by disabling autovacuum, not manually running the above commands, and not restoring from output of the pg_dump command. Performance may degrade quickly under this workaround.”

`https://www.postgresql.org/about/news/postgresql-143-137-1211-1116-and-1021-released-2449/`



Autovacuum

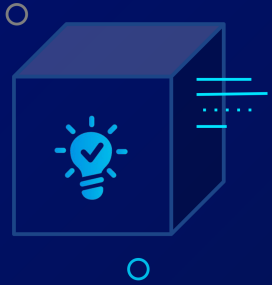
instaclustr

Autovacuum

- PostgreSQL verwendet MVCC und Transaktions-Snapshots
- `DELETE` und `UPDATE` Anweisungen löschen den ursprünglichen Datensatz nicht
- Autovauum räumt auf, wenn ursprüngliche Datensätze von keiner laufenden Transaktion mehr gesehen werden können
- Standardmäßig wird Autovacuum für eine Tabelle aufgerufen, wenn 20% der Datensätze gelöscht oder geändert wurden
- Bei sehr großen Tabellen kann dies ein starke I/O Aktivität von Autovacuum bedeuten

Autovacuum, der Fallstrick

- **Autovacuum braucht viele Ressourcen, also schaltet man es ab…**
  - Postgres ist wieder schneller, bis es langsam wird
- **Die Transaktions-IDs (XIDs) sind als 32bit-Zahlen in den Datensätzen gespeichert**
  - Alle ca. 2 Billionen Transaktion gäbe es einen sog. Wraparound
  - Damit das nicht passiert, muss Postres alle alten Datensätze “einfrieren”
  - Dies erforderte früher extrem viel I/O und ist heutzutage immer noch spürbar
- **Wenn ein Vacuum nicht möglich ist weil langlaufende Transaktonen es behindern wird Postgres drei Millionen Transaktionen vor Erreichen des Überlaufs heruntergefahren**
- **Workarounds: Autovacuum nicht ausschalten sondern tunen, langlaufende Transaktonen und Prepared Transactions beenden**



PUBLIC

instaclustr

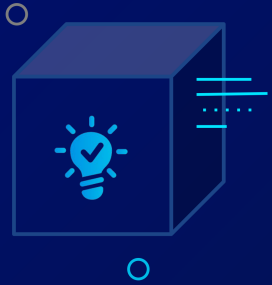
PUBLIC

- `public` **Schema ist das Standard-Schema, wenn kein Schema-Suchpfad oder explizites Schema angegeben ist**
- **Das** `public`**-Schema ist im Standard Schema-Suchpfad**
- `PUBLIC` **ist eine Pseudo-Gruppe, jeder Nutzer ist implizit Mitglied**
- **Erlaubt das einfache Anlegen von eigenen Objekten für eingeloggte Nutzer**

PUBLIC, der Fallstrick

- `PUBLIC` **Gruppe hat weitreichende Nutzungs-Rechte**
  - `CONNECT` auf Datenbanken
  - `USAGE` auf `LANGUAGE PLPGSQL` und `LANGUAGE SQL`
  - `USAGE`, `CREATE` für Schema `public`
  - `EXECUTE` auf Funktionen
- **Erlaubt die Erstellung von Funktionen, die gleichen Namen wie Systemfunktionen haben.**
  - Funktionen in `public` sind vor dem Systemkatalog `pg_catalog` im Schema-Suchpfad
- **Workaround:** `REVOKE ALL ON SCHEMA public FROM PUBLIC;`

`http://joeconway.com/presentations/security-FOSDEM2020.pdf`



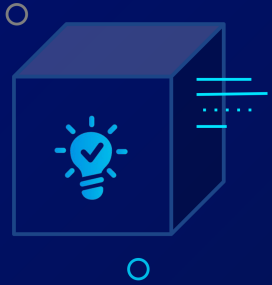
Tablespaces

Tablespaces

- Tablespaces erlauben die Plazierung von Tabellen/Indexe außerhalb des Datenverzeichnisses
- Werden werden in anderen Datenbank-Produkten stark verwendet

Tablespaces, der Fallstrick

- In Postgres sind es lediglich symbolische Links auf ein Verzeichnis
- Hilfreich, wenn Verzeichnis auf anderem (z.B. schnelleren) Dateisystem liegt
- Ansonsten nur administrativer Overhead
- Basebackups benötigen evtl. Tablespace-Mapping
- Tablespaces innerhalb des Datenverzeichnisses werden nicht mehr unterstützt
- Workaround: Tablespaces nur bei dringenden Bedarf verwenden und wenn dann außerhalb des Datenverzeichnisses



pg_hba Zugriff

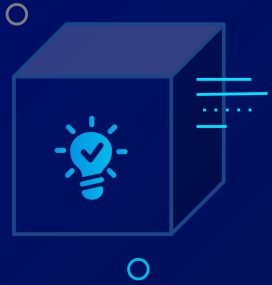
instaclustr

pg_hba Zugriff

- PostgreSQL hat “Host Based Access” (HBA) für das IP-Whitelisting
- `pg_hba.conf` konfiguriert, welche Datenbank-Clients von wo wie zugreifen dürfen
- Viele verschiedene Methoden, sowie sonstige Felder
- Methode `trust` erlaubt den Passwort-losen Zugriff
- Methode `md5` erlaubt den Zugriff mit einem gehashten Passwort
- Method `ldap` erlaubt den Zugriff via (z.B.) Active-Directory

pg_hba Zugriff, die Fallstricke

- **`trust` erlaubt jedem beliebigen Nutzer sich als der angegebene Nutzer einzuloggen**
  - Muss lediglich von der angegebenen Quell-IP Adresse kommen
  - Insbesondere problematisch, wenn `SUPERUSER` wie `postgres` mit `trust` versehen sind
- **Workaround: `trust` nicht verwenden**
- **Kenntnis des `md5` Hashs erlaubt einen direkten Login für eine Benutzerrolle, ohne das Passwort kennen zu müßsen**
  - Mit normalen `libpq`-basierten Programmen nicht möglich
  - MD5-Hash wird mit einem 4-Byte Salt übertragen
- **Woraround: `scram-sha-256` verwenden**
- **`ldap` überträgt das Passwort im Klartext**
- **Workaround: SSL oder `ldaps`-Optionen verwenden**



CREATEROLE

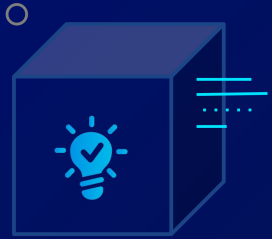
instaclustr

CREATEROLE

- `CREATEROLE` **Recht erlaubt es einem Nutzer, neue Rollen zu erzeugen und Nutzern Gruppen zuzuweisen**
- **Wird oft verwendet, um Nutzer-Management an Mandanten-Ansprechpartner zu delegieren, der keine** `SUPERUSER`**-Rechte haben soll**
- **Jeder Cloud-Provider bietet einen Admin-Account mit** `CREATEROLE` **aber ohne** `SUPERUSER`

CREATEROLE, der Fallstrick

- **Seit Version 11 gibt es die Default-Rolle** `pg_execute_server_program`
- **Ein** `CREATEROLE`**-Nutzer kann sich (oder anderen) dieses Privileg geben**
- **Trivialer** `SUPERUSER`**-Exploit**
- **Insbesondere Problematisch für Upgrades von früheren Instanzen, wo dieses Problem (so) nicht bestand**
- **Workaround:** `DROP ROLE pg_execute_server_program;`



Betriebssystem-Updates

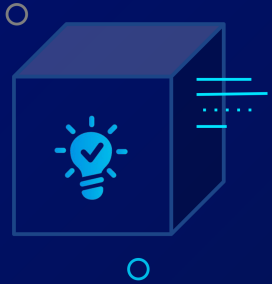
instaclustr

Betriebssystem-Updates

- Betriebssystem-Updates sind nötig, um im Support-Fenster zu bleiben
- Im Prinzip problemlos, sofern die neue Version Postgres-Pakete mit der gleichen Version bereitstellt

Betriebssystem-Updates, der Fallstrick

- **Das Betriebssytem-Update kann die Version der `glibc`-Bibliothek ändern**
  - Evtl. sogar bei Service-Packs
- **Die `glibc` ist unter Linux für die Sortierreihenfolge verantwortlich**
- **Neue `glibc`-Versionen können diese für eine oder mehrere Sprachen ändern**
- **Das führt zu Daten/Index-Korruption**
  - Mit dem `amcheck`-Programm identifizierbar
- **Workaround: `REINDEX` der Text-basierten Spalten**
- `https://wiki.postgresql.org/wiki/Locale_data_changes`



SERIAL

instaclustr

SERIAL

- `SERIAL` **ist ein Datentyp, der im Hintergrund eine Sequenz bereitstellt**
- **Wird üblicherweise zur Generierung von Primärschlüsseln verwendet**
- **Vor Version 12 die einzige Möglichkeit elegant Sequenzen zu verwenden**
  - Inzwischen `GENERATED BY DEFAULT AS IDENTITY`

SERIAL, der Fallstrick

- `SERIAL` **verwendet seit Version 10 eine Sequenz von Typ** `integer` **(32-bit)**
  - Vorher `bigint` (64-bit)
- **Wenn eine Tabelle eine** `SERIAL` **Sequenz als Primärschlüssel verwendet sind "nur" 2 Milliarden Zeilen möglich**
  - `ERROR: integer out of range`
- **Workaround:** `ALTER SEQUENCE <sequence> MINVALUE -2147483648 RESTART -2147483648;`
- **Workaround: Konvertierung nach** `BIGSERIAL`/`bigint` **(nicht trivial bei großen Tabellen)**

instaclustr

www.instaclustr.com

info@instaclustr.com

@instaclustr